

ADVERTIMENT. L'accés als continguts d'aquesta tesi queda condicionat a l'acceptació de les condicions d'ús establertes per la següent llicència Creative Commons:  <https://creativecommons.org/licenses/?lang=ca>

ADVERTENCIA. El acceso a los contenidos de esta tesis queda condicionado a la aceptación de las condiciones de uso establecidas por la siguiente licencia Creative Commons:  <https://creativecommons.org/licenses/?lang=es>

WARNING. The access to the contents of this doctoral thesis it is limited to the acceptance of the use conditions set by the following Creative Commons license:  <https://creativecommons.org/licenses/?lang=en>

UNIVERSITAT AUTÒNOMA DE BARCELONA

Departament d'Arquitectura de Computadors i Sistemes Operatius
Escola d'Enginyeria

Massive cosmological data generation and distribution

By
PAU TALLADA CRESPI

ADVISORS:

NADIA TONELLO
JORGE CARRETERO PALACIOS

TUTOR:

EDUARDO CÉSAR GALOBARDES

*Thesis belonging to the research line of High Performance Computing,
for the degree of Philosophiae Doctor in Computer Science*

UAB
Universitat Autònoma
de Barcelona

Barcelona, abril de 2024

MASSIVE COSMOLOGICAL DATA GENERATION AND DISTRIBUTION

Thesis submitted by Pau Tallada Crespi for the Doctor of Philosophy degree in Computer Science. This doctoral thesis belongs to the High Performance Computing research line, performed in the Computer Architecture and Operative Systems department in Universitat Autònoma de Barcelona, under the supervision of Dr. Nadia Tonello, Dr. Jorge Carretero Palacios and Dr. Eduardo César Galobardes.

12th April 2024

PHD STUDENT:

CO-DIRECTORS:

This work has been granted by the Ministerio de Ciencia e Innovación MCIN AEI/10.13039/501100011033 under contract PID2020-113614RB-C21 and by the Generalitat de Catalunya under contract 2021 SGR 00574.

Abstract

In recent decades, physicists and astronomers have significantly transformed their methodology for investigating the universe's content and evolution. Advanced computing techniques have emerged as indispensable tools to manage the substantial data amassed by contemporary automated telescopes and highly sensitive instruments. Extracting scientific insights from the vast information pool necessitates interdisciplinary collaboration among mechanical and electronic engineers, physicists, astronomers, computer scientists, and software engineers.

This PhD thesis explores the interface of Computer Science and Cosmology within the Port d'Informació Científica (PIC), a High Throughput Computing (HTC) data center. The work encompasses two core domains: (comprehensive) data management and the advancement of (complex) algorithms for cosmological simulations.

In the realm of data management, conventional tools like relational databases are usually employed. In this work, a pioneering stance is taken towards them, exemplified by their central role in the Physics of the Accelerating Universe Survey (PAUS). The design of a comprehensive data management infrastructure within the tight constraints of PAUS is the first contribution in this thesis.

Moreover, given the limitations of relational databases in handling extensive data and evolving usage patterns, this study also delves into alternatives. The challenges in the distribution of cosmological catalogs within the PAUS collaboration lead to the adoption of the Apache Hadoop ecosystem. This investigation culminated in the creation of CosmoHub, an application leveraging Apache Hive—an unprecedented endeavor within astronomy and cosmology—that promotes Open Science principles.

Concurrently, in the domain of algorithm development for cosmological simulations, this thesis describes the effort in developing, optimizing and calibrating an algorithm for the simulation of observed galaxy electromagnetic fluxes. This algorithm, integrated into a much larger set of Python modules within a Spark-driven pipeline operating on a Hadoop cluster, is crucial to the creation of the most extensive and comprehensive virtual galaxy catalogs, serving the European Space Agency's Euclid project.

Resum

En les darreres dècades, físics i astrònoms han transformat radicalment la seva metodologia per investigar el contingut i l'evolució de l'univers. Les tècniques informàtiques avançades s'han convertit en eines indispensables per gestionar la gran quantitat de dades produïdes actualment pels telescopis automatitzats i altres instruments d'alta sensibilitat. Per poder extreure coneixements científics d'aquesta vasta quantitat de dades es requereix la col·laboració interdisciplinària d'enginyers mecànics i electrònics, físics, astrònoms, informàtics i enginyers de programari.

Aquesta tesi doctoral explora la interfície entre l'enginyeria de informàtica i la cosmologia dins del Port d'Informació Científica (PIC), un centre de dades i computació científica d'alt rendiment (HTC). Les contribucions es centren en dues àrees centrals: la gestió integral de les dades i el desenvolupament d'algoritmes complexos per a simulacions cosmològiques.

En l'àmbit de la gestió de dades, es solen emprar habitualment eines com les bases de dades relacionals. En aquesta tesi hem anat un pas més enllà, en situar-la en un posició pionera i central en l'arquitectura de gestió de dades del projecte de cartografiat extragalàctic Física de l'Univers Accelerat (PAUS, de l'anglès). El disseny d'una infraestructura integral de gestió de dades, dins de les estrictes limitacions del projecte PAUS, és la primera contribució d'aquesta recerca.

A més, ateses les limitacions de les bases de dades relacionals a l'hora de gestionar grans volums de dades i el seus patrons d'ús en constant evolució, aquesta tesi també aprofundeix en l'estudi d'alternatives. Els reptes en la distribució de catàlegs cosmològics dins de la col·laboració PAUS ens han portat a l'adopció de l'ecosistema Apache Hadoop. Aquesta línia de recerca va derivar en la creació de CosmoHub, una aplicació impulsada per Apache Hive —un esforç sense precedents en astronomia i cosmologia— i que promou els principis de la Ciència Oberta.

Paral·lelament, en el camp del desenvolupament d'algoritmes per a simulacions cosmològiques, aquesta tesi descriu l'esforç per desenvolupar, optimitzar i calibrar un algoritme per simular els fluxos electromagnètics observats de les galàxies. Aquest algoritme, integrat en un conjunt molt més ampli de mòduls Python dins d'un *pipeline* impulsat per Spark, és fonamental per a la creació dels catàlegs virtuals de galàxies més extensos i complets, que donen suport al projecte Euclid de l'Agència Espacial Europea.

Agraïments

Si bé aquesta tesi porta el meu nom, no hauria estat possible sense el suport i la contribució de tots vosaltres. Si estàs llegint aquestes línies, ben segur has deixat la teva empremta. Com bé va dir Sir Isaac Newton: "*si he vist més lluny que altres, és posant-me sobre les espatlles de gegants*". Doncs jo m'he pogut enfilat damunt d'un bon caramull de gegants!

Als meus pares, als qui tenc tant a agrair que manquen les paraules, pel seu suport i encoratjament constants sense els que no hagués arribat mai tan lluny. A tota la meua família, per les rialles i el bon rotllo, però també el conhort dels moments complicats. A n'Arnau, per haver estat sempre al meu costat. Ets un model a seguir, una persona brillant, un treballador incansable i un idealista que m'has fet millor persona. A na Marta, la meua fidel confident, per la complicitat de tants anys que ens uneix d'una manera tan especial com indescriptible, gràcies per ser com ets!

Als meus directors, per l'ajut incansable i tot el seu suport al llarg d'aquesta travessa. A na Nadia, sempre a punt amb un somriure i unes paraules encoratjadores. A en Jorge, el meu company de trinxeres, treballador infatigable i instigador de tantes aventures. A n'Eduardo, la veu de la mesura i la concòrdia, i el capità que guia a bon port.

Als meus amics, en especial als *palos*, amb qui he compartit tants bons moments i que m'han hagut de sentir remugar a bastament durant aquests anys. I a tots els companys del PIC, que sense la seva infinita experiència i saviesa res del que aquí escric no haguera estat possible.

A tots, gràcies de tot cor!

Pau

Contents

1	Introduction	15
1.1	Scientific motivation	16
1.2	Galaxy Surveys	17
1.2.1	The Physics of the Accelerating Universe	19
1.2.2	Euclid Mission	20
1.3	Cosmological simulations	22
1.3.1	Euclid Flagship simulations	23
1.4	Data management and Infrastructure	24
1.4.1	The Port d’Informació Científica	26
1.5	Contributions	27
1.6	Related work	31
2	PAU Survey data management	35
2.1	Introduction	37
2.2	PAUS data center	39
2.3	The PAUS database (PAUdb): metadata integrity and preservation	40
2.3.1	PAUdb data model	41
2.3.2	Limitations and improvements	42
2.4	Short term storage and data transfer	42
2.5	Analysis pipelines and nightly report availability	43
2.6	Job orchestration tool: BT	46
2.7	Operations and metadata exploitation and distribution: PAUdm web services	47
2.7.1	Operations control	48
2.7.2	Data Quality inspection	49
2.7.3	Nightly Report	49
2.7.4	Files and metadata access and distribution	49
2.7.5	Catalog distribution: CosmoHub origins	51
2.8	Conclusions	52
2.A	PAU database schema	54

3	CosmoHub	59
3.1	Introduction	61
3.2	User requirements	62
3.3	General architecture	64
3.4	Early prototypes	66
3.5	Hadoop Platform	68
3.5.1	Data management	71
3.6	Download formats	72
3.7	User Defined Functions (UDFs)	80
3.7.1	HEALPix	80
3.7.2	Arrays	83
3.7.3	ADQL	84
3.8	Web application	89
3.9	Results	98
3.9.1	Quantitative analysis	98
3.9.2	Scientific applications	101
3.10	Conclusions and future work	102
3.A	Particularly useful applications	104
4	Simulating galaxy fluxes	107
4.1	SciPIC	110
4.2	Computing fluxes for simulations	112
4.3	Input data	116
4.3.1	Galaxy templates and extinction laws	116
4.3.2	Filters	117
4.3.3	Emission lines	119
4.3.4	Milky Way extinction	119
4.4	SciPIC approximated approach	121
4.4.1	SED flux contribution	121
4.4.2	Emission line contribution	122
4.4.3	Milky Way extinction contribution	122
4.5	Results	127
4.6	Conclusions	133
5	Conclusions and future work	135
	References	143

List of Figures

1.1	PAUCam mounted at the primary focus of the WHT	20
1.2	Euclid being launched on July 1st 2023	21
1.3	Visualization of a Euclid Flagship mock galaxy catalog	25
2.1	PAUCam filter exchange and filter tray design	36
2.2	PAU infrastructure and services	39
2.3	Schematic organization of the main PAUdb tables	41
2.4	Daily average transfer speed registered at PIC for PAU Survey data	44
2.5	Dependencies of every PAU data management pipeline	45
2.6	Dependencies of <i>transfer</i> , <i>register</i> and <i>nightly</i> pipelines	45
2.7	PAUS: monthly sum of the number of jobs and wall time	46
2.8	View of the PAUS web interface for operations control	48
2.9	PAUS web interface: Nightly Report	50
2.10	PAUdm catalogs distributed in CosmoHub	51
3.1	CosmoHub’s data model	64
3.2	Hadoop layered ecosystem	67
3.3	Hadoop cluster architecture	70
3.4	Generation of a single CSV.BZ2 download file	75
3.5	Generation of a single FITS/ADSF download file	78
3.6	Generation of a single Parquet download file	79
3.7	Orthographic view of HEALPix partition of the sphere	81
3.8	Cylindrical projection of the HEALPix division of a sphere	82
3.9	UDAF array functions working scheme	84
3.10	CosmoHub’s initial page, showcasing the main projects and features	94
3.11	CosmoHub’s catalog page upper half, steps 0-3	95
3.12	CosmoHub’s catalog page bottom half, steps 4-7	96
3.13	1D histogram over MICECAT2 data	97
3.14	2D heatmap displaying a color-magnitude diagram of MICECAT2	98
3.15	CosmoHub’s activity page	99
3.16	Evolution of size and number of catalogs of CosmoHub over time	99
3.17	CosmoHub’s monthly average processing time	100
3.18	CosmoHub’s cummulative completion ratio as a factor of processing time	100
3.19	Active users in CosmoHub over time	101

4.1	Galaxy flux computation flow	114
4.2	COSMOS SED templates used in SciPIC	116
4.3	Extinction laws used in SciPIC	117
4.4	Euclid Standard Bandpasses	118
4.5	A subset of PAU narrow band filters	118
4.6	The colour excess values of every sky position	120
4.7	O'Donnell extinction function	120
4.8	Error introduced by interpolating across SED bins	123
4.9	Error introduced by interpolating across $E(B-V)$ bins	124
4.10	Error introduced by interpolating across redshift bins	125
4.11	Error introduced by interpolation across all dimensions	126
4.12	Error introduced by the SED flux approximation	128
4.13	Error introduced by the emission lines flux approximation	129
4.14	Error introduced by the Milky way extinction approximation	130
4.15	Error introduced by the combination of all approximations	131

List of Tables

1.1	Most relevant galaxy redshift surveys of last decades	18
2.1	PAUdb tables	55
2.2	PAUdb nightly calibration tables	56
2.3	PAUdb <i>membra</i> and catalogs tables	57
2.4	PAUdb BT tables	58
2.5	PAUdb Public external tables	58
3.1	Custom struct data type to store geometric data	87
3.2	List of CosmoHub API endpoints, grouped by entity	90
3.3	List of public catalogs in CosmoHub	102
4.1	Valid combinations of COSMOS SEDs and extinction laws	117
4.2	Emission lines with their wavelengths both in air and in vacuum	119
4.3	Binning configuration for each parameter and filter set	121
4.4	Percentage of galaxies that fulfil the accuracy requirement	132
4.5	Average flux computation time per galaxy and band, for both methods . . .	133

Chapter 1

Introduction

The primary focus of this thesis is the innovative development of data management tools and the refinement of complex algorithms, conducted within extensive international collaborations aimed at advancing our comprehension of the Cosmos. The scientific concepts behind this work, specifically within the field of cosmology, will be introduced, since they are essential to understand the challenges encountered throughout this research journey.

The field of expertise gap between "pure scientists" and "pure engineers" often results in ~~poor~~poor communication, reduced efficiency and frustration. A new field of specialization has emerged in the last decades, the research software engineers (RSE, Woolston, 2022), with the crucial responsibility to bridge this gap between science and technology, not only to optimize overall efficiency, but to ensure project success. The RSE role has been gaining traction in parallel with the complexity of the challenges faced in the current scientific projects, which are constantly increasing in technical requirements and number of people involved.

RSE are professionals who combine expertise in software development and engineering with domain-specific knowledge and skills (Goth et al., 2023). They work with scientists solve their computing challenges by providing tailored solutions that meet their needs and requirements. RSE's fundamental role in scientific research is to enable projects to make the best use of the available data and resources (Deschamps et al., 2023).

This thesis describes some of my most important contributions to this emerging field of research, with a particular focus on cosmology. In particular, I present my efforts in three different but related areas.

Chapter 2 describes the effort of designing, developing and operating a data management facility for the Physics of the Accelerating Universe Survey (PAUS¹). The main challenge in this contribution was to devise, with very little resources, a comprehensive data management architecture that would cover the entire data life cycle of the survey: from data acquisition at the telescope, its transfer to the scientific data center, the reduction and analysis of the images through the orchestration of thousands of jobs in multiple pipelines, the archival of all the data generated and the final distribution of results to the scientific community.

¹<https://pausurvey.org/>

Chapter 3 explains the design, development and operation of CosmoHub, an application for the interactive exploration and distribution of large structured datasets, focused on astronomical catalogs. This contribution is an evolution of the initial work done to make PAUS results available to the public. The main challenges in this effort were the increasingly growing size of the datasets we have to work with, the absence of a common access pattern that could be exploited to prearrange the data in a more efficient way, and the lack of specific knowledge and/or experience among scientists to deal with relational data.

Chapter 4 details the design, optimization and calibration of an algorithm for the simulation of observed galaxy fluxes². This work is part of a much larger effort to design, develop and operate a pipeline for the generation of synthetic galaxy catalogs from cosmological simulations called Scientific Pipeline at Port d'Informació Científica (SciPIC). In particular, it is crucial for the design, planning, calibration and science exploitation of the European Space Agency (ESA³) Euclid⁴ mission. The main challenge in this contribution was to be able to produce large amounts of simulated fluxes in a short amount of time and within very restrictive accuracy margins.

This introductory chapter is structured as follows: first a scientific motivation about why research along several cosmological topics has been deemed fundamental; second an introduction about galaxy surveys and how they contribute to the advancement of knowledge on those topics; third a presentation of the concept of cosmological simulations and their paramount importance in the planning, execution and exploitation of galaxy surveys; fourth a description of how an adequate data management and infrastructure is essential to handle the massive data volumes of both galaxy surveys and cosmological simulations and crucial to the success of both kind of projects; fifth a summary of the main contributions of this thesis; and a final section to describe the related work and its influence in this thesis.

1.1 Scientific motivation

Among the 12 fundamental questions in astronomy identified as the most pressing ones in the Astronet Roadmap Executive Summary⁵ (a European strategic plan for astronomical research and infrastructure), three of them are directly related to Cosmology:

- What is the nature of dark matter and dark energy?
- Are there deviations from the standard theories and models (general relativity, cosmological model, standard model of particle physics)?
- How do galaxies form and evolve?

Since the discovery of the accelerating expansion of the Universe, a wide range of experiments have been conducted or planned to answer these questions, which have established that the Universe consists of 95% dark energy and dark matter, or that we may need to

²also known as electromagnetic flux, it is the amount of energy captured per unit of time and surface.

³<https://www.esa.int/>

⁴<https://sci.esa.int/web/euclid>

⁵https://www.astronet-eu.org/?page_id=521

rewrite the fundamental laws of physics (Abbott et al., 2019; Alam et al., 2017, 2021; D’Amico et al., 2020; Ivanov et al., 2020; Sánchez et al., 2016; Tegmark et al., 2006, e.g.)). Galaxy surveys have played (Asgari et al., 2021; Porredon et al., 2022) and will play (Amendola et al., 2018; DESI Collaboration et al., 2016a, 2016b; Ivezić et al., 2019) a central role in the discovery and understanding of the constituents of the Universe. The standard model of cosmology and galaxy formation, postulates that galaxies are formed in overdense dark matter regions. This allows galaxy surveys to investigate models of the Universe by comparing either the galaxy distribution (galaxy clustering) or a statistical measure on how space bending light changes their shape (galaxy lensing). Galaxy surveys, therefore, provide multiple probes for testing theories of our Universe, which must be internally consistent and also agree with multiple other cosmological probes, including the cosmic microwave background (CMB, Ferreira 2019; Huterer et al. 2015; Peebles 1980; Percival and White 2009; Planck Collaboration et al. 2016; Weinberg et al. 2013) and local velocity measurements.

In the past two decades cosmology has undergone a transition towards precision science⁵ and computing has played a key role in this transition, in particular with two fundamental related aspects, "Cosmological simulations" and "Infrastructure and (comprehensive) Data management", for both simulated and observed data.

On the one hand, cosmological simulations are indispensable for the design and planning of large-scale extragalactic surveys. Moreover, systematic errors have become more important than statistical errors, and simulations have become one of the fundamental tools for measuring, calibrating, and even eliminating them. This vision is also supported by Astronet:

Simulations of increasing complexity, including higher dynamic range, resolution, and more physics exploiting increases in computing power, are harnessed to interpret results, and sometimes to show vividly the outcomes of the research. [...] Besides these new instruments, future facilities also rely increasingly on computers and data science as well as on simulations to maximize their return.

On the other hand, the quantity and complexity of the data generated, both from observations and from the simulations themselves, have exponentially increased. Both infrastructure and comprehensive data management are crucial for the success and scientific impact of the experiments.

In the following sections we will describe in more detail these two fundamental aspects identified in the latest Astronet’s strategic report, and how the contributions detailed in this work fit in those areas.

1.2 Galaxy Surveys

The accelerating expansion of the Universe (see Riess et al., 1998 and Perlmutter et al., 1999), discovered in the late 1990s by two independent groups through observations of distant supernovae, has been confirmed by many experiments through different cosmological probes. The physical explanation for this accelerating expansion is still a mystery. However, there are many theories explaining it, the most promising ones rely on studying

Date	Survey name	Data volume	Catalog size
1977 - 1982	CfA	N/A	$2.4 \cdot 10^3$
1985 - 1995	CfA2	N/A	$2.0 \cdot 10^4$
1997 - 2002	2dF	N/A	$3.8 \cdot 10^5$
2000 - 2014	SDSS (DR17)	245 TiB	$4.7 \cdot 10^6$
2013 - 2018	DES (DR2)	2 PiB	$6.9 \cdot 10^8$
2023 - 2029	Euclid	~ 30 PiB	$1.0 \cdot 10^{10}$
2024 - 2034	LSST	~ 60 PiB	$3.7 \cdot 10^{10}$

Table 1.1: Most relevant galaxy redshift surveys of last decades, showing how their data volume and catalog size grows over time.

the content of the universe. As far as we know, ordinary matter only accounts for 5% of its total energy, while two other unknown entities, called dark energy and dark matter, account for the remaining 95%. Dark energy is the name given to an unknown form of energy, which is the most accepted hypothesis to explain the accelerating expansion of the universe.

Galaxy surveys are one of the most powerful tools to characterise the nature of dark energy. Their main goal is to determine, with the highest precision possible, the position of distant sources by measuring their redshift⁶ and angular positions. Then cosmologists can create a 3D map of the distribution of galaxies of the observed region of the sky and estimate statistical properties of the large scale structure of the Universe, which can strongly constrain the cosmological parameters that drive the dynamics of the Universe. In practice, due to the sheer volume of data involved and its complexity, such analysis has become nearly impossible without developing innovative solutions for data management and analysis.

Galaxy surveys are usually divided in two main categories depending on the technique used to measure the flux of photons as a function of their energy (or as a function of their wavelength), which constitute the so-called galaxy spectrum. On the one hand, spectroscopic surveys observe each individual galaxy spectra and measure how it has been shifted by its relative velocity. This method delivers very precise estimations of the redshift but, as it requires the study of individual observations for each galaxy, it can only cover very small portions⁷ of the sky. On the other hand, photometric surveys estimate the redshift of a source by analysing the observed flux on a set of images taken using a few⁸ broad-band filters. As multiple sources can be observed in each image, the amount of objects that can be measured is much greater than for spectroscopic surveys. However, as the spectra information collected is sparse (one data point per filter), the precision is not as good.

⁶Cosmological redshift signifies the stretching of light from distant objects due to expanding space, offering a valuable tool for estimating both their distance and the universe's age.

⁷Modern surveys such as the Dark Energy Spectroscopic Instrument (DESI) are starting to change this. Compared to traditional spectroscopic surveys, DESI boasts an impressive 10x increase in observation capacity. This is achieved through its innovative design, featuring 5,000 robotic fiber-positioners that efficiently collect light from multiple objects simultaneously.

⁸Much as DESI is changing the approach for spectroscopic surveys, PAUS is doing the same by using a much larger set of filters than traditional photometric surveys.

The data in Table 1.1 shows some information of the historically most relevant wide area galaxy redshift surveys: CfA⁹, 2dF¹⁰, Sloan Digital Sky Survey¹¹ (SDSS), Dark Energy Survey¹² (DES), Euclid¹³, Legacy Survey of Space and Time¹⁴ (LSST). It is safe to say that in the last decade the observational cosmology has entered into a different data regime.

In the next section subsections I will introduce two very different galaxy surveys on which I have been contributing most of the work described in this thesis.

1.2.1 The Physics of the Accelerating Universe

The Physics of the Accelerating Universe Survey (PAUS) is a galaxy survey aimed at observing an extensive portion of the sky using narrow-band filters integrated within the PAU Camera (PAUCam, Padilla et al. 2019). The main objective of this project is to determine the spectral energy distributions of observed galaxies, surpassing the capabilities of broad-band imaging, thereby facilitating the more accurate determination of their photometric redshifts.

PAUCam, the instrumental centerpiece of this survey, is strategically affixed to the prime focal point of the 4-meter class William Herschel Telescope (see Figure 1.1), located at the "El Roque de los Muchachos" Observatory (ORM) on La Palma island (Spain). It is equipped with six broad-band filters (u, g, r, i, z, and Y) with the same design as the DES filters and 40 narrow-band filters that collectively span the wavelength range from 4500 to 8500 Ångströms. These narrow-band filters exhibit a width of 135 Ångströms, spaced at 100 Ångströms intervals (see Padilla et al., 2019 for more details).

The PAU Survey is a unique approach to measuring galaxy redshifts because it uses these 40 narrow-band filters to achieve precision similar to spectroscopy but covering larger areas like photometric surveys. This makes it particularly useful for calibrating other (much larger) photometric redshift surveys. In addition to calibration, the PAU survey has a number of other unique capabilities that spectroscopic surveys cannot cover, such as measuring the redshifts of galaxies in very crowded regions, or from very faint galaxies.

The target selection strategy of the PAU Survey is aligned with the CFHTLS (Canada-France-Hawaii Telescope Legacy Survey) fields, particularly focusing on areas where deep imaging had previously yielded galaxy shape measurements, as documented in Heymans et al. 2012 and Erben et al. 2013. The combination of the lensing measurements of the source galaxies with the photometric redshifts of the lensing galaxies can enable a variety of studies (e.g., Gaztañaga et al., 2012)

The main technical challenge of this project, besides the constructions of PAUCam, lies in the need to design, implement and operate a data management architecture that covers the entire data lifecycle, from image acquisition at the WHT telescope, to its transfer to the PIC data center, its analysis and storage, and finally, making the results available to the scientific community. The results of this effort are documented in detail in chapter 2, and have also been published in Tonello et al., 2019 and Serrano et al., 2022.

⁹<https://www.cfa.harvard.edu/~dfabricant/huchra/zcat/>

¹⁰<http://www.2dfgrs.net/>

¹¹<https://www.sdss.org/>

¹²<https://www.darkenergysurvey.org/>

¹³<http://www.euclid-ec.org/>

¹⁴<http://www.lsst.org/>

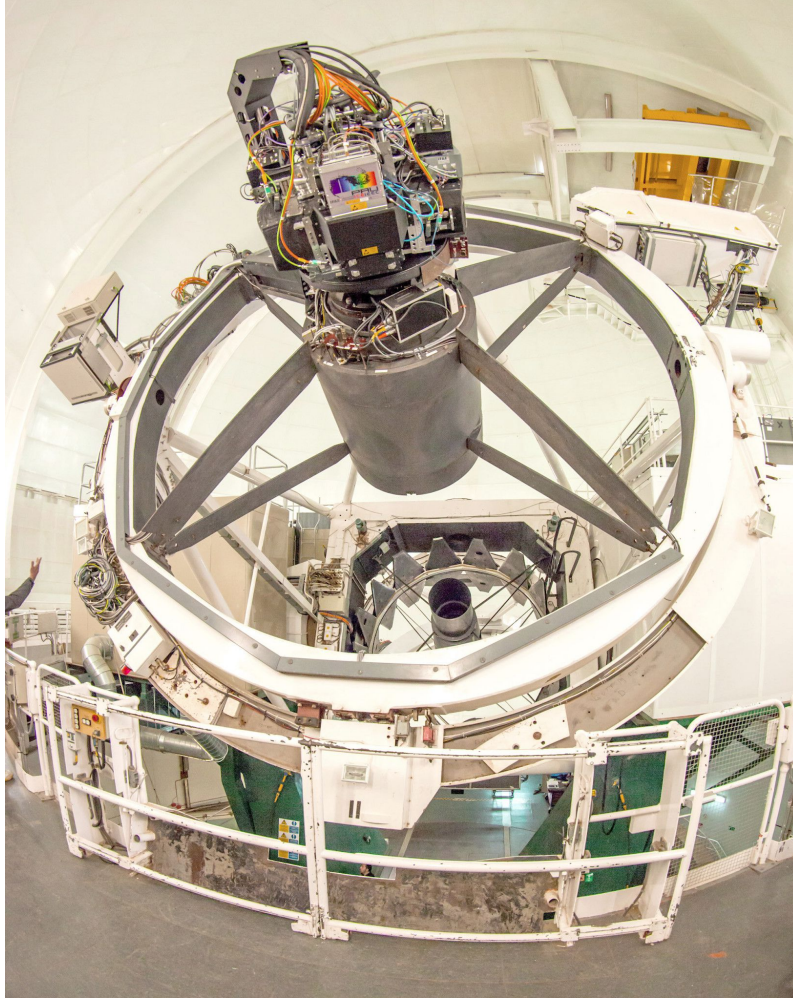


Figure 1.1: PAUCam mounted at the primary focus of the WHT. The 4.2m mirror, with the secondary focus at its center, can be seen with the petals opened and ready to start taking the first images during the commissioning phase.

On the scientific side, the challenge is to design, implement, and calibrate a set of processing pipelines that allow for accurate measurement of the photometric redshift of the objects observed in the images to deliver a resulting 3D map in the form of a catalog. This goal has also been achieved, as attested by Eriksen et al., 2019, Eriksen et al., 2020, Alarcon et al., 2021, Soo et al., 2021, and Navarro-Gironés et al. in prep.

1.2.2 Euclid Mission

The ESA Euclid mission is a space-borne survey mission dedicated to investigate the origin of the Universe's accelerating expansion and the nature of dark energy, dark matter and gravity. Dark energy is a hypothetical form of energy that permeates all space and exerts a negative pressure that drives the accelerated expansion of the universe. Dark matter is a form of matter that does not interact with light or other forms of electromagnetic radiation but can be detected through its gravitational effects on visible matter.



Figure 1.2: Euclid was launched successfully on July 1st 2023 at 11:12 local time from Cape Canaveral SLC-40 launch platform using a SpaceX Falcon 9 rocket.

Euclid is the second medium-class mission of the European Space Agency (ESA), in collaboration with NASA and other international partners, and was named after the ancient Greek mathematician Euclid, who is known for his work on geometry and optics.

Its main goal is to explore the composition and evolution of the dark Universe, which accounts for about 95% of the total energy density of the Universe, but remains largely unknown and mysterious. Euclid will measure how the dark Universe affects the growth of cosmic structures, such as galaxies and clusters of galaxies, over the last 10 billion years of cosmic history.

To achieve this goal, Euclid will use two complementary techniques: weak gravitational lensing and galaxy clustering. Weak gravitational lensing measures how the light from distant galaxies is distorted by the gravity of intervening matter, revealing the distribution

of dark matter in the Universe. Galaxy clustering measures how galaxies are distributed in space and time, revealing the effects of dark energy on the expansion of the Universe.

Euclid will create a huge map of the large-scale structure of the Universe across space and time by observing billions of galaxies out to 10 billion light-years, across more than a third of the sky. It will use a 1.2-meter telescope with two scientific instruments: a visible imager (VIS) and a near-infrared spectrometer and photometer (NISP). VIS will measure the shapes of galaxies for weak gravitational lensing, while NISP will measure the distances and spectra of galaxies for galaxy clustering.

Euclid was recently successfully launched in July 2023 (see Figure 1.2) and it is currently orbiting around the Sun-Earth second Lagrange point (L2), about 1.5 million kilometers from Earth, along two other companions, Gaia and the James Webb Space Telescope (JWST¹⁵). Its observations over 6.5 years will provide unprecedented data, testing our current understanding of cosmology and physics.

The mission's success depends on solving two major challenges. The first challenge is dealing with the huge amount of data that it will produce, which is expected to be more than 100 terabytes over its six-year lifetime. To handle this data, Euclid will use a complex data processing system that includes ground-based data centers and high-performance computing facilities. Chapter 3 describes this thesis' contribution to this challenge: the design, implementation and operation of CosmoHub, an application for the interactive exploration and distribution of massive cosmological datasets. CosmoHub has proven so useful for Euclid that has become the official tool for the analysis, validation and dissemination of their cosmological simulations.

The second challenge is dealing with systematic errors in its measurements, which are errors that are consistent across multiple measurements and can arise from various sources such as instrument calibration, atmospheric effects, astrophysical effects and also on the theory/modelling side. To minimize these errors, Euclid will use a variety of calibration techniques and will cross-check its measurements with other telescopes. One of those techniques is the extensive use of cosmological simulations, describe in the following section, to test different cosmological models and study and characterise the effect of the systematic errors and how they can be neutralised.

1.3 Cosmological simulations

Cosmological simulations are computer-based models that help scientists understand how the universe has evolved over time. These simulations use complex mathematical equations to simulate the behavior of matter and energy in the universe. By simulating the universe, scientists can study processes that occur over long time scales, such as millions or even billions of years, which cannot be directly observed in the universe.

There are different kinds of cosmological simulations, the two most important are only dark-matter n-body simulations and hydrodynamical simulations. Only dark matter n-body simulations model the gravitational interactions between dark matter particles and baryonic matter particles by replaying the evolution of the universe from its early stages to the present day. These simulations are used to study the formation and evolution of

¹⁵<https://webb.nasa.gov/>

large-scale structures such as galaxies, galaxy clusters, filaments and voids, the distribution of dark matter, and the properties of dark energy. They are based on a combination of physical models and numerical algorithms, and their output is a catalog containing billions of mock galaxies, with hundreds of realistic properties such as mass, position, velocity, shape, luminosity, and many others.

Hydrodynamical simulations include the effects of gas dynamics in addition to gravity. They use numerical methods to solve complex equations describing the behavior of matter, energy, and fluids in the universe. Because of the complexity and precision of these techniques, they are restricted to simulating smaller scales and, as such, they are most valuable for studying the formation and evolution of galaxies, including the interstellar medium, star formation, and feedback processes.

The production of these simulations is very complex because of the intrinsic data dependencies of the algorithms involved, which need to evaluate concurrently the gravitational pull of billions of particles. They also require huge amounts of memory and processing power. In order to make their production feasible, they are run on supercomputers using parallel techniques that allow for efficient processing of their data and speed up their calculations.

The importance and relevance of cosmological simulations for galaxy surveys lie in their ability to provide a theoretical framework for interpreting observational data. Cosmological simulations can be used to predict how galaxies should form and evolve under different conditions, allowing astronomers to compare their observations with simulated data. This comparison helps identify systematic errors in observational data and provides a way to test different analysis techniques.

Cosmological simulations are also important for planning galaxy surveys by providing a way to predict what observations should be made to test specific theories or hypotheses. Simulations can be used to optimize survey design by predicting how many galaxies should be observed, what wavelengths should be used, and how long observations should last. This information is critical for planning successful galaxy surveys.

1.3.1 Euclid Flagship simulations

To achieve a level of image quality and resolution never before seen, researchers within the Euclid consortium have embarked on an extensive endeavor. They are employing cutting-edge numerical simulations to accurately model the development and transformation of vast cosmic structures, including galaxies, galaxy clusters, and the intricate filamentary arrangements they create within the universe.

The Euclid Flagship mock, generated by an international team of researchers, is the largest simulated galaxy catalogue ever produced. It uses as input the Euclid Flagship N-body Simulation. Until now, two different Euclid Flagship Simulation productions have been released, FS1 and FS2. Thanks to their very large volume, those simulations are able to portray very distant galaxies we observe today that emitted their light more than 10 giga-years ago, when the universe was much younger. This provides a unique window to investigate how galaxies form and evolve across their entire lifetime.

FS1 (Potter et al., 2017) was run on the Piz Daint supercomputer at the Swiss National Supercomputer Center (CSCS) in 2016 using PKDGRAV3 (Potter & Stadel, 2016).

The simulation was run with the most current cosmological parameters available (Planck Collaboration et al., 2014). An all-sky particle light-cone¹⁶ up to redshift $z = 2.3$, with 2 trillion particles, was produced on the fly. The ROCKSTAR halo finder¹⁷ (see Behroozi et al., 2013) was run on the dark matter particle distribution to identify and generate a halo catalog, able to track substructures and relations with the parent dark matter halos. All-sky lensing convergence maps were built following the approach described in Fosalba, Gaztañaga, et al. (2015), which enable weak lensing effects to be included in the final galaxy catalogue.

FS2 (Stadel et al. in preparation), run in 2020, is an improvement of FS1 and is intended to be the base for the Science Ground Segment simulations (image simulations) and the Science Working Groups studies prior to science exploitation. It features a larger simulation box with 4 trillion dark matter particles that goes up to redshift $z = 3$. This simulation is the largest n-body simulation performed to date and matches the basic science requirements of the mission as it allows us to model the faintest galaxies Euclid will observe (complete down to the Euclid flux limit) and samples a cosmological volume comparable to what the satellite will survey.

The corresponding mock galaxy catalogs are produced from the dark matter halo catalogs of FS1 and FS2 using an improved implementation of the Halo Occupation Distribution (HOD) technique (Carretero et al. 2015, Castander et al. in preparation) called SciPIC. SciPIC (Carretero et al., 2017) is a comprehensive set of Python codes to simulate, in a very efficient and fast way, large galaxy catalogs using as input a dark matter halo population. It runs on top of the PIC Big Data service using Apache Spark. Chapter 4 describes in detail part of all the effort invested in the development, calibration and optimization of SciPIC.

The galaxy catalogs produced from FS1 were used to perform the Euclid Science Performance Verification 2 (SPV2) exercise¹⁸ and to inform the Euclid Mission Critical Design Review. The galaxy catalogs produced from FS2 are being used to perform the Euclid Science Performance Verification 3 (SPV3) analysis¹⁹. We present in Figure 1.3 an example illustration of the mock galaxy catalogue, where a small region of the light-cone is shown, and one can distinguish by eye the emergence of the large-scale structure of the Universe.

1.4 Data management and Infrastructure

As highlighted by Astronet, in the last two decades astronomy has become a data-intensive endeavor, and the large, complex and inter-dependent data sets that are being generated by observatories, space missions and simulations require new tools and approaches for doing science. Scientific questions are driving the need for surveys that cover large regions of the sky, obtain deeper exposures or include multi-frequency/messenger coverage. The sheer

¹⁶A light-cone is a 3D volume that represents the past light history of a single point in space. In other words, it is the set of all points in space that could have emitted light that reaches the observer at the present time.

¹⁷A dark-matter halo is a triaxial structure (usually prolate) that surrounds a galaxy or galaxy cluster. They are the basic entity on top of which galaxy catalogs are simulated. ROCKSTAR is a specialized software to identify and measure the properties of haloes from a dark-matter particle simulation.

¹⁸see https://www.euclid-ec.org/wp-content/uploads/EC-Newsletter_issue08.pdf

¹⁹see <https://www.euclid-ec.org/science-performance-verification-3/>

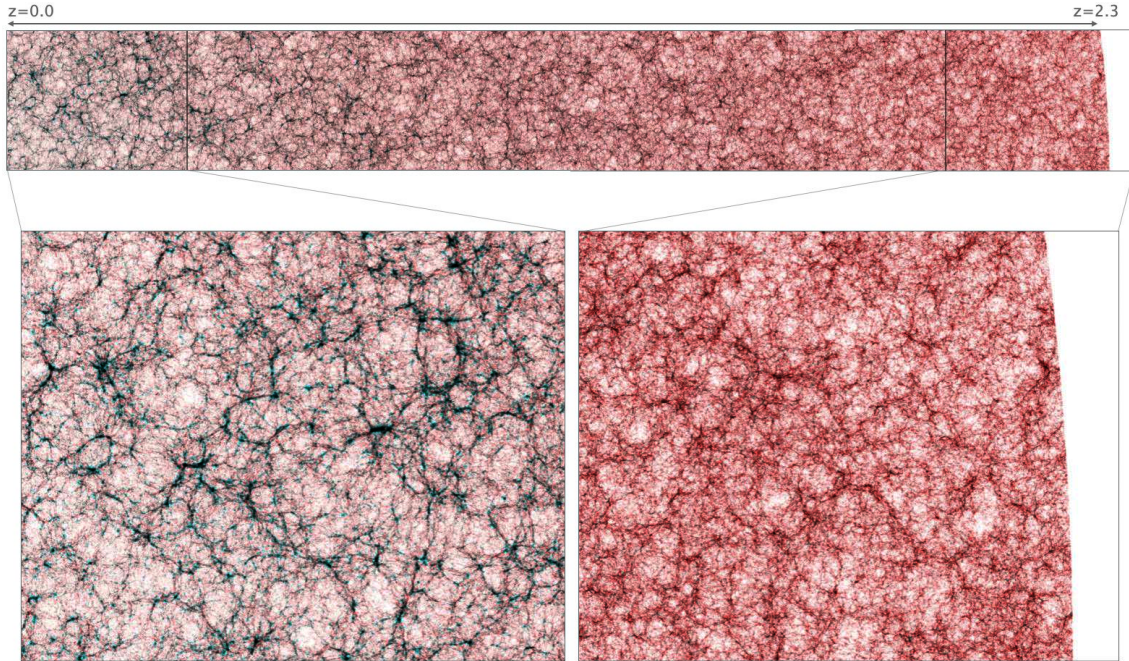


Figure 1.3: False colour visualization of a Euclid Flagship mock galaxy catalog based on FS1. At the top, a depth-limited slice of the full light-cone (roughly 0.3% of the total volume). Central galaxies are coloured in red and satellites in dark blue, where filaments from the large-scale structure are seen in great detail. Bottom left shows a section zoomed on the local universe while bottom right shows the fraction of the furthest galaxies. The differences in large scale structure between closer and farther sections are very noticeable.

data volumes from such experiments is increasing at a high rate with a number of projects and facilities expected to have annual data production of the order of petabytes.

We have also entered the era of multi-messenger astronomy. Going beyond the often-quoted advent of gravitational wave science or the synergy with astroparticle physics experiments, the trend is towards the need to combine data from observations across the electromagnetic spectrum and beyond. This is in addition to the transient object searches of time domain astronomy, leading to high flux event streams where rapid detection, classification and follow-up observations bring new challenges for computing and analysis.

Dedicated simulations, either conducted as numerical experiments or modeling efforts, also reach levels of complexity and volumes which call for an updated assessment of how we run them, process them, and share and distribute their outputs and associated data products.

Users also want to interact with data from diverse sources. The need to jointly analyze datasets from a variety of instruments or facilities with different characteristics, as well as to connect the observational and modeling landscapes further motivates the development of flexible and transparent frameworks where the focus shifts from data towards the applied expertise and tools. This is reinforced by the paradigm shift pertaining to sharing information and the push for "Open Science", where the added value will come from the access to expertise, supported by media and tools that alleviate the difficulty of dealing with varied datasets. This, in turn, motivates new ways to deal with data, using science analysis platforms that enable a "bringing computing to the data" paradigm.

1.4.1 The Port d'Informació Científica

The Port d'Informació Científica (PIC) was founded in 2003 as a data center of excellence for scientific data processing, supporting scientific groups working in projects which require large computing resources for the processing and analysis of massive sets of distributed data. It is operated and maintained through a collaboration agreement between IFAE (Institut de Física d'Altes Energies) and CIEMAT, supported by the Spanish Ministry of Economy and Competitiveness and the catalan Department of Economy and Knowledge.

Located on the campus of the Universitat Autònoma de Barcelona (UAB) in Cerdanyola del Vallès, Catalonia, PIC has two rooms with different characteristics and energy efficiency profiles: a 150 m² air-cooled room and a 25 m² highly energy efficient room which uses immersion cooling based on open bath dielectric fluid tanks for the storage and computing IT equipment. Its facilities include around 12,000 high-throughput cluster computing slots, a tape storage system with approximately 4,500 tape slots and a capacity of 64PB, and 18PB of disk storage within 60 storage servers. PIC also operates a big data service based on a custom-designed 30-node cluster running a Hadoop distribution developed in-house.

The external network is deployed in collaboration between the Catalan NREN (CSUC, Anella Científica), the Spanish NREN (RedIRIS) and the Géant pan-European network, which provide high-throughput connectivity between research institutions and the Internet. Being the Spanish Tier-1 center for CERN, PIC is connected through a dedicated 100 Gb link between PIC, CERN and the other Tier-1s. It is also connected to the LHCONE network to Tier-2s and to La Palma, one of the Canary Islands where the Roque de los Muchachos Observatory (ORM) is located, serving multiple affiliated projects. PIC is also connected to the Internet by another 100 Gbps links.

PIC collaborates with various scientific fields, including particle physics, astrophysics, observational cosmology, gravitational waves, bioimaging, quantum computing and material sciences, among other fields. In particle physics, PIC coordinates and provides support to activities related to the Large Hadron Collider (LHC) in his role as the Spanish Tier-1 infrastructure, such as ATLAS, CMS and LHCb experiments, as well as Neutrino Physics experiments and Voxel Imaging PET (VIP). In astronomy and cosmology, PIC collaborates with several projects dedicated to the research of large scale structure, dark energy, accelerating expansion of the universe, gamma rays and cosmological simulations, such as CTA, MAGIC, DES, PAU, Euclid, and MICE.

Lately, the Council for Science, Technology and Innovation Policy of the Ministry of Science and Innovation has approved the update of the Map of Singular Scientific and Technical Infrastructures (ICTS) of Spain for 2021-2024 with the incorporation of four new infrastructures, including PIC. PIC joined the Spanish Supercomputing Network (RES) in 2020, an infrastructure that connects 14 supercomputing centers in Spain and since 2007 has been providing high-performance computing services to the scientific community. Since the RES is recognized as an ICTS by the Ministry of Science, PIC has also obtained this recognition as a node of this distributed infrastructure. The inclusion of PIC in the ICTS map is an important milestone for the center, as it recognizes its work during almost two decades in large-scale scientific data analysis and opens new avenues for establishing collaborations with research groups facing data analysis challenges, and for funding these activities.

1.5 Contributions

This thesis contributions revolve around the figure of the RSE in a scientific data center and how that role is fundamental as a liaison between scientists and technicians to ensure a correct and efficient information flow that is critical to any research project's success.

In this section I will summarize the objectives, approach and corresponding results. The first one has been the design and implementation of a comprehensive data management system for the PAUS Survey. The second one has been the application of Open Science and Open Data principles for the management of very large cosmological datasets that resulted in the development of an open platform called CosmoHub. And the third one describes a small but very relevant part of a greater effort to optimize and improve the efficiency of an algorithm to simulate electromagnetic galaxy fluxes in order to generate massive synthetic galaxy catalogs.

Note that the work presented in this thesis is deeply indebted to the free and open-source software (FOSS) community, and all its invaluable contributions that foster collaborative innovation. In the spirit of reciprocity, I've made an effort to publish as much source code from my contributions as possible under free licenses, hoping to empower others and enrich the collective knowledge pool. The corresponding source code will be linked accordingly in each section.

PAUS data management and operations

The first contribution in this thesis, described in chapter 2, has been the design, development and operation of a data management and operations architecture that encompasses the entire life cycle of the PAU Survey.

As a small research project, PAUS had several constraints but also some unique advantages. On the one hand, as it started as a project completely lead and develop by Spanish Institutions, both the human and material resources were limited. On the other hand, the lack of a bureaucratic steering body also allowed for a greater degree of freedom on the technical approaches. This effort synthesizes a wide range of tasks and responsibilities, as the scope of the assignment was very broad.

After every PAUCam exposure is captured, it goes through a quick reduction process on-site that gives immediate feedback on several crucial parameters. Any image that does not pass the quality test is discarded, otherwise it is stored and staged for transfer to PIC. It is at this time that we take control of the process. We had to design an architecture to track the images since the moment they are transferred, until they are finally archived after being processed by multiple pipelines.

Furthermore, not only we need to track the images themselves but also the attached metadata that describes them, and all the additional information that will be generated by the analysis pipelines. This metadata is also key to enable efficient processing, reproducibility, and access to the project's data, as it allows scientists to identify datasets of special interest to their research. This architecture has to also allow the ability to perform periodic reprocessing of all the data acquired by the survey, as frequent as every two weeks, enabling iterative algorithm refinement with each cycle, and to provide data releases in a

timely manner. Finally, the status, progress and quality of the processing has to be monitored by the operations personnel, in order to detect as early as possible any irregularities and issue the corrective actions needed.

