UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Arquitectura de Computadors

Economic regulation for multi tenant infrastructures

Xavier León Gutiérrez

PhD Thesis in Computer Architecture
by the Universitat Politècnica de Catalunya

Advisor: Leandro Navarro Moldes

Barcelona, Spain 2013

Abstract

LARGE scale computing infrastructures need scalable and efficient resource allocation mechanisms to fulfil the requirements of its participants and applications while the whole system is regulated to work efficiently. Computational markets provide efficient allocation mechanisms that aggregate information from multiple sources in large, dynamic and complex systems where there is not a single source with complete information. They have been proven to be successful in matching resource demand and resource supply in the presence of selfish multi-objective and utility-optimizing users and selfish profit-optimizing providers. However, global infrastructure metrics which may not directly affect participants of the computational market still need to be addressed –a.k.a. economic externalities like load balancing or energy-efficiency. In this thesis, we point out the need to address these economic externalities, and we design and evaluate appropriate regulation mechanisms from different perspectives on top of existing economic models, to incorporate a wider range of objective metrics not considered otherwise. Our main contributions in this thesis are threefold; first, we propose a taxation mechanism that addresses the resource congestion problem effectively improving the balance of load among resources when correlated economic preferences are present; second, we propose a game theoretic model with complete information to derive an algorithm to aid resource providers to scale up and down resource supply so energy-related costs can be reduced; and third, we relax our previous assumptions about complete information on the resource provider side and design an incentive-compatible mechanism to encourage users to truthfully report their resource requirements effectively assisting providers to make energy-efficient allocations while providing a dynamic allocation mechanism to users.

Resum

Les infrastructures computacionals de gran escala necessiten mecanismes d'assignació de recursos escalables i eficients per complir amb els requisits computacionals de tots els seus participants, assegurant-se de que el sistema és regulat apropiadament per a que funcioni de manera efectiva. Els mercats computacionals són mecanismes d'assignació de recursos eficients que incorporen informació de diferents fonts considerant sistemes de gran escala, complexos i dinàmics on no existeix una única font que proveeixi informació completa de l'estat del sistema. Aquests mercats computacionals han demostrat ser exitosos per acomodar la demanda de recursos computacionals amb la seva oferta quan els seus participants son considerats estratègics des del punt de vist de teoria de jocs. Tot i així, existeixen mètriques a nivell global sobre la infraestructura que no tenen per què influenciar els usuaris a priori de manera directa. Així doncs, aquestes externalitats econòmiques com poden ser el balanceig de càrrega o la eficiència energètica, conformen una línia d'investigació que cal explorar. En aquesta tesi, presentem i descrivim la problemàtica derivada d'aquestes externalitats econòmiques. Un cop establert el marc d'actuació, dissenyem i avaluem mecanismes de regulació apropiats basats en models econòmics existents per resoldre aquesta problemàtica des de diferents punts de vista per incorporar un ventall més ampli de mètriques objectiu que no havien estat considerades fins al moment. Les nostres contribucions principals tenen tres vessants: en primer lloc, proposem un mecanisme de regulació de tipus impositiu que tracta de mitigar l'aparició de recursos sobre-explotats que, efectivament, millora el balanceig de la càrrega de treball entre els recursos disponibles; en segon lloc, proposem un model teòric basat en teoria de jocs amb informació completa que permet derivar un algorisme que facilita la tasca dels proveïdors de recursos per modificar a l'alça o a la baixa l'oferta de recursos per tal de reduir els costos relacionats amb el consum energètic; i en tercer lloc, relaxem la nostra assumpció prèvia sobre l'existència d'informació complerta per part del proveïdor de recursos i dissenyem un mecanisme basat en incentius per fomentar que els usuaris facin pública de manera verídica i explícita els seus requeriments computacionals, ajudant d'aquesta manera als proveïdors de recursos a fer assignacions eficients des del punt de vista energètic a la vegada que oferim un mecanisme d'assignació de recursos dinàmica als usuaris.

Acknowledgements

I don't have enough kind words to thank Prof. Leandro Navarro, my advisor. These past several years, he provided an invaluable scientific advice and prompt and detailed feedback when needed. But most importantly, he has shaped the way I see and feel scientific research. Working under his supervision has been a pleasure.

I am grateful to Prof. Tuan Anh Trinh, from the Budapest University of Technology and Economics, for his collaboration as a co-author on some of the papers related to this thesis. He challenged me to improve my scientific writing and presentation.