PAU database (PAUdb) is the selected approach I envisioned to solve most of these challenges tied to metadata management. Based on a PostgreSQL relational database and tied to a carefully designed data model²⁰, we have been able to store, track and monitor all of the metadata generated throughout PAU Survey. Furthermore, in combination with an Object Relational Mapper (ORM) layer, it enables processing pipelines to access all this information in a very fast, efficient and easy way.

All the images acquired at WHT are transferred to PIC by an automated procedure that runs every morning as observations finish. As soon as the images arrive and the associated metadata are registered in PAUdb, their analysis can start. All the processing of images and metadata is done by a set of highly optimized pipelines that combine external specialized software and custom developed Python algorithms. The pipelines use the ORM layer in order to interact with PAUdb to ensure optimal data access.

The first analysis pipeline to run on the images, and the most important one, is the *nightly* pipeline. This pipeline is fundamental during observation periods as it is responsible for detecting any anomalies on the freshly taken images, so that they can be marked as faulty and be re-observed the following night. Thus, it is imperative that this pipeline has been properly optimized and can run in a timely manner, to allow for the insights of a night to drive the observations of the following and maximize the efficacy of the survey.

Besides the *nightly* pipeline, other pipelines have been developed to further process PAUS' data and deliver the planned scientific results. All these pipelines are divided into individual tasks that get sent to the PIC computing farm to be executed. In order to track the execution of those tasks and ensure the traceability and reproducibility of any analysis, a custom designed tool was developed, named BT.

BT²¹ is a job orchestration framework that allows pipeline developers to design arbitrarily complex workflows, connecting a vast collection of processing steps, and keeping track of their configuration and execution status. It is also integrated with the Quality Control (QC) layer of PAUS, monitoring and storing essential job metrics to track the scientific performance.

Finally, a set of web applications that feed on all this data, allow PAUS scientists and operators to monitor, analyze and supervise the quality of the observations, as well as to enable access and exploration of the resulting galaxy catalogs by the scientific community.

CosmoHub: interactive catalog exploration and distribution

The second contribution in this thesis, described in chapter 3, has been the design, development and operation of CosmoHub, an application for the interactive exploration and distribution of large structured datasets, with special attention to cosmological catalogs. This effort is an evolution of the initial work done to publish PAUS' results through a web portal.

²⁰Source code available at <https://github.com/ptallada/paudm-model>.

²¹Source code available at <https://github.com/ptallada/brownthrower>.

The objectives behind CosmoHub’s design were to provide a centralized distribution point for cosmological catalogs from multiple projects where users could build, plot and download custom subsets of those catalogs using the most common standard data formats, and ensuring its usability even for non-technical users (specifically for those without any Structured Query Language (SQL) knowledge).

The most difficult challenge in this contribution has been the research, deployment and implementation of a data warehouse that was capable of storing very large catalogs (>500 million rows, >500 columns) while keeping the query response times within interactive constraints. This has been specially difficult as the lack of a clear access pattern inhibits most kinds of data access optimization techniques such as indexation, partitioning or sorting.

Although the initial prototype was built on top of PAUdb and used the same relational database to store and query the hosted catalogs, we realized it was not the best tool for this task. While a relational database such as PAUdb is optimized to handle thousands of small queries per second, including the addition, modification and removal of small subsets of rows, CosmoHub’s data workflow is very different. In particular, each catalog is only inserted (or ingested) once at the beginning, is never modified afterwards (only removed if its entirety) and almost all queries request a large subset of rows.

We analyzed several alternative solutions and we finally settled on Apache Hive, a data warehouse based on Apache Hadoop. Hive provides an SQL interface on top of massive datasets stored on a distributed filesystem, such as HDFS (Hadoop Distributed File System). Keeping the SQL interface is important, not only to ensure a smooth transition from PAUdb, but also because SQL is the natural language for relational data access, such as cosmological catalogs.

Hadoop’s design, based on a shared-nothing cluster architecture, provides linear scalability and great fault tolerance. On top of it, Hive is able to bring an impressive performance gain with respect to PAUdb, as the queries are executed in parallel on all the nodes in the cluster, taking advantage of concurrent and distributed data reads that greatly reduce the execution times.

However, using SQL to explore and build custom catalogs is not very user friendly. Therefore, for CosmoHub we have designed and implemented a web application²² that guides the user through all the steps, even those that know nothing about SQL. Using this interface, users are able to list and search within through the available catalogs, build their own subsets by selecting the columns they need, add filters to further restrict the output, quickly explore them by creating histograms or heatmaps in seconds and download the resulting custom catalog if desired in the most standard formats.

As the astronomical community has some specific needs that were not covered by the features available in Hive off-the-shelf, we have designed and developed²³ a set of user defined functions (UDFs) to further extend the capabilities of CosmoHub. In particular, we have implemented three distinct sets of functions that can be called directly through SQL: a set of HEALPix²⁴ functions to interface with this spherical pixelization function, a set of array aggregate functions intended to be used on astronomical spectra or probability density

²²Source code for the backend is available at <https://github.com/ptallada/cosmohub-api>.

²³See <https://github.com/ptallada/pic-hadoop-udf> for the full source code of the implemented UDFs.

²⁴<https://healpix.sourceforge.io/>

functions, and a set of spherical geometric functions that are part of the Astronomical Data Query Language²⁵ (ADQL) standard.

Finally, the custom catalogs requested by the users are delivered in minutes and can be downloaded in several standard file formats. For this purpose, we developed multiple *drivers* to generate the output in the intended format, taking advantage of Hadoop's distributed writes to minimize catalog creation time. CosmoHub is then able to merge the multiple outputs into a single file that is served to the user. This method has been used to implement the following formats²⁶: CSV.BZ2²⁷, FITS^{28,29}, ASDF³⁰ and Parquet³¹.

By now, CosmoHub has become the official catalog distribution platform for several research projects, such as PAUS or the Euclid simulations. It also hosts private data for many other projects such as DES, DESI, MAGIC, MICE, and also public datasets such as those from Gaia³². It currently serves about 150 monthly active users from around the world.

SciPIC: fast and precise simulation of galaxy fluxes

The third and last contribution in this thesis, described in chapter 4, has been the design, implementation, optimization and calibration of a method for simulating galaxy electromagnetic fluxes. This work is part of a large suite of algorithms for the generation of very large synthetic galaxy catalogs, called SciPIC. These catalogs have become fundamental to design and plan both present and future galaxy surveys, to calibrate the reduction algorithms and, in general, to enable the scientific exploitation of all the data they generate. In particular, this effort has been fundamental for the production of mock galaxy catalogs for PAUS and the ESA Euclid mission, among others.

These mock galaxy catalogs, often implemented as massive tables with one row per galaxy and a column per simulated property (up to several hundreds), represent realistic synthetic universes. Galaxy fluxes constitute one of the most important properties on those catalogs and each galaxy can have up to several hundreds, depending on how many instruments (or filters) and observing scenarios are required. Thus, the main challenge in this effort is to generate billions of galaxy fluxes as efficiently as possible, so that the generation of these synthetic universes can be done in a reasonable time frame.

In physics, an observed galaxy flux can be defined as a function of the amount of energy that arrives to an observer with respect to its wavelength. It can be computed as the integral of the observed galaxy electromagnetic spectra convolved with the corresponding filter transmission. In simulations, the observed galaxy spectra is estimated from several

²⁵<https://www.ivoa.net/documents/ADQL/20180112/PR-ADQL-2.1-20180112.html>

²⁶See <https://github.com/ptallada/cosmohub-api/tree/master/cosmohub/api/io/format> for the source code.

²⁷<https://datatracker.ietf.org/doc/html/rfc4180>, and <http://sourceware.org/bzip2/>

²⁸https://fits.gsfc.nasa.gov/fits_documentation.html

²⁹See <https://github.com/ptallada/rearrayserde> for the source code of the Hive FITS binary format serializer/deserializer.

³⁰<https://asdf-standard.readthedocs.io/en/1.6.0/>

³¹<https://parquet.apache.org/>

³²PIC is a Gaia affiliated data center since 2020 and hosts both public and embargoed Gaia data in CosmoHub.

properties present in the galaxy catalog: the galaxy spectral energy distribution (SED), the emission lines contribution and the Milky Way extinction.

However, simulating galaxy fluxes following this approach requires solving several integrals that, even when using highly optimized numerical libraries, impose a high computation cost. In our tests, simulating a single filter for one billion galaxies could take up to 13 months using a single core. Given that the largest synthetic universes can hold tens of billions of galaxies and require up to several hundred fluxes each, it is clear the approach based on integrals cannot be used.

In this contribution we devised an alternative approach³³ based on interpolations and some clever simplifications. The main idea has been to split up the integral computation and take advantage of the fact that some values are constant, some depend only on the filter, and others can be approximated.

In particular, we have tested this method for two very different scenarios: one is a broad-band survey such as Euclid, and the other is a narrow-band survey as PAUS. Our experiments prove that, when properly calibrated, this method is able to compute galaxy fluxes 750 times faster than the integral-based approach within the stringent precision margins that these simulations impose. Thus, making the production of very large mock galaxy catalogs feasible.

1.6 Related work

As in any field of science, the contributions contained in this thesis have not been designed in isolation. They are the result of a great personal effort supported by a broad community of researchers with whom we share a common journey of discovery to push the boundaries of scientific knowledge.

PAUS data management and operations

The design of PAU Survey data management was influenced by other similar contemporary surveys. In particular, the one that had the most influence was the Dark Energy Survey (DES, Morganson et al. 2018), as a great majority of PAUS members were also participating in DES. It shares most of the ideas behind the data model and the reliance on the AstrOmatic³⁴ suite of tools to perform most of the heavy lifting to process the acquired images.

Both PAUS and DES have a data base at the center of their data management architecture. However, the resources to implement this architecture for both projects were very different. PAUS had far less human and material means and, as it was decided from the very beginning, PIC was to centralize all of its operations. In the end, the efficiency and performance of PAUdm surpassed that of DES, as PAUS has been able to perform many more reprocessings at a faster pace.

³³The source code for both approaches is available at https://github.com/ptallada/scipic_fluxes.

³⁴<https://www.astromatic.net/>

Nowadays both PAUS and DES have stopped their observations, but other wide surveys are ready to pick up the baton and continue the mission to unveil the mysteries of the dark universe. The Euclid telescope was just launched in July 2023 and the first science-ready images have already been processed at PIC. The Dark Energy Spectroscopic Instrument (Guy et al., 2023, DESI) has already published an early data release for the first 6 months of observation. And the Legacy Survey of Space and Time (Ivezić et al., 2019, LSST), to be carried out at the Vera C. Rubin Observatory, is expected to see the first light in January 2025.

On a final note, DES is planning to shut down their entire data management architecture due to its high maintenance cost, mainly from the Oracle licenses that power their database. One of the possibilities that are being studied is to migrate the DES catalog archive to CosmoHub, in order to preserve it and keep facilitating its access to the scientific community.

CosmoHub: interactive catalog exploration and distribution

When we envisioned CosmoHub as a data distribution tool between PAUS and MICE projects back in 2012, there were very few data distribution portals. The most important ones were SkyServer (Raddick et al., 2014a, 2014b) and TAO (Bernyk et al., 2016). Those portals were already visionary in their concepts and started a revolution that still echoes today. With vast volumes of data within easy reach, scientific communities gathered and took advantage to further analyze all the data available, in ways they were not imagined yet. Along blossomed citizen science projects such as Galaxy Zoo (Lintott et al., 2011; Willett et al., 2013).

Nowadays, there are many scientific portals that give access to multiple datasets and offer very distinctive features, catering to an ever growing diverse user community:

- SkyServer has matured into a full-fledged science platform called SciServer (Taghizadeh-Popp et al., 2020), that improves on performance while incorporating a large set of new features, including notebooks, long running batch jobs and an ADQL interface to query all its data releases.
- TAO provides web access to cloud-based mock extragalactic survey data, generated using sophisticated semi-analytic galaxy formation models that are coupled to large N-body cosmological simulations. TAO is part of the larger All-Sky Virtual Observatory (ASVO) project, whose goal is to federate and distribute astronomical data to the wider community via a cloud-based data storage system. It supports TAP and ADQL protocols for remote data access.
- The Vera Rubin Observatory has launched the Rubin Science Platform (RSP, O’Mullane et al. 2021) which offers the ability to launch notebooks, a data access portal with a guided interface to query LSST Data Preview releases and the possibility to access the same data over TAP in an automated way.
- The European Space Agency has deployed ESA Datalabs, a science portal featuring notebooks, standard Virtual Observatory³⁵ (VO) APIs for automated access and a collection of commonly used astronomical software like TOPCAT³⁶ and SAOIm-

³⁵<https://www.ivoa.net/>

³⁶<https://www.star.bris.ac.uk/~mbt/topcat/>

ageDS9³⁷ that can be accessed directly from the browser over VNC (Richardson et al., 1998). However, it does not include any guided tool to query the catalogs it hosts.

- The National Science Foundation’s (NSF) NOIRLab has Astro Data Lab (Nikutta et al., 2020), a very comprehensive science portal that features a crossmatch service, an image cutout service, notebooks, a multilayer image visualization tool, an ADQL interface to query about 50 datasets and a complete set of APIs to remotely access the service.

All in all, there seems to be an agreement on the general direction in terms of the functionalities that these science portals implement:

- Exploration of astronomical catalogs with the possibility to display or download subsets of data, based almost exclusively on raw SQL statements. Some of them offer examples or other guidance to assist users with no prior SQL experience.
- Visualization of astronomical images or maps, based on the Hierarchical progressive surveys (HIPS, Fernique, P. et al. 2015) technique. Multiple layers of information can be superimposed to compare between them.
- Notebooks to enable users to run arbitrary analysis over the hosted datasets.
- Standardized APIs to access and interact with the data, including TAP, ADQL, VOTable³⁸, UWS³⁹, SIA⁴⁰ and SCS⁴¹.
- Federated authentication with major academic/research identity providers, such as InCommon⁴² and eduGAIN⁴³, to facilitate user onboarding.
- Collaborative features such as custom groups and the ability to share notebooks, datasets and visualizations within these groups but also to other external users.

Even surrounded by this fierce competitors, CosmoHub still stands proud as a specialized portal for the *interactive* exploration and distribution of *massive* cosmological datasets. Most of the alternatives have imposed restrictions on the time or volume that queries can take. SciServer limits interactive queries to 1 minute, and batch queries to 8 hours. Astro Data Lab has a 10,000 row limit for previews and a 500,000 row limit for batch catalogs. And RSP limits downloads to a maximum of 5 million rows. Oppositely, CosmoHub can offer full unrestricted interactive queries and batch catalogs thanks to its unique design and impressive performance.

Finally, note that CosmoHub has not seen any remarkable changes in design or architecture since its migration to Hadoop in 2016. Even now the most recent science platforms are not up to par with its response times when it comes to dealing with massive datasets. However, time does not pass in vain and some of CosmoHub’s features feel a bit outdated.

³⁷<https://sites.google.com/cfa.harvard.edu/saoimageds9>

³⁸VOTable Format Definition: <https://www.ivoa.net/documents/VOTable/>

³⁹Universal Worker Service Pattern: <https://www.ivoa.net/documents/UWS/>

⁴⁰Simple Image Access: <https://www.ivoa.net/documents/SIA/>

⁴¹Simple Cone Search: <https://www.ivoa.net/documents/latest/ConeSearch.html>

⁴²<https://incommon.org/>

⁴³<https://edugain.org/>

The absence of notebooks and the lack of support for VO APIs is particularly upsetting. Thus, work has already begun to implement these features and many more in a new version of CosmoHub that we hope will be seen as the next point of reference.

SciPIC: fast and precise simulation of galaxy fluxes

When dealing with large datasets it is not unusual that operations that are used regularly cannot be applied as the computational cost becomes unfeasible. The issue resides in the problem scalability. In this scenario, shared with many other fields of science, alternatives must be sought to simplify or circumvent this complexity. Nearly all of the time, gains in speedup come at the cost of losing accuracy, so it is crucial to understand and tune this balance in order to provide meaningful results.

The approach used to optimize the execution time of the algorithm that simulates observed electromagnetic fluxes in SciPIC has nothing out of the ordinary. It is a combination of simple techniques that are very effective in solving the challenge at hand. In simple terms, we replace a convolution by a multiplicative factor and approximate the integrals by carefully calibrated interpolations:

$$\int (A + B) \cdot C \sim \left(\int A + \int B \right) \cdot \int C$$

Similar approaches are taken to generate complex datasets like N-body simulations. In these simulations of a given cubic volume, the position of trillions of particles is randomly assigned and the effect of gravity is reproduced over large scales of time. Solving the Newton equations (general relativity is far too complex for the current computing capabilities) for trillions of particles over billions of years requires an amount of memory and processing power only available in very few supercomputers in the world. The effort they require makes these simulations one of a kind.

However, for some studies, having just a few of these simulations is not enough. For instance, research studying the uncertainty in several cosmological parameters. These studies really benefit from having hundreds even thousands of these simulations. For instance, a method named COmoving Lagrangian Acceleration (COLA, Tassev et al. 2013) is able to produce a similar output than traditional N-body simulations but several orders of magnitude faster. COLA trades accuracy at small-scales in order to gain computational speed without sacrificing accuracy at large scales. This makes it a very valuable tool to cheaply generate large ensembles of accurate mock halo catalogs to study properties like galaxy clustering and weak lensing, essential for performing detailed error analysis for ongoing and future surveys of large scale structure.

Chapter 2

PAU Survey data management

The Physics of the Accelerating Universe Survey (PAUS¹; Martí et al., 2014) observed about 100 deg² of the northern sky for the study of the accelerated expansion rate of the universe. The main scientific contribution has been the measurement of distance (in terms of photometric redshift, or photo-z) for about 1 million galaxies from known catalogs, with an improved resolution, together with the study of galaxy spectral features, clustering, intrinsic alignments and their evolution, among other science cases.

PAUS uses a very innovative approach to try to combine the best characteristics of both spectroscopic and photometric galaxy redshift surveys. By combining photometric measurements of 6 broad-band filters with 40 narrow-band ones one has enough information to estimate each source's redshift with a precision similar to an spectroscopic survey, while being able to observe a much larger area in the same time.

The data acquisition is carried out using an imaging camera, called PAUCam (Padilla et al., 2019), designed and built at the engineering facilities of IFAE, in Barcelona. PAUCam is a community instrument² operating since 2015 at the primary focus of the 4.2 m diameter William Herschel Telescope (WHT) at Observatorio del Roque de los Muchachos (La Palma, Spain). PAUCam (Figure 2.1) is made of 18 4k × 2k CCDs, with a system of 46 optical filters (6 broad band and 40 narrow band), installed in a set of moving interchanging trays (López et al., 2016). Each PAUCam focal plane image consists of about 650 MiB of information, which translates into a mean total data volume of 200 GiB for a typical observing night. PAUCam data are transferred to the Port d'Informació Científica (PIC), the PAUS data center, after each night of observation.

I have been involved in PAUS since its inception around 2009. In particular, I have had a main role in the design, development, testing and operation of the data management

¹The project is governed by a consortium, originally founded by the Spanish institutes Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Instituto de Física Teórica (IFT), Institut de Ciències de l'Espai and Institut d'Estudis Espacials de Catalunya (ICE/IEEC-CSIC), Institut de Física d'Altes Energies (IFAE) and Port d'Informació Científica (PIC), with the later incorporation of several European institutes (Durham University, ETH Zurich, Leiden Observatory, University College London) for its scientific exploitation.

²Community, or visitor instruments at WHT are those instruments whose teams have expressed an interest in collaborative proposals. External applicants can submit a proposal based on PAUCam after contacting the instrument responsible. Further information can be found at <https://www.pausurvey.org/paucam/community-instrument/>

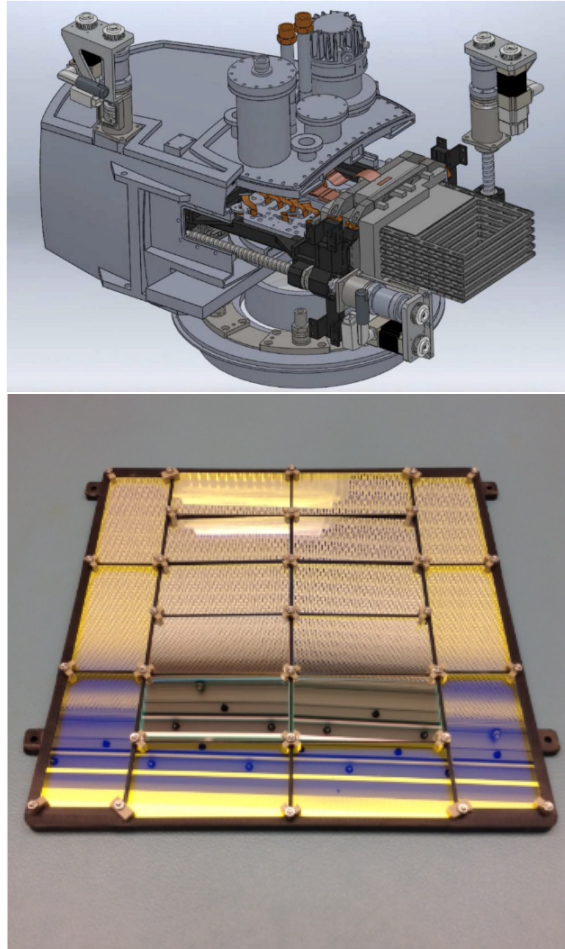


Figure 2.1: Top: PAUCam design showing the filter exchange system and the overall camera vessel. Bottom: one of the filter trays for narrow band filters. Each narrow band filter has a different waveband and covers one of the central CCDs of the PAUCam focal plane. Broad band filters cover the outer CCDs.

for the entire project. This chapter compiles and describes my main contributions to it. Most of its contents were originally published in Tonello et al. (2019), an article in which I am the first technical author. The structure of this chapter is as follows: section 2.1 introduces the main scientific requirements and technical challenges that had to be solved for the success of the survey; section 2.2 describes the technological infrastructure of PIC, the PAUS data center, that inherently imposes additional technical constraints; sections 2.3 through 2.7 explain in detail the design and implementation of a solution for each challenge; and finally, section 2.8 offers some concluding remarks above all this work.

2.1 Introduction

The data management of a galaxy redshift survey is especially demanding due to the large amounts of data and metadata that need to be stored, the strict latency requirements of its processing and the difficulties of providing access to all this information to any external user. Moreover, the limited availability of observation time for the PAU Survey at the WHT and the ambitious scientific goals of the project impose additional constraints that must be taken care of too.

The PAU Survey data management (PAUdm) team is responsible for the design and development of a data management architecture to deliver science-ready data products to the PAUS Collaboration and its scientific community. The main challenges faced are typical of a highly automated system, delivering and processing a considerably high data volume (compared to the one that can be comfortably handled by a single computer).

Successful large projects in analyzing and distributing astronomical data, such as the Sloan Digital Sky Survey (SDSS³), are a clear reference for this work, but are not directly reusable given the rapid evolution of technical and software tools which can be applied to data management.

The scientific requirements and technical challenges imposed by the project are:

- *Metadata integrity and preservation*

PAUdm shall guarantee the preservation and consistency of the PAU Survey's files metadata and of the results of the images reduction, such that they can be accessible for their scientific exploitation. After careful consideration, it was decided to put a relational database at the core of the data management design. This PAUS database, named PAUdb, has been implemented using PostgreSQL⁴.

All of PAU Survey metadata are checked and preserved in PAUdb, including metadata related to the PAUCam images, the reduction results and the settings used for the processing. Most of PAUdm services rely on PAUdb, such as the job orchestration tool (BT), the data distribution portal and several other web services for the successful operation and management of the survey.

Section 2.3 explains the effort to design and deploy PAUdb, define its database schema and implement an object relational layer to interface with all the data analysis code. This work also includes the periodic refactorization and update of the database schema as part of the ever ongoing work to optimize its efficiency and scalability to handle the increasing volume of data.

- *Data transfer and archival*

The PAU Survey's raw and reduced data files shall be archived and preserved in an organized way. Raw image files produced by PAUCam at the WHT are to be transferred to PIC after data taking, usually the morning after a night of observation. A temporary storage in the observatory with capacity for 5 nights of observation allows some margin in case of connection interruptions or other temporary problems.

³<https://www.sdss.org/>

⁴<https://www.postgresql.org/>

Section 2.4 describes the data buffer at the observatory and the definition of a transfer protocol which allowed the automation of the data transfers and its posterior long-term archival.

- *Operations: timely nightly report*

The survey plan for each observation night, determined by the list of sky coordinates and related telescope settings (such as exposure time, filter tray) that should be observed to complete the survey with the desired quality, is guided by a nightly report. This report compiles the result of executing the *nightly* pipeline on the observed images of the previous night. As such, it must be produced during the morning (while the researchers sleep) and be ready by afternoon to guide the target of the following observation night. This imposes a strict time limit, as the entire image reduction analysis has to be completed in less than 6 hours.

Section 2.5 details how the image reduction algorithms have been adapted to run on top of the high throughput computing system at PIC, with optimized efficiency.

The specific algorithms in the nightly pipeline (image detrending and cleaning, astrometric and photometric calibration) as well as the algorithms used for the production of the final catalogs (source extraction and generation of co-added objects spectra) are explained in Castander, F. J. et al (2019) and Gaztañaga, E. et al (2019) respectively.

- *Modular and reproducible analysis*

The data management of PAUS not only has to be able to process the vast amounts of information it generates, but also has to guarantee that those analyses are reproducible and that the codes involved in it can evolve through the project's life cycle.

All the data processing is split in multiple individual steps that, when necessary, can be run isolated to test, debug and optimize. All the analysis pipelines are then built from combining several of those steps in a direct-acyclic-graph (DAG). Section 2.6 describes the design and development of a tool called BT that, based on the metadata stored in PAUdb, aids in the management and orchestration of all these pipelines and their corresponding steps.

- *Operations and metadata exploitation and distribution*

The data management of PAU Survey involves processing and generating a lot of metadata, most of it ends up in PAUdb. Moreover, all files (raw and reduced images as well as analysis results) must be accessible, distributed and published, initially to the collaborators of the project and eventually to the whole scientific community.

In order to facilitate the operations and the data access to its multiple analysis tools and products, several web interfaces have been developed to assist in those tasks. Although they have not been a core part of this thesis, they are briefly described in section 2.7 to illustrate how a well designed data management system can benefit and be further improved by powerful ad-hoc applications that tailor custom use cases. There is one exception though, as the development of the first prototype of a data distribution portal for wide access and distribution of PAUS data catalogs (described in subsection 2.7.5) is indeed part of this thesis and has turn up into a full-featured platform called CosmoHub (see chapter 3).

2.2 PAUS data center

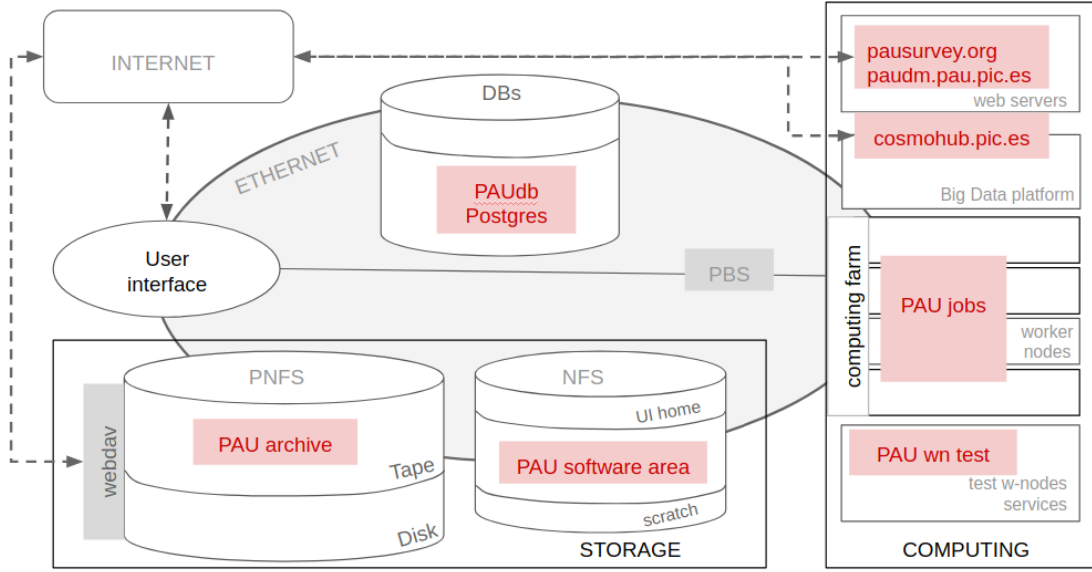


Figure 2.2: PAU infrastructure and services.

PAU data management has been developed on top of the PIC infrastructure, which consists of a storage system, a network, a computing farm, databases and user interfaces. As such, all the challenges described in the previous section must be solved within the technical constraints described here.

The PAU project has access to a series of standard and customized services (user interface, test worker node, web servers on virtual machines and databases), supporting the activities of the data management. The standard user interface system of PIC allows any user of the PAU project to access the PIC infrastructure, with a dedicated 10 GiB "home" space, a shared scratch NFS disk area, and access to the PAU files archive. Interactive access to a test worker node offers an environment which allows developers to test and debug the PAU pipelines before the production phase.

The computing farm is shared among all hosted projects at PIC. 5% of the 12000 available cores (year 2023) have been assigned to PAUdm activities as a yearly average. The software developed for PAU Survey image reduction and analysis is managed through git⁵, accessible from an NFS software area, readable by all the nodes.

Raw images from William Herschel Telescope (WHT) at Observatorio Roque de los Muchachos (ORM) in La Palma, Canary Islands are transferred to PIC through a shared 10 Gb connection. PAU files are permanently stored on disk and tape. On tape, we have configured an automated double copy, in two different cartridges, that guarantees the preservation of PAU Survey data in case of failure of one of them.

A second copy of the raw data produced by PAUCam are stored at the CASU Astronomical Data Center in Cambridge. This serves to mitigate any catastrophic failure of the primary data center at PIC. In addition, this copy can be retrieved from the Isaac Newton

⁵<https://git-scm.com/>

Group Archive⁶ web page one year after the observation, when data are public according to WHT policy.

The virtualized infrastructure of PIC hosts the web services for PAU Survey: the PAU Survey web site, and the internal web service⁷, entirely dedicated to PAUdm activities (see section 2.7).

The PIC storage manager (dCache) integrates WebDAV as a native protocol for data access. This is used for internal PAUdm archive file access. In addition, external observers, that received a time allocation to use PAUCam for their own scientific purposes, use it to download their data. WebDAV will also be used for open access once PAU Survey raw and reduced data become public.

2.3 The PAUS database (PAUdb): metadata integrity and preservation

Preservation of data and metadata, as well as of their consistency, is a critical point of the data management of every scientific project. The PAU Survey project generates a large number of files while data are being taken and processed, and a large amount of metadata as a result of running image analysis pipelines. The data volume is only one of the critical points: files and metadata produced at high velocity need to be accessed concurrently by hundreds of nodes in the PIC computer farm where reduction and analysis codes run.

Several alternatives for implementing a metadata repository were explored, such as a nested structure of ASCII files, relational databases or newer NoSQL solutions. Particular emphasis was given to solutions enabling the following: use of Structured Query Language (SQL), which is already familiar to most of the scientific community; support for relational operations, to ease comparisons between datasets; and an open, mature, stable software capable to deal with the data volume we had foreseen for the project.

We settled on a relational database setup consisting of two twin servers configured one as a replica of the other. Initially, each server had 12 physical cores, 96 GiB of memory and 2 TiB of storage (6-disk RAID array). The main database server was later upgraded to a much powerful hardware, with 24 cores, 64 GiB RAM and 4 TiB of ultra-fast solid-storage. PostgreSQL was selected as the software solution for the relational database due to its stability, performance and broad compatibility with current SQL standards.

The PAU processing pipelines make heavy use of database transactions to ensure the integrity and consistency of the ingested data. The information stored in the PAU database is backed up periodically to minimize data loss in case of malfunction or catastrophic failure. Full backups are made monthly, with critical tables being saved weekly. External catalogs, which are not modified, are backed up just once after ingestion.

⁶<http://casu.ast.cam.ac.uk/casuadc/ingarch/query>

⁷<http://pau.pau.pic.es>

2.3.1 PAUdb data model

The database schema for PAUdb has been designed based on the requirements collected from all PAUdm components, and has been evolving throughout the operations phase to match changes and improvements in the processing pipelines. The PAUdb content is organized in about 40 different tables, each one of them usually maps a to metadata entity in the PAUdm processing pipelines. The data model⁸ has been specifically designed to favour *inserts* instead of *updates* in order to support a large number of concurrent clients, mapping to multiple simultaneously running jobs.

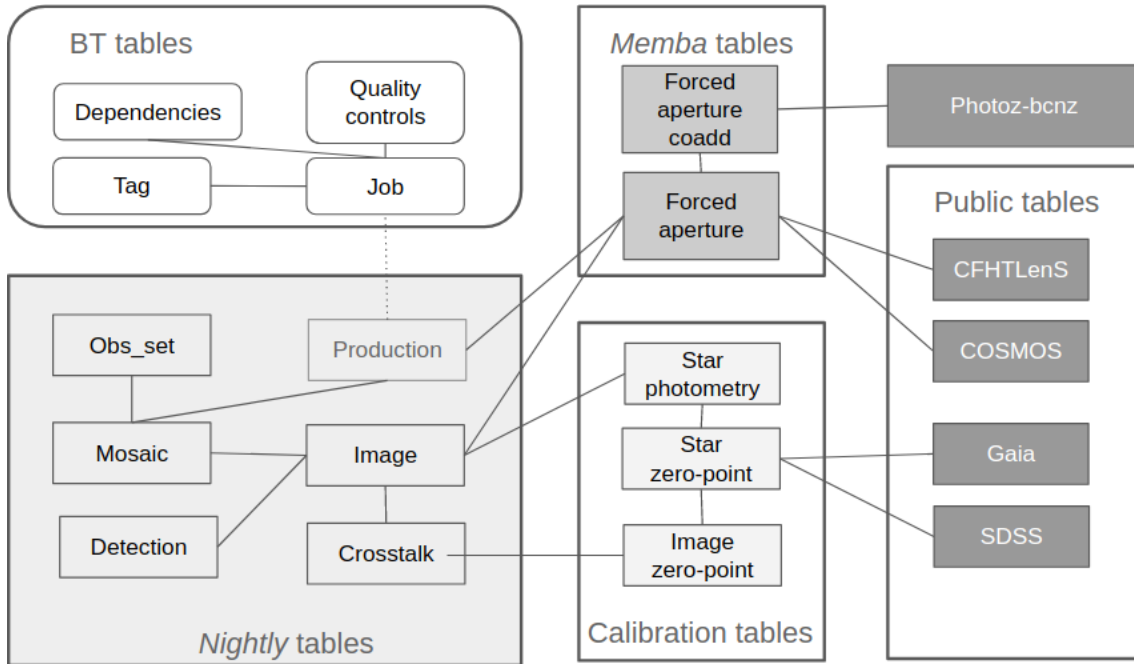


Figure 2.3: Schematic organization of the main PAUdb tables.

Figure 2.3 shows the high level organization of the main PAUdb tables (see tables 2.1, 2.2, 2.3, 2.4 and 2.5 in Appendix 2.A for a complete relation of tables and columns). Some of them are filled and queried during data reduction: they contain the metadata about raw files and files produced by the *nightly* pipeline, as well as calibration factors and intermediate values. The *production* table takes care of code version preservation. Job orchestration tables (for a description of the orchestration tool, see section 2.6) store configuration and metadata related to the tasks run in the computer farm, connected to the tables storing quality checks performed during data reduction. Other tables store the results of the multi-epoch multi-band analysis (*MEMBA*) intermediate and final catalogs. Public catalogs of reference surveys such as Gaia (Gaia Collaboration & et al., 2016), SDSS (Abolfathi & et al., 2018) and CFHTLenS (Heymans et al., 2012) are stored in PAUdb for calibration and analysis purposes.

PAUdb is interfaced using SQL commands or statements. This language enables users to specify, in a declarative manner, the operation they want to perform on the database, such as retrieving a subset of rows or modifying some existing values. Writing complex SQL statements usually requires full knowledge of the database model and understanding

⁸Source code available at <https://github.com/ptallada/paudm-model>.

of how relational databases work. There are also strong security concerns when those statements contain user supplied input, as that may lead to unexpected results such as information leaks, alteration or destruction.

In order to mitigate all those issues and to facilitate access to PAUdb by the pipeline developers, it was decided to proxy all pipeline database operations through an Object Relational Mapper (ORM) layer, allowing developers to interface with the database using the standard constructions present in their programming language. Having Python⁹ as the main programming language for the PAUS processing pipelines, we chose SQLAlchemy¹⁰ as the specific ORM solution because of its complete feature set and its comprehensive documentation.

The database structure, described in the ORM model, allows developers to access the database by importing a specific Python module in their code and using the set of classes, objects and methods defined in it, instead of having to manually construct SQL statements. For instance, querying data from another table linked by a foreign key is as easy as accessing a particular object attribute. Using such an abstraction layer comes with additional benefits, such as being able to change and evolve the database structure without interfering with users, as they only interact with the ORM model.

2.3.2 Limitations and improvements

The implemented relational database works great for the PAUS reduction pipelines. However, while not designed nor optimized to handle high metadata volumes produced by analysis jobs, we found out it worked well enough for most purposes.

To ease out large analysis tasks and to facilitate the publication and distribution of the data, the largest tables handling the products of the image reduction and multi-band analysis are migrated to the PIC Big Data platform. Once there, several services for interactive analysis and distribution, such as CosmoHub (see section 2.7.5 and chapter 3), make those tasks easier, faster and without limitations in data volume.

2.4 Short term storage and data transfer

PAUCam takes exposures of the night sky at the William Herschel Telescope, at Observatorio del Roque de los Muchachos in La Palma (Canary Islands). Each exposure is a Multi-Extension FITS (MEF) (Pence et al., 2010) file that has to be transferred to the PAU data center, preserving in its header the information (metadata) collected by the PAUCam control system related to the sky, weather and telescope conditions when the data was taken.

Raw data produced by PAUCam are saved in MEF files called mosaics, corresponding to one focal plane composed by the 18 PAUCam CCD images. Each extension contains data produced by the readout of one of the four amplifiers of each CCD, for a total of 72 extensions. The organization and collection of the information related to the MEF files is a joint effort between the PAUCam and PAUdm teams. A communication protocol,

⁹<https://www.python.org/>

¹⁰<https://www.sqlalchemy.org/>

assuring automation and consistency, had to be defined and agreed on. The MEF files are written by the PAUCam data acquisition system (Padilla et al., 2019) including all exposure metadata in the header.

Images are organized in observation sets. Each observation set is a group of contiguous exposures taken with the same telescope configuration. Metadata related to each observation set (date, operator comments, project name and list of files names with checksums) are collected in a YAML¹¹ file by the PAUCam data acquisition system. These metadata are not relevant for analysis but are useful for traceability and debugging purposes.

At the end of a night of observation, the telescope operator gives the command to start the archiving procedure. Data files (MEF and YAML) are moved atomically with their extended attributes, such as the adler32 checksum, from PAUCam to the PAUdm storage space located at the observatory.

The main objective of the transfer procedure is to copy all newly created observation sets from the temporary storage at WHT to the PAUdm archive at PIC. The transfer procedure has been evolving since the initial tests before PAUCam commissioning.

The first transfer scheme implemented used a push strategy and involved setting up a pair of Storage Resource Manager (SRM) (Donno et al., 2008) servers at both ends (WHT and PIC), managed by an instance of the File Transfer Service (FTS) (Ayllon et al., 2014). This first setup was heavily influenced by the WLCG infrastructure already functioning at PIC. Later on, this scheme was simplified by dropping the use of FTS (as its use was being deprecated at PIC) and handling ourselves the orchestration of the transfers. This also allowed the replacement of Grid certificates by private, internal self-signed certificates.

In 2018 we completely replaced the transfer procedure for a pull-strategy one based on the `bbcp`¹² tool to handle the bulk transfers, triggered automatically from PIC every morning. This tool was selected because it maximizes the bandwidth utilization even on high-latency wide area network (WAN) links, such as the original 1 Gbps link between PIC and ORM (2000 km apart).

After all the observation sets have been transferred to the PAUdm archive, including checksum verification, *register* jobs are submitted to insert the metadata from each exposure file into PAUdb. Temporary storage at WHT is manually freed as needed (about every 6 months).

Figure 2.4 shows a sample of the network traffic during the observation period of the second semester in 2017 (2017B), as well as the volume of data downloaded. Download speed is stable over periods of days and is around 20 MiB/s on average, in spite of the fact that PAU Survey shares the network link with all the other experiments operating at the observatory.

2.5 Analysis pipelines and nightly report availability

PAUdm operations are entirely carried out on top of the infrastructure available at PIC (see section 2.2). Evolved from a first basic script for image analysis that was designed

¹¹<http://yaml.org/>

¹²<https://www.slac.stanford.edu/~abh/bbcp/>

Network performance - Observation period: 2017B

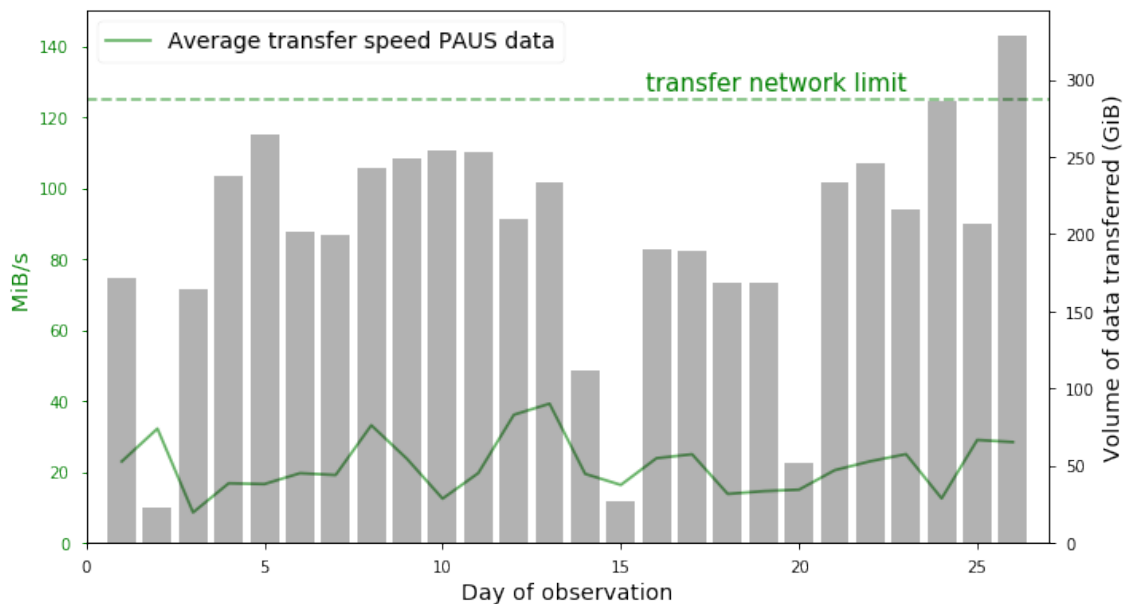


Figure 2.4: The solid line shows the daily average transfer speed registered at PIC for PAU Survey data (observation period 2017B). The bar chart shows the data volume transferred in the same day. The transfer rate performance looks quite stable and independent of the transferred data volume. Fluctuations are due to the fact that the network link is shared with other projects at the observatory.

to work on a single computer, the data reduction code has been acquiring complexity and completeness. The parallelization of the processes in different pipelines has been fundamental. Also, the use of many independent tasks helps scalability, and flexibility of the execution environment. The PAU database has dedicated tables, created in order to automatically activate execution of functions and orchestrate nightly operations.

The code of PAUdm is organized in pipelines written in Python, specifically designed to be run both in a HTC infrastructure or in a local computer. Each pipeline consists of one or more types of tasks. Each task type is connected to others by static dependencies. Tasks of the same type have a defined configurable set of parameters and can run in parallel to guarantee the scalability with the number of files to treat. Each task is composed of three parts: a *prolog*, a *run* and an *epilog*. The *prolog* and *epilog* of the task are able to generate new sub-jobs, with the associated dependencies and configuration. The *run* part executes selected functions on given input file(s).

PAUdm has defined the following main pipelines, schematically shown in Figure 2.5: *register*, *nightly* and *MEMBA*. Additional pipelines are *pixelsim*, for the simulation of raw PAUCam images, *crosstalk*, for the evaluation of the crosstalk effect on raw mosaics and its correction, and *photoz*, for the estimation of the photometric redshift of each source.

The *register* and *nightly* pipelines run automatically after each night of observation. Their dependencies and the level of parallelization of their tasks are shown schematically in Figure 2.6.

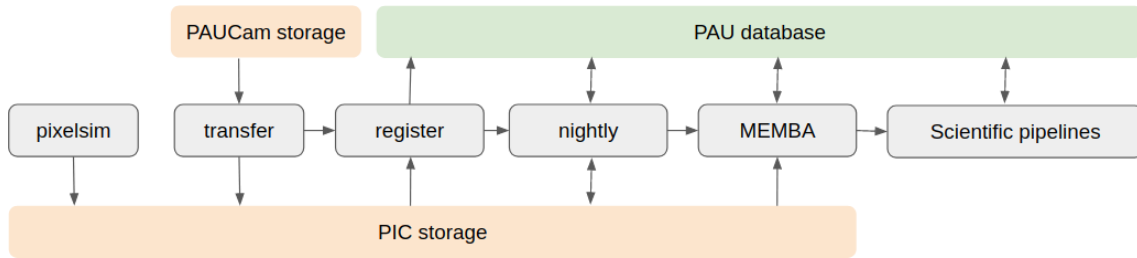


Figure 2.5: PAU data management pipeline dependencies. Gray rectangles identify pipelines and arrows their interfaces with the storage system and the PAU database.

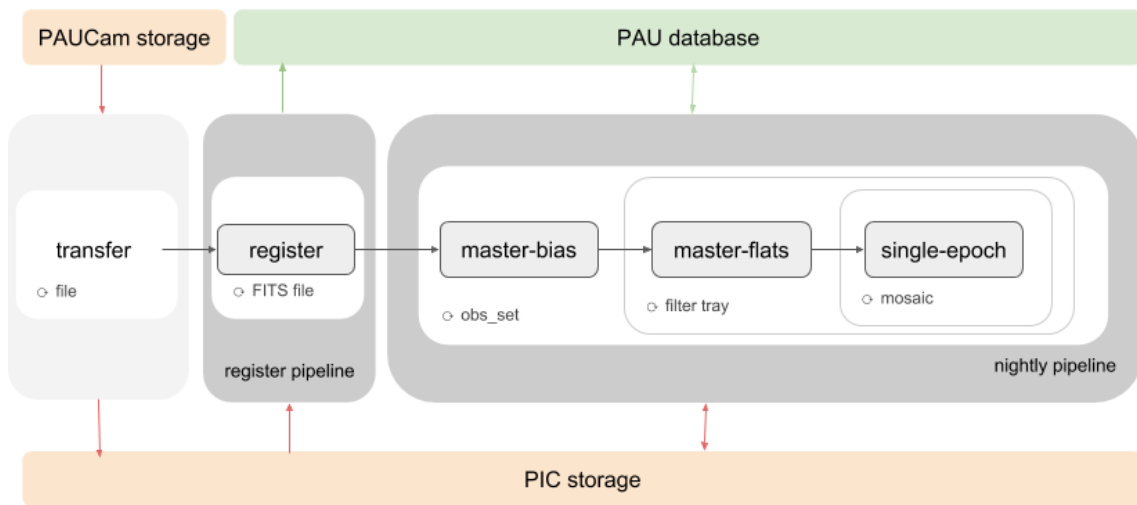


Figure 2.6: PAU data management *transfer*, *register* and *nightly* pipelines, and their relation with storage and the database.

The *register* pipeline is responsible for storing the metadata of all images in PAUdb as they are transferred, and has been parallelized to work every FITS file independently. Each *register* job reads the header of an exposure file and inserts the corresponding metadata in PAUdb: mosaics from the primary headers and individual images from extension headers.

Jobs in the *nightly* pipeline run following a hierarchical structure: for each observation set, the master bias is calculated first, followed by the master flats. Finally the *single_epoch* jobs that process each science mosaic are run in a separate job, as soon as the corresponding master-flats mosaic is available. The *nightly* pipeline also calculates image parameters and calibration factors that determine data quality used by *MEMBA*. The result of each run of this pipeline can be consulted in the nightly report, generated online by querying PAUdb. Each PAUS member can access them from the internal web page (see section 2.7.3).

Figure 2.7 shows part of the computer farm activity for PAUS, from the commissioning of PAUCam to the end of 2017, both in terms of number of jobs (each one of them occupying one slot in a node) and on wall time, i.e. the time spent from job submission to completion.

PAUdm pipelines monthly sum of jobs

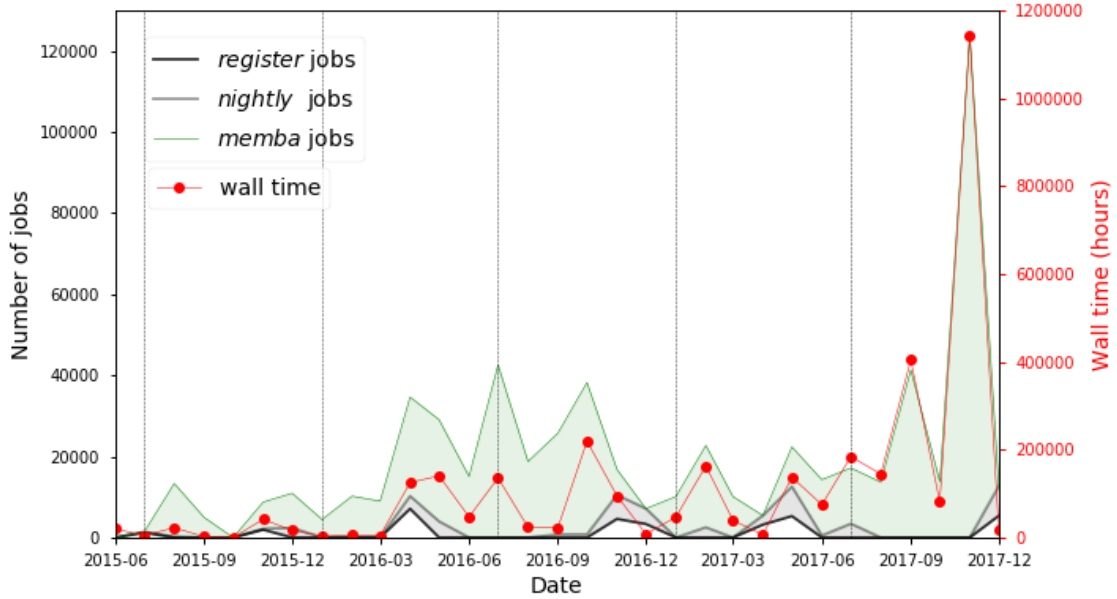


Figure 2.7: Monthly sum of the number of jobs (solid lines) and wall time (red dots), from commissioning on June 2015 to the end of 2017. Only PAU *register*, *nightly* and *MEMBA* jobs have been selected (other analysis and test jobs have been removed for clarity). *Register* jobs have been run during the PAUCam observation periods. *Nightly* jobs follow the *register* pattern. Extra *nightly* code releases have been run out of the observation periods over subfields of the survey for validation purposes. Most of the time spent in the computer farm is due to *MEMBA* jobs, the high number of jobs depending on the high level of parallelization of the pipeline. *Membra* jobs have been run with different configuration and different code complexity, explaining the non-linear dependency of the number of jobs with the wall time.

2.6 Job orchestration tool: BT

The PAUdm algorithms were initially executed as a single monolithic pipeline with a long and convoluted configuration to enable/disable specific steps and to provide the execution parameters. This setup, albeit useful to bootstrap the initial productions of PAUdm, was very hard to extend and maintain. This was improved following the iterations described below.

The first iteration was to split the different processing steps into jobs, with a fixed dependency tree connecting them. This modularization enabled the execution of an entire pipeline in smaller and shorter jobs, improving both the maintainability and the response times. Nevertheless, with the evolution and integration of additional processing steps, this mechanism became too rigid. The dynamic nature of the PAUdm pipelines required a more flexible orchestration method, where dependencies between jobs could be customised per job, or even created at run-time depending on the analysis results.

As a result, it was decided to develop a specific orchestration tool called *brownthrower* (BT¹³). Even though it was developed for the PAUdm pipelines, BT is a generic tool that may be used by other projects. In fact, it has been used for orchestrating several MICE and Euclid job productions. Developed on top of the PIC grid job scheduler (PBS, HTCCondor), it has the advantage of being able to dynamically establish and manage the dependencies between different jobs, therefore allowing the implementation of complex pipelines. In order to achieve an efficient use of computing resources, the processing defined in each job must run in a single batch slot of the PIC computer farm (generally defined as one core and 3-4 GiB of RAM).

The cornerstone of this tool is a relational database, currently in the same database server as PAUdb. It holds all the information about the jobs, such as their input and output data, dependencies, and configuration settings. BT makes heavy use of the transactional nature of the relational database to ensure the consistency and integrity of that information while tracking the status of every job. BT provides two tools to manage jobs: a manager and a runner. *BT manager* is a command line interface to create, configure and submit jobs for execution, query their status and abort them. *BT runner* is the tool that executes the job. Once launched in the proper environment, it starts pulling ready-to-run jobs from the database and executes them sequentially. In practice, *BT runner* is used as a pilot job (Sfiligoi, 2008), with several hundreds of instances running simultaneously on the computing farm, executing multiple jobs in parallel. Finally, the fact that all the jobs, past and present, are stored in a relational database, makes all data available for auditing and accounting purposes. The PAUdm jobs, organized in highly parallelized tasks, are orchestrated thanks to the connection between BT and PAUdb, both for jobs created automatically and manually.

2.7 Operations and metadata exploitation and distribution: PAUdm web services

The PAUdb constitutes the core part of the PAUdm system, providing all the relevant information that can be accessed from a series of tools for the automatic execution of tasks and for user friendly inspection and cross-check. A web interface provides a description of the PAUdb schema, an SQL Browser, and many graphical representations of the DB content, designed according to the users' needs.

The web interface functionality facilitates the PAUdb inspection in a graphic form, for operations control (jobs execution in the computing farm), data quality parameters inspection (calculated during the first images processes), the temporal evolution of the scientific parameters measured in the images.

The web interface is fully accessible to the PAUS members. In addition to the internal usage, it is meant to allow the entire scientific community to easily explore the PAUdb content, once it is published, similarly to what has been implemented for other cosmological surveys and astronomy projects. The advantage of having a single database with multiple access points and different functions guarantees the consistency, the provenance

¹³Source code available at <https://github.com/ptallada/brownthrower>.

and reproducibility of the results exposed. This constitutes the main difference between the previous solutions^{14,15,16} and what has been designed and implemented in this thesis.

2.7.1 Operations control

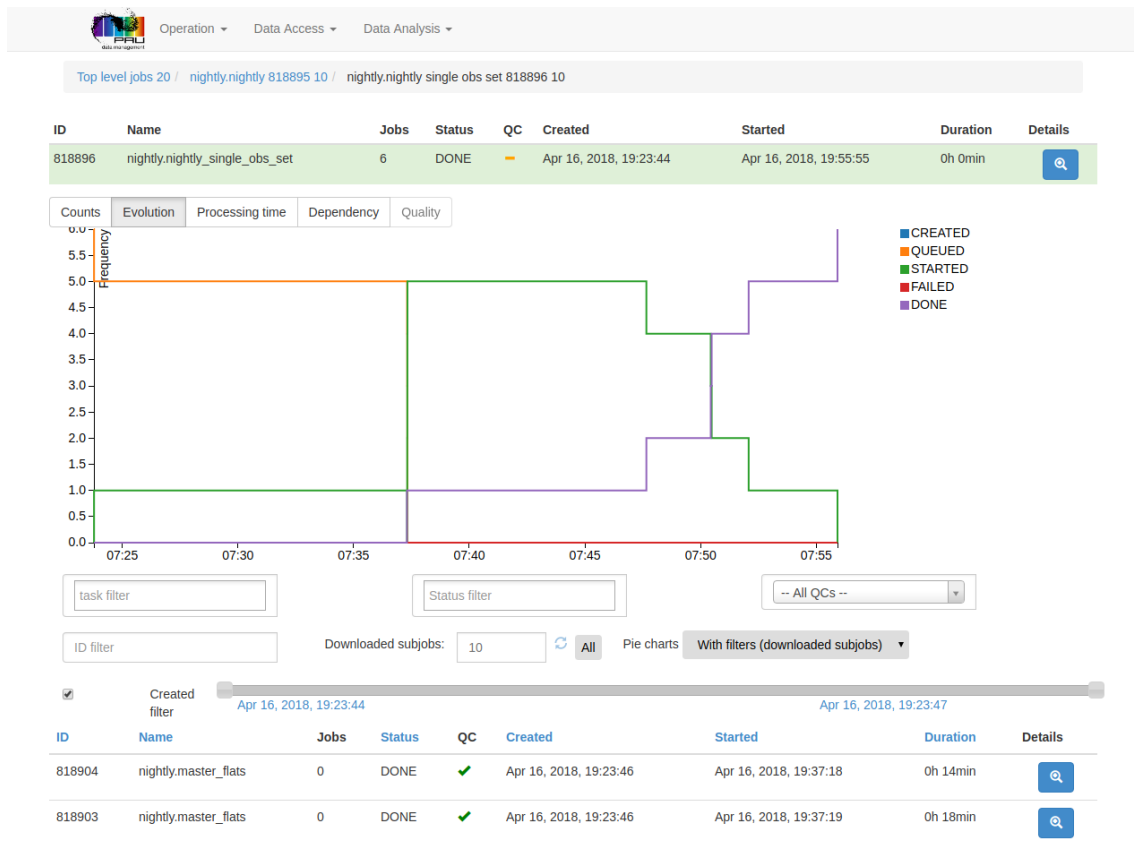


Figure 2.8: View of the PAUS web interface for operations control. Time evolution of jobs processing one observation set.

PAUdb permits the orchestration of the PAUdm pipelines jobs. During operations, pilot jobs are launched in the PIC computing farm and the tasks to be executed are automatically fetched from the PAUdb and executed. The details of each job recorded in PAUdb that can be accessed from the web page are:

- the configuration,
- input,
- output,
- log messages,

¹⁴<https://www.sdss.org>

¹⁵<https://ivoa.net/>

¹⁶<https://www.darkenergysurvey.org/the-des-project/data-access/>

- and, in case of an error, the traceback.

In particular, the job configuration and its error traceback allow for a quick look in case of a failed job.

Jobs are hierarchically structured. Each job can have sub-jobs, with static dependencies, as a result of the parallelization of the pipeline execution. From the web, the job hierarchy is maintained and sub-jobs details can be accessed clicking on the top level jobs.

A series of graphical views are associated with the operation control web page. The execution status chart, for example, gives a quick overview of the time evolution of the jobs status of a certain pipeline (Figure 2.8).

2.7.2 Data Quality inspection

During the execution of the tasks and in the *epilog* of the parent tasks, a series of quality checks are performed with results assigned to the executed job. The quality checks can be either numeric values of parameters, to be compared to a certain value range, and/or plots for visual check. Quality controls plots are generated by parent jobs, summarizing the evolution of interesting parameters calculated in their subtasks.

The results of the quality checks are registered in the PAUdb and linked to the corresponding job, the plots are stored in a dedicated disk space accessible both from the nodes executing the jobs and from the web server. These results can be viewed through the job details page.

The quality inspection is especially interesting during the process of validation and debugging. During normal operations, the quality checks result is monitored, shown as a green, red or yellow (in case of partial, but not critical failure) flag in the main job operations page.

2.7.3 Nightly Report

The main purpose of the Nightly Report web page is to provide feedback to the survey and science program planning process. Each PAUS member can access the nightly report and check the results of the basic pipelines running on the data taken each night, through an automatic query to PAUdb.

It provides an overview (see Figure 2.9) of all the image parameters taken in a given range of observation nights and stored as raw metadata in PAUdb. Additional metadata of scientific interest, like seeing, sky background, detrending status etc. are available to be queried by the page after the nightly tasks have finished successfully, while the information is safely stored in PAUdb. These parameters, displayed as plots, show the time evolution of observing conditions and data quality over the night.

2.7.4 Files and metadata access and distribution

The PAUdb webpage allows the user to access the PAU files in archive, the PAUdb schema and the metadata through an SQL search window. This tool permits running a direct



Figure 2.9: PAUS web interface: Nightly Report page example with plots.

query to the PAUdb tables using SQL, a language widely used in the astronomical community.

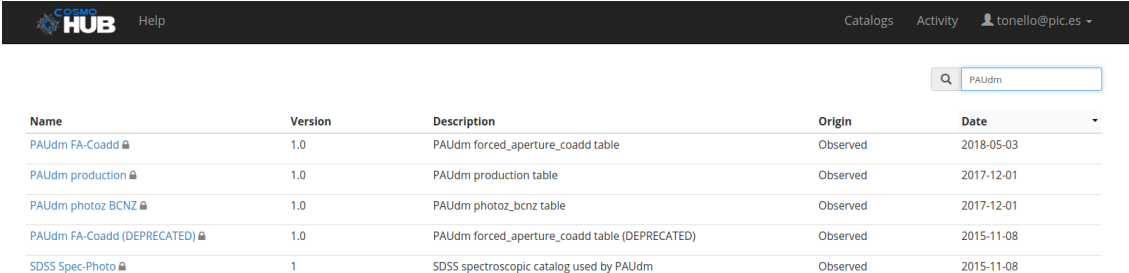
Construction of queries is supported by the publication of the full db schema: the name of all tables, fields, types, descriptions, and of available indexes. The PAUdb webpage is a powerful tool to query the database, visualize the results on-line in form of tables or plots (histogram, or scatter), create new fields applying functions or as a combination of multiple fields, and download query results (up to 10k rows) in CSV format (Shafranovich, 2005). This tool has been largely superseded by CosmoHub, that provides more features and performance, as explained in section 2.7.5.

The production table, where information of the general configuration of the pipelines execution is stored, is also published, with its full content (see Figure 2.3 for a schema of the PAUdb). Each entry of production table is associated with a pipeline and linked to its input pipeline through the field `input_production_id`, to guarantee full traceability.

Other functionalities and tools developed accessing PAUdb content under scientists request:

- Web visualization of PAUS analysis results (the so-called photometric spectra).
- Full comparison of the analysis output with the original telescope images
- Comparison of the analysis results with the results of external surveys (pre-loaded in PAUdb).
- PAUS catalogs distribution of massive cosmological datasets without any SQL knowledge being required (the origin of CosmoHub ¹⁷)

2.7.5 Catalog distribution: CosmoHub origins



Name	Version	Description	Origin	Date
PAUdm FA-Coadd	1.0	PAUdm forced_aperture_coadd table	Observed	2018-05-03
PAUdm production	1.0	PAUdm production table	Observed	2017-12-01
PAUdm photoz BCNZ	1.0	PAUdm photoz_bcnz table	Observed	2017-12-01
PAUdm FA-Coadd (DEPRECATED)	1.0	PAUdm forced_aperture_coadd table (DEPRECATED)	Observed	2015-11-08
SDSS Spec-Photo	1	SDSS spectroscopic catalog used by PAUdm	Observed	2015-11-08

Figure 2.10: PAUdm catalogs distributed in CosmoHub.

Spatial queries run directly on PAUdb, implemented using Q3C (Koposov & Bartunov, 2006) and PgSphere extensions over PostgreSQL, are not very efficient due to the size of the tables, even when indexes are available. When users request a large number of rows, databases ignore indexes and traverse the entire table, as it is faster than hoping between the index and the referenced table data. Moreover, the filtering criteria may be so complex that implementing every combination as an index becomes unfeasible.

One of the tools we have adopted to access and distribute data for the PAUS collaboration is CosmoHub, a web platform based on Big Data technologies developed at PIC to

¹⁷<https://cosmohub.pic.es/>

perform interactive exploration and distribution of massive cosmological datasets without any SQL knowledge being required. The latest release has been built on top of Apache Hive, a data warehouse based on Apache Hadoop which facilitates reading, writing and managing large datasets. CosmoHub is hosted at the Port de Informació Científica (PIC) and provides support not only to PAU Survey, but also to several international cosmology projects such as the ESA Euclid space mission, the Dark Energy Survey (DES) and the Marenstrum Institut de Ciències de l'Espai Simulations (MICE).

CosmoHub allows users to access value-added data such as sky or survey properties maps, to load and explore pre-built datasets, and to create customized object catalogs through a guided process. All those datasets can be interactively explored using an integrated visualization tool which includes 1D histogram and 2D heatmap plots. They can also be downloaded in standard formats. For an in-depth description of CosmoHub, see chapter 3.

We currently ingest into CosmoHub three tables of PAUdb (see Figure 2.10), containing metadata (productions information) and analysis results (table production and forced_aperture_coadd) and derived quantities (photometric redshift, or photo-z) results coming from pipelines using MEMBA data. Data from external surveys, used to calibrate, to perform forced photometry or to cross-check PAUS data (such as COSMOS, CFHTLenS or SDSS) are also accessible from CosmoHub for a direct comparison, in addition to mock galaxy catalogs created by the MICE collaboration, which are used for calibrating and testing the different pipelines. Access to this information allows the PAU Survey collaborators to explore and download the PAUdm *MEMBA* output catalogs for scientific exploitation.

2.8 Conclusions

The PAU Survey project considered that the data management solutions to adopt were crucial to determine its success, as it has been demonstrated by the previous and current generation Surveys (from SDSS on). PAUS data management is in charge of the data transfer from the observatory to the data center, the deployment and execution of the *nightly* pipeline and other analysis pipelines, as well as the organization and distribution of the final results and the metadata produced during the data processing.

PAUdb, implemented on a PostgreSQL database, is the successful solution adopted by the project for PAUdm operation and orchestration. It has been designed to be the pillar of the data management, not only to preserve information but also its consistency, from the raw metadata to images, object catalogs and results reproducibility. It also keeps track of the archive status, the processes that have been run, and it has been developed with the idea of enabling data accessibility as well as distribution of final observed catalogs. The source code of the data model and its ORM layer is available at <https://github.com/ptallada/paudm-model>.

PostgreSQL has been proven to be a good choice for frequent data insertion, deletion and update. Our implementation has limits when over several hundred concurrent connections are reached, limiting parallel execution to about 250 concurrent jobs. This, however, does not impact routine PAUdm operations.

BT¹⁸, a locally developed tool, is used for orchestration of PAUdm tasks on the HTC infrastructure at PIC. It should be noted that this development was undertaken when Grid computing was not fully mature and most of the tools available today were not in existence. If such development had to take place today, we would have probably used an already existing tool such as Celery¹⁹. A posteriori, the size and complexity of the PAU Survey project justified this development, which has provided many advantages and custom features such as the next day feedback to the observation scheduling.

The performance of the automatic transfer procedure has fulfilled the PAUS project needs. The data downloading process from the observatory in La Palma to PIC finalizes in a few hours, with outstanding reliability during normal operations, and full recovery in the rare cases of failure. While more sophisticated tools would allow the same reliability with higher automation, the data volume to be transferred per night, the concentration of the transfer in a few weeks per years, and the relatively short lifetime of the project favoured a simpler solution.

The high level of parallelization of the pipelines in independent jobs, the number of computing nodes and the distributed file system available at PIC for the PAUdm jobs process allows us to fulfill the requirement of having nightly results and data quality available on time for optimizing the survey program. The HTC infrastructure enables reprocessing of all survey observations within a few days.

Web services have been implemented for PAU data and metadata access, retrieval and analysis. They have received a very positive feedback from collaborators, due to their usability and information content. The data services implemented are giving key support to the first scientific productions exploiting the PAUS data (see Cabayol et al. 2019; Stohert et al. 2018; Tortorelli et al. 2018).

The first version of CosmoHub (see chapter 3 for an in-depth description of the subsequent implementation) over PAUdb was an excellent solution for catalog exploration and distribution. The process of migrating PAUdb tables to this platform is still manual. The synchronization of contents between PAUdb and CosmoHub not only improves the handling of PAUS catalogs, but also their findability, once the standardization of CosmoHub content in the Virtual Observatory paradigm is completed.

¹⁸Source code available at <https://github.com/ptallada/brownthrower>.

¹⁹<https://docs.celeryq.dev>

2.A PAU database schema

In the following we show the names, description and column list of each of the main tables of the PAUdb schema. Foreign keys, which determine the relation between tables, are marked in *italic*.

Table 2.1: PAUdb tables

Table name	Description	Columns
production	Tracks the different processing production runs for all pipelines	comments, created, id, <i>input_production_id</i> , <i>job_id</i> , pipeline, release, software_version
mosaic	List of mosaic exposure images (raw and reduced)	airmass, amb_temp, archivepath, astro_chi2, astro_contrast, astro_href_sigma, astro_nstars, astro_nstars_highsn, astro_ref_cat, astro_ref_sigma, astro_status, comment, date_creat, date_obs, dec, detrend_status, equinox, exp_num, exp_time, extinction, extinction_err, filename, filtertray, filtertray_tmp, guide_enabled, guide_fwhm, guide_var, humidity, id, instrument, kind, mean_psf_fwhm, merged_mosaics, nextend, <i>obs_set_id</i> , obs_title, photo_status, pressure, <i>production_id</i> , psf_model_status, ra, rjd_obs, telfocus, time_creat, time_obs, wind_dir, wind_spd, <i>zp_phot_id</i>
image	List of images associated with the mosaics (CCD and single amplifier images)	amp_num, bkg_mean, bkg_std, ccd_num, cosmic_ratio, dec_max, dec_min, filter, gain, id, image_num, max_readnoise, <i>mosaic_id</i> , naxis1, naxis2, n_extracted, psf_fit, psf_fwhm, psf_stars, ra_max, ra_min, rdnoise, saturation_ratio, transparency, waveband, wavelength, zp_nightly, zp_nightly_err, zp_nightly_stars
obs_set	List of Observation Sets registered in the database	id, instrument, log, night, notes, obs_set, operator, rjd_start, rjd_stop
project	Description of projects observing with PAUCam	contact_email, contact_name, created_at, description, id, name
crosstalk_ratio	Crosstalk correction to be applied to each amplifier	amp_num_dest, amp_num_orig, ccd_num_dest, ccd_num_orig, <i>production_id</i> , ratio
detection	Detections measured directly on the image after the nightly data reduction	id, image_id , insert_date, band, background, class_star, spread_model, spreaderr_model, flux_auto, flux_err_auto, flux_psf, flux_err_psf, flux_model, flux_err_model, flags, elongation, dec, ra, x, y, zp_offset

Table 2.2: PAUdb nightly calibration tables

Table name	Description	Columns
phot_method	Photometric methods	background_method, background_parameter, comments, extraction_code, extraction_method, extraction_parameter, id, scatterlight_method, scatterlight_parameter
phot_zp	Photometric zero-points	band, date, id, <i>production_id</i> , zp
template	SED of star templates references	filename, id, template_index, template_lib, template_name
star_photometry	Calibration stars fluxes	bg, bg_err, flags, flux, flux_err, id, <i>image_id</i> , <i>phot_method_id</i> , ref_cat, ref_id, x_image, y_image
star_template_zp	SED template associated with each star	id, <i>star_zp_id</i> , <i>template_fit_band_id</i> , zp, zp_error, zp_weight
star_zp	Star zero points calculated in the nightly calibration	calib_method, id, <i>star_photometry_id</i> , zp, zp_error, zp_weight
image_zp	Image zeropoint measurements for each photometry-calibration method	<i>calib_method</i> , <i>id</i> , <i>image_id</i> , <i>phot_method_id</i> , transparency, zp, zp_error

Table 2.3: PAUdb *memba* and catalogs tables

Table name	Description	Columns
forced_aperture	Contains the single measurements using force photometry in <i>memba</i> for each band and pass, and for each reference source	annulus_a_in, annulus_a_out, annulus_b_in, annulus_b_out, annulus_ellipticity, annulus_median, annulus_samples, annulus_sigma, aperture_a, aperture_b, aperture_theta, aperture_x, aperture_y, flag, flux, flux_error, image_ellipticity, <i>image_id</i> , pixel_id, <i>production_id</i> , ref_id
forced_aperture_coadd	Contains the combined measurements using force photometry in <i>memba</i> for each band and for each reference source	band, chi2, flux, flux_error, n_coadd, <i>production_id</i> , ref_id, run
sdss_spec_photo		_class, dec, extinction_g, extinction_i, extinction_r, extinction_u, extinction_z, fiberID, fiberMagErr_g, fiberMagErr_i, fiberMagErr_r, fiberMagErr_u, fiberMagErr_z, fiberMag_g, fiberMag_i, fiberMag_r, fiberMag_u, fiberMag_z, mjd, mode, modelMagErr_g, modelMagErr_i, modelMagErr_r, modelMagErr_u, modelMagErr_z, modelMag_g, modelMag_i, modelMag_r, modelMag_u, modelMag_z, objID, plate, ra, specObjID, subClass, survey, tile, z, zErr, zWarning
memba_ref_cat	Reference catalog used for a given <i>memba</i> production	<i>production_id</i> , ref_cat
photoz_bcnz	Photometric redshifts	chi2, ebv, n_band, odds, <i>production_id</i> , pz_width, ref_id, zb, zb_mean

Table 2.4: PAUdb BT tables

Table name	Description	Columns
dependency	Tracks the dependency between Brownthrower jobs (Operation table)	<i>super_id, parent_id, child_id</i>
job	Tracks the list of Brownthrower computing jobs (Operation table)	config, description, id, input, name, output, status, <i>super_id</i> , token, ts_created, ts_ended, ts_queued, ts_started
quality_control	quality control entries measured during the data reduction process	check_name, id, <i>job_id</i> , max_value, min_value, plot_file, qc_pass, ref, time, units, value
tag	Configurable tags associated with a job (tracebacks, logs, etc.)	<i>job_id</i> , name, value

58

Table 2.5: PAUdb Public external tables

Table name	Description
cosmos	External table from zCOSMOS. Sources with accurate redshifts for forced photometry and validation
sdss_spec_photo	External table from SDSS DR12 (SpecPhoto view). Sources with spectra for forced photometry and validation
spec_conv	Convolved fluxes derived from spectra observations from external surveys (i.e. SDSS, COSMOS, DEEP2, etc.
usnostars	Stars from USNO catalog
yalestars	Stars from Yales catalog
cfhtlens	CFHTLenS catalogue for Forced Photometry
deep2	Deep2 catalog
gaia_dr2	Gaia DR2 catalog for astrometry and calibration

Chapter 3

CosmoHub

When managing huge datasets, the main challenges that scientists have to deal with are the interactive exploration, customization and distribution of data. Typical cosmological surveys can produce datasets of several terabytes in size. The needs of the scientists to deal with such huge datasets, and the novel affordable technological and computational solutions, lead me to propose and finally guide the implementation of CosmoHub, that will be explained in this chapter.

CosmoHub (<https://cosmohub.pic.es>) is a web platform based on Hadoop to perform interactive exploration, customization and distribution of massive cosmological datasets, with over 60 TB of catalogued information and 50 billion astronomical objects, without any Structured Query Language (SQL) knowledge required. Hosted at the Port d'Informació Científica (PIC), it currently provides support to a worldwide community of more than 1400 scientists and several leading edge collaborations. It was born as an extension of the solution implemented for the PAU Survey, with the intention to give a generic service, suited for managing cosmological catalogs coming from galaxy surveys and other similar projects. I investigated the possible solutions, prepared the technical design and the implementation plan of the platform, and developed all of its core features, while the deployment and web frontend has been carried out in collaboration with the staff¹ at PIC.

The main objectives of CosmoHub are to allow users to access and download custom subsets of the datasets hosted in it and to do some quick exploration using its integrated plotting tools, all of this using a simple guided interface. It is the first platform of its kind to introduce Hadoop as the storage and computing platform of choice for cosmological datasets. Its frontend has been designed around modern web standards, with special emphasis on simplicity and usability to appeal to a broader audience. As it can be accessed through a web browser, it does not require any kind of deployment or installation on the final users computers.

¹Special mention to Jordi Casals, for the design and implementation of the frontend; Marc Caubet, Carles Acosta and Agustín Bruzzese for the operation of the Hadoop cluster; and Ricard Cruz for the infrastructure maintenance.

CosmoHub hosts a collection of catalogs from a wide array of cosmological projects, such as COSMOS Laigle 2015², Gaia³ DR3⁴, the Dark Energy Survey (DES⁵) DR2⁶, the Physics of the Accelerating Universe Survey (PAUS) PAUS+Cosmos photo-z⁷, the MICE numerical simulations⁸, and the official Euclid Flagship galaxy catalogs⁹, among many others. Each of those catalogs belongs to one or several research groups or projects. Public catalogs do not require any specific membership or permission to access its data, while access to restricted data is regulated still manually by the data owner.

Since its migration to Hadoop in 2016, CosmoHub has demonstrated its ability to deliver cutting-edge capabilities for researchers in the field, empowering them to explore and analyze cosmological data with unprecedented efficiency. The platform has witnessed a steady increase in both user engagement and data volume, up to 150 monthly active users and more than 60 TiB of catalogued data. However, this has not impacted its response times, as 96.8% of all interactive exploration plots are generated in less than two minutes, while 97.4% of the custom catalogs are delivered within a timeframe of 30 minutes.

This chapter describes CosmoHub and the technological decisions that drove its design and development and is structured as follows. Section 3.1 introduces the scientific context, section 3.2 enumerates the main user requirements that had to be addressed, section 3.3 outlines the general architecture of the solution implemented, section 3.4 dwells into the earlier iterations of the platform, section 3.5 characterizes the Hadoop platform that powers CosmoHub, section 3.6 details the implementation of the multiple download data formats, section 3.7 depicts how we developed several sets of User Defined Functions to extended the functionality, section 3.8 portrays the design and development of the web application, section 3.9 reports the main results of this work and section 3.10 summarizes and concludes.

²<http://cosmos.astro.caltech.edu/page/photom>

³<https://www.cosmos.esa.int/web/gaia>

⁴<https://www.cosmos.esa.int/web/gaia/dr3>

⁵<https://des.ncsa.illinois.edu>

⁶<https://des.ncsa.illinois.edu/releases/dr2>

⁷<https://pausurvey.org/pauscosmos-photo-z-catalog/>

⁸<http://maia.ice.cat/mice/>

⁹https://www.euclid-ec.org/?page_id=4133

3.1 Introduction

Experimental astronomy has entered in recent years into a new data regime, mainly due to the construction and development of ground —and space— based sky surveys¹⁰ in the whole electromagnetic spectrum, from gamma rays and X-rays, ultraviolet, optical, and infrared to radio bands. This trend will increase with the next generation of projects, for example: (i) the future 3.2 GigaPixel LSST camera (LSST Dark Energy Science Collaboration, 2012) will take images every 30 seconds and the data rate will be about 15 terabytes per night¹¹, (ii) the complete Euclid survey (Laureijs et al., 2012) represents hundreds of thousands images and several tens of petabytes of data; the final Euclid source catalog will contain about 10^{10} entries¹².

A substantial part of the success of a scientific project can be measured by the impact its results have on the scientific community. Also, having powerful tools to facilitate exploration and distribution of data is key to boost their usage. With open science principles in mind, cosmology surveys need to adopt different solutions to share and distribute their data, including analysis tools.

One of the most successful and innovative galaxy surveys is the Sloan Digital Sky Survey (SDSS) (York et al., 2000). The enormous success of this project is due to —besides the quality of the data— the fact that its results are fully public and easily accessible¹³. They have put great effort into facilitating scientific exploitation by any user, regardless of their technical expertise (see Szalay et al. (2002) and Raddick et al. (2017)).

Web interfaces are a common tool to provide access to cosmological datasets. For instance Cosmosim¹⁴ provides results from cosmological simulations performed within different projects. The Millennium Run Observatory¹⁵ (Overzier et al., 2013) is a theoretical virtual observatory which uses virtual telescopes to *observe* semi-analytic galaxy formation simulations. The Theoretical Astrophysical Observatory¹⁶ (TAO, Bernyk et al. 2016) also gives access to multiple popular simulated datasets but, in addition, the datasets can be funneled through higher-level modules to build custom mock galaxy catalogues and images.

Most recent surveys have also created dedicated portals to manage access to their data releases. For example, the Dark Energy Survey (The Dark Energy Survey Collaboration, 2005) has produced the DES Science Portal (Gschwend et al., 2018). Future surveys like LSST are putting a tremendous effort into designing adequate tools to access and analyze the massive amounts of data they will generate (Jurić et al., 2017).

CosmoHub origins can be traced back to the beginnings of the Physics of the Accelerating Universe Survey (PAUS) project in 2013. At that time, PAUS had started to produce its first simulated data that needed to be distributed to its collaborators. In order to facilitate the distribution, a pilot web interface called "PAUdm Simulations Portal" was commissioned and integrated into the official PAUdm operations web. This prototype offered access to the PAUdm database hosted in a PostgreSQL server.

¹⁰See <http://www.astro.ljmu.ac.uk/~ikb/research/galaxy-redshift-surveys.html> for a non-complete list of galaxy surveys

¹¹<https://www.lsst.org/about/dm>

¹²<https://www.euclid-ec.org/>

¹³<https://www.sdss.org/dr15/>

¹⁴<https://www.cosmosim.org/>

¹⁵<http://galformod.mpa-garching.mpg.de/mrobs/>

¹⁶<https://tao.asvo.org.au/tao/>

The amount of data stored in the prototype grew substantially. Most of it came from external data used in PAUS pipelines, such as SDSS star catalogs and MICE mock galaxy catalogs. In time, MICE started ingesting more of its own data in the same database, in order to be able to use the web interface. Through several iterations we ended up implementing some dataset exploration features.

After a few iterations, we decided to promote this PAUS dataset exploration tool into a full fledged independent product that could give service to lots of projects with similar requirements. This implementation started as early as 2013, with the express idea of offering a web application for the interactive exploration and distribution of massive cosmological datasets. It also had to leverage an intuitive user interface so users with no Structured Query Language (SQL, Chamberlin and Boyce (1974)) knowledge could visualize and download customized subsets of the data without any difficulty. This first version used the same PostgreSQL as PAUdb to handle the data, following a similar approach as the one adopted by other projects (SDSS uses Microsoft SQL Server, while DES uses an Oracle Database). A few years later, we started struggling with performance issues due to the increasing amounts of data we were managing, and we decided to revisit our design choices.

In order to overcome the performance and scalability limitations of PAUdb when processing CosmoHub’s queries, we explored alternative setups. The most promising ones were based on distributed architectures that enabled linear scaling of both storage and processing power and, in addition, their data operation semantics were better suited for CosmoHub’s data workflow. The results of all this effort are described in Tallada et al., 2020, a re-engineered version of CosmoHub released in late 2016 and powered by Apache Hive, a data warehouse solution based on Apache Hadoop which facilitates reading, writing and managing large datasets. First described in Carretero et al., 2017, CosmoHub is one of the earliest implementations of a storage and computing platform for cosmological datasets based on Apache Hive. Some other works have also used other Apache Hadoop components to deal with massive datasets, e.g. Apache Spark in Plaszczynski et al., 2019.

At the time of writing this thesis, over 60 TiB of catalogued information and 50×10^9 astronomical objects from a dozen different projects can be interactively explored using an integrated visualization tool which includes 1D histogram and 2D heatmap plots. Interactive visualization of datasets of thousands of millions of objects can be done in less than a minute, and customized subsets can be generated in a timescale of minutes.

3.2 User requirements

Most of the CosmoHub’s objectives originated from our experience in designing and developing the PAU Survey data management (PAUdm) (Tonello et al., 2019), and from the interactions of the PAU Survey project with other peer projects such as MICE (Fosalba, Crocce, et al., 2015, Crocce et al., 2015, Fosalba, Gaztañaga, et al., 2015, Carretero et al., 2015 and Hoffmann et al., 2014) and DES. The following list defines the set of key requirements for CosmoHub.

- *Centralized data distribution*

Having a unique point of data distribution enables having a single, authoritative version of the data, reducing the risk of duplicated, corrupted or deprecated replicas. A unique entry point also facilitates the enforcement of access controls and policies.

Also, when relying on a common platform, it is important to minimize the risk of having a single point of failure. Therefore, this platform has been configured in a high-availability setup where service is not disrupted by eventual failures of its individual components (see section 3.3).

- *Easy to use*

Usability is also a key for the success of this kind of platform. The easier it is to use, the more users it will engage and, therefore, the data published on it will reach a wider audience, increasing their impact on the scientific community.

Interfaces should be clean and simple enough such that any user may use the service without prior training. In detail, the following two issues should be addressed:

- *No Structured Query Language (SQL) knowledge must be required*

In a data distribution service, SQL is the most common interface for interacting with the data. SQL is a declarative language that provides a set of constructs to select, project, filter and retrieve subsets of information from a database. As an industry standard, most (if not all) vendors of data warehouses offer a SQL interface to interact with their own services.

Exposing an SQL interface is problematic for at least two reasons: First, while SQL knowledge is common in technical circles, many scientific users are not trained with it. And second, SQL is an industry standard, but has different vendor implementations that deviate from official specifications. These differences originate from adding complementary features or because their implementation predates the official specification. This means that even users with training on data warehouses might encounter problems because the SQL they know does not match the exact flavour used in a given solution.

- *Standard file formats*

The astronomical community has grown used to a standard set of formats for data interchange. Consequentially, specific tools for managing, processing and visualizing data stored in these formats have been developed and are widely used. Therefore, we must support those formats to enable our users to keep using the tools they already master and make the interoperability with them as straightforward as possible.

This usability objective has been considered for end users, but also for system administrators. They should be able to deploy and keep the service operational, avoiding that eventual hiccups pose a threat to service availability.

- *Custom datasets*

One of the main challenges of managing large datasets is to be able to efficiently generate small customized subsets fitting the scientists' needs.

Allowing the generation and download of custom subsets enables users to minimize data storage and transfer costs. At the same time, processing costs are also reduced as the selection and filtering part is offloaded onto the service, which has plenty of resources optimized for this task.

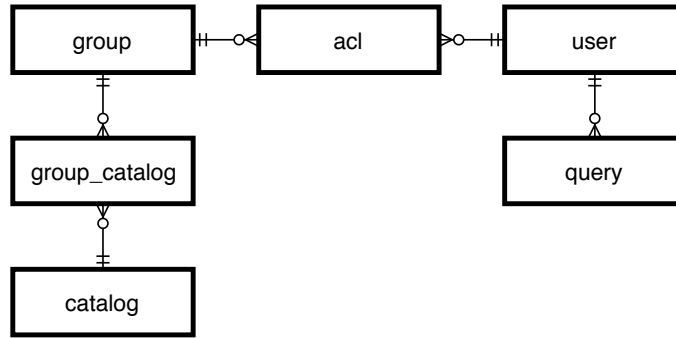


Figure 3.1: CosmoHub’s data model, which stores information about its catalogs, groups, users and access control lists (acl), among other metadata. Users can access all the data in CosmoHub of those groups they are member of. Users request access to the groups they are affiliated to when they register in the platform and those requests are validated/granted by an admin of the project/group.

As a downside, the service must cope with the additional storage required for all (potentially overlapping) subsets that the users have requested, including some means to eventually expire or purge them.

- *Quick exploration*

The ability to create custom datasets is not useful if the user does not know the exact criteria to define them or is not confident enough on the properties of the resulting subset. Having some functionality to interactively explore and preview the results of subset generation is very helpful.

This quick exploration tool can be offered in different ways, for example row sampling or simple visualizations such as scatter plots and histograms. Once the users are certain that the subset matches their expectations, they can proceed with the download, as needed.

3.3 General architecture

This section describes the general design of the architecture of CosmoHub, from a technologically agnostic viewpoint:

- *Target audience*

Prospective users of CosmoHub are the thousands of scientists around the globe collaborating in astronomical projects that manage and/or produce large amounts of catalogued data. Most of these data end up released to the public and, if a replica is available in CosmoHub, any registered user may access, explore and download it.

Furthermore, these projects usually have private datasets for internal processes, such as release validation or calibration, which are only available to project members.

- *Metadata database*

CosmoHub provides access to a collection of catalogs or astronomical datasets hosted in our data warehouse. Each catalog is defined by a name, a short description and a

summary of its characteristics. Each catalog is mapped to a single table in our data warehouse containing a set of columns, which are described by its data type and a short description.

Each catalog belongs to one or several projects or research groups. Users can access all data in CosmoHub associated to the groups they are member of. Users request access to the groups when registering in the service. Each group has a set of users with special privileges who are in charge of validating and granting the corresponding membership requests. Access to specific groups or projects can be updated at any time through a web interface. Users receive an email notification whenever their privileges change. A catalog may also be public and, as such, not require any specific membership to access it.

Figure 3.1 shows the relational data model of all this metadata. In particular, in addition to storing the information about catalogs, groups and users, it also includes the relationships between them such as project membership or group administration privileges.

- *Web interface*

CosmoHub is designed as a web application. This solution only needs a web browser, that usually comes preinstalled on any computer, and an internet connection. Moreover, this also allows to reuse all the user experience of the web semantics and graphical metaphors that most users are already used to.

- *Guided process*

In order to facilitate the usage of the platform to those users unfamiliar with SQL, the custom subset building interface has been designed as a guided series of steps that can be followed very easily even by the most inexperienced user.

The web interface guides users through a sequence of steps, allowing them to select catalogs they are interested in, then the columns they need, adding filtering criteria (if needed) and choosing the download format.

They can also plot and preview the dataset they are building with the integrated plotting tool, also implemented with intuitive and easy to use web forms to configure each type of plot (see section 3.8 for details).

- *SQL expert mode*

CosmoHub also offers the possibility to unleash the full power and capabilities of SQL. The "expert" mode allows to write an SQL query directly and passes it to the underlying database for its execution. This feature allows more experienced users to define additional computed columns using standard functions and operators, specify arbitrary groupings or even perform joins. The latter is very interesting because it allows matching and combining data from several different catalogs.

Furthermore, these capabilities can be extended by implementing additional user defined functions (UDFs) through a set of JAR files. For now, users cannot add their own UDFs for security reasons and are instead directed to contact us if they have a specific need. The list of implemented UDFs (described in section 3.7) is linked in the "expert" mode and also available in the Help section¹⁷.

¹⁷<https://cosmohub.pic.es/help>

- *(Simple) visualization*

Users may get a quick insight on the data they are selecting by using the four integrated visualizations that CosmoHub offers:

- Table overview shows 20 rows of the subset.
- Scatter plot, to visualize trends and relations between different columns. It also supports plotting different series of data, but is limited to plot only 10k points, so users only see partial results/behaviour of the full subset. Therefore, the plot may not be representative of the subset as a whole.
- 1D histograms and 2D heatmaps, both with automatic hints on column names, and bin ranges.

1D histograms and 2D heatmaps are aggregated plots implemented on the backend. They get the full picture of the custom subset. For performance reasons they are limited to less than 10k uniform bins, which is by far enough for most applications.

- *Batch custom subsets*

When users finish exploring a subset, they can select a download format and request its generation and delivery. Among the formats, special attention must be put in supporting Flexible Image Transport System (FITS) (Wells & Greisen, 1979), which is one of the most popular data formats in astronomy.

The custom subset is built in the background by the underlying database engine. When the custom subset is ready and stored, an email will be sent to the user with a link that they must follow in order to start the download.

- *Security*

CosmoHub hosts private datasets and also serves as the authoritative catalog repository for a number of projects. As such, it is of utmost importance to ensure the integrity of any stored dataset and to prevent any unauthorized accesses to them. All SQL queries, including those coming from the "expert" mode, are screened by the API. In this process, each SQL sentence is parsed by Hive and the list of catalogs to be accessed is retrieved. Then, each catalog is checked against the user's privileges stored in CosmoHub's metadata (see Figure 3.1). If a user tries to access a catalog without the required permissions, the action is logged and an error is returned.

Furthermore, in order to better ensure the integrity of the datasets, all sentences that are not pure SELECTs are filtered at the API level too.

3.4 Early prototypes

From the beginning, CosmoHub was designed in a way that no specific technical knowledge was required in order to exploit its functionalities. In particular, users were already able to formulate queries without any SQL knowledge through a guided process. Also, users were able to directly download Value-Added-Data (prebuilt catalogs or other information needed to analyze the data, such as filter curves or dust maps), they could visualize general data trends using simple plots and, of course, download custom subsets which were created asynchronously in the computing facilities at PIC.

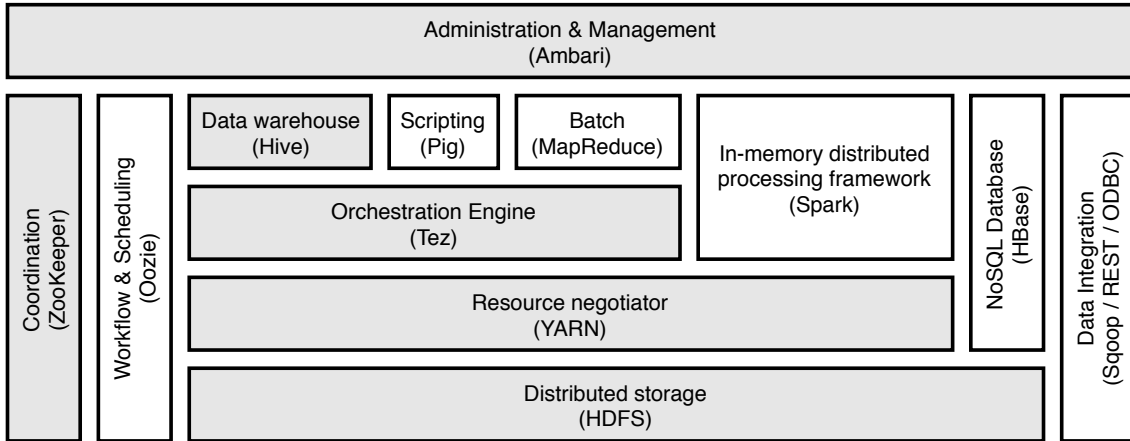


Figure 3.2: Hadoop layered ecosystem, showing how multiple components stack on top of others to provide a broad set of features and services. In grey, the components used for CosmoHub.