I would also like to thank Prof. Torsten Eymann for the ideas and the discussion we had during the first baby-steps of my doctoral dissertation as well as the doctoral course about economics and research he taught at UPC; they have been really useful.

Thanks to Prof. Joan Manuel Marquès who, without knowing, introduced me to research by encouraging me to read my first scientific paper on peer-to-peer systems and got me interested in research as a career.

Since this thesis is the end product of a long education and previous work in past European and Spanish projects, I would also like to take the opportunity to thank the people with whom I worked or learnt valuable assets and helped me enjoy working on research. Thanks to Pablo Chacín for endless discussions about our mutual research interests. His ideas and comments have also influenced to shape this thesis. During my first year, I also greatly benefited from working with Xavier Vilajosana who was already working on economic mechanisms and helped me get in touch with this line of research.

I also had the pleasure to collaborate with the PlanetLab team at Princeton University during a research stay in the US. Specially, I would like to thank Sapan Bathia and Marco Yuen for making my stay over there enjoyable and showed me a different perspective on how to do research.

On a more personal note, I would like to thank my family. My parents Angel and Mari Carmen who provided me unconditional support at any time and under any circumstance. I will owe you forever. Also thanks to Laia for his patience with me and constant support during the writing process of this dissertation.

Last but by any means least, I would like to thank the friends I made during all these years at the department of Computer Architecture. I don't know how to thank you all for the time we spent together and for spicing things up when need.

Contents

List of Publications

[P1] Xavier León, Tuan Anh Trinh, and Leandro Navarro. Modeling resource usage in planetary-scale shared infrastructures: Planetlab's case study. *Comput. Netw.*, 55(15):3394–3407, October 2011. (page 24, 29)

[P2] Xavier León, Tuan Anh Trinh, and Leandro Navarro. Using economic regulation to prevent resource congestion in large-scale shared infrastructures. *Future Generation Computer Systems*, 26(4):599 – 607, 2010. (page 25, 57)

[P3] Xavier León and Leandro Navarro. Limits of energy saving for the allocation of data center resources to networked applications. In Proceedings of the*INFOCOM IEEE Conference*, pages 216–220, 2011. (page 26, 81)

[P4] Xavier León and Leandro Navarro. A stackelberg game to derive the limits of energy savings for the allocation of data center resources. *Future Generation Computer Systems*, 29(1):74 – 83, 2013. (page 26, 81)

[P5] Xavier León and Leandro Navarro. Incentives for Dynamic and Energy-aware Capacity Allocation for Multi-tenant Clusters. under review *International Conference on the Economics of Grids, Clouds, Systems, and Services. GECON '13*, 2013. (page 27, 105)

Other Publications

[P6] X. Leon, X. Vilajosana, R. Brunner, R. Krishnaswamy, L. Navarro, F. Freitag, and J.M. Marques. Information and regulation in decentralized marketplaces for p2p-grids. In *Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2008. WETICE '08. IEEE 17th*, pages 133–138, 2008.

[P7] Pablo Chacin, Xavier Leon, Rene Brunner, Felix Freitag, and Leandro Navarro. Core services for grid markets. In Thierry Priol and Marco Vanneschi, editors, *From Grids to Service and Pervasive Computing*, pages 205–215. Springer US, 2008.

[P8] Ruby Krishnaswamy, Leandro Navarro, René Brunner, Xavier León, and Xavier Vilajosana. Grid4all: Open market places for democratic grids. In *Proceedings of the 5th international workshop on Grid Economics and Business Models*, GECON '08, pages 197–207, Berlin, Heidelberg, 2008. Springer-Verlag.

Chapter 1

Introduction

Computational resources are becoming easier to share as a result of technological advances in network, operating system and middleware infrastructures. These new kind of infrastructures have enabled a trend towards openness, accessibility and community contribution where resources, services and applications are hosted and consumed on the Internet.

The scientific computing community has led the advancement of highly scalable wide-area network distributed computing farms –e.g computational grid infrastructures and computing clouds– capable of solving complex problems needed by enterprises or academic institutions –e.g optimizing work flow in hospitals, data mining services, weather modeling and forecasting, etc.– by means of large pools of commodity resources.

Traditional computing job schedulers have focused on optimising utilization as well as throughput and response time and have only given users minimal control over service quality for individual jobs. Besides, priorities among users and jobs were typically configured by system administrators through ad-hoc policies and cannot easily adapt to changes in demand and supply without supervision. These kind of ad-hoc approaches lacks scalability as they usually are based on centralized schedulers and with minimum flexibility on how users specify their jobs.