After a year, the performance of the database server—designed specifically to host only PAUS data—started to suffer. The amount of data hosted kept growing (from 1 to 10 TB), mainly due to catalogs ingested from external projects, the storage space became tight and response times degraded (queries over large catalogs could take up to 2 days to complete).

In this situation, the first affected feature were the interactive plots, which at that time were limited to 10k rows and queries taking less than 2 minutes to complete. With the increasing size of the catalogs, most queries did not fulfill the response time requirements even using custom indexes.

I redesigned the database part and proposed to migrate to another instance of PostgreSQL database in a separate server, with much more storage space and similar processing power. In that way, we mitigated two problems: the limited storage available and the competition of computing resources with the main PAUdm processing pipelines.

Nevertheless, the problem of solving the long response times was hard to tackle. In our experience, traditional relational databases such as PostgreSQL can deal with huge datasets, as long as you work with them in small chunks. But when the requested data is above a certain threshold, the PostgreSQL query optimizer does not use indexes as it is not efficient anymore. That is the main reason why most of the queries ended up performing a sequential scan of the entire table, resulting in response times that could take several hours and even a few days. Lastly, although used very sparsely, modifying any table schema and removing large amounts of rows were extremely inefficient operations.

For these reasons PostgreSQL was not the right tool for CosmoHub’s data workflow and we explored different possibilities (see section 3.5) to solve the problem. The best solution we agreed on was to use Apache Hive, a data warehouse based on Hadoop.

3.5 Hadoop Platform

Assessing alternatives From the experience gained from prototypes, we knew that choosing the right data storage and processing platform was fundamental to achieve our objectives. Therefore, we researched and tested several alternatives that we thought could be useful. There are multiple solutions in the market to handle large structured datasets in a manner that is scalable and has good performance, such as NoSQL (or non-relational) databases and distributed relational databases. We only took into consideration the open source alternatives due to technical and economic reasons.

We knew from the beginning that one of the key requirements of CosmoHub would be the ability to hold multiple large datasets, and to be able to analyze, compare and cross-match them. The particular architecture of NoSQL solutions such as HBase¹⁸, Cassandra¹⁹, MongoDB²⁰ or Redis²¹) normally does not allow for efficient joins between datasets. Furthermore, each of them implement a different language for interacting with the data, with partial SQL support, at best. For these reasons, we decided to discard the NoSQL solutions.

For the distributed relational databases, we studied two approaches. First, we tested clustered implementations of traditional databases, such as Postgres-XL and Greenplum. These solutions rely on sharding and replicating the datasets onto multiple nodes in a computer cluster. Queries are then split and routed to the proper nodes, which execute them assisted by a central coordinator. Due to the reliance on partitions and indexes, this kind of solutions are optimal for scaling out large datasets.

These kind of solutions are mostly engineered to the CRUD (Create, Retrieve, Update and Delete) paradigm they have inherited, where each operation usually involves a small subset of the total number of rows. In contrast, in the typical CosmoHub workflow, datasets are ingested and deleted always as a whole, never updated, and usually retrieved in large subsets, or as aggregations of large subsets. In addition to the critical differences in data workflow design, it was not straightforward to implement or integrate new data formats on these solutions.

Next, we tested solutions based on the Hadoop platform, such as Apache Hive (Thursoo et al., 2009) and Apache Impala (Bittorf et al., 2015). Hive is an open-source data warehouse which has gained a lot of momentum since 2013, mostly thanks to the Hortonworks²² Stinger²³ and Stinger.next²⁴ initiatives. Impala is a massively parallel processing (MPP) SQL query engine for data stored in Hadoop, and its most important contributor is Cloudera²⁵.

When evaluating these alternatives, we found that Impala timings were inconsistent, and in some cases, the results were incorrect. More up to date correctness reports have

¹⁸<https://hbase.apache.org>

¹⁹<https://cassandra.apache.org/>

²⁰<https://www.mongodb.com/>

²¹<https://redis.io>

²²<https://hortonworks.com/>

²³<https://es.hortonworks.com/blog/100x-faster-hive/>

²⁴<https://es.hortonworks.com/blog/stinger-next-enterprise-sql-hadoop-scale-apache-hive/>

²⁵<https://www.cloudera.com/>

replicated the same findings^{26 27 28}. Furthermore, the administration tools from the Cloud-era Hadoop distribution were not free, which the Hortonworks' were open source. Consequently, we decided on a solution based on Apache Hive on top of Hortonworks due to its stability, extensibility, comprehensive documentation and availability of free administration tools.

Software stack Apache Hive is one of the multiple components in the Hadoop ecosystem. Figure 3.2 displays a typical Hadoop architecture showing several of those components layered in a stack. CosmoHub heavily relies on several of these components, specially on Apache Hive, which is a data warehouse software that facilitates reading, writing, and managing large structured datasets located in distributed storage. From the administrator's point of view, migrating from the previous setup based on PostgreSQL to Apache Hive is easy as both have the same interface (SQL) with the data.

Apache Hive sits on top of Apache Tez (Saha et al., 2015), an application framework which allows the execution of complex directed-acyclic-graphs (DAG) of tasks for processing data. The scaffolding provided by Tez allows the orchestration and optimization of Hive tasks, even at runtime, boosting its performance.

The computing needs of every Hadoop platform are delivered by Yet Another Resource Negotiator (YARN) (Vavilapalli et al., 2013). YARN enables the execution of arbitrary tasks on top of containers executed in cluster nodes. Their resources are delimited and isolated, as to not interfere or starve each other. Resources are managed using scheduling queues, where each user and group can get a fair share of resources as per configuration.

The keystone of the Hadoop platform, the Hadoop Distributed File System (HDFS) (Shvachko et al., 2010) lays at the base of the stack. HDFS is a high performance distributed filesystem that makes use of the storage of the nodes in a Hadoop cluster, merging it into a single name-space for increased capacity and performance. Concisely, it works by splitting the files in fixed-size blocks and then replicating those blocks between the available nodes in the cluster. This architecture allows very good resilience and scalability.

Additional components take care of security, user authentication and authorization, as well as the administration and configuration of the different components. The market offers several distributions that include most of those components in the form of self-contained packages that facilitate an easy installation and configuration and, in addition, commercial support. After testing several alternatives, we decided on using Hortonworks Data Platform (HDP) as the software solution. The currently installed version is HDP 3.1.4.

Hardware architecture Figure 3.3 shows the hardware setup of the Hadoop platform used for CosmoHub. The current cluster is composed of 16 compute nodes and 3 head nodes. The 16 compute nodes are grouped in 4 dual-twin servers. Each node is equipped with two 14-core CPUs, 128 GiB of RAM, two 6 TB SATA disks and a single 960 GB SSD disk. The SATA drives are configured with a small RAID-1 partition to host the OS, and the rest of space is used for HDFS storage (without RAID). The SSD drive is devoted to

²⁶<https://mr3.postech.ac.kr/blog/2018/10/30/performance-evaluation-0.4/>

²⁷<https://mr3.postech.ac.kr/blog/2019/03/22/performance-evaluation-0.6/>

²⁸<https://mr3.postech.ac.kr/blog/2019/06/26/correctness-hivemr3-presto-impala/>

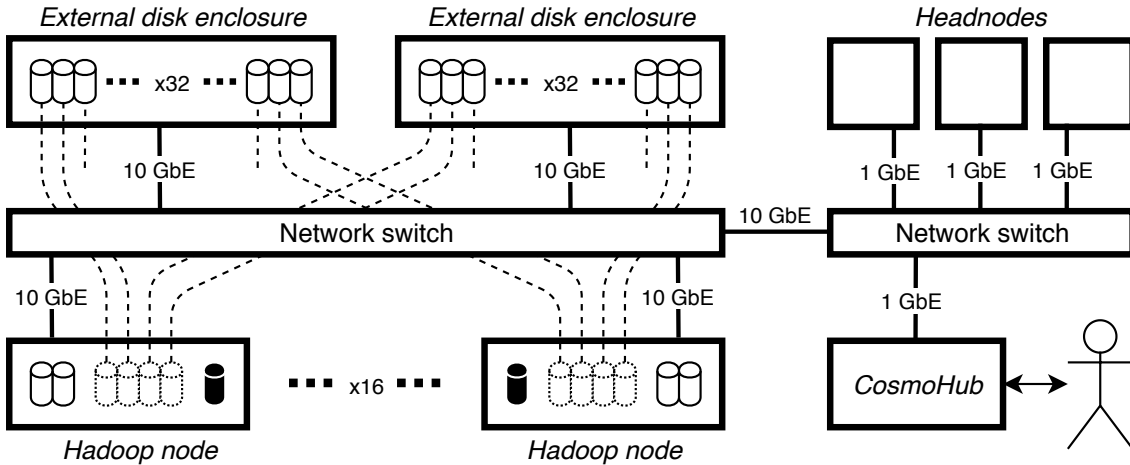


Figure 3.3: Hadoop cluster architecture, showing the configuration of the external disks and their logical links, the physically separated network for the nodes and external disk enclosures and the connection with the core network along with the headnodes and CosmoHub server.

cache intermediate files in order to boost the efficiency of shuffle operations (Kambatla & Chen, 2014), such as CosmoHub joins. The compute nodes are located at our innovative oil-submersion cooling facility, which is very power efficient (Acín et al., 2015).

Because of the dense form factor used for the compute nodes, there is not much space for expansion. Thus, in order to expand the storage capacity, we provisioned two external disk enclosures, with 36 SATA disks of 2 TB. Each compute node mounts 4 of those disks using iSCSI, two from each enclosure, accounting for 32 of the 36 drives in each enclosure. The remaining 4 drives are left as spare. The iSCSI disks appear as if they were local disks and are used for HDFS storage.

All in all, the compute nodes sum 448 cores, 2 TiB of RAM, 192 TiB of local SATA storage space and 128 TiB of external iSCSI storage space. After taking into account the CPU and RAM reserved for Hadoop services in the compute nodes, and the overhead that replicas introduce in HDFS storage, the actual resources available for processing are approximately ~ 400 cores, 1.8 TiB RAM, ~ 60 TiB of local SATA storage and ~ 40 TiB of external iSCSI storage space.

Each compute node and disk enclosure has a 10 GbE link to the same network switch. This switch is kept separated from the rest of the network infrastructure so that traffic peaks or saturation do not affect other critical services at PIC. An additional 10 GbE link connects this switch to the rest of the PIC network, where the head nodes and the CosmoHub application server are located.

The head nodes are equipped with two 8-core CPUs, 32 GiB of RAM and 2 TB of local storage. They are configured in a high-availability setup. In case of a failure, the service running in another head node is able to promote itself to master and keep the service functioning.

3.5.1 Data management

CosmoHub hosts multiple datasets of different projects, origins, cardinalities and complexities, and all of them are stored on HDFS. Most of the time, incoming raw data is published in some kind of ASCII²⁹ format, like CSV. The main problem with this kind of formats is that they are not able to describe the corresponding data types in a native way. In some cases, the data files are augmented with additional headers or they come with attached documentation of the types and meaning of the different columns in the dataset. However, alternative binary formats —such as FITS— are also common, and lately others such as Hierarchical Data Format 5 (HDF5) (Folk et al., 2011) have been gaining traction. A decisive advantage of these binary formats is that they can store the machine representation of the values and they carry along detailed metadata, including the description of data types and columns. Therefore, they are preferred in order to preserve as much information as possible when ingesting a dataset into CosmoHub.

Raw data The original upstream data, in whatever raw format is provided, is copied into HDFS and then converted into a native Hive format that is suitable for efficient query processing. Optimized Row Columnar (ORC³⁰) and Parquet³¹ are two contending formats in this area.

Features like columnar-based structure, push-down predicate (PPD) capabilities, column statistics or bloom filters are very useful for query efficiency, as they enable to skip entire sections of rows that do not contain data of interest. While both formats are very similar and support nearly the same set of features, we decided to use ORC because it is the one that is recommended by Hortonworks, the chosen Hadoop platform distribution for CosmoHub.

Although Hive already performs very well just using the features described above, we tried to reduce even more the execution times combining them with other capabilities. For instance, even though joins are not an officially supported feature, we use a combination of partitions and buckets to improve its efficiency. Hive had also limited support³² for single-table indexes that we deemed unnecessary for CosmoHub’s purposes.

Interactive exploration Queries intended for visualization purposes should finish in a delimited time to satisfy the interactivity requirement. As transferring large amounts of data to the browser for plotting is unfeasible, we use histograms and heatmaps to visualize large datasets. CosmoHub rewrites the queries that feed the histogram and heatmap plots in order to pre-aggregate the results and deliver to the browser only the data points to be plotted. This minimizes network traffic and lessens the load on the client.

Furthermore, in order to speed up even more the execution of interactive queries, two resources queues have been set up in YARN. One queue is for batch tasks such as generating custom catalogs or other non-CosmoHub related jobs, while the other is devoted solely to the execution of interactive queries. This last queue has absolute priority over the resources

²⁹<https://tools.ietf.org/html/rfc20>

³⁰<https://orc.apache.org/specification/ORCv1/>

³¹<https://parquet.apache.org/>

³²Indexes are no longer available in latest versions (v3.0 and later)

and can even preempt resources from the other queue if needed in order to ensure the fastest execution time possible.

The results (see section 3.9.1) show that, in nearly all cases, the time it takes for queries to finish is low enough so that the interactive use is feasible.

Custom catalogs Custom catalogs are user-defined subsets of other catalogs that serve a specific purpose for the requester. They are generated from an SQL statement and, due to the distributed nature of Hadoop, the output is written concurrently as a set of independent files, each one of them created by a different task. Oozie³³ is used to orchestrate the execution of the corresponding SQL query.

In previous iterations of CosmoHub, as the database engine was not distributed, the output was generated and served as a single-file download. This feature was optimal for our users as it simplified and streamlined the handling of the custom catalogs. Thus, in order to have a smooth transition, we wanted to keep the same single-file download functionality.

However, per the specific characteristics of the Hadoop platform, preserving this behaviour proved to be a tough challenge, as we had to conceive and implement a fast and simple way to merge a set of files into a single coherent download stream.

The following subsection describes in great detail the different formats supported in CosmoHub and how they were implemented to enable the single-file download feature.

3.6 Download formats

Custom catalogs are generated by Hive, following a user request. The parallel nature of Hive query execution makes this process very fast, as the data is processed and written simultaneously using multiple nodes in the Hadoop cluster. However, this also means that, at the end of this process, the resulting catalog is stored scattered among multiple files. Then, these files, *or the data contained in these files*, have to be delivered to the requesting user.

The main limitation encountered is that neither web browsers nor users like to deal with multiple files. First, web browsers do not support multiple file downloads out-of-the-box, although there are some proposals³⁴ to change this, with very limited support³⁵ yet. Currently, the most commonly found workaround is to generate and serve a ZIP³⁶ file containing the set of files to be downloaded. But this process adds a non-trivial amount of processing power per download, both when creating the ZIP file to be served, and then when it is decompressed by the user after the download has finished. Also, it doubles the storage requirements at the user side, where the ZIP file and the decompressed data must coexist, even if just for some seconds before the ZIP file can be deleted. And second, dealing with multiple files may be cumbersome, or not even possible, depending on the application or tool that is used to process the data. Moreover, even though the input data processed to generate a custom catalog may be extensive (around 100GiB - 10TiB), the

³³<https://oozie.apache.org/>

³⁴File System API: <https://fs.spec.whatwg.org/>

³⁵<https://caniuse.com/native-filesystem-api>

³⁶ZIP file format specification: <https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT>

resulting custom catalogs are usually much smaller (100MiB - 10GiB). Dealing with this amount of data in a single file is much more easier and less error prone.

Thus, we wanted to preserve the ability to deliver the custom catalogs as a single file download, as with the prior version of CosmoHub. In order to minimize the response time, it would be extremely interesting to generate this file on-the-fly while it is being sent to the user browser, avoiding materializing the combined file to disk before sending it, to be able to start the download process as soon as the query finishes.

However, not all formats are acceptable. For the CosmoHub platform, one of the key requirements is to support the most commonly used formats in cosmology and astronomy. Also, the different formats must be compatible and interoperable with the ecosystem of software and tools already in place in those fields. The corresponding source code for the formats implemented below can be consulted at <https://github.com/ptallada/cosmohub-api/tree/master/cosmohub/api/io/format>.

csv.bz2

The CSV³⁷ format, albeit standardized "de jure" as recently as 2005, it predates the appearance of personal computers for at least a decade. It can be traced back, at least, as far as 1972. Its simple definition and implementation have turned it into the most common data interchange format for structured data, with broad support among nearly all software and tools.

Accordingly, it is fundamental that CosmoHub offers CSV as one of the available download formats for custom catalogs. Hive has native support for delimited text formats and can write the results of a query in CSV format. Note that as explained previously, as queries are executed in parallel in the Hadoop cluster, the result data is stored distributed among a set of output files. Merging the output data into a single file is trivial for a format as simple as CSV. One just needs to concatenate all the individual files in sequence to generate a single file that is also a CSV format compliant.

CSV format, as it is based on textual data, is not very efficient when representing information which is not text-based. In particular, numerical data, which accounts for 99% of all the data stored in CosmoHub, has an evident storage overhead. As an example, computers just use 4 bytes when storing an integer value. The same value can take up to 11 bytes when stored in CSV, nearly 3 times more. And we still have to add the storage space needed for all the delimiters.

The simplest way to mitigate this overhead is to apply a compression algorithm to get a more compact representation of the same data. In fact, most compression algorithms work great on text data and are able to achieve high compression ratios. However, not all compression algorithms work the same. On the one hand, it must be supported by Hive. On the other hand, we shall be able to merge several compressed files into a single one as easily as possible while keeping the CSV format interoperability.

³⁷RFC 4180: <https://www.rfc-editor.org/rfc/rfc4180.html>

Hive has built-in support for two compression algorithms: GZip³⁸ (file extension `.gz`), and BZip2³⁹ (file extension `.bz2`), although more can be added, such as XZ⁴⁰, following a relatively simple process⁴¹.

From all the algorithms that were tested, only the ones aforementioned above permitted the simplest merging operation possible: concatenating a sequence of compressed files results in a single valid compressed file. In other words, given three CSV files A , B and C , and let $|$ be the concatenation operation, then:

$$\text{compress}(A) | \text{compress}(B) | \text{compress}(C) \equiv \text{compress}(A|B|C) \quad (3.1)$$

In order to choose the best algorithm we analyzed several performance and timing reports^{42,43,44}. When testing, we encountered some problems when decompressing a file containing multiple Gzip streams, so it was discarded. In the end we settled for BZip2 as it has a good balance between compression ratio and speed, is available out-of-the-box in Hadoop and has broader support in software and tools.

Figure 3.4 the process of producing a custom catalog in CSV.BZ2 format. Once the user requests the custom catalog in CosmoHub, its corresponding query is submitted to Hive. The execution is carried out by multiple nodes in the Hadoop cluster which process the input data and write their corresponding output into CSV.BZ2 files. Once the query has finished, the user is notified and the custom catalog is ready to be delivered.

When the user triggers the download process, the Python API sums the size of all the output files and sets some headers. There are two kinds of headers: a CSV header is added dynamically to the file with the information of the author, date of creation and query used to generate the catalog, and also HTTP headers to set the filename, its length (to let the browser show the download progress) and to advertise resuming capabilities. Afterwards, the Python API reads and forwards the contents of each of the individual files, one by one, to the same download stream until all the content has been delivered. The download process, if interrupted, can be resumed at any point: the Python API just needs to start reading the output files from the requested position, taking into account the space taken by the headers.

FITS

In astronomy and cosmology, one of the most popular and broadly used formats is FITS^{45,46} (Flexible Image Transport System). The term "image" in the standard's name is loosely applied and FITS files often contain only non-image data, such as tabular data. It was initially developed in the late 1970s and standardized as early as 1981. Since then, several revisions have been published, the latest is v4.0 in August 2018. FITS was designed for

³⁸<https://www.gzip.org/>

³⁹<http://sourceware.org/bzip2/>

⁴⁰<https://tukaani.org/xz/>

⁴¹<https://github.com/yongtang/hadoop-xz>

⁴²<https://www.rootusers.com/gzip-vs-bzip2-vs-xz-performance-comparison/>

⁴³<https://tukaani.org/lzma/benchmarks.html>

⁴⁴https://linuxreviews.org/Comparison_of_Compression_Algorithms

⁴⁵https://fits.gsfc.nasa.gov/fits_documentation.html

⁴⁶<https://www.loc.gov/preservation/digital/formats/fdd/fdd000317.shtml>

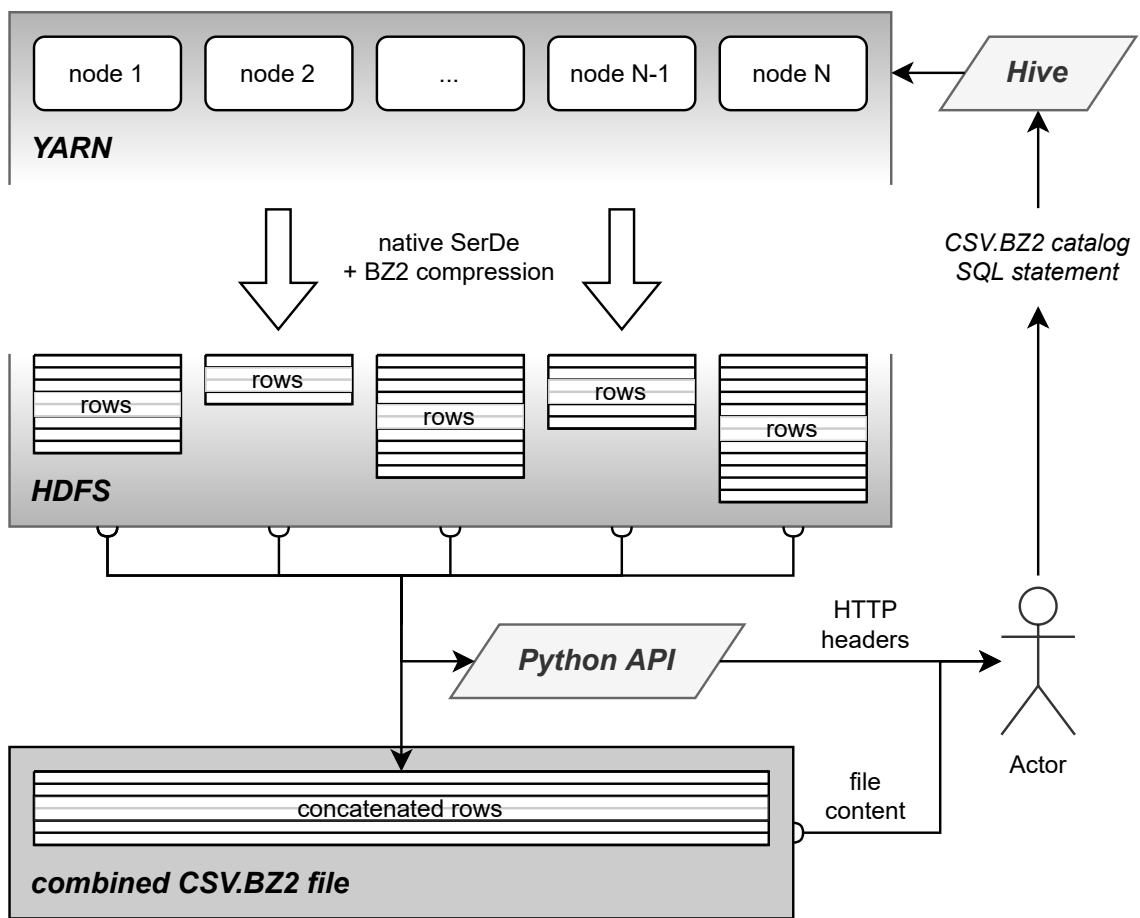


Figure 3.4: Dataflow of the generation of a single CSV.BZ2 download file.

long-term archival use, and special care is taken so that the format is backwards compatible as new features are added.

However, and due to some design decisions taken decades ago, FITS suffers some limitations (see Thomas et al., 2015) that get in the way of some modern use cases. For CosmoHub, the most important ones are:

- Only ASCII characters are supported, both for metadata (i.e. author names, file comments, column names and descriptions) and actual columns with text data. This pitfall, which comes from the historical context when it was defined, severely restricts the use of any language other than English for representing this information. The first publication of Unicode (Unicode Staff, 1991) was in 1991, 10 years later than the first FITS standard.
- The number of rows must be specified in the header. As innocuous as this design decision might appear, it has a profound impact on those use cases which rely on streaming or sequential writes. In our case, as the number of rows that will be generated by a custom catalog is not known beforehand, it is impossible to generate the corresponding header for the FITS file until the end. But by that time, and due to the way Hive and Hadoop work, the output files have already been written and cannot be modified.

Several other formats appeared, partly to overcome these limitations, such as VOTable (Ochsenbein et al., 2004), Hierarchical Data Format 5 (HDF5; Folk et al., 2011) or Advanced Scientific Data Format (ASDF; Greenfield et al., 2015).

For some time, I thought it would not be possible to support FITS as a download format in CosmoHub, or at least that its implementation would not be as optimal as the CSV.BZ2 alternative. The obvious workaround would be to generate the output files in another format, and then convert and merge them all into a single FITS file afterwards. This course of action implies an additional step that would add processing and storage overheads, as well a significant delay before being able to start the download process.

After careful thinking, we came up with a technique that would allow us to support the FITS format in CosmoHub without sacrificing any processing or storage space. These files have the following structure: a primary header, followed by a binary table header, and finally the row data. This three parts need to be padded to a size that is multiple to 2880 bytes⁴⁷.

The problem is that the number of rows must be specified in the binary table header. However, there is a particular characteristic of the FITS format that we can exploit to work around this problem. Tabular data is stored row by row in a specific binary representation, and the size of each row does not depend on any of its values, but only on the column data types.

This fixed-size row length means that the number of rows can be computed by dividing the size of the row data by the width of each row. The second value can be easily obtained by inspecting the schema returned from the catalog query. For the first, we need a way for Hive to generate the output files data using the same binary representation. Hive has

⁴⁷An obsolete requirement meant to align with the magnetic tape block size of earlier times.

specific APIs to extend its functionality, and one of those APIs tackles the ability to implement additional output formats. In particular, we had to implemented a custom SerDe⁴⁸ (Serializer/Deserializer) to define and implement how row values are to be represented and stored on disk. For the implementation of this SerDe, we used the nom.tam FITS Java library⁴⁹, which made this development much simpler. The corresponding source code of this implementation is available at <https://github.com/ptallada/recarrayserde>. With this SerDe in place, finally had support for FITS as download format in a very optimal way.

ASDF

ASDF aims to take the best qualities that made FITS so successful, such as the binary data representation for fast reading and the clear text human-editable metadata. Among its particular features it allows streaming row data (that is, without the need to specify in advance the number of rows it contains).

ASDF files are divided in several sections, the ones involved in this use case are the header (which contains the metadata) and the data sections. The schema is defined in the header and is serialized as a YAML Ain't Markup Language (YAML) (Ben-Kiki et al., 2009) document. In order to facilitate interoperability and migration from FITS, ASDF uses the exact same row binary format in the data sections. This fact made it very easy to add ASDF support in CosmoHub, as we just developed a method to write the rows using the FITS binary format. In other words, we can generate a valid ASDF file by combining the output files we already have with the corresponding ASDF header.

In order to generate the correct ADSF header we took advantage of the ASDF⁵⁰ Python library. The header can be constructed just providing the query schema (which we already need to compute the row width). With this approach, we have been able to support an additional download format with very little extra effort.

Figure 3.5 shows the data flow when generating a catalog either in FITS or ASDF format. Once the user requests the catalog, its corresponding query is submitted to Hive. The participating nodes process the input data and write their output using the custom SerDe that was developed, which writes down the rows following the FITS binary representation. Once the query finishes and the user starts the download, the Python API sums the size of all the output files and divides it by the width of each row to get the total number of rows in the custom catalog. Then, depending on the chosen format, either a FITS header or an ASDF is generated.

After this, we already have all the information to set the required HTTP headers and start forwarding the data to the user: first the header (with some padding for FITS), followed by the concatenated contents of all the output files (finished with more padding for FITS). If the download process is interrupted, just as for the CSV.BZ2 format, the process can be resumed just by reading the output files from the requested position, taking into account the size taken by the headers (and padding).

⁴⁸<https://cwiki.apache.org/confluence/display/hive/serde>

⁴⁹<http://nom-tam-fits.github.io/nom-tam-fits/>

⁵⁰<https://github.com/asdf-format/asdf>

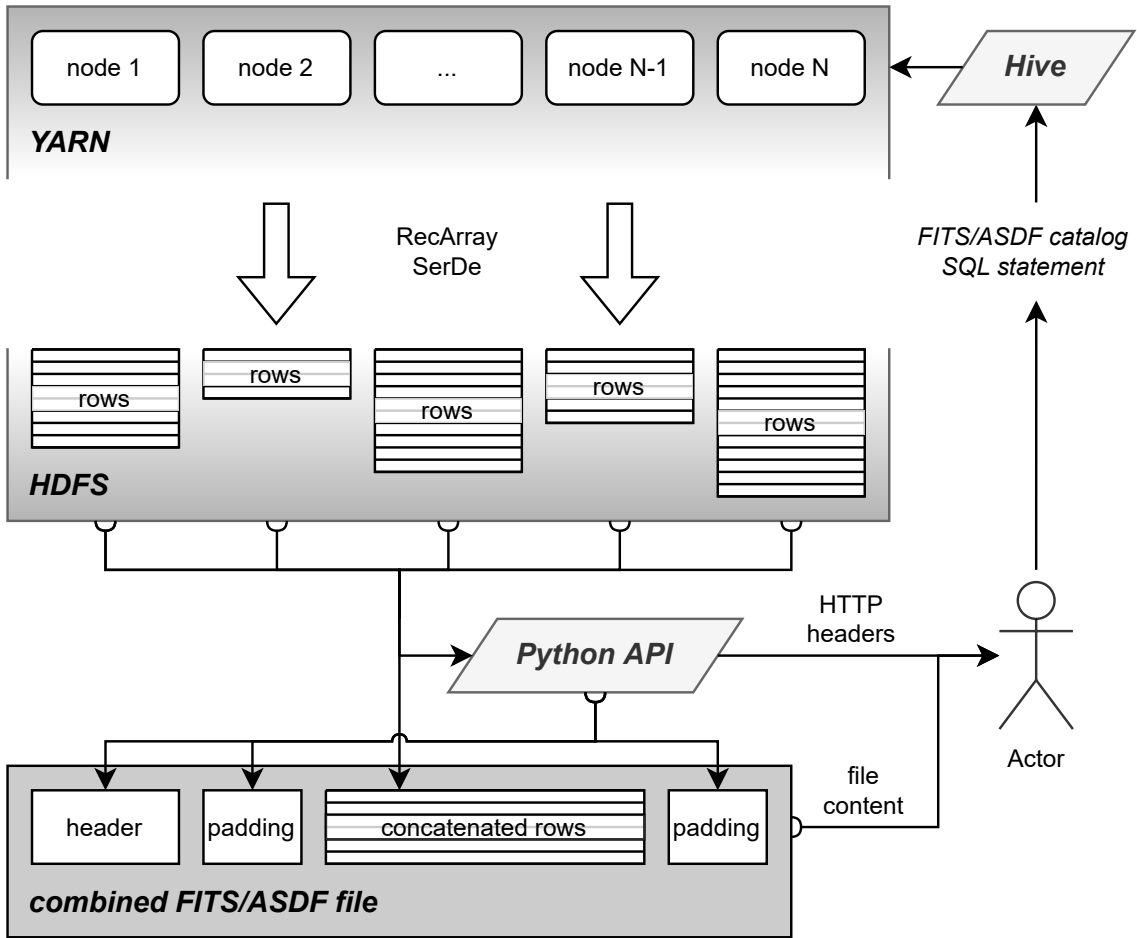


Figure 3.5: Generation of a single FITS/ASDF download file.

Parquet

Apache Parquet⁵¹ is the last download format that has been added in CosmoHub. Parquet is an open source column-oriented data storage format based on a record-shredding and assembly algorithm⁵² described in Melnik et al. (2010). As a columnar format, the values in each column are stored in contiguous memory locations, bringing multiple advantages such as efficient reads that skip unneeded column and column-wise compression optimized to each specific data type.

Parquet files can be identified by a special marker (*magic number*) at the beginning and ending. After the first marker, there are multiple *row groups* storing the row data and a footer containing all the row group and file metadata. As all the metadata is written at the end of the file, this format is perfect for use cases than involve streaming an undetermined amount of rows. Parquet uses Thrift⁵³ (Agarwal et al., 2007) and, in particular, its TCompactProtocol⁵⁴ to represent both the row data and metadata in an efficient way.

⁵¹<https://parquet.apache.org/>

⁵²<https://github.com/julienledem/redelm/wiki/The-striping-and-assembly-algorithms-from-the-Dremel-paper>

⁵³<https://thrift.apache.org/>

⁵⁴<https://github.com/apache/thrift/blob/master/doc/specs/thrift-compact-protocol.md>

Hive has native support for Parquet, so the output files of a custom catalog can be written in this format. Yet, in order to facilitate the download process, all their contents still have to be merged into a single file. As the metadata is written in the footer at the end of each file, it would seem feasible to create this single file by concatenating all the *row groups* of each file and then adding a footer with also the concatenation the metadata for all row groups. We want to avoid at all costs to parse the contents of the row groups, as that would add too much processing overhead to the download process.

After some failed attempts merging files written in Python, the Parquet developers confirmed⁵⁵ that both the row groups and the metadata contain pointers to absolute file offsets that would get misaligned if its contents are simply concatenated. However, files generated by Hive are much simpler and they only contain two problematic pointers, both in the file metadata. Therefore, for these files, it was possible to concatenate the row groups without any parsing, and then generating the file metadata with the correct pointer values.

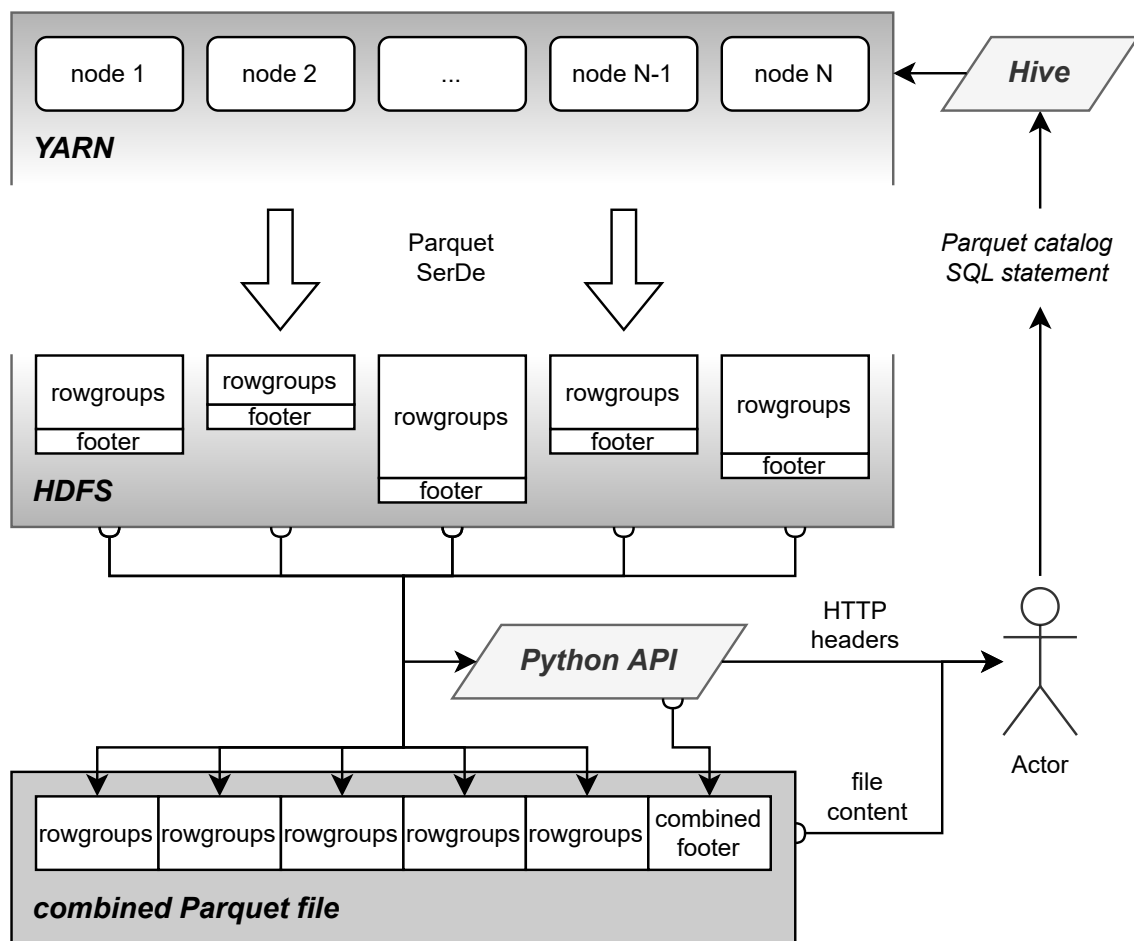


Figure 3.6: Generation of a single Parquet download file.

Figure 3.6 shows the data flow when generating a catalog in Parquet. As the catalog query is being processed by Hive, the participating nodes write the output in Parquet format. Once this process has finished, the catalog is ready to be downloaded. When the download process is triggered the Python API lists all the files, parses their footer and also gets the length of all its row groups. With this information, we are able to construct a file

⁵⁵<https://www.mail-archive.com/dev@parquet.apache.org/msg16464.html>

metadata by concatenating every file metadata while shifting each pointer by the length of all its preceding row groups.

The merged file is then constructed and delivered by concatenating the magic marker, the contents of all row groups, the merged metadata, and the final marker again. As with the other formats, several HTTP headers are used to specify the download size, file name and to enable transfer resuming. For the latter, the merged file metadata must be recreated as explained and then the contents can be read from the requested position to resume the download.

3.7 User Defined Functions (UDFs)

This section describes the use of an extension mechanism called *User Defined Functions*, introduced in section 3.3, in order to further expand the capabilities of the CosmoHub platform. These are functions developed by a user of Apache Hive to implement custom routines and procedures that can later be called directly inside any SQL statement. This is very useful to add support for additional features not included by built-in SQL functions.

Hive defines three types of *User Defined Functions*: regular User Defined Functions (UDF, from now on), are functions that work on a single row and produce also a single row as output. Most of all the functions available in Hive are of this kind (e.g. `sin`, `log`, `trim`, `current_date`). User Defined Aggregate Functions (UDAF) work on a set of rows and produces a single row as output. They should be used along the `GROUP BY` construction and are used to *summarize* the values of multiple rows (e.g. `count`, `sum`, `max`, `avg`). Finally, User Defined Tabular Functions (UDTF) work on a single row as input and return multiple rows as output. There are very few of these functions built in Hive, the most commonly used is `explode`⁵⁶⁵⁷. While regular UDFs are somewhat simple to develop, both UDAFs and UDTFs require the implementation of complex classes with several methods and require extensive testing to ensure that they work correctly inside Hive.

The following subsections describe the work done to implement several sets of these functions to extend the functionalities Hive as they cover some important uses cases for CosmoHub users. Their corresponding source code can be found at <https://github.com/ptallada/pic-hadoop-udf>.

3.7.1 HEALPix

HEALPix (Hierarchical Equal Area isoLatitude Pixelation of a sphere), described in Gorski et al. (1999), describes a set of pixelization schemes⁵⁸ to hierarchically subdivide the sphere in equal surface regions (see Figure 3.7). It was originally designed to aid in the statistical analysis of massive full-sky datasets such as the ones used for measuring the cosmic microwave background (CMB; Wandelt et al., 1998) anisotropy. Nowadays, its use cases cover

⁵⁶[https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF#LanguageManualUDF-Built-inTable-GeneratingFunctions\(UDTF\)](https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF#LanguageManualUDF-Built-inTable-GeneratingFunctions(UDTF))

⁵⁷<https://cwiki.apache.org/confluence/display/hive/languagemanual+lateralview>

⁵⁸The particular HEALPix projection that is used in astronomy is defined by the parameters $N_\theta = 3$ and $N_\phi = 4$.

the study of magnified positions due to gravitational lensing or visualization of hierarchical sky images, among many others.

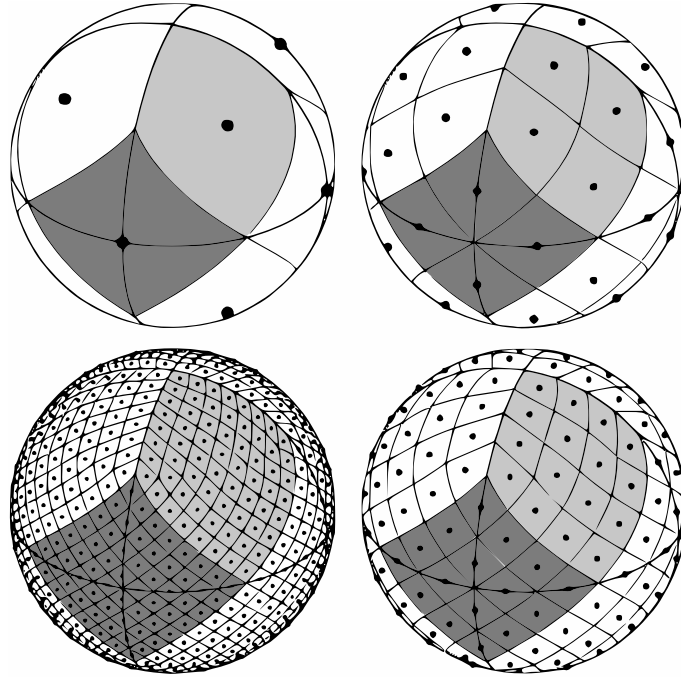


Figure 3.7: Orthographic view of HEALPix partition of the sphere, showing subsequent subdivisions using resolution parameter $N_{side} = 1, 2, 4, 8$. Image source: The HEALPix Primer (Gorski et al., 1999).

HEALPix provides two different numbering or ordering schemes (see Figure 3.8) for assigning numbers to each pixel: RING, where pixels are simply numbered counting from north to south along each latitude ring; and NESTED, where pixels are arranged into twelve tree structures corresponding to base-resolution pixels.

In order to facilitate its implementation as UDFs, we have made use of the official HEALPix Java library⁵⁹. The UDFs listed below implement a large subset of the features offered by this library. These functions take some common parameters that describe the desired resolution order (subdivision level), the ordering or numbering scheme (RING or NESTED) and the units for position coordinates (either co-latitude and longitude in radians, or longitude and latitude in degrees).

ang2pix(order, theta|ra, phi|dec, nest, lonlat)

Given the resolution order and a position in the sky returns the corresponding pixel number. If `nest` is True, return the pixel in NESTED ordering, RING otherwise. If `lonlat` is True, position is specified by longitude and latitude in degrees, otherwise colatitude and longitude in radians.

ang2vec(theta|ra, phi|dec, lonlat)

Given a position in the sky returns the corresponding 3D position vector on the unit sphere. If `lonlat` is True, position is specified by longitude and latitude in degrees, otherwise colatitude and longitude in radians.

⁵⁹<https://healpix.sourceforge.io/>

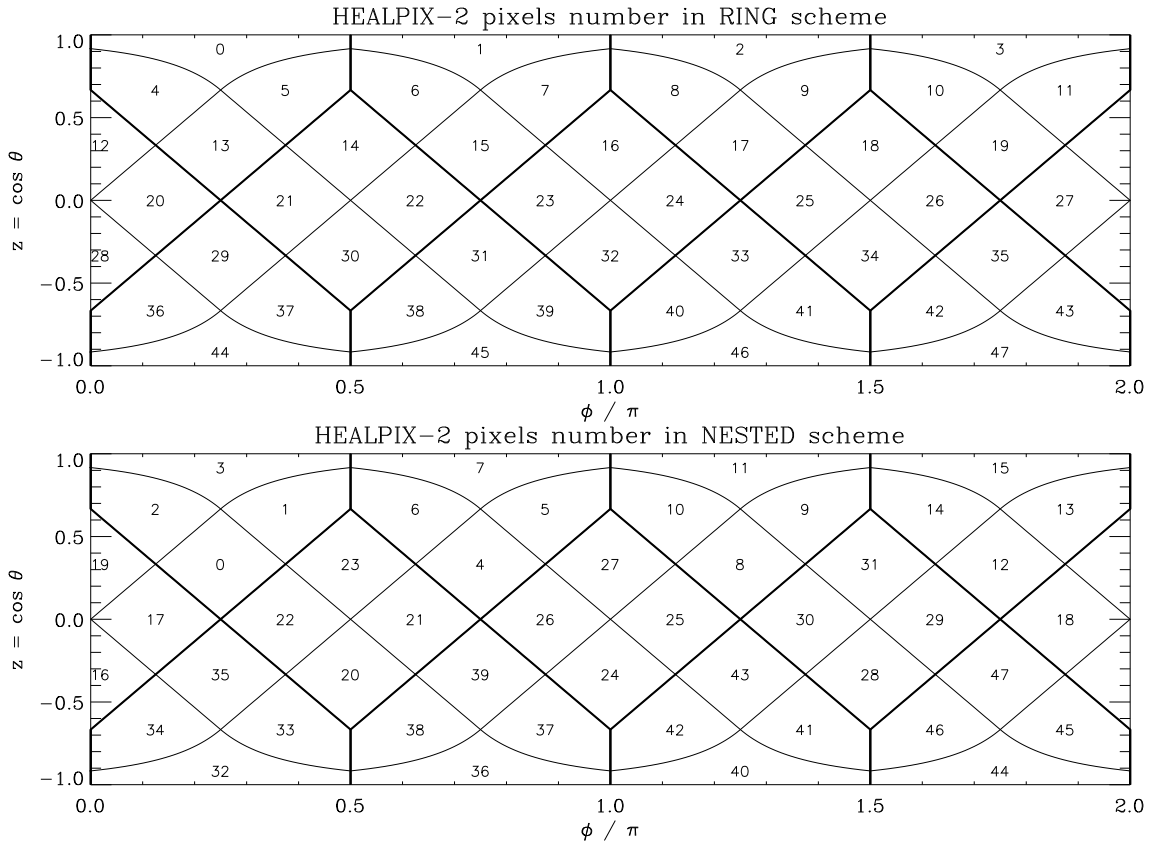


Figure 3.8: Cylindrical projection of the HEALPix division of a sphere using both pixel numbering schemes (RING and NESTED). Image source: The HEALPix Primer (Gorski et al., 1999).

pix2ang(order, ipix, nest, lonlat)

Given a resolution order and a pixel number returns the corresponding position in the sky. If `nest` is True, the pixel index is specified in NESTED ordering, RING otherwise. If `lonlat` is True, position is returned as longitude and latitude in degrees, otherwise colatitude and longitude in radians.

pix2vec(order, ipix, nest)

Given a resolution order and a pixel number returns the corresponding 3D position vector on the unit sphere. If `nest` is True, the pixel index is specified in NESTED ordering, RING otherwise.

vec2ang(vector, lonlat)

Given a 3D position vector returns the corresponding position in the sky. If `lonlat` is True, position is returned as longitude and latitude in degrees, otherwise colatitude and longitude in radians.

vec2pix(order, x, y, z, nest)

Given a resolution order and the 3D vector components of a position returns the pixel number. If `nest` is True, the pixel index is specified in NESTED ordering, RING otherwise.

neighbours(order, theta|ra|ipix, phi|dec, nest, lonlat)

Given a resolution order and either a position or a pixel number returns the pixel

number of the SW, W, NW, N, NE, E, SE and S neighbours. The output is always an array of 8 elements, some of them may be -1 is that pixel does not have a neighbour in that direction. If `nest` is True, the pixel index is specified in NESTED ordering, RING otherwise. If `lonlat` is True, position is specified by longitude and latitude in degrees, otherwise colatitude and longitude in radians.

nest2ring(order, ipix)

Given a resolution order and a pixel, converts it from NESTED ordering to RING ordering.

ring2nest(order, ipix)

Given a resolution order and a pixel, converts it from RING ordering to NESTED ordering.

nside2npix(nside)

Return the number of pixels for a given NSIDE.

npix2nside(npix)

Return the NSIDE for the given number of pixels.

nside2order(nside)

Return the resolution order for the given NSIDE.

order2npix(order)

Return the number of pixels for the given resolution order.

maxpixrad(order, degrees)

Given a resolution order returns the maximum angular distance between any pixel center and its corners. If `degrees` is True, the angular distance is returned in degrees, otherwise in radians.

3.7.2 Arrays

Another common way of storing structured data is in form of arrays, such as probability density functions (PDF). In particular in Astronomy, galaxy catalogs may store the spectral energy distribution (SED) or the redshift PDF of their objects. An interesting scientific use case would be to enable the aggregation of this kind of array data in order to be accessed and analyzed as a single entity. For instance, in Asorey et al., 2016 they state that using the entire redshift PDF instead of just the peak or the mean value delivers results with less bias. For those kind of studies, having the ability to easily aggregate the data using a single query would help speed up the analysis.

Unfortunately, Hive includes very few functions to deal with array data, and none of them supports this use case. Therefore, we developed the following list of UDAFs that operate element-wise on arrays that have matching element types and cardinalities (see Figure 3.9 for more details).

array_min(data[])

Returns the lowest value for each position in the array.

array_max(data[])

Returns the highest value for each position in the array.

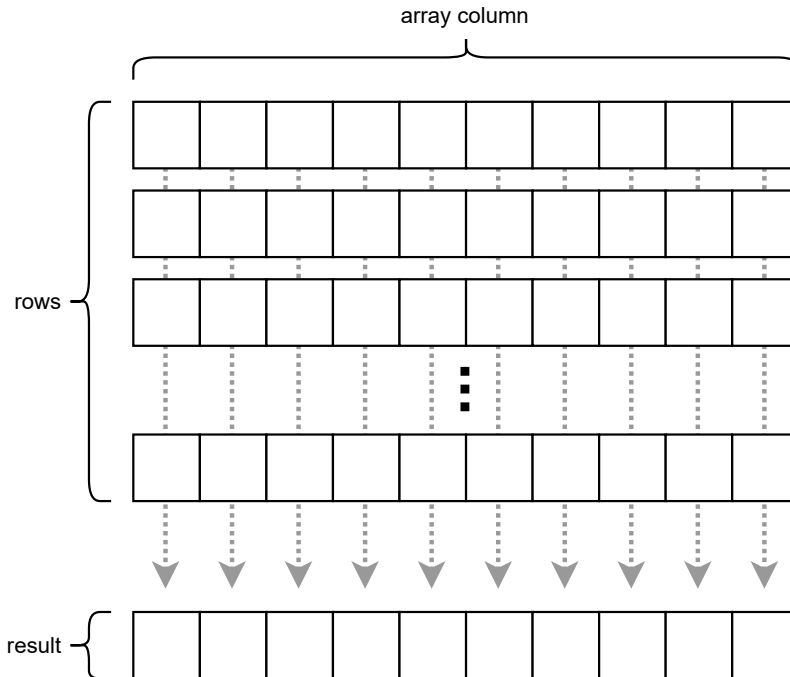


Figure 3.9: UDAF array functions generate their result by aggregating all values in each position. The resulting array has the same cardinality as those from input.

array_sum(data[])

Returns the sum of all the values for each position in the array.

array_count(data[])

Returns the number of elements different than NULL for each position in the array.

array_avg(data[])

Returns the average of all the values for each position in the array.

array_stddev_pop(data[])

Returns the population standard deviation for each position in the array.

array_stddev_samp(data[])

Returns the sample standard deviation for each position in the array.

array_variance(data[]) / array_var_pop(data[])

Returns the population variance for each position in the array.

array_var_samp(data[])

Returns the sample variance for each position in the array.

3.7.3 ADQL

This subsection describes the work done to add support to run Astronomical Data Query Language⁶⁰ (ADQL) statements in CosmoHub.

⁶⁰<https://www.ivoa.net/documents/ADQL/20180112/PR-ADQL-2.1-20180112.html>

The Virtual Observatory⁶¹ (VO) describes the concept of accessing astronomical datasets distributed worldwide in a simple and transparent way. The VO provides scientific projects and data centers with a well-defined framework to publishing and delivering services using their data, and enables users to explore, analyze and visualize all these data using common tools and software.

The International Virtual Observatory Alliance⁶² (IVOA) is an organization that coordinates and provides the collaborative environment needed to develop and agree on the technical standards required to make the VO possible. The IVOA has defined multiple standards, but for this appendix, we are interested in just two: Table Access Protocol⁶³ (TAP) and Astronomical Data Query Language (ADQL).

TAP defines a protocol for interacting with tabular data, such as astronomical catalogs. It enables access to the actual table data, as well as associated metadata for the corresponding table and database. TAP service providers must support queries written in ADQL, a special flavour of SQL specifically designed to work with astronomical data, although they can add other alternatives.

ADQL is loosely based on SQL-92⁶⁴ with some minor differences in both syntax and basic arithmetic and trigonometric functions. HiveQL, the specific SQL flavour of Hive, is also based on SQL-92 and already supports most those features, albeit some of them use a slightly different syntax. For these cases we are using a technique called *transpiling*⁶⁵ to rewrite ADQL statements into HiveQL specific syntax. In order to facilitate this, we have tested and extended a Python SQL parser, transpiler and optimizer named `sqlglot`⁶⁶.

ADQL also defines a set of geometric data types and functions to enable storing and processing different kinds of geometries. In particular, it defines the following types: `POINT`, a single sky coordinate; `CIRCLE`, defined by a sky coordinate and a radius in degrees; `POLYGON`, defined by a sequence of consecutive vertices; and `REGION`, an arbitrarily complex geometric region that is expressed as an STC-S⁶⁷ string. ADQL then defines a set of functions to construct these data types and to operate with them.

The following subsections describe the features that are missing in Hive to completely support ADQL and our approach at solving them.

Syntax differences

SELECT TOP N

This archaic syntax comes from older versions of SQL Server and it is not supported any more in actual databases. The goal was to transpile these constructions, but `sqlglot` was unable to parse. After submitting a pull request⁶⁸ to add support for it, we can now rewrite the queries to use the standard syntax:

```
SELECT ... LIMIT N
```

⁶¹<https://ivoa.net/about/what-is-vo.html>

⁶²<https://www.ivoa.net/>

⁶³<https://www.ivoa.net/documents/TAP/>

⁶⁴<http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt>

⁶⁵https://en.wikipedia.org/wiki/Source-to-source_compiler

⁶⁶<https://github.com/tobymao/sqlglot>

⁶⁷<https://www.ivoa.net/documents/STC-S/>

⁶⁸<https://github.com/tobymao/sqlglot/pull/177>

FROM A JOIN B USING C

This construction is not compatible with HiveQL but it can be transpiled to an equivalent form. Support for this syntax was added in sqlglot per request⁶⁹ and the queries are rewritten as follows:

```
FROM A JOIN B ON A.C = B.C
```

FROM A NATURAL JOIN B

EXCEPT [ALL]

INTERSECT [ALL]

These three operators are not compatible with Hive. However, there are equivalent constructions for all of them that deliver the same results using a compatible syntax. Unfortunately, these cases cannot be transpiled automatically as they require additional additional schema information not present in the query. Also, adding native support for them in Hive would be very complex and out of the scope of this thesis. For this, we have decided to compromise on the ADQL specification and not support this constructions.

Standard functions

log(x)

ADQL defines this function to compute the natural logarithm of any number. We transpile this function to the equivalent in Hive, named LN(x), as LOG(x, n) is reserved to compute logarithms of arbitrary bases.

atan2(y, x)

This function was one of the first we implemented as a UDF, years ago. However, UDFs are not defined globally⁷⁰ and to call them must be prefixed by the schema in which they have been defined. To improve this, we have compiled a custom version of Hive where we have included this function, and others, as built-ins so they can be called directly as ADQL requires.

truncate(x, n)

This function is used to truncate the input to the specific number of decimal places. However, ADQL standard specifies *double* as the return type, and it is unclear if an implementation following this specification to the letter would have problems with irrepresentable floating point numbers. In our case, we have decided to implement this feature through a UDF using the *Decimal* data type which does not suffer from representation issues.

⁶⁹<https://github.com/tobymao/sqlglot/issues/178>

⁷⁰Work ongoing in <https://issues.apache.org/jira/browse/HIVE-17986>

Field	Type	Example
tag	tinyint	POINT(0), CIRCLE(1), POLYGON(2), REGION(3)
coords	array<double>	[12.4, 76.3, 84.9, 34.1, 37.4, 72.3]
rs	array<byte>	0x991fdebdb1b7b6b2c2bdec41d30475a4...

Table 3.1: Description of the custom struct data type to store geometric data.

Geometric types

ADQL defines several data types to store and operate with specific geometries. Initially, we implemented them in Hive using a custom `union` data type that could store the corresponding parameters needed to represent each geometry. However, as the support for `union` data types in Hive is incomplete, we had to revise the implementation and refactor it based on a custom `struct` data type (see Table 3.1 for its structure and implementation details). Here follows the specific geometries defined by ADQL:

- A `POINT` is defined by a set of two floating point values representing its coordinates in degrees: right ascension and declination.
- A `CIRCLE` is defined by a set of three floating point values representing the coordinates of its center (right ascension and declination) and its radius, all of them in degrees.
- A `POLYGON` is defined by three or more pairs of floating point coordinates defining the sequence of vertices of the polygon. However, one also need the order or orientation⁷¹ on the sphere when traversing those vertices to be able reconstruct the proper shape. Most libraries dealing with spherical geometry use the CCW (counter-clockwise) convention, in which the interior of the polygon lays *on the left side* along the sequence of vertices. ADQL specification does not mention any kind of ordering, which is a grave omission, so we decided to go along the broad consensus of the industry.
- A `REGION` can represent an arbitrary region on the sky. ADQL defines it as a complex geometric shape described by an STC-S⁷² string, but is unclear whether its support is mandatory. Moreover, implementing support for STC-S is very complex, as there is hardly any support at all in external libraries, and it also has not been designed for efficient operations.

In order to make its implementation viable and its operations efficient, I decided to base the design of the `REGION` geometry as a Multi-Order Coverage (MOC⁷³) map. In this case, the geometry is represented by an opaque byte array. More details about its implementation are given later.

Geometric functions

ADQL defines the following operations involving the geometries just described.

⁷¹<https://gis.stackexchange.com/questions/119150/order-of-polygon-vertices-in-general-gis-clockwise-or-counterclockwise>

⁷²<https://ivoa.net/documents/STC-S/>

⁷³<https://healpix.sourceforge.io/doc/html/java/healpix/essentials/Moc.html>

area(geom)

Returns the area of the specified geometry in square degrees.

box(point, width, height) / box(ra, dec, width, height)

Returns a POLYGON geometry that represents a box on the sky defined by the coordinates of its center, width and height, in degrees. The edges are segments of great circles parallel to the coordinate axes at the center position.

centroid(geom)

Returns the centroid or *center of mass* of the provided geometry as a POINT.

circle(point, radius) / circle(ra, dec, radius)

Returns a CIRCLE geometry defined by its center coordinates and radius, in degrees.

contains(geom, geom)

Returns 1 whether the first geometry parameter is wholly contained within the second one, 0 otherwise.

coord1(point)

Given a POINT, returns its first coordinate value (right ascension).

coord2(point)

Given a POINT, returns its second coordinate value (declination).

coordsys(geom)

This function has not been implemented as it has been marked as deprecated⁷⁴ in the latest ADQL specification.

distance(point, point) / distance(ra1, dec1, ra2, dec2)

Returns the angular distance between two sky positions, in degrees.

intersects(geom, geom)

Returns 1 whether the specified geometries overlap, 0 otherwise.

point(ra, dec) / point(hpix_29_nest)

Returns a POINT geometry defined by its coordinates in degrees, or from a HEALPix pixel index of order 29 (the smallest one). The second variant is an extension to the ADQL specification that we implemented to enable dealing with positions represented using HEALPix pixels.

polygon(ra1, dec1, ra2, dec2, ra3, dec3, ...)**polygon(point1, point2, point3, ...)**

Returns a POLYGON geometry defined from the sequence of its vertices (in CCW order).

Region extension

As described before, we decided to support the REGION geometry by implementing it as a MOC map. This special kind of HEALPix map allows defining footprints combining pixels of different sizes. One particular property of the NESTED ordering is that it allows

⁷⁴https://www.ivoa.net/documents/ADQL/20180112/PR-ADQL-2.1-20180112.html#tth_sEc4.2.4

describing any pixel, of any size, by the integer range that encloses all the subdividing pixels in the HEALPix hierarchy. As a result, any combination of arbitrarily-sized pixels that form a MOC can be represented by a simple set of corresponding integer ranges.

HEALPix Java library has a specific class called `RangeSet`⁷⁵ that allows very efficient operations when dealing with this data structure, as well as a very optimal interpolative-coding compression⁷⁶ that saves up a lot of storage. The resulting binary array is what gets stored in the custom geometry data type. Based on this special properties, we have implemented the following additional functions to complement the features of CosmoHub when dealing with `REGION` data types:

region(geom)

Converts the provided geometry into a `REGION`.

complement(geom)

Returns the complement of the provided geometry.

intersection(region)

This is an aggregate function that, given a column of regions, it returns the overall intersection of them.

union(region)

This is an aggregate function that, given a column of regions, it returns the overall union of them.

3.8 Web application

This section describes the web framework of CosmoHub, composed of two components. On the one hand the server backend that powers the operation of the service for which I conducted its entire design and development. On the other hand the frontend that exposes all of CosmoHub's functionalities to the user which was developed in collaboration with PIC colleagues and in which I was involved in the design and testing phase.

Python backend

CosmoHub allows users to process large datasets to obtain additional information. In order to decouple the results of those interactions from the way they are presented, all operations are carried out through a set of API calls. In particular, all actions available to the user are implemented in an API that follows the REST (Fielding, 2000) paradigm. Operations are grouped in several endpoints depending on the data model entity they act on (see Figure 3.1). Database access is proxied through an Object Relational Mapper (ORM) layer in order not to tie the implementation of each action to the data model specifics, enabling data model evolution. All actions return JSON⁷⁷ responses, which are consumed by the web frontend. A complete list of endpoints and their description is available in Table 3.2. The full source code for the Python backend is available at <https://github.com/ptallada/cosmohub-api>.

⁷⁵<https://healpix.sourceforge.io/doc/html/java/healpix/essentials/RangeSet.html>

⁷⁶[https://healpix.sourceforge.io/doc/html/java/healpix/essentials/Moc.html#toCompressed\(\)](https://healpix.sourceforge.io/doc/html/java/healpix/essentials/Moc.html#toCompressed())

⁷⁷<https://tools.ietf.org/html/rfc7159>

URL	Method	Description
/user	GET	Retrieve current user profile
	PATCH	Update profile data (i.e. email, password)
	POST	Register a new user
	DELETE	Remove a user account
/groups	GET	Retrieve the list of groups
/acls	GET	Retrieve the list of users and their memberships
	PATCH	Modify a user's membership
/catalogs	GET	Retrieve the list of catalogs accessible to the current user
/catalogs/{id}	GET	Retrieve detailed information of a catalog
/catalogs/syntax	GET	Perform an SQL syntax check
/downloads/datasets/{id}/readme	GET	Download the <i>README</i> file for a dataset
/downloads/files/{id}/readme	GET	Download the <i>README</i> file for a value-added file
/downloads/files/{id}/contents	GET	Download the contents of a value-added file
/downloads/queries/{id}/results	GET	Download the output of a custom catalog
/queries	GET	Retrieve the list of custom catalogs for the current user
	POST	Request the generation of a custom catalog
/queries/{id}/cancel	POST	Abort the generation of a custom catalog
/queries/{id}/done	GET	Callback to notify the completion of a custom catalog
/contact	POST	Send a message to the CosmoHub Team

Table 3.2: List of REST API endpoints, grouped by entity. For each one, its URL pattern, the HTTP method and a brief description is shown.

Most catalogs in CosmoHub belong to a single project although, in some special cases, they can be associated with several projects. Only users which are members of those projects are able to access their corresponding data. In order to prevent unauthorized uses of CosmoHub, all requests are authenticated and the user privileges are checked against the database. The API accepts two authentication methods, basic and token.

With HTTP Basic authentication⁷⁸, each request must include a username and password combination. This information is looked up in the user database and, if no match is found, the request is denied. The main inconvenience with this mechanism is that each request requires a round-trip to the database. In order to soften the load on the database, a JSON Web Token⁷⁹ (JWT) is attached to every response. This token contains signed information about the authenticated user and, when supplied on future requests, it allows the backend to verify the identity of the user without any database involvement.

Regarding the interactive exploration feature of CosmoHub, as Hive queries are potentially executed on all nodes in the Hadoop platform, a full table scan usually takes about a minute. Combined with sampling, results can be obtained even faster. This performance allowed us to implement interactive exploration of large datasets using histograms and heatmaps. As also mentioned in 3.5.1, data for these plots is pre-aggregated on the Hive side using a specially constructed query and only the data points to be plotted are sent to the browser.

Additionally, in order to provide some feedback to the user during the execution of interactive queries, an extension⁸⁰ was developed for the Python DB-API interface for Hive (PyHive) in order to extract the progress of an ongoing query. This information is relayed using a websocket⁸¹ connection, which enables bidirectional communication between the browser and the backend. Through this channel, users receive periodic progress updates about a query and can also request its cancellation.

CosmoHub is deployed in three different instances corresponding to the production, pre-production and test environments. Each one of them runs on separate identical virtual machines, with 4 cores, 4 GiB of RAM and 10 GiB of storage each. Only the production instance is accessible to the outside through <https://cosmohub.pic.es>.

The main software components used for building the backend stack are *Flask*⁸² as the Python Web Server Gateway Interface (WSGI⁸³) framework, *Flask-RESTful*⁸⁴ as the REST framework and *gevent*⁸⁵ as the coroutine networking library. This stack runs on top of uWSGI⁸⁶ behind an *NGINX*⁸⁷ proxy.

⁷⁸<https://tools.ietf.org/html/rfc7617>

⁷⁹<https://tools.ietf.org/html/rfc7519>

⁸⁰<https://github.com/dropbox/PyHive/pull/136>

⁸¹<https://tools.ietf.org/html/rfc6455>

⁸²<http://flask.pocoo.org/>

⁸³<https://www.python.org/dev/peps/pep-3333/>

⁸⁴<https://flask-restful.readthedocs.io>

⁸⁵<http://www.gevent.org/>

⁸⁶<https://uwsgi-docs.readthedocs.io>

⁸⁷<https://www.nginx.com/>

For the data access layer, *SQLAlchemy*⁸⁸ is used as the ORM component and the combination of *psycopg*⁸⁹ and *psycogreen*⁹⁰ are used as the PostgreSQL driver and coroutine adapter library, respectively. Finally, *astropy*⁹¹ is used to implement FITS as a download format and the ASDF⁹² python library to implement ASDF format.

Web frontend

The web interface’s main objective is to enable the user to access all of CosmoHub’s features which, as described in the previous section, are available through a set of REST endpoints.

Usability is a strong requirement, as the interface should be intuitive enough so that any user can interact with it with no prior training. We took special care to follow and exploit characteristic web semantics —such as forms, hyperlinks or scrolling, among others— to aid the user at every step. In the end, designing a clean and simple interface, preferably self-explanatory, is key to permitting open science.

CosmoHub’s frontend has been developed using modern, widely-supported, community technologies, such as AngularJS⁹³, Bootstrap⁹⁴, WebSockets⁹⁵, Plot.ly⁹⁶ and Wordpress⁹⁷.

AngularJS is a Javascript framework, developed and maintained by Google⁹⁸, that extends HTML to implement Model-View-Controller capabilities into web browsers, making user interactions dynamic, faster and more fluid. It has a large collection of official and third-party plugins and is specially designed to interact with API based applications, such as CosmoHub. Being open-source, well maintained and with a broad community of users and developers are key aspects for choosing it as the base of the frontend.

Bootstrap is an open-source HTML, JavaScript and Cascading Style Sheets (CSS) library created and maintained by Twitter⁹⁹ for responsive web design. It is one of the most used styling frameworks in the entire web development community, providing users with clear and coherent interfaces. It comes with a handy and easy to configure column-based layout, plus some predefined style elements. These features are extensively used in CosmoHub.

WebSockets is a technology for bi-directional communication between web browsers and web servers. This technique involves *upgrading* a stateless HTTP request into a persistent TCP connection that can be subsequently used to transfer information from both parties. Some features such as real-time progress monitoring require websockets to work properly.

⁸⁸<https://www.sqlalchemy.org/>

⁸⁹<http://initd.org/psycopg/>

⁹⁰<https://bitbucket.org/dvarrazzo/psycogreen>

⁹¹<https://www.astropy.org/>

⁹²<https://asdf-standard.readthedocs.io>

⁹³<https://angularjs.org/>

⁹⁴<https://getbootstrap.com/>

⁹⁵<https://tools.ietf.org/html/rfc6455>

⁹⁶<https://plot.ly/>

⁹⁷<https://wordpress.com/>

⁹⁸<https://google.com>

⁹⁹<https://twitter.com>

Plot.ly is an open-source plotting library based on the widely used D3.js¹⁰⁰ web visualization framework. It greatly simplifies the programming needed to implement all sorts of charts and dashboards, such as those used for interactive exploration.

Finally, WordPress is one of the most used content management systems (CMS). Although it is mostly associated with blogs, CosmoHub uses it as a backend for editing the content of dynamic sections, such as the news feed.

When a user visits CosmoHub (<https://cosmohub.pic.es>), it is presented with the initial page shown in Figure 3.10. This front page describes its goals, showcases its main features and holds references to the rest of public contents.

User management Authentication is required in order to access CosmoHub's main features. For this, users have to enter their credentials in the login form. Public catalogs can be accessed by any user. However, access to private data from additional projects has to be manually validated by project administrators. All personal data from registered users is stored and processed following GDPR¹⁰¹ regulations.

Interactive exploration Just after logging in, users are presented with the catalog selection table. The headers on the top of each column allow to sort the list for each field, while the top search box allows user to restrict the listing to only those entries containing the specified words in their name or description.

Selecting any entry brings them to the catalog page (Figures 3.11 and 3.12), where they can build their own custom catalogs or subsets for interactive exploration and/or download. In order to guide them, the subset construction process is divided in a series of steps, which can be traversed through scrolling or using the navigation bar fixed at the top.

The first piece of information a user encounters is a complete description of the catalog, usually provided by the catalog owner. Just below, an optional section called "Value Added Data" contains links and documentation to additional data that complements this catalog and that may be useful to analyze it, such as filter curves or extinction maps.


Users can use a predefined dataset (Step 0) or create a new one from scratch (Step 1). Predefined datasets are curated options with specific purposes, although users can modify them to suit their needs. There are two types of predefined datasets: *basic* datasets use the guided interface to configure the subset, while *expert* datasets resort to setting up directly the SQL statement in expert mode. In order to build a custom catalog from scratch, users start choosing the set of columns they need from Step 1. The search box on the top right allows users to filter the columns display looking for partial matches on names and comments.

Steps 2 and 3 represent two methods available to users for restricting the number of rows contained in their custom catalog. On the one hand, with row sampling (Step 2), Hive can be configured to only read a fraction of the files that store the catalog's data. In an attempt to deliver statistically unbiased subsets, rows are divided in those files not

¹⁰⁰<https://d3js.org/>


¹⁰¹<https://gdpr-info.eu/>

[Login / Sign Up](#)




Build your own Universe


Interactive data analysis of massive cosmological data without any SQL knowledge




Billions of observed and simulated galaxies



Superfast queries means superfast results







Features to make you work faster and easier



Online plotting preview and data download

[Learn more](#)

CosmoHub currently provides support to four international cosmological projects





Contact us to upload your data and be able to easily distribute and analyse it

Super fast queries means super fast results

CosmoHub portal uses the Apache Hive infrastructure, which provides a very fast data query and is built on top of Hadoop.

Filter a catalog with more than half a billion galaxies by simple or complex conditions to generate a subset in minutes.

CosmoHub is developed and maintained by the following research groups

Features

- ✓ Observed and simulated cosmological data
- ✓ Value-added data ready to download
- ✓ Load prebuilt datasets
- ✓ Create your own custom catalogs through a guided process without any knowledge of SQL language
- ✓ Easily filter and sample the catalog
- ✓ Amazingly fast online analysis of the data
- ✓ Different charts: scatter, 1D-histogram, contour or heatmap
- ✓ Three different file formats to download the selected sample: csv, fits and asdf
- ✓ Download the selected sample into your computer

Contact us!

If you want to know more about the project

If you want to become an end user

If you want to add your datasets to the platform

Or if you just want to say hi!

Email address

Subject

Message

[About](#) - [News](#) - [Legal](#) - [Twitter](#)

Figure 3.10: Initial page, showcasing the main projects and features. Note that some content has been edited for presentation purposes.

MICECAT 1 Datasets Columns Sampling Filters Query Analysis Format Request

This catalog is a release (v0.4_r1.4) of the parent MICE-Grand Challenge Galaxy and Halo Light-cone Catalog.

The catalog was generated using a hybrid Halo Occupation Distribution (**HOD**) and Halo Abundance Matching (**HAM**) prescriptions to populate Friends of Friends (FOF) dark matter halos from the MICE-GC simulation.

This catalog used as input the light-cone of the MICE-GC N-body run simulation.

The input cosmological parameters are $\Omega_m=0.25$, $\sigma_8 = 0.8$, $n_s=0.95$, $\Omega_b=0.044$, $\Omega_c=0.75$, $h=0.7$. Further details on the simulation can be found at the [MICE web page](#)

The catalog was built to follow local observational constraints:

- The luminosity function (Blanton et al. (2003))
- The galaxy clustering as a function of luminosity and colour (Zehavi et al. 2011)
- The color-magnitude diagram (NYU dr7 catalog)

Value Added Data *Directly download useful or necessary files to analyse the catalog*

Name	Version	Description	Size	Download
SEDS_EXTINCTION_LAWS	1.0	Cosmos SEDs, Extinction laws and reference filter	494.57 KB	Download Readme

Catalog Playground *Create and analyze your own sample of the catalog following some basic steps*

Step 0: Datasets · Load a particular sample of the catalog ?

You can find below particular sets of the catalog.
If you click in the 'Load' button you will jump to the Analysis Step 5.

Name	Version	Type	Description	Rows	Load
Clustering demo 2	1.0	Basic	Volume limited sample for a luminosity bin	492,210	Load Readme
Clustering demo 3	1.0	Expert	Volume limited sample for a luminosity bin and a color cut	505,528	Load Readme

Step 1: Columns · Select the fields you need ?

id object id
 ra right ascension (degree)
 dec declination (degree)
 ra_mag magnified right ascension (degree)
 dec_mag magnified declination (degree)
 z true redshift
 d_c comoving distance (Mpc/h)
 z_v observed redshift (including peculiar velocity)
 d_c_v comoving distance corresponding to z_v (Mpc/h)

Step 2: Sampling · Select a subset and get faster results ?

Size: 1/256 1/1 ~ 205.22 M rows

Seed: 1 Random

Step 3: Filters · Add conditions to refine your search ?

z > 0.4 ✕
z < 0.6 ✕

Figure 3.11: Catalog page upper half, showing catalog's description, valued added data and steps 0 to 3.

Step 4: Query · Review ?

```
SELECT 'abs_mag_r', 'gr_restframe'
FROM m1cecatv1_0_hpix
WHERE 'z' > 0.9 AND 'z' < 1.1
```

Step 5: Analysis · Explore the selected data ?

<input checked="" type="button" value="Table"/>	Scatter	Histogram	Heatmap
---	---------	-----------	---------

X axis: <input type="text" value="abs_mag_r"/>	X min: <input type="text" value="-23,2900009155273"/>	X max: <input type="text" value="-18,8999996185303"/>	Bins: <input type="text" value="100"/>
Y axis: <input type="text" value="gr_restframe"/>	Y min: <input type="text" value="-0,189999997615814"/>	Y max: <input type="text" value="1,72000002861023"/>	Bins: <input type="text" value="100"/>

Function:

Step 6: Format · Select a file type ?

CSV.BZ2	Bzip2 compressed Comma-separated values file (please check Help #4 if using Pandas DataFrame)
FITS	The Flexible Image Transport System is the most commonly used digital file format in astronomy
ASDF	The Advanced Scientific Data Format pretends to be the successor for the immensely successful FITS format

Step 7: Request · Review citation guides ?

How to cite CosmoHub

If you have used in your work any plots or data produced through CosmoHub please include in the Acknowledgments section the following snippet:

This work has made use of CosmoHub.

CosmoHub has been developed by the Port d'Informació Científica (PIC), maintained through a collaboration of the Institut de Física d'Altes Energies (IFAE) and the Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), and was partially funded by the "Plan Estatal de Investigación Científica y Técnica y de Innovación" program of the Spanish government.

In addition, please cite the following proceedings:

Carretero et al. 2018 *CosmoHub and SciPIC: Massive cosmological data analysis, distribution and generation using a Big Data platform* | [PDF](#) - [BibTeX](#)

How to cite this catalog

If you make use of the MICE mocks for scientific publications, we kindly ask you to cite the following papers:

"The MICE grand challenge lightcone simulation - I. Dark matter clustering". Fosalba, P.; Crocce, M.; Gaztañaga, E.; Castander, F. J., *MNRAS*, 448, 2987 (2015)

"The MICE Grand Challenge Lightcone Simulation II: Halo and Galaxy catalogues". Crocce, M.; Castander, F. J.; Gaztanaga, E.; Fosalba, P.; Carretero, J., *MNRAS*, 453, 1513 (2015)

"The MICE Grand Challenge light-cone simulation - III. Galaxy lensing mocks from all-sky lensing maps". Fosalba, P.; Gaztañaga, E.; Castander, F. J.; Crocce, M., *MNRAS*, 447, 1319 (2015)

"An algorithm to build mock galaxy catalogues using MICE simulations". Carretero, J.; Castander, F. J.; Gaztañaga, E.; Crocce, M.; Fosalba, P., *MNRAS*, 447, 646 (2015)

"Measuring the growth of matter fluctuations with third-order galaxy correlations", Hoffmann K., Bel J., Gaztanaga E., Crocce M., Fosalba P., Castander F.J., *MNRAS*, 447, 1724 (2015)

How to distribute this catalog

There are no special restrictions on use or distribution of this data.

I have read the instructions on how to cite CosmoHub and this catalog in my publications.

Figure 3.12: Catalog page bottom half, showing steps 4 to 7.

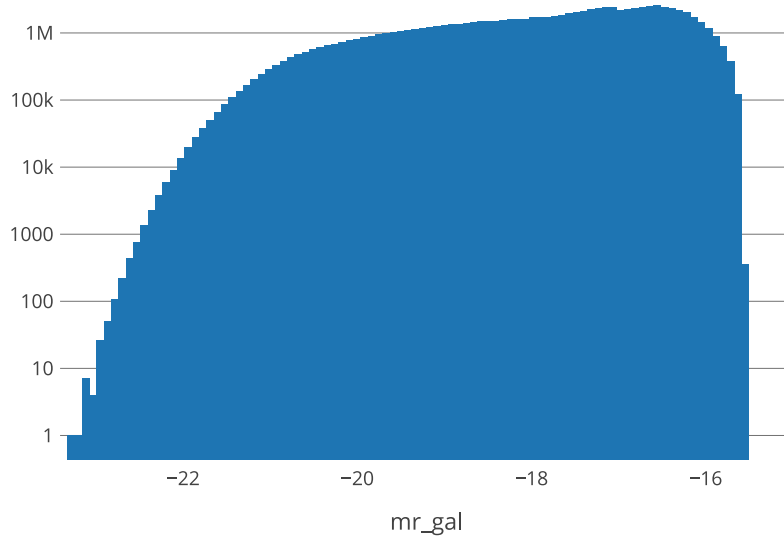


Figure 3.13: 1D histogram, displaying the number of galaxies in MICECAT2 with true redshift between 0.4 and 0.6, grouped by absolute magnitude in hundred uniformly sized bins.

following any actual property but a pseudo-random value, usually a surrogate key. In addition, users may specify in Step 3 any arbitrary criteria to further restrict the resulting rows. If multiple criteria are specified its effects are combined, thus only rows fulfilling all criteria are returned.

Step 4 displays the corresponding SQL statement constructed from the options selected in the previous steps. If the guided interface capabilities are not enough, or users are proficient using SQL, the *Expert* mode can be enabled by clicking a button. From that point on, the guided interface will be disabled and the SQL sentence can only be modified by manually editing it in the text area.

Once the custom catalog has been defined, users can interactively explore its properties using any of the 4 visualization tools in Step 5. (i) The table preview shows the first 20 rows in the subset and it is mainly used to have a glance at the results. (ii) The scatter plot allows to display the relationship between several properties. However, as it cannot aggregate data, it is limited to ten thousand points. (iii) 1D histograms can be generated from any column using a configurable number of uniformly sized bins. An example is shown in Figure 3.13. (iv) 2D heatmaps, as scatter plots, display the relation between two properties using rectangular bins. As with 1D histograms, bin ranges are filled in from column statistics. Also, the metric can be selected between *COUNT*, *AVG*, *MAX* or *MIN*, in order to display the number of rows, average, maximum or minimum value for each bin, respectively. Figure 3.14 shows an example.

After filling in the required fields, pressing the *Play* button will start the process to generate the visualization. All plots can have customized display options such as axis scaling (linear or logarithmic), axis direction (increasing or decreasing) or switching to a cumulative plot. Also, visualizations can be zoomed in and out, exported as a Portable Network Graphics (PNG) image or downloaded as a CSV file for additional processing.

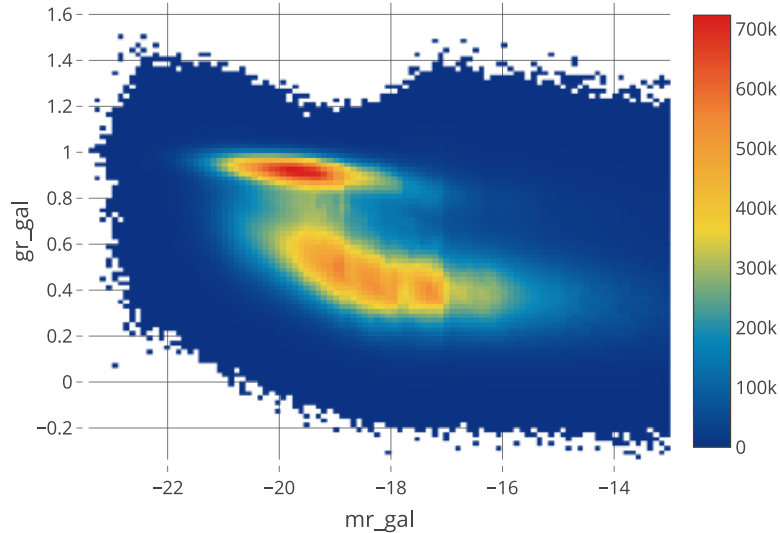


Figure 3.14: 2D heatmap, displaying a color-magnitude diagram of MICECAT2.

Custom catalogs Whenever users are satisfied with the properties of their custom catalog, they can request it to be generated and stored in a specific format to be later downloaded. Step 6 shows the supported formats: CSV with BZip2 compression, FITS and ASDF. Finally, after selecting the desired download format. Once requested, the custom catalog is assigned a unique identifier, so that users may track its progress

When a custom catalog is completed, users receive an email directing them to the Activity page (see Figure 3.15) in order to download it. Users may also use this page to follow the progress of their custom catalog requests. Finished catalogs are kept a minimum of 30 days and are eventually deleted to maintain enough free storage space.

3.9 Results

In this section we present some of the most relevant results and successful use cases of the version of CosmoHub as described in this article, which was commissioned in October 2016. The results are separated into two sections, the first one contains a quantitative analysis of the volume, timing and performance of the service as a whole, while the second section contains specific scientific applications where CosmoHub is being used.

3.9.1 Quantitative analysis

Since this version opened for public use, CosmoHub has been constantly growing in all relevant metrics, such as number of users, number of catalogs and volume of published data (see Figure 3.16). Note that the information in this figure only accounts for data published through CosmoHub, excluding custom catalogs.

Until 2022, more than 1500 new users have opened an account, and more than 12,000 custom catalogs and nearly 20,000 interactive queries have been delivered. Data growth has been limited by the available storage space.

ID	Query	Date	Status	Results
5351	SELECT `unique_gal_id`, `ra_gal`, `dec_gal`, `z_cgal`, `des_asahi_full_g_true`, `des_asahi_full_r_true`, `des_asahi_full ... Show full query	▶ 2019-07-01 11:14:45 ✓ 2019-07-01 11:16:07	SUCCEEDED	csv.bz2 (775.18 MiB)
5099	SELECT CAST((r_gal+delta_r)*cos((ra_gal)*(pi()/180))*cos(dec_gal* (pi()/180)) AS FLOAT, CAST((r_gal+delta_r)*sin((ra_ga ... Show full query	▶ 2019-06-03 15:22:50 ✓ 2019-06-03 15:39:15	FAILED	
4852	SELECT * FROM zest_v1_0 TABLESAMPLE (BUCKET 1 OUT OF 256)	▶ 2019-05-03 14:41:19 ✓ 2019-05-03 14:42:26	DELETED	Info
4781	SELECT `unique_gal_id`, `ra_gal`, `dec_gal`, `z_cgal`, `des_asahi_full_g_true`, `des_asahi_full_r_true`, `des_asahi_full ... Show full query	▶ 2019-04-24 15:27:32 ✓ 2019-04-24 15:32:02	DELETED	Info
4780	SELECT coadd_object_id, ra, dec, sof_cm_mag_corrected_g, sof_cm_mag_corrected_r, sof_cm_mag_corrected_i, sof_cm_mag_corr ... Show full query	▶ 2019-04-24 12:39:53 ✓ 2019-04-24 12:42:34	DELETED	Info

Figure 3.15: Activity page, used to follow custom catalog creation progress and to download them when they are ready.

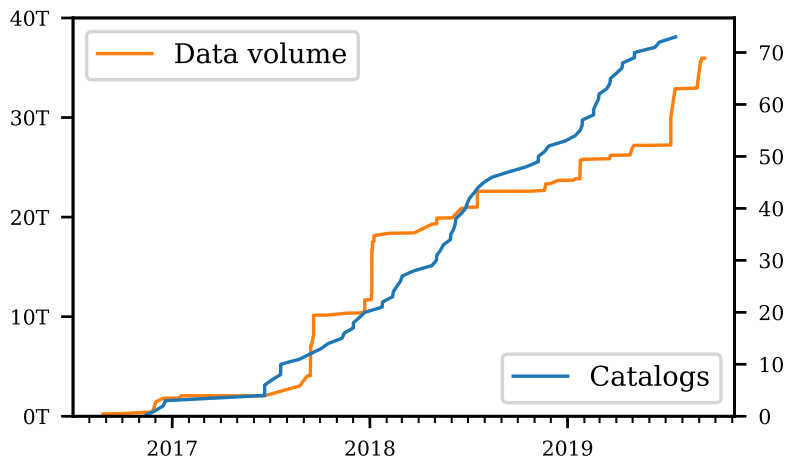


Figure 3.16: Evolution over time of the number of available catalogs and size of published data.

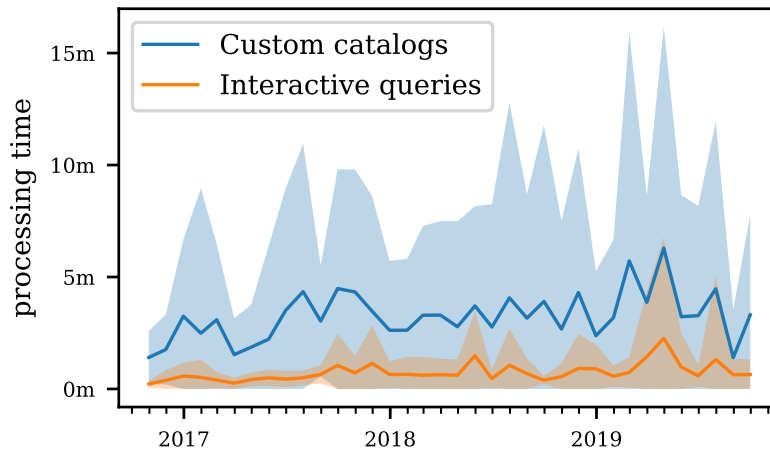


Figure 3.17: Processing time in minutes (monthly average) for batch catalogs and interactive queries. Shaded area shows the standard deviation.

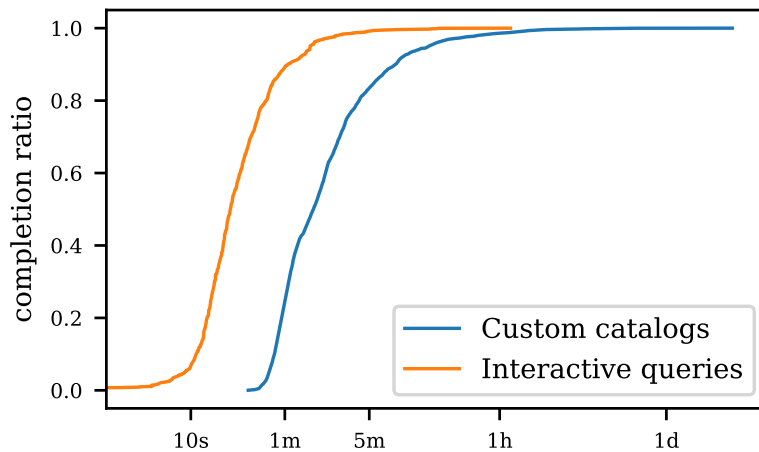


Figure 3.18: Cumulative completion ratio as a factor of processing time for batch catalogs and interactive queries.

At the same time, performance in terms of response time has been stable. Figure 3.17 shows that the average response time for the execution of interactive queries and the generation of custom catalogs has barely increased over time. Statistical fluctuations are due to resource contention with concurrent queries and scheduling overheads, among others.

It is worth noticing that, since its commissioning, the architecture and configuration of the Hadoop platform has seen several reorganizations, although the only resource that has been increased is the storage capacity. Therefore, by improving and tuning the platform, we have been able to cope with the growth in users and data volume and to keep the response time stable.

Figure 3.18 provides information about the distribution of response times by plotting the completion ration for both interactive queries and custom catalogs. The completion ration is defined as the fraction of queries completed after a given elapsed time. Note that the orchestration of the different tasks on the cluster nodes has a minimum overhead of about 10-12 seconds. Only queries that can be answered directly from statistics, such as

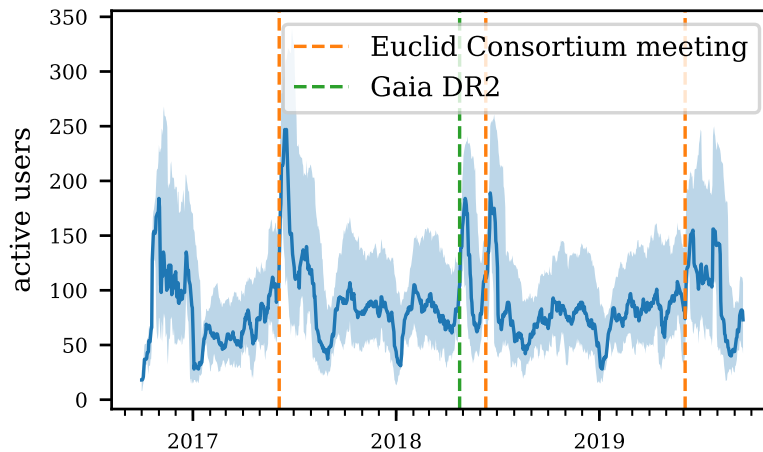


Figure 3.19: Active users over time, with relevant milestones highlighted. Upper and lower bounds correspond to active users within a window of 28 days and 7 days respectively, while middle line is for a 14 days window.

selecting the number of rows or the maximum value of a column without filters, return in a shorter time.

About 66.9% of all interactive queries finish in less than 30 seconds, while 96.8% of them finish in less than 2 minutes. These response times enable users to interactive explore any dataset, without worrying about its volume or the complexity of the query. On the other hand, for custom catalogs, 71.0% are produced in less than 3 minutes, while 97.4% of them finish in 30 minutes. Although not as relevant as for interactive queries, keeping a low response time for the generation of custom catalogs is also important to keep a good user experience.

Figure 3.19 shows the evolution of active users over time. Dates of particular scientific events are overlaid. A clear correlation can be seen, particularly with the Euclid Consortium meeting held yearly every June, where interest on newly released data boosts user activity. A peak in activity can also be seen coinciding with Gaia Data Release 2 (DR2¹⁰²). The catalog was mirrored in CosmoHub in less than 12h and we were able to release it almost simultaneously to the official announcement at the European Space Astronomy Centre (ESAC¹⁰³).

3.9.2 Scientific applications

CosmoHub supports multiple international cosmology projects, the most relevant in terms of users and data volume being Euclid¹⁰⁴. Euclid simulations require the production of extremely large datasets. In their Flagship simulation¹⁰⁵ (paper in preparation), with an estimated final size of 20 TiB and nearly 30×10^9 objects, they are using as input the largest dark-matter halo catalog up to date, with 5.5 TiB and 40×10^9 objects. CosmoHub is currently providing access to the entire halo catalog and to several productions of mock

¹⁰²<https://www.cosmos.esa.int/web/gaia/dr2>

¹⁰³http://www.esa.int/About_Us/ESAC

¹⁰⁴<http://sci.esa.int/euclid/>

¹⁰⁵https://www.euclid-ec.org/?page_id=4133

Catalog	Rows	Fields	Size
ALHAMBRA			
— S/G classified	422 K	7	10.4 MiB
— photometric redshifts	441 K	113	149.5 MiB
CFHTLens	29 M	129	7.5 GiB
COSMOS 2015	1.2 M	537	1.6 GiB
Gaia DR1	1,143 M	62	174.9 GiB
Gaia DR2	1,693 M	98	486.5 GiB
KiDS DR4	100 M	306	89.6 GiB
MICECAT 1	205 M	91	59.9 GiB
MICECAT 2	500 M	122	211.1 GiB
PAUS-COSMOS EDR	6.5 K	126	2.8 MiB
PAU-MillGas Lightcone	7.4 M	34	9.0 GiB
Zest	132 K	71	20.9 MiB

Table 3.3: List of public catalogs in CosmoHub (as of 2019).

galaxy catalogs (these account for about half of the data stored in CosmoHub). The Euclid cosmological simulations validation team makes heavy use of the exploration capabilities of CosmoHub in order to validate data prior to its release to the full collaboration. Then, Euclid scientists use CosmoHub to download customized subsets of the data for further analysis and processing.