Computational markets have been proposed as an economics-inspired solution to the problem of allocation in large-scale systems where resources are shared among different users. The general idea is to allow users to express their preferences in terms of prices representing their resource requirements

and value or priority to be computed. They have been proven to efficiently allocate resources in a decentralised way in the presence of selfish utility-optimising resource consumers and selfish profit-optimising resource providers. The advantage of this approach is the self-adjustment made by market participants –resource consumers and providers– to improve their outcomes without centralized control or scheduling.

However, these computational markets may fail to address external system-wide metrics not directly sensed by users of such markets. This way, an *externality*[1] in economic terms is defined as the cost which results from an activity and which affects an otherwise uninvolved party who did not choose to incur that cost. An example of externality is resource congestion in contributive systems –e.g. experimental testbeds like PlanetLab. In this case, a specific resource provider may unintentionally support a higher workload than other resource providers if users have correlated preferences for a given resource of this provider. Another example is energy-awareness in data centers. Users may not be aware of the economic costs related energy consumption and thus, a resource provider may incur in higher costs if it is not able to feed energy-related information into the computational market.

As in real life free-market economies, these externalities are addressed by central governments and regulators which impose restrictions (regulations) on the system that are designed to act as incentives to influence users to behave in a certain way.

Problem statement

The main question addressed in this thesis is how economic regulation from the macro economic standpoint can be integrated into large-scale computational infrastructures to allow resource providers modify the behavior of resource consumers to not only make economically and computationally efficient allocation decisions, but to improve system-wide metrics which are not considered by economically rational users like resource congestion or energy-efficiency.

[1] A comprehensive discussion on this subject as seen by economists is found at [1]

Methodology

The results in this thesis were obtained mainly by three means following the design science approach by Ken Peffers et al. [2]:

- **Trace analysis.** In order to successfully capture, model and present the existing problems in the area of resource allocation in large-scale infrastructures, we decided to study in depth one of the largest open operational infrastructures nowadays, PlanetLab. We used techniques from statistical modelling and time series analysis to find patterns and statistical properties about the behaviour of applications on this infrastructure which were used in subsequent works also part of this thesis.
- **Theoretical modeling.** System modeling based on real traces allows us to analytically study and evaluate our enhancements to the micro-economic framework and how our proposals may impact existing models. To this end, game theory is a useful tool to provide bounds on the performance and efficiency of a system in the equilibrium no matter the strategies used by participants in a game –i.e when resource providers and consumers in our scenario have no incentive to behave differently.
- **Simulation evaluation.** Given the complexity and the lack of a suitable infrastructure to implement, deploy and test our proposals, we find that simulation is the best approach to evaluate the contributions presented in this thesis. To this end, simulation tools were developed to narrow down the problem and parameter space that was most interesting for the problem in place. It allowed us a great deal of flexibility to scale up our system without bounds, and the results obtained could also more easily be compared to previous work as well as purely theoretical models.

Thesis organization

This thesis is organized as follows. In chapter 2 we summarize the system model upon which our contributions are build and the necessary background to aid the reader to interpret our contributions. Chapter 3 presents the state of the art on economics-inspired resource allocation systems to point out the

gaps that this thesis try to fill. Chapter 4 presents the main research questions addressed in this thesis and summarizes the main contributions to provide the reader a coherent view of the separate contributions detailed on chapters 5, 6, 7 and 8. Finally, we present our conclusions and future directions on chapter 9.

Chapter 2

Foundation

In this chapter, we discuss the foundational concepts and theory of the work presented in this thesis. We first describe the scenarios where our solutions may be applied covering from large-scale experimental infrastructures to multi-tenant clusters. Then, fundamental theory related to game theory and strategic behavior is also presented to guide the reader through the contributions on following chapters. Finally, we present the system model upon which our contributions are based to better situate the work presented later on this thesis.

2.1 Scenarios of shared infrastructures

The problems envisaged in this thesis come as a result of sharing resources among several users or applications with diverse and potentially conflicting needs. With the advance on huge monolithic computer mainframes shared by tens of users, scheduling decisions started to become a complex problem to deal with and, thus, the first solutions based on economic principles were proposed [3][4].

However, over the last two decades, the scientific computing community has led the advancement of highly scalable wide-area network distributed computing farms –e.g. networked testbeds, grid infrastructures and computing clouds– capable of solving complex problems needed by enterprises or academic institutions like optimizing workflows in hospitals, data mining services, weather modeling and forecasting.