Up to 2023, 84 publications have acknowledged CosmoHub contribution to their results. Projects such as PAU Survey¹⁰⁶ and MICE¹⁰⁷ use CosmoHub as the official and primary channel for the distribution of their data (Eriksen et al., 2019 and Cabayol et al., 2019). Other projects such as DES¹⁰⁸ and Gaia¹⁰⁹ have a replica of their most important releases, and several publications have made use of them (see Serenelli, Aldo et al., 2019 and Sevilla-Noarbe et al., 2018). The subset of public catalogs in CosmoHub (as of October 2019) is shown in Table 3.3.

Finally, other applications not based solely on the distribution of data are also present, such as the one from the DES clustering science working group. They made intensive use of the exploration capabilities to define and test many different arbitrary galaxy subsamples to estimate the Baryon Acoustic Oscillations (BAO) feature in the galaxy distribution (Crocce et al., 2019). Some particularly useful applications are shown in 3.A. Lastly, some projects have also cited CosmoHub as a state-of-the-art reference to their own data publication procedures, such as Heitmann et al., 2019 and Nelson et al., 2019.

3.10 Conclusions and future work

CosmoHub enables the interactive exploration and distribution of large cosmological datasets on top of Hadoop. This chapter describes the main features and capabilities of Cos-

¹⁰⁶<https://www.pausurvey.org/>

¹⁰⁷<http://maia.ice.cat/mice/>

¹⁰⁸<https://www.darkenergysurvey.org/>

¹⁰⁹<http://sci.esa.int/gaia/>

moHub from the user’s point of view but, more importantly, it also details all the research and decisions made regarding its design and implementation.

Regarding the design (section 3.3), it is focused on satisfying a set of needs (enumerated in section 3.2) gathered from the scientific community, while the experience gained through the early prototypes (described in section 3.4) helped pave the way to achieve its current success. In particular, the easy to use requirement has been met by implementing CosmoHub as a web application, with a guided process to remove any SQL knowledge dependency, and the support for common astronomical data formats such as CSV and FITS. Also, the ability to produce visualizations to get insight over billions of rows in just a few seconds fulfills the interactive exploration requirement. Several projects such as PAUS, MICE and the Euclid simulation group have selected CosmoHub as their primary data distribution service, which was also one of our objectives.

The decision to delegate CosmoHub’s data processing to Hadoop and Hive (see section 3.5) has proven to be wise, as the resulting implementation is reliable, high performing and usable with powerful features. Also, the great scalability of the platform has allowed to keep response times low at all times (see section 3.9.1), in spite of the constant increase in data volume. In the end, CosmoHub is providing a useful service to the scientific community with a high quality of service, as proven by the use of CosmoHub by some of the most relevant projects in cosmology (see section 3.9.2).

When CosmoHub entered into service in late 2016, it was the first project to apply Hadoop to the analysis and distribution of large cosmological datasets. Over these years we have learnt a lot from both our own experience and user’s feedback. In fact, we are already working on the next iteration of CosmoHub which will include a lot of improvements based on this experience:

(i) Regarding the Hadoop platform, upgrade it to a custom developed Hadoop distribution. The most exciting new features include the possibility to reduce replica overhead using erasure coding, the ability to access a read-only view of externally provided storage and the implementation of materialized views in Hive to speed up join queries.

(ii) From CosmoHub application’s perspective, add the ability for users to upload their own catalogs and to publish and share them with other users, extend and optimize the visualization tools performance, and improve the general responsiveness of the user interface. We also want to implement some VO protocols such as the Table Access Protocol (TAP) to enable programmatic access to launch SQL queries and retrieve results, and to integrate with Jupyter Notebooks to be able to create arbitrary plots or perform custom analysis. There are plans to provide guided analysis tasks to compute the 2-point galaxy correlation function, to generate mock galaxy catalogs using the Halo Occupation Distribution (HOD) model or to compute the galaxy photometric redshift, among others.

With all this future work under way, we are prepared to keep pushing forward and to help put in place the next generation of services for managing large volumes of structured scientific data.

3.A Particularly useful applications

This appendix describes in detail several representative use cases that make use of the custom catalog generation capabilities in CosmoHub. Each application includes the full SQL statement that was used, along with the time it took to complete. The timings measured in this section, unlike the results in section 3.9, were performed having exclusive use of the entire Hadoop platform.

MICECAT1 clustering sample

In this application, we want to generate a subset of MICECAT1 in order to compute the projected 2-point correlation function on it. Thus, we selected the right ascension, the declination and the comoving distance columns, and we filtered on a redshift shell (with z between 0.3 and 0.4) and also on a magnitude range (with absolute magnitude on the r band between -22 and -21). The creation of this custom catalog takes only 14 seconds and generates a CSV.BZ2 file of 4.99 MiB containing 492,210 rows.

```
SELECT 'ra', 'dec', 'd_c'
FROM micecat_v1
WHERE 'z' > 0.3 AND 'z' < 0.4
      AND 'abs_mag_r' < -21 AND 'abs_mag_r' > -22
```

DES Y1A1 BAO main sample

Another interesting application was the generation of the BAO sample for DES. This sample, described in Crocce et al. (2019), is a subset of DES Y1 data that, according to the article, represents “red galaxies with a good compromise of photo- z accuracy and number density, optimal for the BAO measurement”. The query below implements the criteria shown in Table 1 of the paper. The fast response times of CosmoHub were particularly useful to interactively refine the parameters of the sample, which is now available also as a predefined dataset within one of the DES private catalogs. An interactive query to visualize the number of objects as a function of the photometric redshift takes 31 seconds. Exporting the sample into a CSV.BZ2 file of 55.6 MiB containing 2.7 million objects takes 39 seconds.

```
SELECT
  coadd_objects_id, ra, dec,
  mean_z_bpz_hiz, z_mc_bpz_hiz, t_b_hiz, odds_hiz
FROM des_y1
WHERE (mag_auto_i > 17.5) AND (mag_auto_i < 22)
      AND (mag_auto_i < 19.0 + 3.0*mean_z_bpz_hiz)
      AND (ra < 15 or ra > 290 or dec < -35)
      AND (flags_badregion <= 3 and flags_gold = 0)
      AND (spread_model_i + (5.0/3.0)*spreaderr_model_i > 0.007)
      AND ((mag_auto_i - mag_auto_z) + 2.0*(mag_auto_r - mag_auto_i) > 1.7)
      AND ((mag_auto_g - mag_auto_r) BETWEEN -1. and 3.)
      AND ((mag_auto_r - mag_auto_i) BETWEEN -1. and 2.5)
      AND ((mag_auto_i - mag_auto_z) BETWEEN -1. and 2.)
```

GAIA DR2 HEALPix partial map

The custom catalog feature can also be used in conjunction with the FITS format to create HEALPix maps. For instance, in this application the following query was used to create a partial map with explicit indexing¹¹⁰ estimating the average of the *Standard error of parallax (Angle/mas)* for each pixel. The pixel identifier is taken from the `_hpix_12_nest` column. The generation of this custom catalog produces a FITS file of 156 million rows and 1.74 GiB in size in 65 seconds.

```
SELECT '_hpix_12_nest', AVG(parallax_error)
FROM gaia_dr2
GROUP BY '_hpix_12_nest'
```

Euclid True Universe FITS file

CosmoHub stores and distributes a large amount of data for the Euclid Cosmological Simulations Working Group (CSWG) which is then used as input for different image simulator pipelines. This application uses a complex SQL statement to generate an individual input from the catalogs stored in CosmoHub. The resulting FITS file has the correct format, the proper field names and the correct units. This allows Euclid scientists to easily test their codes on a smaller scale, while at the same time enabling them to iterate and provide feedback much faster. In this particular example, the FITS file generated contains 1823344 objects, occupies 406.9 MiB and was produced in 22s.

```
SELECT CAST(((gal.halo_id * 10000) + gal.galaxy_id) AS bigint) AS SOURCE_ID,
       CAST(gal.ra_gal AS float) AS RA,
       CAST(gal.dec_gal AS float) AS DEC,
       CAST(gal.ra_gal_mag AS float) AS RA_MAG,
       CAST(gal.dec_gal_mag AS float) AS DEC_MAG,
       CAST(gal.observed_redshift_gal AS float) AS Z_OBS,
       CAST(gal.abs_mag_r01_evolved AS float) AS TU_MAG_R01_SDSS_ABS,
       CAST(-2.5*log10(gal.sdss_r01) - 48.6 AS float) AS TU_MAG_R01_SDSS,
       CAST(gal.sed_cosmos AS float) AS SED_TEMPLATE,
       CAST(ROUND(gal.ext_curve_cosmos) AS smallint) AS EXT_LAW,
       CAST(gal.ebv_cosmos AS float) AS EBV,
       CAST(gal.logf_halp_alpha_model3_ext AS float) AS HALPHA_LOGFLAM_EXT,
       CAST(gal.logf_hbeta_model3_ext AS float) AS HBETA_LOGFLAM_EXT,
       CAST(gal.logf_o2_model3_ext AS float) AS O2_LOGFLAM_EXT,
       CAST(gal.logf_o3_model3_ext AS float) AS O3_LOGFLAM_EXT,
       CAST(gal.logf_n2_model3_ext AS float) AS N2_LOGFLAM_EXT,
       CAST(gal.logf_s2_model3_ext AS float) AS S2_LOGFLAM_EXT,
       CAST(gal.bulge_fraction AS float) AS BULGE_FRACTION,
       CAST(gal.bulge_length AS float) AS BULGE_LENGTH,
       CAST(gal.disk_length AS float) AS DISK_LENGTH,
       CAST(gal.disk_axis_ratio AS float) AS DISK_AXIS_RATIO,
       CAST(gal.disk_angle AS float) AS DISK_ANGLE,
       CAST(gal.kappa AS float) AS KAPPA,
       CAST(gal.gamma1 AS float) AS GAMMA1,
```

¹¹⁰See https://healpix.sourceforge.io/data/examples/healpix_fits_specs.pdf for more information about how HEALPix data is stored as FITS.

```

CAST(gal.gamma2 AS float) AS GAMMA2,
CAST(gal.mw_extinction AS float) AS AV,
CAST(gal.euclid_vis_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_VIS,
CAST(gal.euclid_nisp_y_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_Y_NISP,
CAST(gal.euclid_nisp_j_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_J_NISP,
CAST(gal.euclid_nisp_h_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_H_NISP,
CAST(gal.blanco_decam_g_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_G_DECAM,
CAST(gal.blanco_decam_r_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_R_DECAM,
CAST(gal.blanco_decam_i_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_I_DECAM,
CAST(gal.blanco_decam_z_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_Z_DECAM,
CAST(gal.cfht_megacam_u_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_U_MEGACAM,
CAST(gal.cfht_megacam_r_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_R_MEGACAM,
CAST(gal.jst_jpcam_g_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_G_JPCAM,
CAST(gal.pan_starrs_i_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_I_PANSTARRS,
CAST(gal.pan_starrs_z_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_Z_PANSTARRS,
CAST(gal.subaru_hsc_z_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_Z_HSC,
CAST(gal.gaia_g_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_G_GAIA,
CAST(gal.gaia_bp_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_BP_GAIA,
CAST(gal.gaia_rp_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_RP_GAIA,
CAST(gal.lsst_u_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_U_LSST,
CAST(gal.lsst_g_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_G_LSST,
CAST(gal.lsst_r_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_R_LSST,
CAST(gal.lsst_i_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_I_LSST,
CAST(gal.lsst_z_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_Z_LSST,
CAST(gal.lsst_y_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_Y_LSST,
CAST(gal.kids_u_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_U_KIDS,
CAST(gal.kids_g_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_G_KIDS,
CAST(gal.kids_r_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_R_KIDS,
CAST(gal.kids_i_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_I_KIDS,
CAST(gal.2mass_j_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_J_2MASS,
CAST(gal.2mass_h_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_H_2MASS,
CAST(gal.2mass_ks_el_model3_odonnell_ext*1.e23 AS float) AS TU_FNU_KS_2MASS,
CAST(SHIFTRIGHT(gal.hpix_29_nest, (29-5)*2) AS bigint) AS hpix_5_nest
FROM cosmohub.flagship_mock_sc456 AS gal
WHERE SHIFTRIGHT(hpix_29_nest, (29-5)*2) = 7155

```

Chapter 4

Simulating galaxy fluxes

Cosmological simulations are intricate computational models that replicate the evolution of the universe over vast cosmic scales of space and time. These simulations employ sophisticated numerical techniques to solve the complex equations governing fundamental forces like gravity, fluid dynamics, and radiation. By dividing a simulated "box" of space into a grid and tracking variables like matter density and energy distribution at each point, these simulations offer a dynamic portrayal of cosmic processes, providing a virtual laboratory to explore the universe's history.

Cosmological simulations hold immense significance in observational cosmology due to their unique ability to connect theoretical models with real-world observations. By generating synthetic data that mirrors astronomical phenomena, simulations aid in understanding how galaxies, dark matter, dark energy, and other cosmic components interact, form structures, and evolve across billions of years.

In this context, the simulation of the observed galaxy electromagnetic fluxes, the amount of energy captured per unit of time and surface, is crucial because it enables the interpretation and calibration of astronomical data obtained from telescopes and other observational instruments. By accurately simulating how galaxies emit and interact with different types of electromagnetic radiation across various wavelengths (such as visible light, radio waves, X-rays), they allow astronomers to understand how galaxies and their components contribute to the observed fluxes, aiding in the identification of different astrophysical processes, such as star formation, supernovae, and active galactic nuclei. Furthermore, these simulations are fundamental to design and optimize observational strategies, improving our ability to analyze and interpret the vast amount of data collected from telescopes.

The computation of the observed electromagnetic flux of an astronomical object given its spectral energy distribution (SED) –the energy emitted by an object as a function of the wavelength– is one of the fundamental properties provided in simulated galaxy catalogs.

The simulation of the observed electromagnetic flux of an astronomical object given its spectral energy distribution (SED) –the energy emitted by an object as a function of the wavelength– requires computing an integral, either in wavelength or frequency, of the object SED, multiplied by the optical system response function. This is not a complicated

operation, but it takes 36 milliseconds¹ to compute the flux for one galaxy in a single band. Therefore, when this process has to be performed for a large number of galaxies (i.e. billions of objects) the computational time required becomes prohibitive without optimized parallelization. Moreover, when the SED comes from a simulation it may have separate components affecting in different ways the final result, such as the stellar continuum, the emission line fluxes or the extragalactic absorption. In those cases, it can be more efficient to compute each contribution separately, but this also adds complexity to the process.

The Euclid Flagship mock galaxy catalogue^{2,3} is the largest synthetic mock galaxy catalogue ever produced, containing of the order of 5 billion objects with up to 400 properties, including their fluxes in more than 30 energy (or frequency) bands. The scale of this catalog means that it is completely unfeasible to compute the fluxes using a traditional approach using integrals, as the processing time would be around 170 years using a single core⁴.

This chapter explains in detail the methodology we have followed to design, develop, calibrate and validate an approximate method that enables the computation of the galaxy fluxes in a much faster way, while ensuring the results still fulfill the scientific precision constraints. This work is part of a larger suite of algorithms called SciPIC (Carretero et al., 2017), that has been successfully applied to the generation of Euclid and PAUS mock galaxy catalogs for their instrument and scientific simulations.

In the approach presented in this chapter, the computation of each galaxy flux is divided in three independent steps. The combination of the results from each step produces the desired value for the flux. This method has been heavily optimized and is calibrated to deliver its results as fast and accurate as possible. Using this approach, it takes 47 microseconds on average¹ to estimate the observed flux in any band, lowering the time needed using integrals by a factor 750 in equal computing conditions.

In the case of simulations for the Euclid project, this means that the fluxes of the entire Flagship mock galaxy catalog with its 5 billion sources and 30 wavelength bands of the Euclid satellite instruments (including the visible imager (VIS) and the near infrared 3-filter (Y, J and H) photometer (NISP-P) instruments) can be estimated in less than 12 hours when using 300 cores on our PIC Big Data platform. However, most dataset productions are internal validation releases that are much smaller and take less than an hour to be produced. For PAUS simulations, albeit its galaxy mock catalogs are smaller (~ 3 billion sources), they require a larger set of wavelength bands (~ 90 filters) so they end up taking a similar amount of time.

This method must also be precise as well as fast in order for its results to be scientifically valid. The effective scientific applicability of this method has been tested for both Euclid and PAUS simulations, that define strict requirements in terms of precision for their measures. For Euclid, when comparing the magnitude⁵ difference between the standard integrals and the approximated approach, 99.8626% of the simulated galaxies fulfill the requirement of having a $\Delta m < 0.01$. The method has also been optimized and

¹at PIC, using a reference computing node.

²<https://sci.esa.int/web/euclid/59348-euclid-flagship-mock-galaxy-catalogue>

³https://www.esa.int/About_Us/ESAC/Euclid_flagship_mock_galaxy_catalogue

⁴ $510^9 \text{ objects} \cdot 30 \text{ bands/object} \cdot 0.036 \text{ seconds/band} = 5.410^9 \text{ seconds} = 62,500 \text{ days} = 171,12 \text{ years}$

⁵As galaxy fluxes are so minuscule, astronomers prefer using logarithmic units such as magnitudes (m). Also, magnitudes align better with how our eyes perceive brightness changes as non-linear. See section 4.2 for the exact definition.

used to simulate the observed flux for PAUS using its narrow band filters, fulfilling the survey requirement $\Delta m < 0.01$ for 99.9996% of the simulated galaxy sample. Note that this method is very generic and, with the proper calibration, can be applied to simulate fluxes for any survey.

The structure of this chapter is as follows. Section 4.1 introduces SciPIC, a set of codes and algorithms for the generation of synthetic mock galaxy catalogs using a specific kind of cosmological simulations, and which this contribution is part of. Section 4.2 describes the traditional approach to estimate the apparent magnitude using integrals having as input the parameters from the cosmological simulations without any approximation, produced for cosmological surveys such as Euclid and PAUS. Section 4.3 shows the specific input data used in the design of the approximate computation method. Section 4.4 explains the optimization and calibration of the approximate method needed for Euclid and PAUS mock galaxy simulations and integrated in SciPIC. Section 4.5 displays the scientific performance and accuracy results using a sample of galaxies. Finally, section 4.6 collects the conclusions of this chapter.

4.1 SciPIC

SciPIC, named after Scientific Pipeline at PIC, is a suite of algorithms integrated into a powerful pipeline dedicated to the generation of massive synthetic galaxy catalogs based on halo catalogs coming from n-body dark matter cosmological simulations. These algorithms implement multiple recipes, each one of them specifically tailored to simulate a distinct set of dependent galaxy properties.

Originally, most of the algorithms were designed, developed and applied to the production of the Marenstrum Institut de Ciències de l'Espai (MICE) galaxy catalogs (Carretero et al., 2015). Afterwards, I have led and contributed greatly to the refactoring, optimization, calibration and integration of all these codes into SciPIC, coordinated by a team of scientific advisors.

This effort was fundamental in order for them to perform as efficiently as possible, thus allowing the pipeline to scale up to producing catalogs of billions of entries, such as the Euclid Flagship mock galaxy catalog⁶ (see Castander et al. in preparation), among other datasets. With up to 10 billion objects up to redshift⁷ $z = 3$ and more than 1250 properties each, it is one of the largest (if not the largest) and most complete synthetic galaxy catalogs ever produced. It is based on the Flagship simulation (see Stadel et al. in preparation), also the largest dark-matter particle simulation produced at the time of this writing, and its corresponding halo catalog⁸ with more than 130 billion objects and about 80 properties. Also, it has also been used in the production of synthetic galaxy catalogs for the PAU Survey project (see Cabayol et al., 2023).

The combination of all these state-of-the-art algorithms in a single pipeline is a very distinctive key feature that greatly improves the scientific applications of its results and the corresponding impact of the derived research. The resulting pipeline is divided in a series of sequential steps that can be enabled individually in order to customize the resulting catalog. Each of those steps involves a different algorithm that computes a distinct set of properties, and several of them already have a corresponding paper describing their design, implementation and scientific results.

This chapter describes in depth the galaxy flux computation algorithm (step 9 from the list below), for which a short article was published (see Tallada-Crespí et al. (2023)) explaining the main approach and results. This chapter is the evolution of the original extended article that describes in greater detail the design, implementation, calibration and testing done to the algorithm to ensure its suitability for the simulation of galaxy fluxes for both Euclid and PAUS surveys.

The input of the SciPIC pipeline is specified as an SQL query producing the columns required depending on the enabled steps. It can also be used to patch or improve an already existing catalog by recomputing a subset of its properties. As a reference, here follows the description of all the steps listed in the order they get computed:

⁶https://www.esa.int/About_Us/ESAC/Euclid_flagship_mock_galaxy_catalogue

⁷In this context, redshift is used to refer to how far does the simulation reach, in terms of time and space. In more "traditional" units, we could say the catalog contains galaxies up to 11.5 giga years old, or 21.1 giga light-years away.

⁸As dealing with individual particles of a dark-matter simulation is computationally unfeasible, neighbouring particles within a certain distance are aggregated into a structure called dark-matter halo. This also allows the computation of additional properties such as morphology or density profiles, among others.

1. **Chunking:** All rows from the input catalog are split into subsets and transformed into an iterator of `Pandas DataFrame` objects (one per subset). Then, the following steps can take advantage of the column-oriented layout of this structure to perform very efficient vectorized operations in order to compute their set of properties.
2. **Halo properties:** Several additional properties are derived from the input halo catalog, such as true redshift, concentration-virial mass relation, smoothed mass or Euler angles.
3. **Galaxy assignment:** A specific number of galaxies are assigned to each host halo using both the Halo Occupation Distribution (HOD) model and the Sub-Halo Abundance Matching (SHAM) method, depending on the mass of the halo and a luminosity function (Blanton et al., 2003, 2005, see). We also assign each of them a *color* –represented by a flux ratio between two wavelength bands–, positions and velocities, among other main properties.
4. **Spectral Energy Distribution (SED)** parameters: Based on the luminosity, color and distance, we assign each galaxy an SED template from a well-known library, a corresponding intrinsic extinction curve and color excess $E(B - V)$. See section 4.3 for the definition of these terms.
5. **Lensing:** Parameters describing the deformation of the galaxy shape due to gravitational effects –also known as lensing– are assigned to each galaxy from corresponding lensing maps provided along the dark-matter simulation. These parameters are assigned from very large arrays, covering the full sky and 200 redshift steps, that do not fit in memory. In order to make the mapping feasible and efficient, the input is organized by redshift step and sky region. Thus, if this step is enabled, prior to computing these properties the mock catalog is spilled to disk, registered as a temporary table, and read again in order to regroup the data in the optimal way to minimize memory consumption.
6. **Magnified positions:** The apparent positions of the galaxies on the sky are *magnified* by applying the lensing distortions computed in the previous step.
7. A **rotation** can be applied to positions, velocities and orientations, both from host halos as well as galaxies, in order to place the simulated area in the desired region on the sky.
8. Multiple **stellar properties** are computed, such as metallicity, stellar mass or the luminosity of emission lines, among others.
9. **Fluxes:** We simulate the apparent and absolute electromagnetic flux of each galaxy as a combination of the galaxy SED, the emission lines flux and the attenuation due to the Milky Way dust extinction.
10. The **morphological parameters** that define the shape and orientation of each galaxy are derived.
11. The **photometric noise** estimation is carried out by applying different recipes depending on the filter (or wavelength band) to account for uncertainties in the flux measurement.

12. The **intrinsic alignment** of each galaxy is assigned. More details can be found in Hoffmann et al. (2022). Note that this is a fundamental and novel property that very few catalogs have.
13. An estimation of the **photometric redshift** using a deep learning code based on the fluxes and noise derived in step 9 and 11, respectively. The method is explained in detail in Eriksen et al. (2020).
14. Finally, the resulting catalog is **stored to disk and registered as a table**. After the proper validation process, it is published in CosmoHub to make it accessible to its final users.

4.2 Computing fluxes for simulations

Synthetic galaxy catalogs are produced based on data coming from simulations of the universe and provide a multitude of properties for each galaxy. One of the most interesting properties in observational cosmology are the galaxy spectral flux densities (or electromagnetic fluxes, in short). From them, we can derive many other properties such as the color or redshift.

In most simulated galaxy catalogs, fluxes are computed for a combination of instruments and filters (see section 4.3.2), corresponding to a set of telescopes and wavelength ranges on which they perform their observations. Fluxes denote the rate at which energy is transferred by electromagnetic radiation, per unit surface area and per unit wavelength. In SI units, it is measured in $\text{W} \cdot \text{m}^{-3}$, although in astronomy it is much more practical to use cgs units such as $\text{erg} \cdot \text{s}^{-1} \cdot \text{cm}^{-2} \cdot \text{Hz}^{-1}$.

However, astronomers usually work with AB magnitudes. Given a flux in cgs units, its corresponding AB magnitude (m_{AB}) is defined for a monochromatic frequency (or wavelength) as:

$$m_{\text{AB}} = -2.5 \log_{10} f(\nu) - 48.6 \quad (4.1)$$

where $f(\nu)$ is the energy flux density in cgs units. This definition can be extended to the magnitude measured with a filter of transmission $T(\nu)$ for a photon counting device as Fukugita et al., 1996:

$$m_{\text{AB}} = -2.5 \log_{10} \frac{\int f(\nu) T(\nu) \frac{d\nu}{\nu}}{\int T(\nu) \frac{d\nu}{\nu}} - 48.6 \quad (4.2)$$

which can also be computed in wavelength as

$$m_{\text{AB}} = -2.5 \log_{10} \frac{\int f(\lambda) T(\lambda) \frac{\lambda}{c} d\lambda}{\int T(\lambda) \frac{d\lambda}{\lambda}} - 48.6 \quad (4.3)$$

using the conservation of energy as $f(\nu) d\nu = f(\lambda) d\lambda$ and the relation $\nu = c/\lambda$, where c is the speed of light.

The relation between the apparent magnitude⁹, m , and the absolute magnitude¹⁰, M , in the same band is given by the following expression using the distance modulus relation:

$$m = M + 5 (\log_{10} (D_L) - 1) + K_{\text{corr}} \quad (4.4)$$

where D_L is the luminosity distance¹¹ in megaparsecs (Mpc) and K_{corr} is the K-correction¹² which is needed because, for a galaxy observed at a given redshift, the absolute (M) and apparent (m) magnitudes do not sample the same wavelength region of the SED (Hogg et al., 2002).

The galaxy simulations used provide, for each galaxy, all the necessary information to estimate the apparent magnitude in any band: the absolute AB magnitude in the $^{0.1r}$ band of the Sloan Digital Sky Survey (SDSS)¹³ ($M_{0.1r} - 5 \log_{10} h$), the redshift and the SED. The luminosity distance (D_L) can be estimated using Equation 4.4 given the galaxy redshift and the assumed cosmological model¹⁴. Then, the K-correction can be estimated in the $^{0.1r}$ band of SDSS using the following expression:

$$K_{\text{corr}} = 2.5 \log_{10} \left[\frac{\int f^{\text{rest}}(\lambda) T(\lambda) \lambda d\lambda}{\int f^{\text{obs}}(\lambda) T(\lambda) \lambda d\lambda} \right] \quad (4.5)$$

where $f^{\text{rest}}(\lambda) = SED_{\text{rest}}(\lambda)$ is the galaxy template in rest-frame, $T(\lambda)$ is the system transmission and $f^{\text{obs}}(\lambda) = SED_{\text{obs}}(\lambda)$ is the *redshifted* observed galaxy SED. The way to compute $SED_{\text{obs}}(\lambda)$ is to shift the wavelengths to $\lambda \cdot (1 + z_{\text{obs}})$ and decrease the *flux_density* as $flux_density / (1 + z_{\text{obs}})$, where z_{obs} is the observed redshift. Note that Equation 4.5 is a simplified version of the ones described in Hogg et al., 2002.

The Euclid Flagship galaxy mock catalogue does not directly provide the SED of a galaxy, but the values of the templates to build an SED from a library of templates and the emission lines fluxes. The SED is further split into the stellar continuum contribution and the internal extinction. For more details, see sections 4.3.1 and 4.3.3

See https://github.com/ptallada/scipic_fluxes for the full source code of the standard approach based on integrals and figure 4.1 for the full computational flow.

We start by computing the extinguished SED from the unextinguished galaxy SED template using the following expression:

$$SED^{\text{ext}}(\lambda) = SED^{\text{un-ext}}(\lambda) \cdot \text{ext_factor}(\lambda) \quad (4.6)$$

⁹Measures the brightness of a galaxy as seen from Earth, including the attenuation by the distance. It can also include other effects such as intergalactic dust or magnification due to lensing.

¹⁰Measures the intrinsic brightness of a galaxy, regardless of its distance from Earth. It is calculated based on the amount of electromagnetic flux that the galaxy would emit if it were placed at a distance of 10 parsecs from Earth.

¹¹The relation between an object's observed brightness and its cosmological distance. It is defined as the distance at which a point source would have to be placed in order to have the same observed brightness as the object in question.

¹²A factor that is applied to the observed magnitude of an astronomical object to account for the redshift of the object

¹³In Blanton et al., 2003, the 0.1 band passes are defined as the SDSS bandpasses shifted to match their rest-frame shape at redshift $z = 0.1$.

¹⁴ Λ CDM model with $\Omega_m = 0.319$, $\Omega_b = 0.049$, $\Omega_\Lambda = 0.681$, $\sigma_8 = 0.83$, $n_s = 0.96$ and $h = 0.67$

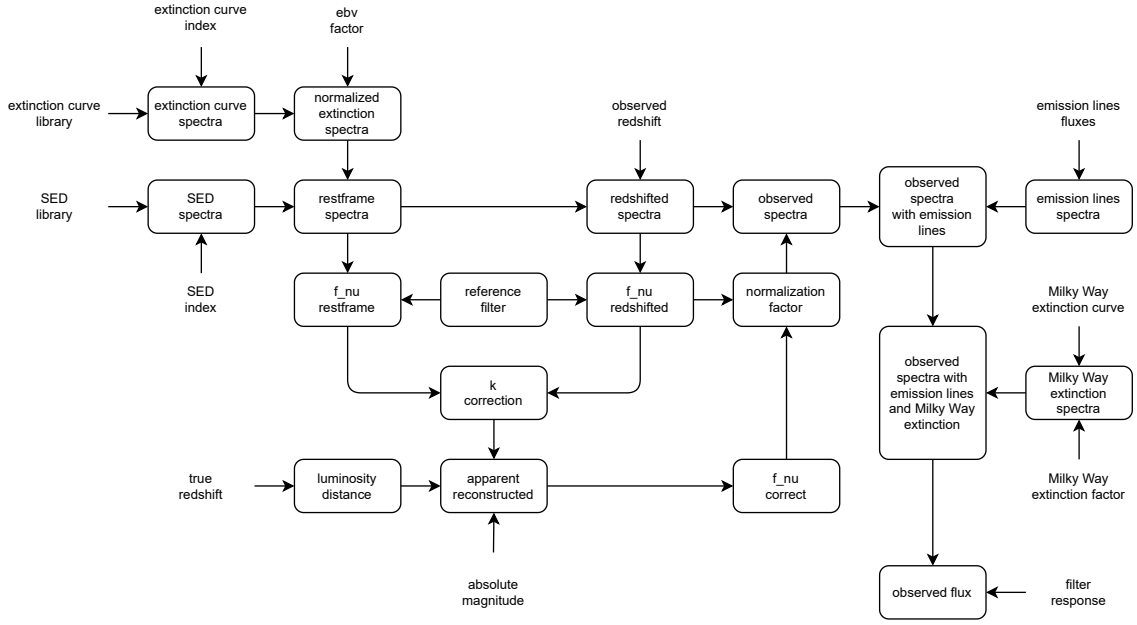


Figure 4.1: Galaxy flux computation flow. Each calculation is denoted by a rounded square with the result as its label, and the values it depends on are showed by arrows. The external input values are displayed without any enclosing box.

where $ext_factor(\lambda)$ is given by the expression:

$$ext_factor(\lambda) = \left(\frac{ext_curve(\lambda)}{ext_curve(\lambda)[0]} \right)^{(E(B-V)/0.2)} \quad (4.7)$$

where:

- $SED^{ext_curve}(\lambda)$ is the extinction curve for that particular galaxy
- $SED^{ext_curve[0]}(\lambda)$ is a reference function in frequency that does not apply any extinction to the galaxy
- $E(B - V)$ is the colour-excess¹⁵ given in the catalogue.

Note that this expression is somewhat peculiar because it is driven by the way the extinction is provided in the input catalogue and the format in which the extinction templates are given.

As explained above, the apparent AB magnitude in the $^{0.1}r$ band of SDSS can be estimated using Equation 4.4. Then, fluxes can be computed from magnitudes using the inverse of the AB magnitude definition given in Equation 4.1.

$$^{0.1}r\ flux_observed = 10^{-0.4(^{0.1}m_r + 48.6)} \quad (4.8)$$

¹⁵It measures the reddening of light due to interstellar dust, defined as the difference between the observed (B-V) color and its intrinsic (B-V) color

Then, the observed AB magnitude is computed from Equation 4.3. In fact, one can alternatively reformulate Equation 4.3 in flux (in $\text{erg s}^{-1} \text{cm}^{-2}$ units) instead of in AB magnitudes, as shown in Equation 4.9.

$${}^{0.1r}flux_integral = \frac{\int f(\lambda)T(\lambda)\frac{\lambda}{c} d\lambda}{\int T(\lambda)\frac{d\lambda}{\lambda}} \quad (4.9)$$

Note that here we use the extinguished SED, $f(\lambda) = SED^{ext}(\lambda)$, of Equation 4.6. With both values, ${}^{0.1r}flux_observed$ and ${}^{0.1r}flux_integral$, the normalization factor of the SED is given by:

$$norm_factor = \frac{{}^{0.1r}flux_integral}{{}^{0.1r}flux_observed} \quad (4.10)$$

With the normalization factor we can compute the normalized SED. Then, using Equation 4.9 one can compute the flux of an object in any particular filter with transmission curve $T(\lambda)$, where $f(\nu) = SED_{obs}^{ext,norm}$ is the observed spectral energy distribution already normalized and with the intrinsic extinction applied. And again, Equation 4.1 is used to convert from fluxes to magnitudes to get the apparent magnitude in the AB magnitude system in the same particular filter with transmission curve $T(\lambda)$.

So far, the emission line contribution has not been taken into account. Here we present a simple method to model this contribution. For the purpose of this work it is not relevant the accuracy of the emission line model since the focus is on the optimization of the method. Each emission line is modeled with a Gaussian function, with its mean value being the wavelength of the emission line and its standard deviation (in km/s) is estimated using the Tully-Fisher relation (Tully & Fisher, 1977):

$$\log_{10}(\sigma_{el}) = ((-0.10 + 0.01 \cdot z_{obs}) \cdot (M_{0.1r} - 5 \log_{10} h - 0.3) - 0.05 \cdot z_{obs}) \quad (4.11)$$

The Gaussian function representing the emission line is extinguished, *redshifted* and normalized using the flux of the emission line given in the input catalogue.

If the science use case requires it, we can include the Milky Way extinction¹⁶ (see section 4.3.4) by applying the extinction law correction (which is a also function of λ) to the normalized and intrinsically extinguished spectral energy distribution, $SED_{obs}^{ext,norm}(\lambda)$. The extinction due to the Milky Way is applied in the same way as the galaxy intrinsic extinction. There are two main measurements of the Milky Way extinction values, those of Schlegel et al., 1998 and those of Planck Collaboration et al., 2014¹⁷.

¹⁶It refers to the dimming of light from stars and other astronomical objects due to interstellar dust in the Milky Way galaxy, also known as galactic extinction.

¹⁷<https://www.cosmos.esa.int/web/planck/home/>

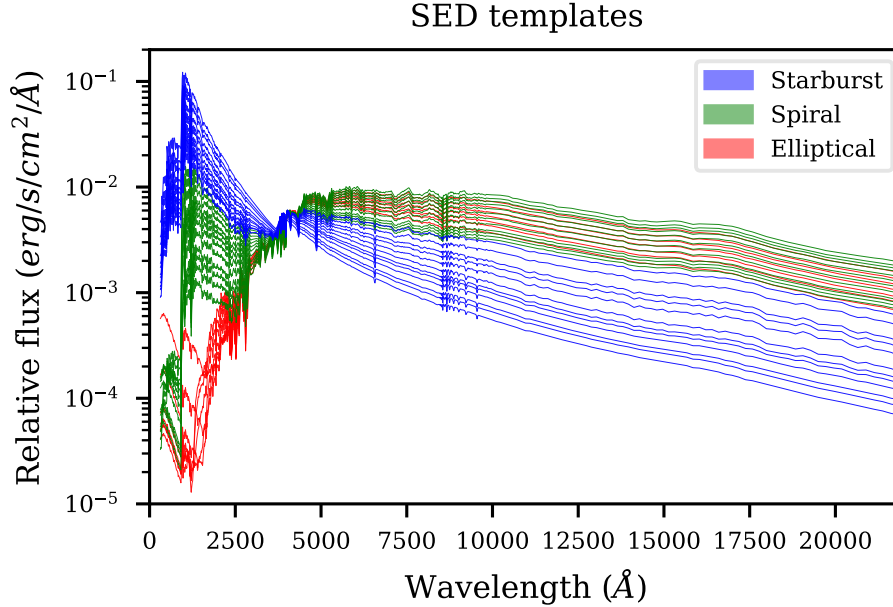


Figure 4.2: The 31 COSMOS SED templates used for the Euclid Flagship mock galaxy catalog. The 7 elliptical models are shown in red. Green lines correspond to 11 spirals and 1 lenticular, and blue lines represent the 12 Starburst templates.

4.3 Input data

In this section we describe the particular data used within SciPIC pipelines to estimate the flux for the Euclid Flagship and PAUS mock galaxy catalogs: the COSMOS2015¹⁸ spectral energy distributions and extinction laws, the different filters and the model for the Milky Way extinction curve and its strength.

4.3.1 Galaxy templates and extinction laws

The galaxy SED templates used are the COSMOS templates from Ilbert et al., 2009. There are in total 31 different SEDs corresponding to different kind of galaxies (starburst, spiral and elliptical). Figure 4.2 shows all the COSMOS2015 spectral energy distributions.

Galaxy intrinsic extinction is also taken into account when estimating the flux. In particular, five different extinction laws from Ilbert et al., 2009 are used. The galaxy SED or template is modified with a factor coming from the corresponding extinction law for each galaxy as expressed in Equations 4.6 and 4.7, where $ext_curve[0]$ means no extinction and ext_curve refers to the rest of the extinction curves. The different extinction laws used are shown in Figure 4.3.

The SciPIC flux pipeline assigns to each galaxy an $sed_template$, an extinction law (ext_curve) and its strength (colour excess, $E(B-V)$). One or more extinction laws can be applied to the COSMOS SED templates (see Table 4.1). The total combination of COSMOS SEDs and extinction laws is 47. A $sed_template$ refers to a linear interpolation

¹⁸A comprehensive survey of the distant universe that combines data from multiple telescopes and surveys for over half a million objects in the two-square-degree COSMOS field.

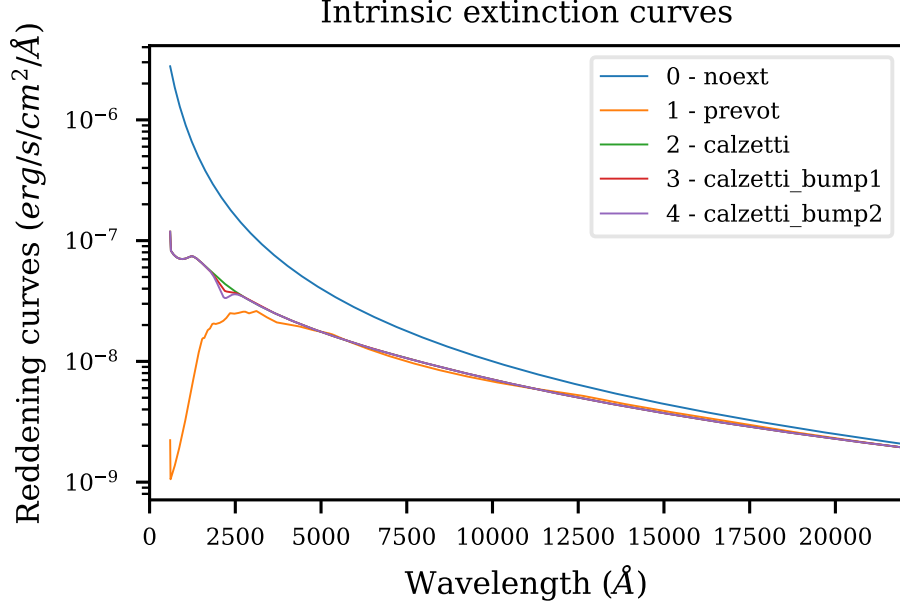


Figure 4.3: Different extinction laws used for the Euclid Flagship mock galaxy catalog.

COSMOS SED	Extinction curve
0-6	0
7-9	0
10-18	1
19-22	1
23-30	2
23-30	3
23-30	4

Table 4.1: Possible combinations of COSMOS SEDs and the different extinction laws.

between two COSMOS SEDs. For example, if a galaxy has $sed_template = 10.3$, then $sed_template = 0.7 \cdot SED[10] + 0.3 \cdot SED[11]$.

4.3.2 Filters

The Euclid Flagship mock galaxy catalog provides fluxes in more than 30 different filters. In addition of the four different filters of the Euclid mission (see Figure 4.4), simulations include fluxes for other surveys, such as DES or Rubin-LSST, in order to compute accurate photometric redshifts to achieve their science goals. The PAU Camera mounts a system of 40 narrow-band filters, consecutive in wavelength and with effective wavelengths separated by 10nm. The narrow-band system spans the wavelength range from 450nm to 850nm. The camera is also equipped with 6 broad-band filters, mimicking the u, g, r, i, z, Y filter set used in the DES camera. For this dissertation we show the results for 6 different narrow band PAU filters covering all its wavelength range (see Figure 4.5). We have tested that selecting another set of filters does not modify the results and the conclusions are equivalent.

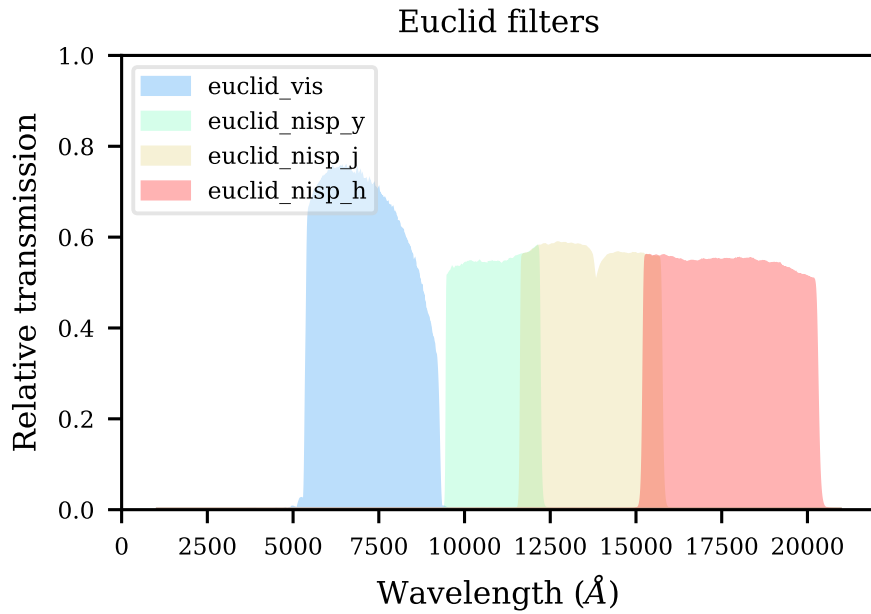


Figure 4.4: Euclid Standard Bandpasses for the Euclid experiment filters. The bandpasses represent the total system throughput.

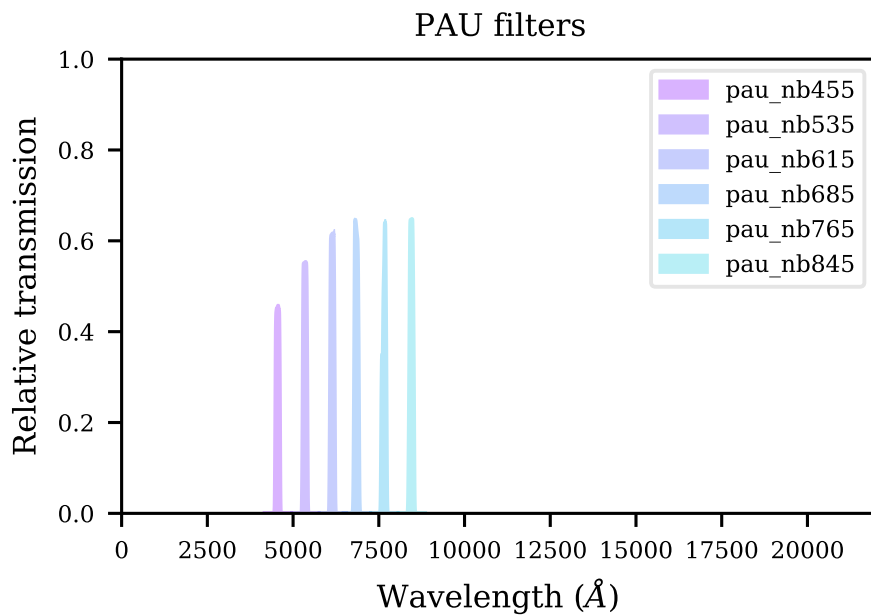


Figure 4.5: The 6 different narrow band PAU filters tested for this chapter: nb455, nb535, nb615, nb685, nb765 and nb845. In the naming convention used, «nb» stands for narrow-band, and the number corresponds to the central lambda of the filter.

Extinction line	Wavelength	
	Vacuum	Air
H α	6564.6	6562.8
H β	4862.7	4861.3
OII (1)	3727.0	3726.0
OII (2)	3729.8	3728.8
OIII (1)	5008.2	5006.8
OIII (2)	4960.2	4958.9
NII (1)	6585.2	6583.4
NII (2)	6549.8	6548.0
SII (1)	6718.2	6716.4
SII (2)	6732.6	6730.8

Table 4.2: Emission lines with their corresponding wavelengths both in air and in vacuum, named after the element that is emitting and the transition energy or ionization state of the atom.

4.3.3 Emission lines

Simulated galaxy spectra also contains the contribution of some of the most prominent emission lines. The input catalog specifies the logarithm of the observed emission line flux in $erg \cdot cm^{-2} \cdot s^{-1}$ including already the intrinsic extinction. Table 4.2 shows the ten emission lines we compute with their corresponding wavelengths both in air and in vacuum¹⁹. These are the strongest, and therefore most important to model.

4.3.4 Milky Way extinction

In order to simulate fluxes as observed in a given position on the sky, we need to include the Milky Way extinction at the position of each galaxy. This effect, also known as interstellar extinction, refers to the absorption and scattering of light as it travels through the dusty interstellar medium within our own galaxy, the Milky Way. This dust, composed mainly of tiny grains of carbon, silicon, and other elements, effectively dims and reddens the light from objects behind it, making them fainter and redder than they truly are. We use the latest release²⁰ from Planck Collaboration et al. (2014) to obtain the colour excess $E(B-V)$ that we apply to our simulations. See Figure 4.6 for the corresponding extinction value in galactic coordinates of our Milky Way.

The O’Donnell (1994) extinction curve is used to model the wavelength dependence of the Milky Way extinction. In particular we use the `extinction`²¹ Python library, which contains an implementation for this function. Figure 4.7 shows the normalized shape of the Milky Way extinction curve.

¹⁹This distinction is made to accomodate both terrestrial observatories (which are affected by the air in the atmosphere) and space-borne ones (which operate in vacuum).

²⁰https://irsa.ipac.caltech.edu/data/Planck/release_1/all-sky-maps/maps/HFI_CompMap_ThermalDustModel_2048_R1.20.fits

²¹<https://extinction.readthedocs.io/>

Milky way dust map

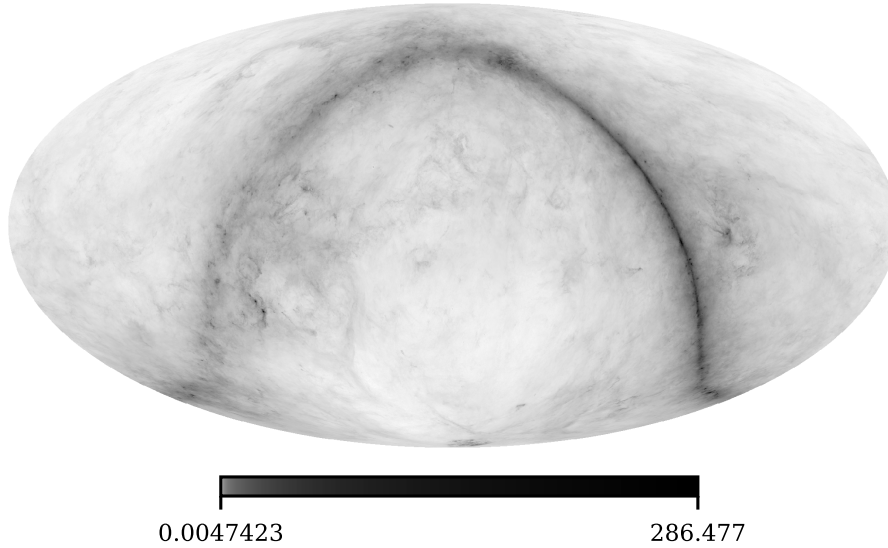


Figure 4.6: The colour excess values from the latest release of Planck Collaboration et al. (2014) in ecliptic coordinates.

Milky way dust extinction

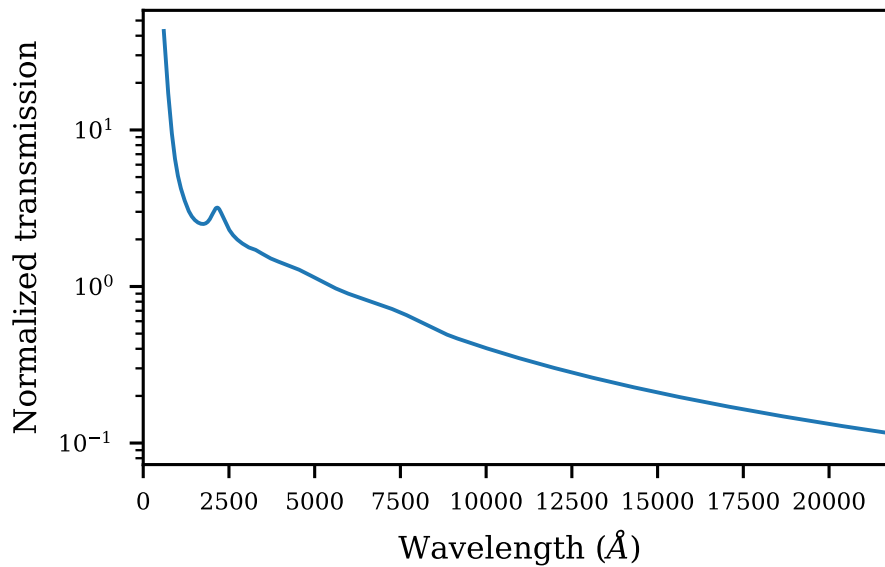


Figure 4.7: O'Donnell extinction function with $A_V = 1$ and $R_V = 3.1$.

	Range	Bin width	
		Euclid	PAU
sed template	[0, 30]	0.5	0.5
ext curve	[0, 4]	1	1
E(B-V)	[0, 0.5]	0.01	0.01
true redshift	[0.0, 2.3]	0.02	0.005

Table 4.3: Binning configuration for each parameter and filter set.

4.4 SciPIC approximated approach

Given an SED and a transmission function, galaxy fluxes can be computed exactly solving the integrals in Equation 4.9. However, when this method has to be applied to a large galaxy catalog, it becomes prohibitive in terms of computation time. As introduced in section 3.1, and as a reference, estimating the flux of one billion galaxies in a single band can more than 13 months²² in a single core. With the objective of generating simulated galaxy fluxes in a reasonable time while minimizing accuracy loss, we have developed and optimized a method to estimate an approximated value by splitting its computation in several steps using the following expression:

$$flux = (flux_{SED} + flux_{EL}) \cdot factor_{MW} \quad (4.12)$$

where $flux_{SED}$ is the SED flux contribution, $flux_{EL}$ is the emission lines flux contribution and $factor_{MW}$ is a factor applied to approximate the Milky Way extinction. The validity and accuracy of the method will be show in section 4.5.

4.4.1 SED flux contribution

The value in this first step is generated by an interpolation function for each filter that, given the SED, the extinction curve and its strength ($E(B-V)$), and the observed redshift, yields an interpolated continuum flux which already includes the intrinsic extinction. This result is then normalized using an absolute magnitude and the true redshift of the galaxy. This function is implemented using the `RegularGridInterpolator`²³ class from the `scipy` python package²⁴.

The interpolation function is built from a grid of coordinates for each dimension and the corresponding multidimensional matrix of flux values. Each filter of each survey has its own corresponding interpolation function, as the binning for its parameter space has been optimized to fulfill the accuracy requirements. In particular, both PAU and Euclid surveys require that the error in AB magnitude introduced by the approximation has to be smaller than 1%. Table 4.3 show the details about the binning configuration of each filter set.

²²36 milliseconds per flux per band on average, based on actual measures of an optimized implementation of section 4.2 method.

²³<https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.RegularGridInterpolator.html>

²⁴<https://scipy.org>

Each resulting matrix of fluxes has 1.8 million entries, in the case of Euclid filters, or 7.1 million entries for PAU ones. All these values are computed using the standard approach described in section 4.2. Even though the computation is also done in parallel on top of the PIC Big Data platform using Apache Spark, it still takes about 3 hours to generate all the matrices for the interpolation functions. In order to optimize this process and be able to reuse them, they are stored and cached on disk. Whenever an interpolation function is needed, it is first looked for in the cache, and only computed if it is not present. Changes in its parameters such as the binning of the filter transmission also trigger its computation to account for the new values.

By construction, the interpolation function yields the exact computed flux when the provided coordinates match any of the grid vertices, otherwise it returns an interpolated flux in which the error introduced by the approximation increases the farther the coordinates are from a vertex. Figures 4.8, 4.9 and 4.10 show this behaviour and characterize the deviance introduced by the interpolation function, after carefully adjusting the binning spacing. In this particular case, we never interpolate over the extinction curves, so no error is introduced there. The accumulated error combining the effect of all dimensions is shown in Figure 4.11, and it is always below the limit imposed by the surveys (less than 0.01 difference in *AB magnitude*).

4.4.2 Emission line contribution

An interpolation matrix is also used to include the flux contribution of the different emission lines. Each line is modeled using a normalized Gaussian function: the mean value is the wavelength of the line and the standard deviation (sigma) depends on the absolute magnitude and the observed redshift, as shown in Equation 4.11.

The interpolation matrices are built as described in section 4.4.1 but, opposite to SED interpolation matrices, they are not cached due to their reduced dimensions and the very little time it takes to compute them. In this case, the function yields a *not normalized* contribution for each emission line and filter using as input the absolute magnitude and the observed redshift. Each emission line flux contribution is normalized using the respective emission line absolute flux from the input catalog, and then added all together to the interpolated flux to obtain ($flux^{interp} + flux^{el}$).

The error in magnitude introduced in this step by the interpolation cannot be measured at construction time, as was done in section 4.4.1. Instead, it has been characterized using a sample of synthetic galaxies (see section 4.5 for more details).

4.4.3 Milky Way extinction contribution

The last step to estimate the galaxy flux as would be observed by a telescope is to apply the extinction due to how the light attenuates when passing through our Milky Way. The colour excess ($E(B-V)$) correction value is given by the map presented in section 4.3.4. The extinction (or reddening) law is assumed to be the function from O'Donnell (1994). As already mentioned, we use the `extinction` python library. We follow the work of Cox (2000) and Bolzonella et al. (2000) to estimate the observed flux after passing through the Milky Way:

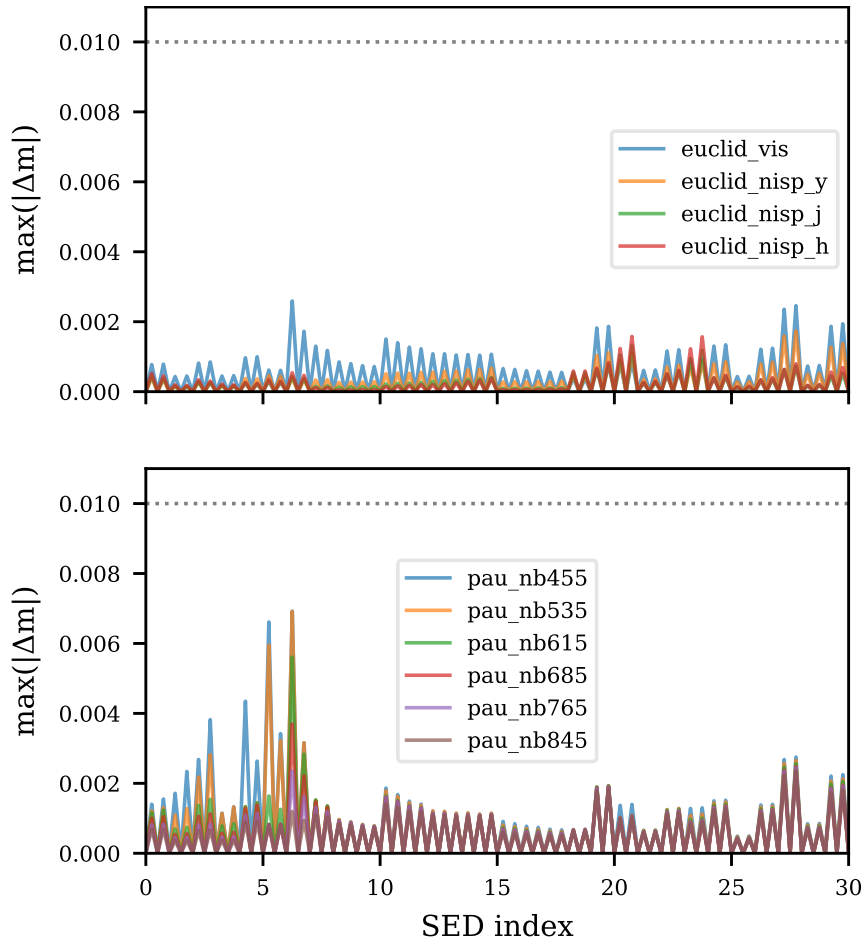


Figure 4.8: Maximum absolute magnitude difference introduced by interpolating across adjacent SED templates. In order to keep the error below the acceptable threshold, we had to double the binning along the SED index axis.

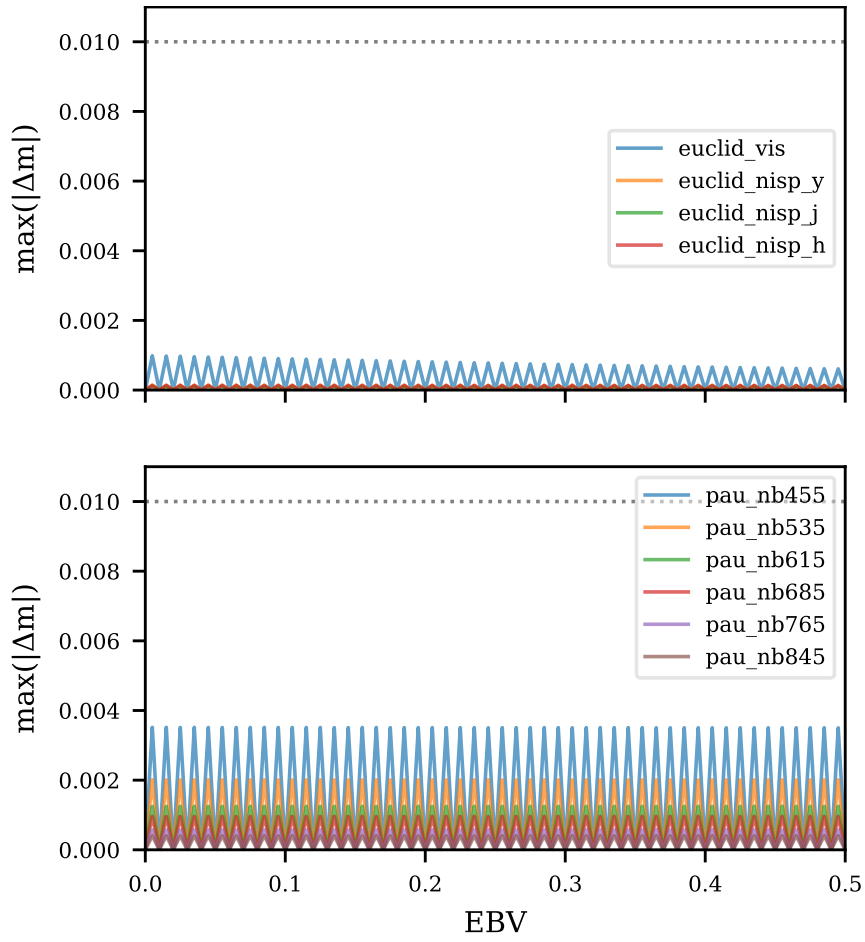


Figure 4.9: Maximum absolute magnitude difference introduced by interpolating across $E(B-V)$ bins.

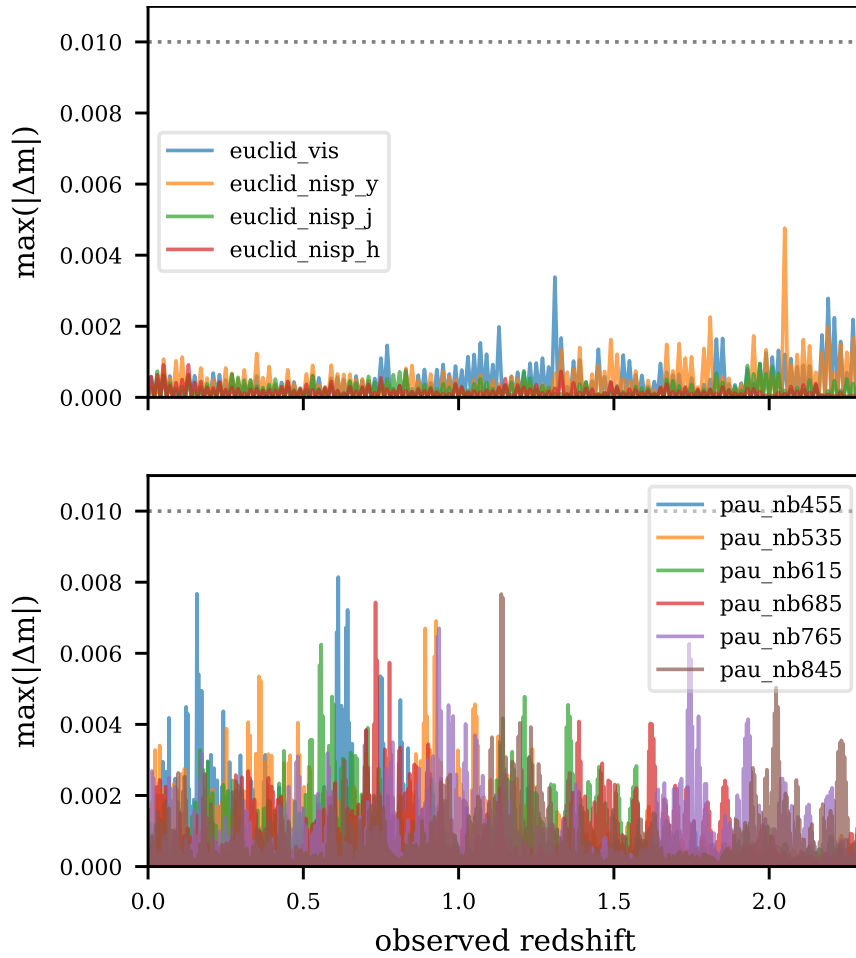


Figure 4.10: Maximum absolute magnitude difference introduced by interpolating across redshift bins.

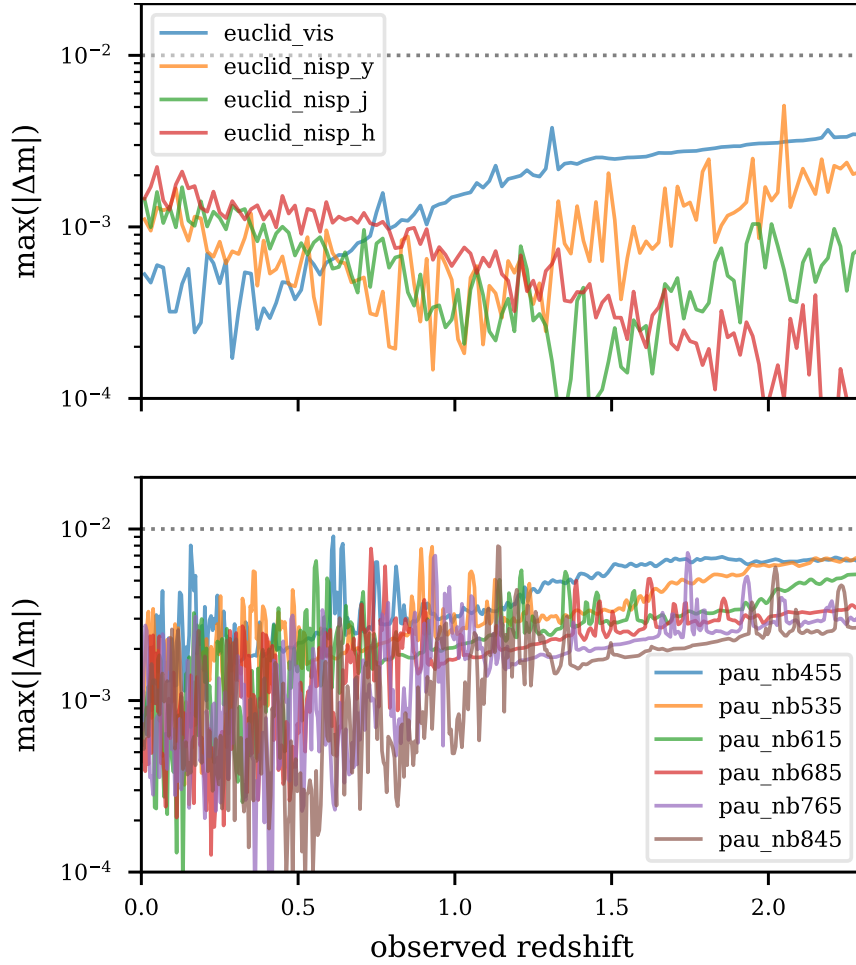


Figure 4.11: Maximum absolute magnitude difference introduced by interpolating across all dimensions (except extinction curve) as a function of redshift. After carefully defining the binning of each dimension for each filter, the delta in magnitude is always below the threshold for any combination of parameters.

$$F_o(\lambda) = F_i(\lambda) \cdot 10^{-0.4 \cdot A(\lambda)} \quad (4.13)$$

where $F_o(\lambda)$ is the observed flux as a function of λ , $F_i(\lambda)$ is the flux before passing through the Milky Way and $A(\lambda)$ is the extinction at a wavelength λ , which is related to the colour excess, $E(B-V)$, and with the reddening curve $k(\lambda)$ by:

$$A(\lambda) = k(\lambda)E(B - V) = k(\lambda)\frac{A_V}{R} \quad (4.14)$$

where R is a constant.

We approximate Equation 4.13 assuming the extinction in each filter is a multiplicative factor. In other words, we consider that the extinction curve (in Figure 4.7) is flat across the filter (in Figures 4.4 and 4.5). The following expression estimates this average value within each filter:

$$\overline{odonne\ell} = \frac{\int SED_{odonne\ell}^{norm}(\lambda) \cdot T(\lambda) \cdot \frac{\lambda}{c} d\lambda}{\int T(\lambda) \cdot \frac{d\lambda}{\lambda}} \quad (4.15)$$

where $SED_{odonne\ell}^{norm}$ is the chosen function to model the extinction curve of the Milky Way and $T(\lambda)$ is the filter transmission curve. The corresponding colour excess $E(B - V)$ at the position of each galaxy is retrieved from Planck Collaboration et al. (2014) dust maps and used to derive the final extinction factor:

$$factor^{MW} = 10^{\left(E(B-V) \cdot \left(\frac{r_v}{A_v}\right) \cdot \overline{odonne\ell} \cdot 0.4\right)} \quad (4.16)$$

that is finally multiplied to the galaxy flux to get the desired result (See Equation 4.12).

4.5 Results

The method presented in this chapter needs to be fast as well as accurate to fulfill the scientific requirements of both Euclid and PAUS simulations. In order to measure both, a mock catalog of 2.21 million synthetic galaxies was provided as input and the galaxy fluxes were computed using both the standard method and the approximated method. In particular, for these results we used the broad band filters from Euclid and a selection of narrow band filters from the PAU survey that covered its whole range of wavelengths, as described in section 4.3.2.

With respect of the accuracy, the following set of plots displays the difference between the fluxes computed using both approaches. Figure 4.12 shows the magnitude difference in the SED flux between both methods. This figure is the equivalent to Figure 4.11 but using the sample of galaxies, instead of blindly testing the whole space of parameters. As such, the errors in Figure 4.12 are less than or equal to those in Figure 4.11, and always below the required threshold.

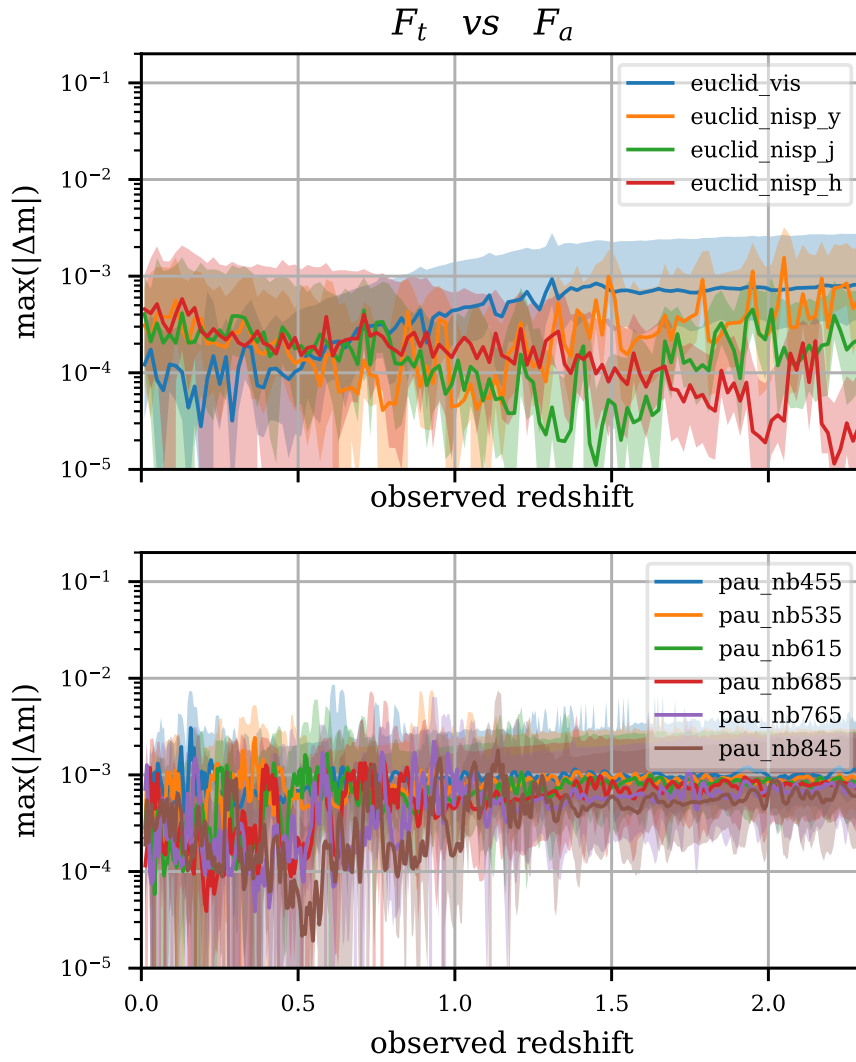


Figure 4.12: Maximum magnitude difference introduced by the SED flux approximation. The solid line indicates the average delta, the top shaded area indicates the maximum, and the shaded bottom indicates the standard deviance, for each redshift bin.

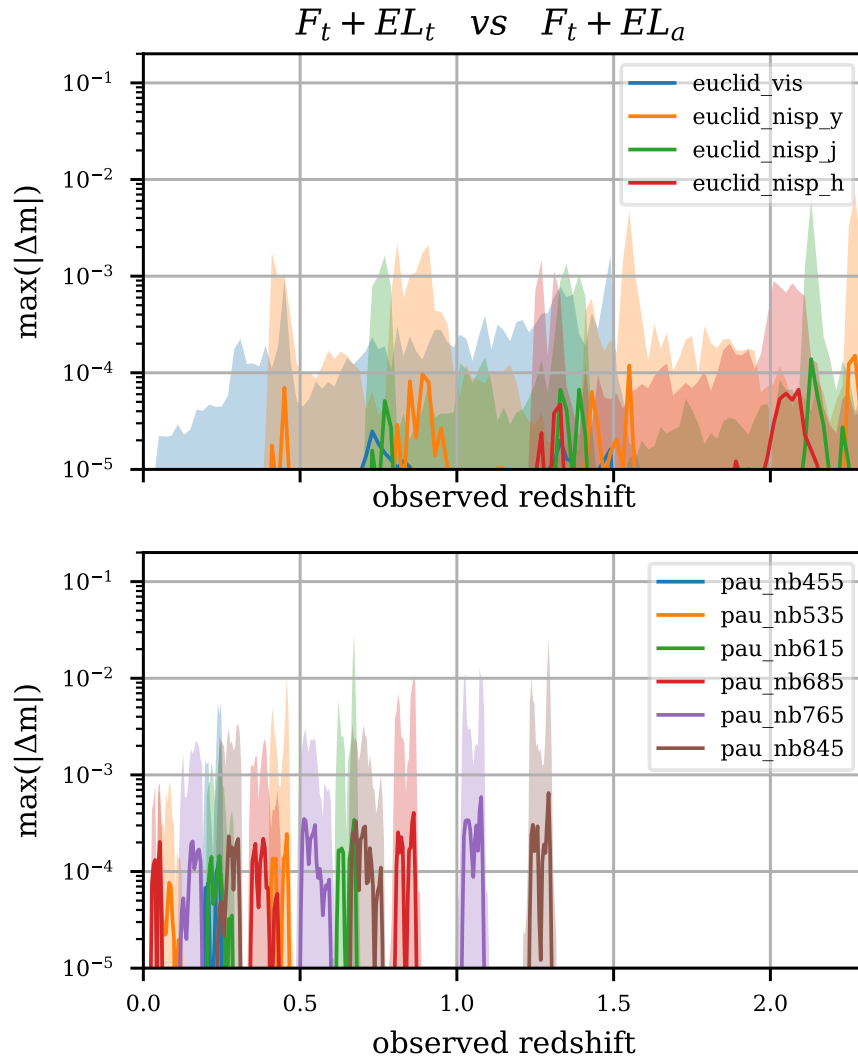


Figure 4.13: Maximum magnitude difference introduced by the emission lines flux approximation. The solid line indicates the average delta, the top shaded area indicates the maximum, and the shaded bottom indicates the standard deviance, for each redshift bin.

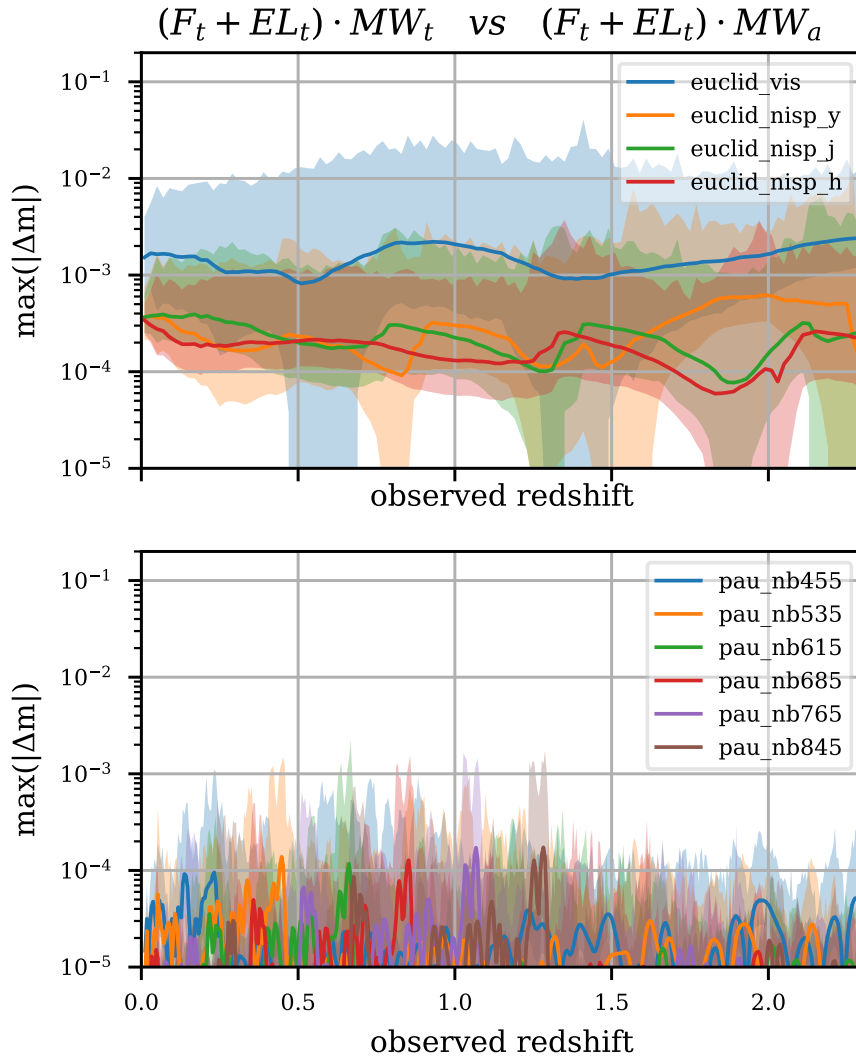


Figure 4.14: Maximum magnitude difference introduced by the Milky way local extinction using a flat approximation. The solid line indicates the average delta, the top shaded area indicates the maximum, and the shaded bottom indicates the standard deviance, for each redshift bin.

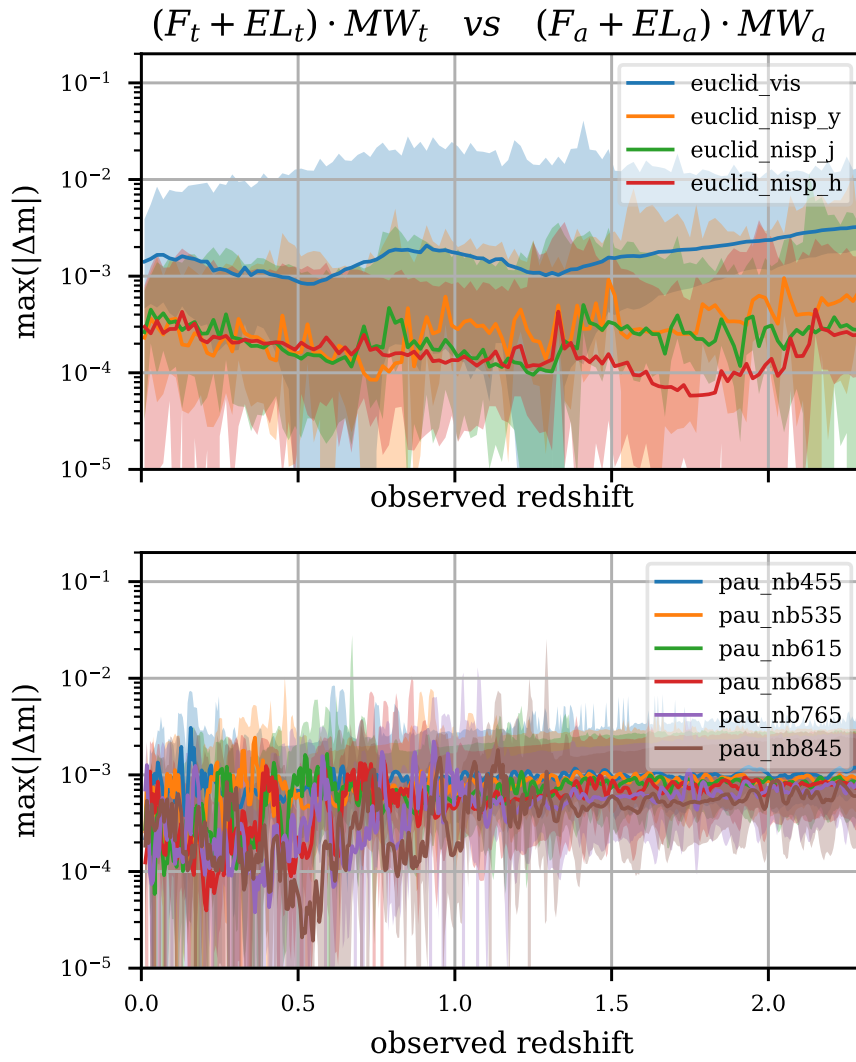


Figure 4.15: Maximum magnitude difference introduced by the combination of all approximations. The solid line indicates the average delta, the top shaded area indicates the maximum, and the shaded bottom indicates the standard deviance, for each redshift bin.

	SED	el	M.W.	Combined
euclid_nisp_h	100.000	100.000	100.000	100.000
euclid_nisp_j	100.000	100.000	100.000	99.999
euclid_nisp_y	100.000	100.000	99.999	99.999
euclid_vis	100.000	100.000	99.848	99.863
pau_nb455	100.000	100.000	100.000	99.999
pau_nb535	100.000	99.999	100.000	99.999
pau_nb615	100.000	99.999	100.000	99.999
pau_nb685	100.000	99.999	100.000	99.999
pau_nb765	100.000	99.999	100.000	99.999
pau_nb845	100.000	99.999	100.000	99.999

Table 4.4: Percentage of galaxies that fulfil the accuracy requirement (an AB magnitude difference below 0.01) for each approximated step (from left to right: SED flux interpolation, emission line flux interpolation, Milky Way dust extinction), as well as for the combination of all steps.

Figure 4.13 shows the magnitude difference in the galaxy flux introduced by the emission lines contribution approximation. For the Euclid bands all fluxes are below the required threshold. For all the PAUS bands, the threshold is exceeded only for 0.00014% of the galaxies on average.

Figure 4.14 shows the magnitude difference introduced by the flat approximation of the local Milky Way extinction. This is the step which introduces most of the error, and the one that is most difficult to optimize. The reason is that there is no binning that can be adjusted here, but a constant factor per filter. In particular, very broad band filters are affected the most as a flat approximation struggles to fit the O’Donnell function across the wavelength range. The effect is clearly noticeable in the widest of the filters (Euclid VIS) in which 0.15% of the galaxies exceed the delta threshold. The rest of the bands are unaffected. If needed, this could be reduced by splitting affected filters in several subfilters, computing the flux for each of them and adding all up.

Figure 4.15 shows the result of combining the error contribution from the three steps: the SED flux, the emission lines and the Milky Way extinction. In the end, only 0.137% of galaxies in the Euclid VIS band exceed the error in magnitude, while for the rest of the bands is less than 0.0003%. Table 4.4 summarizes all these numbers.

Performance wise, the standard method takes around 35.5 milliseconds on average to produce a flux (including the emission lines and the Milky Way extinction). The same results can be delivered in 0.047 milliseconds (about 750 times faster) using the approximated approach. Timings are stable even when across filters, as shown in Table 4.5.

Finally, note that the accuracy of this method has also been tested by an independent and completely different external implementation led by the Euclid OU-SIM team. OU-SIM is the organizational unit responsible for delivering simulations for all Euclid’s purposes, such as VIS data, NIR photometry, NIR spectroscopy and external ground based photometry. In their approach, they reconstruct the full SED and then compute the flux using integrals. However, as (1) the number of bands they need is very limited, (2) the computation is distributed to multiple data processing centers around the world using

Band	Processing time (in ms)		
	Standard	Approximated	Speedup
pau_nb455	35.68	0.0478	745
pau_nb535	35.63	0.0477	745
pau_nb615	35.61	0.0477	746
pau_nb685	35.60	0.0476	746
pau_nb765	35.63	0.0478	744
pau_nb845	34.61	0.0467	740
euclid_nisp_h	35.6	0.0472	753
euclid_nisp_j	35.7	0.0471	756
euclid_nisp_y	35.2	0.0468	752
euclid_vis	35.5	0.0476	745

Table 4.5: Average flux computation time per galaxy and band for both the standard and the approximated method. The computation has been performed on top of the PIC Big Data platform comprised of 12 nodes, where each node is equipped with an AMD Threadripper 1920X @ 2.2 GHz CPU, 128 GiB RAM DDR4 @ 2666 MHz, 12 x 3 TiB SATA HDDs and 2 x 1 TiB NVMe SSDs.

thousands of CPUs and, (3) they do not simulate as much area, their computation takes several weeks but is still manageable.

4.6 Conclusions

This chapter describes the method that has been designed and developed for estimating the observed galaxy fluxes for both the Euclid and PAU survey simulations. In particular, the Euclid Flagship mock galaxy catalog with its 9 billion objects and more than 200 properties, including fluxes in 30 different bands, is the largest synthetic galaxy catalog to be produced to date. The traditional approach based on solving several integrals per observed flux is not adequate when the number of fluxes to be computed is very large, as the total processing time becomes prohibitive. As a reference, producing the fluxes for the Euclid Flagship mock galaxy catalog using the standard approach based on integrals would take about 300 years in a single CPU, at about 36 milliseconds per flux on average, which is totally unfeasible.

Compared to the standard method, this approach delivers a staggering 750x boost in speedup, achieving an average of 0.047 milliseconds per flux on average (all within the same computing environment) while still fulfilling the accuracy requirements of both Euclid and PAU surveys. Combining the speed of the approximated method with the scaling capabilities of the Apache Spark framework it has been developed onto, the fluxes for the Euclid Flagship mock galaxy catalog can be delivered in less than 12 hours on a 300-core computing cluster. Complete source code for this contribution can be found at https://github.com/ptallada/scipic_fluxes.

In conclusion, if it was not for the method and calibration presented in this chapter, it would not have been possible to produce neither the Euclid Flagship galaxy mocks nor the PAUS simulated catalogs.

Chapter 5

Conclusions and future work

Physicists and astronomers have radically changed their approach to study the universe, its content and evolution, over the past decades. The need for advanced computing techniques has become the key to deal with the outstanding data volume collected by the modern automatized telescopes and high sensitive instruments. The infrastructures that support their operation also have to keep scaling up their capacity and efficiency to meet the requirements of the experiments. As the complexity of those infrastructures increase, scientists alone can no longer be in charge of them. This responsibility has been progressively delegated to dedicated facilities run by specialized personnel, such as data processing centers and research software engineers.

In parallel, the ability to extract the scientific insights from the multitude of information collected by those instruments has become a multidisciplinary work between mechanical and electronic engineers, physicists, astronomers, mathematicians, computer scientists, and software engineers. Communication between scientific teams and their technical counterparts has become critical for the success of their projects. In the last decade, a specialized role called Research Software Engineer (RSE) has arisen to commit to this duty (Woolston, 2022). However, after 10 years of the role being defined, there are still many handicaps that need to be addressed. In particular, how to train, integrate and provide a career progression for that role (Cosden et al., 2022), and how software needs to be truly recognized as a fundamental research output on its own (Jay et al., 2020).

This thesis focuses on my work as a Research Software Engineer (RSE) at the Department of Cosmology and Astrophysics at PIC. It delves into my efforts to address various data management challenges for large volumes of structured data using efficient and diverse approaches, all within the scientific computing domain, with a specific focus on cosmology projects. Notably, my contributions have significantly impacted two of the most prominent galaxy redshift surveys today: PAUS and Euclid. These efforts have been recognized through publications in peer-reviewed journals (Tonello et al., 2019; Tallada et al., 2020; Tallada-Crespí et al., 2023).

In the following sections I will summarize the main conclusions of each chapter and I will conclude with the lines of work planned and in progress.

PAU Survey data management

Based on the experience from preceding surveys such as SDSS or DES, PAUS acknowledged from the very beginning that the design of a proper data management architecture¹ was key to achieving their scientific objectives. This architecture had to cover all steps of data processing, starting from the transfer of images acquired by PAUCam, handling all their processing by multiple analysis pipelines and ending with the archival and distribution of the results.

Given that no open global solution was existing, an original management was needed to fulfill the project requirements. In this regard, the decision of putting PAUdb, a relational database, at the center of all the data management architecture was a bold move at that time, but one that delivered great advantages: (1) it ensures the consistency, integrity and traceability of all the metadata related to the operations of PAUS and, (2) through the ORM layer², it also provides very efficient metadata access for all the processing pipelines.

The automatic transfer procedure of images from the WHT, albeit simple, fulfilled every PAUS project need. The downloading process finalizes in a few hours, with outstanding reliability during normal operations, and full recovery in the rare cases of failure.

The orchestration of all analysis jobs on top of PIC's HTC infrastructure is carried out by a custom developed tool called BT³. It tracks the configuration, execution status and dependencies of every job to maximize throughput and ensure that the *nightly* pipeline results are available in time for the next observing night. It also monitors the quality of data products and guarantees the reproducibility of any results. Paired with the high level of parallelization of the pipelines in independent jobs, the number of computing nodes and the distributed file system available at PIC, BT also enables the reprocessing of all survey observations within a few days.

A suite of web services has also been developed on top of PAUdb to facilitate seamless access, retrieval, and analysis of PAUS data and metadata. Together with the automated data management services, they enable to comply with the Open Science and Open Data principles, guaranteeing the reproducibility of scientific results. These web services have been met with positive feedback from collaborators, who have also commended their user-friendliness and the richness of the information they provide.

Finally, for the distribution of results in the form of catalogued data, a web portal called CosmoHub was implemented. Several prototypes were developed on top of PAUdb that later became the seed of a much larger effort to develop a truly scalable platform for handling massive structured datasets of several interoperable data sources, shared with the scientific community.

Summarizing, my contributions to PAUS have been instrumental in enabling the project's scientific achievements. Hence, in recognition of my role in PAUS's success, I have been granted the status of PAUS *builder*. This distinction allows me to sign as co-author on all PAUS-related publications⁴. To date, PAUS has produced 30 articles in high-impact

¹highly automated, reliable and fully traceable

²Source code available at <https://github.com/ptallada/paudm-model>.

³Source code available at <https://github.com/ptallada/brownthrower>.

⁴<https://pausurvey.org/pausurvey/publications/>

peer-reviewed journals and 18 communications with written proceedings, demonstrating its significant impact within the scientific community.

CosmoHub

Stemming from the necessity to share simulated data between MICE and PAUS, and later to distribute PAUS results to the public community, CosmoHub was designed as a generic platform to enable interactive exploration and distribution of very large structured datasets, with a special focus on massive cosmological catalogs.

After careful evaluation of numerous alternatives, the decision to select Apache Hadoop and Apache Hive as the data warehouse platform and to construct CosmoHub upon this foundation has proven to be a resounding success. This robust, highly performant, and user-friendly platform has empowered CosmoHub with a suite of impressive features that have significantly enhanced its user-facing functionalities. Also, its unrivaled scalability has allowed to keep response times low at all times in spite of the constant increase in users and data volume.

Usability was also a major goal in CosmoHub's design. The implementation of a guided interface has been key in bridging the gap for researchers with limited SQL expertise. Furthermore, the development and integration of a comprehensive suite of UDFs⁵ has considerably expanded and simplified the platform's capabilities, catering to those specific yet prevalent use cases. CosmoHub also boasts a variety of visualization options, enabling users to effortlessly explore and gain insights from any dataset within seconds. Additionally, customized subsets can be materialized and downloaded in a matter of minutes. By embracing the most prevalent data formats⁶, including *de-facto* standards like CSV, FITS⁷, ASDF and Parquet, CosmoHub seamlessly integrates with the broader scientific toolkit, fostering smooth data exchange and analysis.

Since the redesign and migration to Hadoop was completed in October 2016, CosmoHub became the first project to apply Hadoop to the analysis and distribution of large cosmological datasets. It has kept growing at a steady pace and it is now providing a useful service to the scientific community to many relevant projects, even becoming the official data distribution service for several of them such as PAUS, MICE and the Euclid Flagship mock galaxy simulations. In 2020, PIC became an affiliated Gaia data center, and CosmoHub step up to host a mirror of every major Gaia public data release since. In 2021, CosmoHub was also accepted in re3data, a global registry of research data repositories from different academic disciplines.

In all these years, CosmoHub has grown significantly in terms of data storage and user base while keeping the response times stable. This is a summary of the most important numbers in these areas nowadays:

Data storage The amount of data hosted by CosmoHub has grown steadily over the years, reaching about 60 TB of catalogued data by the end of 2023. This vast

⁵Their full source code can be found at <https://github.com/ptallada/pic-hadoop-udf>.

⁶See <https://github.com/ptallada/cosmohub-api/tree/master/cosmohub/api/io/format> for the corresponding source code.

⁷See <https://github.com/ptallada/recarrayserde> for the complete source code of the Hive FITS binary format serializer/deserializer.

repository of cosmological data is a testament to the platform’s growing popularity and utility within the scientific community.

User base CosmoHub has over 1800 registered users, 150 of them are active at least once a month, demonstrating its widespread adoption among researchers working in various fields of cosmology. These users actively engage with the platform, creating over 50,000 interactive plots and downloading over 20,000 custom catalogs to date.

Response times CosmoHub has been specifically designed to prioritize fast response times, particularly for interactive plots, ensuring a seamless user experience while exploring large datasets. For interactive data exploration, 67% of queries are completed within 30 seconds, and 97% are finished within 2 minutes. Similarly, for custom catalog generation, 71% of catalogs are generated within 3 minutes, and 97% are completed within 30 minutes.

Summing up, these impressive growth metrics and performance indicators underscore CosmoHub’s position as a valuable resource for cosmological research. The platform’s ability to handle large datasets, accommodate a diverse user base, and deliver fast query response times has undoubtedly contributed to its success and widespread adoption within the scientific community. The full source code for the web backend is available at <https://github.com/ptallada/cosmohub-api>.