These kind of infrastructures have enabled a trend towards openness, accessibility and community contribution in some cases where resources, services and applications are hosted and consumed on the Internet, and shared by hundreds or thousands of users.

Grid and Cloud Computing

At the beginning of this century, computational resources were becoming more common on scientific environments to implement, deploy, execute and validate results from the scientific community.

These initial scientific infrastructures were called the *Grid* which, according to a famous definition by Foster [5], is a system that "coordinates resources that are not subject to centralized control using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service." Non-trivial here means that services beyond pure information sharing, as typical in the World Wide Wed, are offered. What is interesting in these infrastructures from this thesis perspective is that they typically involve large-scale resource infrastructures that are able to deliver computational services shared by a dynamic community of users and providers across a large geographic area.

With the advancements on virtualization which allowed the use of the same physical hardware by different users in a completely isolated environment, and the need from enterprises to reduce IT costs, large infrastructure operators began to commercialize computing resources – be it computation or storage – to external users to make the most of their resources. To denote the shift of paradigm and to set it apart from current infrastructures like the Grid, the idea of a *cloud* infrastructure was born.

Thus, the idea behind the *cloud* is to maximize the effectiveness of the shared resources, not only shared by multiple users, but dynamically re-allocated as per demand as well. What this means is that not only applications can scale-up or down the number of resources they consume according to their demand but that the same resources can be dynamically allocated to different applications over time. In these kind of environments, energy-related costs are one of the largest contributions to the overall cost of operating a data center.

Networked testbeds infrastructures for experimentation

Grid computing and its evolution into the *cloud* have their main purpose on high performance computing and maximize the effectiveness of the shared resources respectively without compromising the quality offered to users.

At the same time, there is a huge interest in creating large-scale distributed computational and communication infrastructures capable of mimicking real conditions found in the current Internet. An experimental testbed is a platform for experimentation of large projects in development. They provide an open platform to conduct strict, transparent, and replicable testing of scientific theories, computational tools, new technologies, and the next generation large-scale services –content distribution networks, peer-to-peer file sharing or network measurements.

Examples of current large-scale experimental infrastructures are the US-initiated PlanetLab[6] and GENI [7], OneLab2 [8] and Federica in Europe, and AKARI [9] in Japan –known in the European context as Future Internet Research and Experimentation facilities (FIRE). In these environments, researchers are able to test and deploy their new tools without the hazards and risks of testing in a live or production environment.

A key characteristic shared by all these infrastructures is their trend to openness in terms of usage: researchers are free to use the infrastructure as long as they adhere to some pre-established user acceptable policy or norms of good conduct.

The same infrastructure is used and shared by hundreds of experimental applications which at the same time show a highly heterogeneous behavior. Thus, resource allocation and quality of service provided to researchers is a key concern to expose a stable enough infrastructure to conduct repeatable experimentation.

Software and hardware general model

The solutions presented throughout this thesis are based on several assumptions about the type of applications that our mechanisms are designed to deal with, as well as the type of hardware that they run on.

The mechanisms we propose are designed to deal with applications that are able to scale horizontally or *scale out*. In other words, these applications are able to deal with the addition of multiple and independent computers together to provide more processing power.

Examples of such applications may be distributed peer-to-peer applications –like peer-to-peer content distributed networks– where the addition or removal of a portion of the computing elements does not modify the semantics of the application or its behavior, but only its overall performance. The model proposed by map reduce for computing jobs also admit this definition as they are composed of several smaller tasks that can be added or removed from the overall job to be executed by different computing elements depending on the size of the data input.

This restrictions comes from the fact that economic mechanisms applied to computer infrastructures usually involve scaling up or down the resource allocation because of the change in resource supply and demand in highly dynamic environments. Thus, applications need to adapt to such adjustments in a transparent manner.

For the same reason and because applications may need to move from one resource to another depending on the state of the computational economy, the hardware we consider for our scenarios is basically homogeneous in terms of functionality. However, resources may be heterogeneous in terms of hardware characteristics like CPU power, memory, network capacity, or energy consumption.

An example of functional homogeneous hardware are the nodes provided by the PlanetLab infrastructure which are presented to the user as a set of lightweight virtual machines running the same operating system. Another example are the nodes present in most data centers, which may be replaced from time to time due to hardware failures. These resources are usually offered to the user through different abstractions like virtual machines following the Platform As A Service paradigm or computing time slots in the case of the map reduce paradigm. However, the hardware characteristics may differ from one node to another in both cases.