Finally, note that CosmoHub has been developed according to the same principles of data Findability, Accessibility, Interoperability and Reusability (FAIR, Wilkinson et al. 2016), promoted by the European Open Science Cloud (EOSC⁸) and strongly supported by the European Commission, to maximize the distribution and exploitation of scientific data generated by public funds.

SciPIC: fast and precise simulation of galaxy fluxes

Galaxy surveys need very complex cosmological simulations for their design, operation and science exploitation. One of the most important kinds of simulations are the mock galaxy catalogs, a sort of synthetic universe, containing billions of observable objects with an extensive list of properties.

SciPIC is a comprehensive multi-stage pipeline for the generation of synthetic galaxy catalogs that has been under heavy development at PIC for the last decade. Based on previous work by Carretero et al. (2015), it contains dozens of algorithms carefully tuned to be able to generate these massive catalogs as efficiently as possible.

One of those algorithms is responsible for the simulation of observed electromagnetic fluxes in multiple filters. This operation requires solving several integrals whose computing requirements, when combined with very large catalogs, end up being unaffordable.

Advised by an experienced team of cosmologists, I designed, implemented and optimized an approximated method. The main idea behind has been to separate the integrals into different components that can be computed separately using interpolations and other techniques. Although this approach is not mathematically equivalent, the balance between

⁸<https://eosc.eu/eosc-about/>

computing requirements and accuracy can be tuned with the proper calibration. The corresponding source code, both for the original integral-based approach, and the approximated one can be found at https://github.com/ptallada/scipic_fluxes.

The resulting method is able to perform up to 750 times faster than the approach based on integrals. It also has been calibrated to deliver accurate results within a restrictive 1% error threshold. Furthermore, when integrated with the parallel execution capabilities of Spark and the PIC Big Data platform, massive productions containing billions of objects can be delivered in a single day. For instance, it has been used extensively to produce mock galaxy catalogs for Euclid and PAUS. One of the most distinctive achievements has been the production of several releases of the Euclid Flagship mock galaxy catalog, the largest synthetic universe to date, with up to 5 billion objects and more than 400 properties each.

Future work

The successful collaboration between software engineering and cosmology and the fundamental role of RSE, both for the scientific projects supported and for developing and maintaining the core services of the PIC, is demonstrated by the work planned and funded for the next period.

PAU Survey has not taken any new data for several years now as the WHT is currently devoted full-time to another instrument⁹. Furthermore, it is a very real possibility that it will not be able to observe any more. Hence, work on the data management infrastructure has been cut down to the bare minimum, maintenance only. However, scientists are still optimizing and calibrating their analysis pipelines and reprocessing all the acquired data. Several publications are expected in 2024 based on these new results.

We are currently in the process of commissioning a new Hadoop cluster that will enable both CosmoHub and SciPIC to scale up on their capabilities. This new cluster is composed of 30 nodes summing 720 cores, 15 TiB of RAM, 60 TiB of NVMe and 4.3 PB of storage combined. Two thirds of this cluster are already deployed and will enter into production in February 2024. The last third of the cluster has yet to be awarded in a public tender, in a process that is suffering severe delays from the contracting office at IFAE.

This new cluster is being tested and deployed using a new custom Hadoop distribution that has been under development at PIC for the past year. While we have been using and open-source Hadoop distribution (Hortonworks Data Platform) for the last seven years, its acquisition by Cloudera and subsequent access restriction to both binaries and updates has left us little with no other choice. Our own distribution, named Shepherd, includes a broad set of components from the Hadoop ecosystem required by CosmoHub and SciPIC. In particular, it comes with a much needed Spark update that brings important performance improvements and also clears the path to upgrade to recent Python versions.

This distribution has been presented to several Spanish forums to foster the creation of a local community around it, including those data centers that already operate a Hadoop cluster such as the Centro de Supercomputación de Galicia (CESGA) and Consorci de Serveis Universitaris de Catalunya (CSUC).

⁹<https://www.ing.iac.es/astronomy/instruments/weave/weaveinst.html>

CosmoHub is already being tested on top of this new infrastructure. We are seeing notable performance improvements, both for the interactive visualizations and the custom catalog generation. This speedup comes not only from the increase in computing power, but also from enhancements in the query execution algorithms and the correction of multiple bugs.

The future of CosmoHub has already been defined in two separate lines of work: one focused on the expansion of its core features, and the other on broadening its scope to become a multi-messenger interoperable science portal. Work on the first line has already started in several design documents that cover all the planned features and how they can be implemented. From a technical point of view, these are the main features already planned:

- The implementation of a WebDAV translation layer on top of WebHDFS to facilitate large data transfers to/from the outside.
- The adoption of Arrow as the data communication protocol between CosmoHub's backend and its web frontend to improve the efficiency and density of interactive visualizations.
- The integration of CosmoHub's user accounts into the PIC centralized user database in order to simplify the maintenance and take advantage of features like a consistent authorization layer and the Single Sign-On.
- The migration to a new technological stack: the backend will be rewritten using FastAPI¹⁰, and the frontend will use Nuxt¹¹, PrimeVue¹² and UnoCSS¹³. The APIs will be documented using OpenAPI¹⁴ and AsyncAPI¹⁵.

The user-facing features we have planned are the following:

- There will be a public section for unauthenticated users to browse and interact with public catalogs. They will be able to interactively explore the datasets but will not be able to request custom catalogs.
- Sharing visualizations and custom catalogs will be as easy as distributing a link to the intended parties.
- Users will be able to upload their own catalogs, to analyze them privately or to put them public for everyone to see.
- CosmoHub will fully embrace the IVOA recommendations¹⁶ and implement several important VO protocols such as ADQL, UWS, TAP and VOTable. We are studying if we also implement VoSpace. This will allow users to interact with CosmoHub's data using standard tools and libraries like TOPCAT¹⁷, astroquery¹⁸ or pyvo¹⁹.

¹⁰<https://fastapi.tiangolo.com/>

¹¹<https://nuxt.com/>

¹²<https://primevue.org/>

¹³<https://unocss.dev/>

¹⁴<https://www.openapis.org/>

¹⁵<https://www.asyncapi.com/>

¹⁶<https://www.ivoa.net/documents/>

¹⁷<https://www.star.bris.ac.uk/~mbt/topcat/>

¹⁸<https://astroquery.readthedocs.io>

¹⁹<https://pyvo.readthedocs.io>

The evolution of CosmoHub into a multimessenger science platform is still in the very early stages of design. Several use cases are being explored, with the general idea to combine current cosmological datasets (which mostly cover the visible portion of the electromagnetic spectrum), with data from gravitational waves, gamma rays and neutrinos.

Regarding SciPIC, we are invested in several lines of work at this moment:

- A more refined calibration of the HOD algorithm through an iterative minimization approach, testing a broad range of parameters. This effort will result in the production of more accurate realizations of the number of satellite galaxies in our simulations.
- The implementation of a Spark version of TreeCorr²⁰, a very optimized code for the computation of 2-point and 3-point correlation functions. By leveraging the parallel processing capabilities of Spark, this development will make it possible to manage very large datasets in much shorter times. See Plaszczyński et al. (2021) for a similar approach.
- The development and integration of an n-dimensional pdf transfer code for the generation of survey-specific galaxy samples inspired by color-matching algorithms (Pitié & Dahyot, 2005) in photography and image manipulation.
- With respect to the simulation of observed galaxy electromagnetic fluxes, there are still several optimizations we can pursue, such as the computation of these fluxes directly from absolute color differences. This will help further reduce the computation requirements and/or increase the precision of the results.

In this regard, I want to emphasize that in the last two years, our team has successfully applied to several competitive funding calls to consolidate and expand these lines of work. In total, we have received 1.5 million euros until 2025, which clearly reflect the importance and the impact of our contributions. Here follows the calls and the amount we were granted:

- 1,000,000 euros from Plan Complementario, for the design, deployment and operation of a multi-messenger science platform.
- 400,000 euros from Equipamiento Científico y Técnico for the acquisition and deployment of a new Hadoop cluster to increase the performance and storage both for CosmoHub and SciPIC.
- 89,000 euros from Plan Nacional, for the development of new algorithms and techniques to improve the generation of massive synthetic mock galaxy catalogs for multiple surveys.

Finally, the role of RSEs in Spanish public research institutions is still in its early stages of development. The lack of a clear role hinders their ability to secure funding and receive proper recognition for their contributions. Existing funding opportunities often prioritize hardware or postdoctoral positions, leaving RSEs with limited funding avenues. Moreover, the lack of a defined role can make it challenging for RSEs to secure authorship in research papers, diminishing their visibility. Ultimately, this ambiguity undermines the recognition

²⁰<https://rmjarvis.github.io/TreeCorr>

that RSEs rightly deserve for their fundamental role in scientific research. However, there are some initiatives underway to define a technological career for RSEs, but these are still incipient. The first openings for RSE positions are expected to be allocated in 2024. We are closely following the progress of these initiatives, as our own future is inextricably linked to their success.

References

- Abbott, T. M. C., Alarcon, A., Allam, S., Andersen, P., Andrade-Oliveira, F., Annis, J., Asorey, J., Avila, S., Bacon, D., ... Zuntz, J. (2019). Cosmological constraints from multiple probes in the dark energy survey. *Phys. Rev. Lett.*, *122*, 171301. <https://doi.org/10.1103/PhysRevLett.122.171301>
- Abolfathi, B., & et al. (2018). The Fourteenth Data Release of the Sloan Digital Sky Survey: First Spectroscopic Data from the Extended Baryon Oscillation Spectroscopic Survey and from the Second Phase of the Apache Point Observatory Galactic Evolution Experiment. *The Astrophysical Journal Supplement Series*, *235*, 42. <https://doi.org/10.3847/1538-4365/aa9e8a>
- Acín, V., Delfino, M., Herbera, A., & Hernández, J. (2015). Free cooling on the Mediterranean shore: Energy efficiency upgrades at PIC. *J. Phys. Conf. Ser.*, *664*(5), 052009. <https://doi.org/10.1088/1742-6596/664/5/052009>
- Agarwal, A., Slee, M., & Kwiatkowski, M. (2007, April). *Thrift: Scalable cross-language services implementation* (tech. rep.). Facebook. <http://thrift.apache.org/static/files/thrift-20070401.pdf>
- Alam, S., Ata, M., Bailey, S., Beutler, F., Bizyaev, D., Blazek, J. A., Bolton, A. S., Brownstein, J. R., Burden, A., ... Zhao, G.-B. (2017). The clustering of galaxies in the completed SDSS-III Baryon Oscillation Spectroscopic Survey: cosmological analysis of the DR12 galaxy sample. *Monthly Notices of the Royal Astronomical Society*, *470*(3), 2617–2652. <https://doi.org/10.1093/mnras/stx721>
- Alam, S., Aubert, M., Avila, S., Balland, C., Bautista, J. E., Bershad, M. A., Bizyaev, D., Blanton, M. R., Bolton, A. S., ... Zheng, Z. (2021). Completed sdss-iv extended baryon oscillation spectroscopic survey: Cosmological implications from two decades of spectroscopic surveys at the apache point observatory. *Phys. Rev. D*, *103*, 083533. <https://doi.org/10.1103/PhysRevD.103.083533>
- Alarcon, A., Gaztanaga, E., Eriksen, M., Baugh, C. M., Cabayol, L., Casas, R., Carretero, J., Castander, F. J., De Vicente, J., ... Tallada-Crespí, P. (2021). The PAU Survey: an improved photo-z sample in the COSMOS field. *Monthly Notices of the Royal Astronomical Society*, *501*(4), 6103–6122. <https://doi.org/10.1093/mnras/staa3659>
- Amendola, L., Appleby, S., Avgoustidis, A., Bacon, D., Baker, T., Baldi, M., Bartolo, N., Blanchard, A., Bonvin, C., ... Zlosnik, T. (2018). Cosmology and fundamental physics with the Euclid satellite. *Living Reviews in Relativity*, *21*(1), Article 2, 2. <https://doi.org/10.1007/s41114-017-0010-3>
- Asgari, M., Lin, C.-A., Joachimi, B., Giblin, B., Heymans, C., Hildebrandt, H., Kannawadi, A., Stölzner, B., Tröster, T., ... Valentijn, E. (2021). KiDS-1000 cosmology: Cosmic shear constraints and comparison between two point statistics. *Astronomy & Astrophysics*, *645*, Article A104, A104. <https://doi.org/10.1051/0004-6361/202039070>

- Asorey, J., Carrasco Kind, M., Sevilla-Noarbe, I., Brunner, R. J., & Thaler, J. (2016). Galaxy clustering with photometric surveys using PDF redshift information. *Monthly Notices of the Royal Astronomical Society*, *459*(2), 1293–1309. <https://doi.org/10.1093/mnras/stw721>
- Ayllon, A. A., Salichos, M., Simon, M. K., & Keeble, O. (2014). FTS3: New data movement service for WLCG. *Journal of Physics: Conference Series*, *513*(3), 032081. <https://doi.org/10.1088/1742-6596/513/3/032081>
- Behroozi, P. S., Wechsler, R. H., & Wu, H.-Y. (2013). The ROCKSTAR Phase-space Temporal Halo Finder and the Velocity Offsets of Cluster Cores. *Astrophysical Journal*, *762*, Article 109, 109. <https://doi.org/10.1088/0004-637X/762/2/109>
- Ben-Kiki, O., Evans, C., & Ingerson, B. (2009). YAML ain't markup language (YAML) version 1.2. *YAML.org*.
- Bernyk, M., Croton, D. J., Tonini, C., Hodkinson, L., Hassan, A. H., Garel, T., Duffy, A. R., Mutch, S. J., Poole, G. B., & Hegarty, S. (2016). The theoretical astrophysical observatory: Cloud-based mock galaxy catalogs*. *The Astrophysical Journal Supplement Series*, *223*(1), 9. <https://doi.org/10.3847/0067-0049/223/1/9>
- Bittorf, M., Bobrovitsky, T., Erickson, C., Hecht, M. G. D., Kuff, M., Leblang, D. K. A., Robinson, N., Rus, D. R. S., Wanderman, J., & Yoder, M. M. (2015). Impala: A modern, open-source sql engine for hadoop. *Proceedings of the 7th Biennial Conference on Innovative Data Systems Research*.
- Blanton, M. R., Hogg, D. W., Bahcall, N. A., Brinkmann, J., Britton, M., Connolly, A. J., Csabai, I., Fukugita, M., Loveday, J., ... Weinberg, D. H. (2003). The Galaxy Luminosity Function and Luminosity Density at Redshift $z = 0.1$. *Astrophysical Journal*, *592*(2), 819–838. <https://doi.org/10.1086/375776>
- Blanton, M. R., Lupton, R. H., Schlegel, D. J., Strauss, M. A., Brinkmann, J., Fukugita, M., & Loveday, J. (2005). The properties and luminosity function of extremely low luminosity galaxies*. *The Astrophysical Journal*, *631*(1), 208. <https://doi.org/10.1086/431416>
- Bolzonella, M., Miralles, J.-M., & Pelló, R. (2000). Photometric redshifts based on standard SED fitting procedures. *aap*, *363*, 476–492.
- Cabayol, L., Sevilla-Noarbe, I., Fernández, E., Carretero, J., Eriksen, M., Serrano, S., Alarcón, A., Amara, A., Casas, R., ... Tortorelli, L. (2019). The PAU survey: star-galaxy classification with multi narrow-band data. *MNRAS*, *483*, 529–539. <https://doi.org/10.1093/mnras/sty3129>
- Cabayol, L., Eriksen, M., Carretero, J., Casas, R., Castander, F. J., Fernández, E., Garcia-Bellido, J., Gaztanaga, E., Hildebrandt, H., ... Tramacere, A. (2023). The PAU Survey and Euclid: Improving broadband photometric redshifts with multi-task learning. *A&A*, *671*, Article A153, A153. <https://doi.org/10.1051/0004-6361/202245027>
- Carretero, J., Castander, F. J., Gaztañaga, E., Croce, M., & Fosalba, P. (2015). An algorithm to build mock galaxy catalogues using MICE simulations. *Monthly Notices of the Royal Astronomical Society*, *447*(1), 646–670. <https://doi.org/10.1093/mnras/stu2402>
- Carretero, J., Tallada, P., Casals, J., Caubet, M., Castander, F., Blot, L., Alarcón, A., Serrano, S., Fosalba, P., ... Delfino, M. (2017). CosmoHub and SciPIC: Massive cosmological data analysis, distribution and generation using a Big Data platform. *Proceedings of the European Physical Society Conference on High Energy Physics. 5-12 July*, Article 488, 488.

- Castander, F. J. et al. (2019). The PAU Survey: Data Reduction of Narrow Band Images. *in preparation*.
- Chamberlin, D. D., & Boyce, R. F. (1974). SEQUEL: A structured english query language. *ACM SIGMOD*, 249–264. <http://www.almaden.ibm.com/cs/people/chamberlin/sequel-1974.pdf>
- Cosden, I. A., McHenry, K., & Katz, D. S. (2022). Research Software Engineers: Career Entry Points and Training Gaps. *Computing in Science and Engineering*, 24(6), 14–21. <https://doi.org/10.1109/MCSE.2023.3258630>
- Cox, A. N. (2000). *Allen’s astrophysical quantities*. Springer.
- Crocce, M., Castander, F. J., Gaztañaga, E., Fosalba, P., & Carretero, J. (2015). The MICE Grand Challenge lightcone simulation – II. Halo and galaxy catalogues. *MNRAS*, 453(2), 1513–1530. <https://doi.org/10.1093/mnras/stv1708>
- Crocce, M., Ross, A. J., Sevilla-Noarbe, I., Gaztanaga, E., Elvin-Poole, J., Avila, S., Alarcon, A., Chan, K. C., Banik, N., & Carretero, J. (2019). Dark Energy Survey year 1 results: galaxy sample for BAO measurement. *MNRAS*, 482(2), 2807–2822. <https://doi.org/10.1093/mnras/sty2522>
- D’Amico, G., Gleyzes, J., Kokron, N., Markovic, K., Senatore, L., Zhang, P., Beutler, F., & Gil-Marín, H. (2020). The cosmological analysis of the sdss/boss data from the effective field theory of large-scale structure. *Journal of Cosmology and Astroparticle Physics*, 2020(05), 005. <https://doi.org/10.1088/1475-7516/2020/05/005>
- Deschamps, J., Dalle Nogare, D., & Jug, F. (2023). Better Research Software Tools to Elevate the Rate of Scientific Discovery – or why we need to invest in research software engineering. *arXiv e-prints*, Article arXiv:2307.03934. <https://doi.org/10.48550/arXiv.2307.03934>
- DESI Collaboration, Aghamousa, A., Aguilar, J., Ahlen, S., Alam, S., Allen, L. E., Allende Prieto, C., Annis, J., Bailey, S., . . . Zu, Y. (2016a). The DESI Experiment Part I: Science, Targeting, and Survey Design. *arXiv e-prints*, Article arXiv:1611.00036. <https://doi.org/10.48550/arXiv.1611.00036>
- DESI Collaboration, Aghamousa, A., Aguilar, J., Ahlen, S., Alam, S., Allen, L. E., Allende Prieto, C., Annis, J., Bailey, S., . . . Zu, Y. (2016b). The DESI Experiment Part II: Instrument Design. *arXiv e-prints*, Article arXiv:1611.00037. <https://doi.org/10.48550/arXiv.1611.00037>
- Donno, F., Abadie, L., Badino, P., Baud, J.-P., Corso, E., Witt, S. D., Fuhrmann, P., Gu, J., Koblitz, B., . . . Zappi, R. (2008). Storage resource manager version 2.2: Design, implementation, and testing experience. *Journal of Physics: Conference Series*, 119(6), 062028. <https://doi.org/10.1088/1742-6596/119/6/062028>
- Eriksen, M., Alarcon, A., Gaztanaga, E., Amara, A., Cabayol, L., Carretero, J., Castander, F. J., Crocce, M., Delfino, M., . . . Tortorelli, L. (2019). The PAU Survey: early demonstration of photometric redshift performance in the COSMOS field. *Monthly Notices of the Royal Astronomical Society*, 484(3), 4200–4215. <https://doi.org/10.1093/mnras/stz204>
- Eriksen, M., Alarcon, A., Cabayol, L., Carretero, J., Casas, R., Castander, F. J., De Vicente, J., Fernandez, E., Garcia-Bellido, J., . . . Tallada, P. (2020). The PAU Survey: Photometric redshifts using transfer learning from simulations. *Monthly Notices of the Royal Astronomical Society*, 497(4), 4565–4579. <https://doi.org/10.1093/mnras/staa2265>
- Fernique, P., Allen, M. G., Boch, T., Oberto, A., Pineau, F.-X., Durand, D., Bot, C., Cambrésy, L., Derriere, S., . . . Bonnarel, F. (2015). Hierarchical progressive surveys - multi-resolution healpix data structures for astronomical images, catalogues, and

- 3-dimensional data cubes. *Astronomy & Astrophysics*, 578, A114. <https://doi.org/10.1051/0004-6361/201526075>
- Ferreira, P. G. (2019). Cosmological tests of gravity. *Annual Review of Astronomy and Astrophysics*, 57, 335–374.
- Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* [Doctoral dissertation] [AAI9980887]. University of California, Irvine.
- Folk, M., Heber, G., Koziol, Q., Pourmal, E., & Robinson, D. (2011). An overview of the hdf5 technology suite and its applications. *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*, 36–47.
- Fosalba, P., Crocce, M., Gaztañaga, E., & Castander, F. J. (2015). The MICE grand challenge lightcone simulation – I. Dark matter clustering. *MNRAS*, 448(4), 2987–3000. <https://doi.org/10.1093/mnras/stv138>
- Fosalba, P., Gaztañaga, E., Castander, F. J., & Crocce, M. (2015). The MICE Grand Challenge light-cone simulation - III. Galaxy lensing mocks from all-sky lensing maps. *Monthly Notices of the Royal Astronomical Society*, 447(2), 1319–1332. <https://doi.org/10.1093/mnras/stu2464>
- Fukugita, M., Ichikawa, T., Gunn, J. E., Doi, M., Shimasaku, K., & Schneider, D. P. (1996). The Sloan Digital Sky Survey Photometric System. *Astronomical Journal*, 111, 1748. <https://doi.org/10.1086/117915>
- Gaia Collaboration & et al. (2016). Gaia Data Release 1. Summary of the astrometric, photometric, and survey properties. *AAP*, 595, A2. <https://doi.org/10.1051/0004-6361/201629512>
- Gaztañaga, E., Eriksen, M., Crocce, M., Castander, F. J., Fosalba, P., Marti, P., Miquel, R., & Cabré, A. (2012). Cross-correlation of spectroscopic and photometric galaxy surveys: cosmology from lensing and redshift distortions. *Monthly Notices of the Royal Astronomical Society*, 422(4), 2904–2930. <https://doi.org/10.1111/j.1365-2966.2012.20613.x>
- Gaztañaga, E. et al. (2019). The PAU Survey: Photometric Calibration of Narrow Band Images. *in preparation*.
- Gorski, K. M., Wandelt, B. D., Hansen, F. K., Hivon, E., & Banday, A. J. (1999). The HEALPix Primer. *arXiv e-prints*, Article astro-ph/9905275.
- Goth, F., Alves, R., Braun, M., Jael Castro, L., Chourdakis, G., Christ, S., Cohen, J., Erxleben, F., Grad, J.-N., ... Wittke, S. (2023). Foundational Competencies and Responsibilities of a Research Software Engineer. *arXiv e-prints*. <https://doi.org/10.48550/arXiv.2311.11457>
- Greenfield, P., Droettboom, M., & Bray, E. (2015). Asdf: A new data format for astronomy. *Astronomy and Computing*, 12, 240–251. <https://doi.org/https://doi.org/10.1016/j.ascom.2015.06.004>
- Gschwend, J., Rossel, A. C., Ogando, R. L. C., Neto, A. F., Maia, M. A. G., da Costa, L. N., Lima, M., Pellegrini, P., Campisano, R., ... Walker, A. R. (2018). DES science portal: Computing photometric redshifts. *Astronomy and Computing*, 25, 58–80. <https://doi.org/10.1016/j.ascom.2018.08.008>
- Guy, J., Bailey, S., Kremin, A., Alam, S., Alexander, D. M., Allende Prieto, C., BenZvi, S., Bolton, A. S., Brooks, D., ... Zou, H. (2023). The Spectroscopic Data Processing Pipeline for the Dark Energy Spectroscopic Instrument. *Astronomical Journal*, 165(4), Article 144, 144. <https://doi.org/10.3847/1538-3881/acb212>
- Heitmann, K., Uram, T. D., Finkel, H., Frontiere, N., Habib, S., Pope, A., Rangel, E., Hollowed, J., Korytov, D., & Larsen, P. (2019). HACC Cosmological Simulations: First Data Release. *arXiv e-prints*, Article arXiv:1904.11966, arXiv:1904.11966.

- Heymans, C., Van Waerbeke, L., Miller, L., Erben, T., Hildebrandt, H., Hoekstra, H., Kitching, T. D., Mellier, Y., Simon, P., . . . Velander, M. (2012). CFHTLenS: the Canada-France-Hawaii Telescope Lensing Survey. *MNRAS*, *427*, 146–166. <https://doi.org/10.1111/j.1365-2966.2012.21952.x>
- Hoffmann, K., Bel, J., Gaztanaga, E., Crocce, M., Fosalba, P., & Castander, F. (2014). Measuring the growth of matter fluctuations with third-order galaxy correlations. *MNRAS*. <https://doi.org/10.1093/mnras/stu2492>
- Hoffmann, K., Secco, L. F., Blazek, J., Crocce, M., Tallada-Crespí, P., Samuroff, S., Prat, J., Carretero, J., Fosalba, P., . . . Castander, F. J. (2022). Modeling intrinsic galaxy alignment in the mice simulation. <https://doi.org/10.48550/ARXIV.2206.14219>
- Hogg, D. W., Baldry, I. K., Blanton, M. R., & Eisenstein, D. J. (2002). The K correction. *ArXiv Astrophysics e-prints*.
- Huterer, D., Kirkby, D., Bean, R., Connolly, A., Dawson, K., Dodelson, S., Evrard, A., Jain, B., Jarvis, M., . . . Zhao, G. (2015). Growth of cosmic structure: Probing dark energy beyond expansion. *Astroparticle Physics*, *63*, 23–41. <https://doi.org/10.1016/j.astropartphys.2014.07.004>
- Ilbert, O., Capak, P., Salvato, M., Aussel, H., McCracken, H. J., Sanders, D. B., Scoville, N., Kartaltepe, J., Arnouts, S., . . . Zucca, E. (2009). Cosmos Photometric Redshifts with 30-Bands for 2-deg². *Astrophysical Journal*, *690*.
- Ivanov, M. M., Simonović, M., & Zaldarriaga, M. (2020). Cosmological parameters from the boss galaxy power spectrum. *Journal of Cosmology and Astroparticle Physics*, *2020(05)*, 042. <https://doi.org/10.1088/1475-7516/2020/05/042>
- Ivezić, Ž., Kahn, S. M., Tyson, J. A., Abel, B., Acosta, E., Allsman, R., Alonso, D., AlSayyad, Y., Anderson, S. F., . . . Zhan, H. (2019). LSST: From Science Drivers to Reference Design and Anticipated Data Products. *Astrophysical Journal*, *873(2)*, Article 111, 111. <https://doi.org/10.3847/1538-4357/ab042c>
- Jay, C., Haines, R., & Katz, D. S. (2020). Software must be recognised as an important output of scholarly research. *arXiv e-prints*, Article arXiv:2011.07571. <https://doi.org/10.48550/arXiv.2011.07571>
- Jurić, M., Ciardi, D., & Dubois-Felsmann, G. (2017). LSST Science Platform Vision Document. *LSST Document LSE-319* <https://lse-319.lsst.io>.
- Kambatla, K., & Chen, Y. (2014). The truth about mapreduce performance on ssds. *Proceedings of the 28th USENIX Conference on Large Installation System Administration*, 109–117. <http://dl.acm.org/citation.cfm?id=2717491.2717499>
- Koposov, S., & Bartunov, O. (2006, July). Q3C, Quad Tree Cube – The new Sky-indexing Concept for Huge Astronomical Catalogues and its Realization for Main Astronomical Queries (Cone Search and Xmatch) in Open Source Database PostgreSQL. In C. Gabriel, C. Arviset, D. Ponz, & S. Enrique (Eds.), *Astronomical data analysis software and systems xv* (p. 735, Vol. 351).
- Laureijs, R., Gondoin, P., Duvet, L., Saavedra Criado, G., Hoar, J., Amiaux, J., Auguères, J.-L., Cole, R., Cropper, M., . . . Valenziano, L. (2012). Euclid: ESA’s mission to map the geometry of the dark universe. *Space Telescopes and Instrumentation 2012: Optical, Infrared, and Millimeter Wave*, *8442*, Article 84420T. <https://doi.org/10.1117/12.926496>
- Lintott, C., Schawinski, K., Bamford, S., Slosar, A., Land, K., Thomas, D., Edmondson, E., Masters, K., Nichol, R. C., . . . Vandenberg, J. (2011). Galaxy Zoo 1: data release of morphological classifications for nearly 900 000 galaxies. *Monthly Notices of the Royal Astronomical Society*, *410(1)*, 166–178. <https://doi.org/10.1111/j.1365-2966.2010.17432.x>

- López, L., Padilla, C., Cardiel-Sas, L., Ballester, O., Grañena, F., Majà, E., & Castander, F. J. (2016). The PAU camera carbon fiber cryostat and filter interchange system. *Ground-based and Airborne Instrumentation for Astronomy VI, 9908*, 99084G. <https://doi.org/10.1117/12.2232294>
- LSST Dark Energy Science Collaboration. (2012). Large Synoptic Survey Telescope: Dark Energy Science Collaboration. *arXiv e-prints*.
- Martí, P., Miquel, R., Castander, F. J., Gaztañaga, E., Eriksen, M., & Sánchez, C. (2014). Precise photometric redshifts with a narrow-band filter set: the PAU survey at the William Herschel Telescope. *Monthly Notices of the Royal Astronomical Society, 442*(1), 92–109. <https://doi.org/10.1093/mnras/stu801>
- Melnik, S., Gubarev, A., Long, J. J., Romer, G., Shivakumar, S., Tolton, M., & Vassilakis, T. (2010). Dremel: Interactive analysis of web-scale datasets. *Proc. VLDB Endow.*, 3(1-2), 330–339. <https://doi.org/10.14778/1920841.1920886>
- Morganson, E., Gruendl, R. A., Menanteau, F., Carrasco Kind, M., Chen, Y. .-, Daues, G., Drlica-Wagner, A., Friedel, D. N., Gower, M., . . . DES Collaboration. (2018). The Dark Energy Survey Image Processing Pipeline. *Publications of the ASP, 130*(989), 074501. <https://doi.org/10.1088/1538-3873/aab4ef>
- Nelson, D., Springel, V., Pillepich, A., Rodriguez-Gomez, V., Torrey, P., Genel, S., Vogelsberger, M., Pakmor, R., Marinacci, F., & Weinberger, R. (2019). The IllustrisTNG simulations: public data release. *Computational Astrophysics and Cosmology, 6*(1), Article 2, 2. <https://doi.org/10.1186/s40668-019-0028-x>
- Nikutta, R., Fitzpatrick, M., Scott, A., & Weaver, B. A. (2020). Data Lab-A community science platform. *Astronomy and Computing, 33*, Article 100411, 100411. <https://doi.org/10.1016/j.ascom.2020.100411>
- Ochsenbein, F., Williams, R., Davenhall, C., Durand, D., Fernique, P., Hanisch, R., Giarretta, D., McGlynn, T., Szalay, A., & Wicenec, A. (2004). Votable: Tabular data for the virtual observatory. In P. J. Quinn & K. M. Górski (Eds.), *Toward an international virtual observatory* (pp. 118–123). Springer Berlin Heidelberg.
- O'Donnell, J. E. (1994). R v-dependent Optical and Near-Ultraviolet Extinction. *Astrophysical Journal, 422*, 158. <https://doi.org/10.1086/173713>
- O'Mullane, W., Economou, F., Huang, F., Speck, D., Chiang, H.-F., Graham, M. L., Allbery, R., Banek, C., Sick, J., . . . Dubois-Felsmann, G. P. (2021). Rubin Science Platform on Google: the story so far. *Proceedings of ADASS XXXI*, Article arXiv:2111.15030. <https://doi.org/10.48550/arXiv.2111.15030>
- Overzier, R., Lemson, G., Angulo, R. E., Bertin, E., Blaizot, J., Henriques, B. M. B., Marleau, G. .-, & White, S. D. M. (2013). The Millennium Run Observatory: first light. *MNRAS, 428*(1), 778–803. <https://doi.org/10.1093/mnras/sts076>
- Padilla, C., Castander, F. J., Alarcón, A., Aleksic, J., Ballester, O., Cabayol, L., Cardiel-Sas, L., Carretero, J., Casas, R., . . . de Vicente, J. (2019). The Physics of the Accelerating Universe Camera. *Astronomical Journal, 157*(6), Article 246, 246. <https://doi.org/10.3847/1538-3881/ab0412>
- Peebles, P. J. E. (1980). *The large-scale structure of the universe*.
- Pence, W. D., Chiappetti, L., Page, C. G., Shaw, R. A., & Stobie, E. (2010). Definition of the Flexible Image Transport System (FITS), version 3.0. *AAP, 524*, A42. <https://doi.org/10.1051/0004-6361/201015362>
- Percival, W. J., & White, M. (2009). Testing cosmological structure formation using redshift-space distortions. *Monthly Notices of the Royal Astronomical Society, 393*, 297–308. <https://doi.org/10.1111/j.1365-2966.2008.14211.x>

- Perlmutter, S., Aldering, G., Goldhaber, G., Knop, R. A., Nugent, P., Castro, P. G., Deustua, S., Fabbro, S., Goobar, A., . . . Project, T. S. C. (1999). Measurements of Ω and Λ from 42 High-Redshift Supernovae. *ApJ*, *517*, 565–586. <https://doi.org/10.1086/307221>
- Pitié, F., & Dahyot, R. (2005). N-dimensional probability density function transfer and its application to colour transfer. *Proceedings of the IEEE International Conference on Computer Vision*, *2*, 1434–1439. <https://doi.org/10.1109/ICCV.2005.166>
- Planck Collaboration, Abergel, A., Ade, P. A. R., Aghanim, N., Alves, M. I. R., Aniano, G., Armitage-Caplan, C., Arnaud, M., Ashdown, M., . . . Zonca, A. (2014). Planck 2013 results. XI. All-sky model of thermal dust emission. *Astronomy and Astrophysics*, *571*, Article A11, A11. <https://doi.org/10.1051/0004-6361/201323195>
- Planck Collaboration, Ade, P. A. R., Aghanim, N., Arnaud, M., Ashdown, M., Aumont, J., Baccigalupi, C., Banday, A. J., Barreiro, R. B., . . . Zonca, A. (2016). Planck 2015 results. XIII. Cosmological parameters. *Astronomy and Astrophysics*, *594*, Article A13, A13. <https://doi.org/10.1051/0004-6361/201525830>
- Plaszczynski, S., Peloton, J., Arnault, C., & Campagne, J. E. (2019). Analysing billion-objects catalogue interactively: ApacheSpark for physicists. *Astronomy and Computing*, *28*, Article 100305, 100305. <https://doi.org/10.1016/j.ascom.2019.100305>
- Plaszczynski, S., Campagne, J. E., Peloton, J., & Arnault, C. (2021). Scaling pair count to next galaxy surveys. *Monthly Notices of the Royal Astronomical Society*, *510*(2), 3085–3097. <https://doi.org/10.1093/mnras/stab3640>
- Porredon, A., Crocce, M., Elvin-Poole, J., Cawthon, R., Giannini, G., De Vicente, J., Carnero Rosell, A., Ferrero, I., Krause, E., . . . Weller, J. (2022). Dark energy survey year 3 results: Cosmological constraints from galaxy clustering and galaxy-galaxy lensing using the maglim lens sample. *Phys. Rev. D*, *106*, 103530. <https://doi.org/10.1103/PhysRevD.106.103530>
- Potter, D., & Stadel, J. (2016, September). PKDGRAV3: Parallel gravity code.
- Potter, D., Stadel, J., & Teyssier, R. (2017). PKDGRAV3: beyond trillion particle cosmological simulations for the next era of galaxy surveys. *Computational Astrophysics and Cosmology*, *4*, Article 2, 2. <https://doi.org/10.1186/s40668-017-0021-1>
- Raddick, J., Souter, B., Lemson, G., & Taghizadeh-Popp, M. (2017). SciServer: An Online Collaborative Environment for Big Data in Research and Education. *American Astronomical Society Meeting Abstracts #229*, *229*, Article 236.15, 236.15.
- Raddick, M. J., Thakar, A. R., Szalay, A. S., & Santos, R. D. C. (2014a). Ten Years of SkyServer I: Tracking Web and SQL e-Science Usage. *Computing in Science and Engineering*, *16*(4), 22–31. <https://doi.org/10.1109/MCSE.2014.34>
- Raddick, M. J., Thakar, A. R., Szalay, A. S., & Santos, R. D. C. (2014b). Ten Years of SkyServer II: How Astronomers and the Public Have Embraced e-Science. *Computing in Science and Engineering*, *16*(4), 32–40. <https://doi.org/10.1109/MCSE.2014.32>
- Richardson, T., Stafford-Fraser, Q., Wood, K., & Hopper, A. (1998). Virtual network computing. *IEEE Internet Computing*, *2*(1), 33–38. <https://doi.org/10.1109/4236.656066>
- Riess, A. G., Filippenko, A. V., Challis, P., Clocchiatti, A., Diercks, A., Garnavich, P. M., Gilliland, R. L., Hogan, C. J., Jha, S., . . . Tonry, J. (1998). Observational Evidence from Supernovae for an Accelerating Universe and a Cosmological Constant. *AJ*, *116*, 1009–1038. <https://doi.org/10.1086/300499>
- Saha, B., Shah, H., Seth, S., Vijayaraghavan, G., Murthy, A., & Curino, C. (2015). Apache tez: A unifying framework for modeling and building data processing applications.

- Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 1357–1369. <https://doi.org/10.1145/2723372.2742790>
- Sánchez, A. G., Scoccimarro, R., Crocce, M., Grieb, J. N., Salazar-Albornoz, S., Vecchia, C. D., Lippich, M., Beutler, F., Brownstein, J. R., ... Zhao, G.-B. (2016). The clustering of galaxies in the completed SDSS-III Baryon Oscillation Spectroscopic Survey: Cosmological implications of the configuration-space clustering wedges. *Monthly Notices of the Royal Astronomical Society*, *464*(2), 1640–1658. <https://doi.org/10.1093/mnras/stw2443>
- Schlegel, D. J., Finkbeiner, D. P., & Davis, M. (1998). Maps of Dust Infrared Emission for Use in Estimation of Reddening and Cosmic Microwave Background Radiation Foregrounds. *Astrophysical Journal*, *500*(2), 525–553. <https://doi.org/10.1086/305772>
- Serenelli, Aldo, Rohrmann, René D., & Fukugita, Masataka. (2019). Nature of blackbody stars. *A&A*, *623*, A177. <https://doi.org/10.1051/0004-6361/201834032>
- Serrano, S., Gaztañaga, E., Castander, F. J., Eriksen, M., Casas, R., Alarcon, A., Bauer, A., Cabayol, L., Carretero, J., ... de Vicente, J. (2022). The PAU Survey: Narrow-band image photometry. *arXiv e-prints*, Article arXiv:2206.14022.
- Sevilla-Noarbe, I., Hoyle, B., Marchã, M. J., Soumagnac, M. T., Bechtol, K., Drlica-Wagner, A., Abdalla, F., Aleksić, J., Avestruz, C., & Balbinot, E. (2018). Star-galaxy classification in the Dark Energy Survey Y1 data set. *MNRAS*, *481*(4), 5451–5469. <https://doi.org/10.1093/mnras/sty2579>
- Sfiligoi, I. (2008). glideinWMS—a generic pilot-based workload management system. *Journal of Physics: Conference Series*, *119*(6), 062044. <https://doi.org/10.1088/1742-6596/119/6/062044>
- Shafranovich, Y. (2005, October). *Common format and mime type for comma-separated values (csv) files* (RFC No. 4180) (<http://www.rfc-editor.org/rfc/rfc4180.txt>). RFC Editor. RFC Editor. <http://www.rfc-editor.org/rfc/rfc4180.txt>
- Shvachko, K., Kuang, H., Radia, S., & Chansler, R. (2010). The hadoop distributed file system. *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, 1–10. <https://doi.org/10.1109/MSST.2010.5496972>
- Soo, J. Y. H., Joachimi, B., Eriksen, M., Siudek, M., Alarcon, A., Cabayol, L., Carretero, J., Casas, R., Castander, F. J., ... Tallada-Crespí, P. (2021). The PAU Survey: narrow-band photometric redshifts using Gaussian processes. *Monthly Notices of the Royal Astronomical Society*, *503*(3), 4118–4135. <https://doi.org/10.1093/mnras/stab711>
- Stothert, L., Norberg, P., Baugh, C. M., Alarcon, A., Amara, A., Carretero, J., Castander, F. J., Eriksen, M., Fernandez, E., ... Tortorelli, L. (2018). The PAU Survey: spectral features and galaxy clustering using simulated narrow-band photometry. *MNRAS*, *481*, 4221–4235. <https://doi.org/10.1093/mnras/sty2491>
- Szalay, A. S., Gray, J., Thakar, A. R., Kunszt, P. Z., Malik, T., Raddick, J., Stoughton, C., & vandenBerg, J. (2002). The SDSS SkyServer: Public Access to the Sloan Digital Sky Server Data. *arXiv e-prints*.
- Taghizadeh-Popp, M., Kim, J., Lemson, G., Medvedev, D., Raddick, M., Szalay, A., Thakar, A., Booker, J., Chhetri, C., ... Rippin, M. (2020). Sciserver: A science platform for astronomy and beyond. *Astronomy and Computing*, *33*, 100412. <https://doi.org/https://doi.org/10.1016/j.ascom.2020.100412>
- Tallada, P., Carretero, J., Casals, J., Acosta-Silva, C., Serrano, S., Caubet, M., Castander, F. J., César, E., Crocce, M., ... Tonello, N. (2020). CosmoHub: Interactive exploration and distribution of astronomical data on Hadoop. *Astronomy and Computing*, *32*, 100391. <https://doi.org/10.1016/j.ascom.2020.100391>

- Tallada-Crespí, P., Carretero, J., Serrano, S., Castander, F. J., César, E., Eriksen, M., Fosalba, F., Gaztañaga, E., Merino, G., ... Torradeflot, F. (2023). Fast approximate method for computing simulated galaxy fluxes on top of Apache Spark. *Accepted for publication in the Astronomical Society of the Pacific Conference Series*.
- Tassev, S., Zaldarriaga, M., & Eisenstein, D. J. (2013). Solving large scale structure in ten easy steps with cola. *Journal of Cosmology and Astroparticle Physics*, 2013(06), 036. <https://doi.org/10.1088/1475-7516/2013/06/036>
- Tegmark, M., Eisenstein, D. J., Strauss, M. A., Weinberg, D. H., Blanton, M. R., Frieman, J. A., Fukugita, M., Gunn, J. E., Hamilton, A. J. S., ... York, D. G. (2006). Cosmological constraints from the sdss luminous red galaxies. *Phys. Rev. D*, 74, 123507. <https://doi.org/10.1103/PhysRevD.74.123507>
- The Dark Energy Survey Collaboration. (2005). The Dark Energy Survey. *arXiv e-prints*.
- Thomas, B., Jenness, T., Economou, F., Greenfield, P., Hirst, P., Berry, D., Bray, E., Gray, N., Muna, D., ... Homeier, D. (2015). Learning from fits: Limitations in use in modern astronomical research. *Astronomy and Computing*, 12, 133–145. <https://doi.org/https://doi.org/10.1016/j.ascom.2015.01.009>
- Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., & Murthy, R. (2009). Hive: A warehousing solution over a map-reduce framework. *Proc. VLDB Endow.*, 2(2), 1626–1629. <https://doi.org/10.14778/1687553.1687609>
- Tonello, N., Tallada, P., Serrano, S., Carretero, J., Eriksen, M., Folger, M., Neissner, C., Sevilla-Noarbe, I., Castander, F., ... Tortorelli, L. (2019). The PAU Survey: Operation and orchestration of multi-band survey data. *Astronomy and Computing*, 27, 171–188. <https://doi.org/https://doi.org/10.1016/j.ascom.2019.04.002>
- Tortorelli, L., Della Bruna, L., Herbel, J., Amara, A., Refregier, A., Alarcon, A., Carretero, J., Castander, F. J., De Vicente, J., ... Tonello, N. (2018). The PAU Survey: a forward modeling approach for narrow-band imaging. *Journal of Cosmology and Astro-Particle Physics*, 2018, 035. <https://doi.org/10.1088/1475-7516/2018/11/035>
- Tully, R. B., & Fisher, J. R. (1977). A new method of determining distances to galaxies. *Astronomy and Astrophysics*, 54, 661–673.
- Unicode Staff, C. (1991). *The unicode standard: Worldwide character encoding* (1st). Addison Wesley Longman Publishing Co., Inc.
- Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., ... Baldeschwieler, E. (2013). Apache hadoop yarn: Yet another resource negotiator. *Proceedings of the 4th Annual Symposium on Cloud Computing*, 5:1–5:16. <https://doi.org/10.1145/2523616.2523633>
- Wandelt, B. D., Hivon, E., & Gorski, K. M. (1998). Topological Analysis of High-Resolution CMB Maps. *arXiv e-prints*, Article astro-ph/9803317.
- Weinberg, D. H., Mortonson, M. J., Eisenstein, D. J., Hirata, C., Riess, A. G., & Rozo, E. (2013). Observational probes of cosmic acceleration. *Physics Reports*, 530(2), 87–255. <https://doi.org/10.1016/j.physrep.2013.05.001>
- Wells, D. C., & Greisen, E. W. (1979). Fits-a flexible image transport system. *Image Processing in Astronomy*, 445.
- Wilkinson, M., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., Bonino da Silva Santos, L. O., ... Mons, B. (2016). The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 3. <https://doi.org/10.1038/sdata.2016.18>
- Willett, K. W., Lintott, C. J., Bamford, S. P., Masters, K. L., Simmons, B. D., Castells, K. R. V., Edmondson, E. M., Fortson, L. F., Kaviraj, S., ... Thomas, D. (2013). Galaxy Zoo 2: detailed morphological classifications for 304,122 galaxies

- from the Sloan Digital Sky Survey. *Monthly Notices of the Royal Astronomical Society*, 435(4), 2835–2860. <https://doi.org/10.1093/mnras/stt1458>
- Woolston, C. (2022). Why science needs more research software engineers. *Nature*, 2022, 1516–2. <https://doi.org/10.1038/d41586-022-01516-2>
- York, D. G., Adelman, J., Anderson, J. E., Jr., Anderson, S. F., Annis, J., Bahcall, N. A., Bakken, J. A., Barkhouser, R., Bastian, S., ... SDSS Collaboration. (2000). The Sloan Digital Sky Survey: Technical Summary. *aj*, 120, 1579–1587. <https://doi.org/10.1086/301513>