2.2 Economic principles for computational markets: some background

When managing service levels on the scenarios presented above, we would like to make sure that the system cannot be abused by strategic and selfish users, who could starve out competing resource consumers. We review the economic principles behind computational markets that study how mechanisms can be developed to ensure an overall healthy system even with strategic users.

Tragedy of the commons

Free access and unrestricted demand for finite resource ultimately ends up the resources through over-exploitation. This occurs because the benefits of exploitation are received by individuals –each of whom is motivated to maximize the usage of the resource– while the costs of the exploitation are distributed among all those to whom the resource is available. This, in turn, causes demand for the resource to increase leading to the point in which the resource is exhausted and becomes useless. This problem is a clear example of the well known problem of the "Tragedy of the commons" [10] or free-riding.

Consider the example of farmers grazing their cows on a grassy area. This area can support up to one hundred cows. One hundred farmers each bring a cow, and the eating is good enough to provide benefit to all of them. But a strategic farmer may think that bringing an extra cow to the common area must increase its benefit by doubling the income and only putting a 1% drain on the common grassy area. The tragedy comes when all the farmers realize that the common area cannot support such amount of cows. Substitute in this example the common grassy are by a computer cluster, a farmer by a user and a cow by an application, and it is not hard to imagine the same problem from a computational point of view if some incentive is not provided to users to constrain the cluster usage.

Game theory as an analytical tool

Game theory [11] is the study of strategic decision making and a useful tool that allows to make statements about the outcome of a system from the theoretical point of view.

In Game theory, a number of *players* and their possible *actions* with associated individual *preferences* model a *game*. Other players' actions affect the utility or payoff a player receive from a game.

Thus, players receive a certain payoff from choosing an action from their action profile and given the actions chosen by the other players. The key insight provided by game theoretic analysis is which is the optimal strategy or action to take given other players' actions.

When no player can obtain a higher utility by changing the action chosen given that every other player chose an optimal action given their strategy, the game is in a *Nash equilibrium*. It is important to note that a Nash equilibrium does not make any statements about uniqueness of the solution, and many games can indeed have multiple Nash equilibrium. However, it provides an statement about the steady state of a game, when no player has an incentive to change its action.

These strategy profiles which produces the most favorable outcome for a player, taking other players' strategies as given is called the *best response*. This concept is interesting because best response strategies may be implemented by a software agent to act on behalf of a user or application. Thus, an agent can react to changes made by other agents on the game. This situation usually leads to the question of whether the best response strategy is a Nash equilibrium of a given game.

Mechanism design as a design tool

Mechanism design –sometimes called reverse game theory– is a field in game theory studying solution concepts for a class of private information games. The distinguishing features of these games are: from one side a game *designer* chooses the game structure rather than inheriting one; and, from the other side, that the designer is interested in the game's outcome.

Thus, the mechanism designer starts with a set of desirable properties that a specific game should possess and then, a game is constructed so that participants are influenced to behave in a certain way that fulfils such initial properties. Such a game is called a *game of mechanism design* and is usually solved by motivating agents to disclose their private information.

A central property derived from mechanism design is *incentive compatibility*. A mechanism is incentive compatible if all the participants consider their best interest to truthfully reveal any private information inquired by the mechanism. This way, participants of the game have no incentive to lie and try to game the mechanism to its own interest.

Why markets then?

Markets can effectively aggregate information from multiple sources in large, dynamic and complex systems where there is not a single source with complete information. Prices summarize demand succinctly and uniformly and can be communicated globally to other participants. Thus, the vision is that selfishly utility-optimizing resources consumers and selfishly profit-optimizing resource providers will move the system into an equilibrium state where resources are allocated efficiently to applications.

There are a couple of fundamental elements of computational markets that help to meet this vision. First, provide the right incentive to users to specify truthful priorities of their resource needs and, secondly, to dynamically set resource prices to find the level where demand equals supply –i.e. the market equilibrium price.

As we will review on chapter 3, most computational markets use some kind of currency to meet the first element. A currency is *a unit of exchange, facilitating the transfer of goods and services. It is a form of money, where money is defined as a medium of exchange rather than a store of value.* Therefore one function of currencies is facilitation and regulation since each country or region with a currency has some institutions or monetary authority exerting control over the amount of circulating currency – the central bank or the ministry of finance.

Limiting each user to a finite budget of a given currency is the main solution to deal the tragedy of the commons effect. How this budget is initially assigned and how users are able to spend it is a specific matter of each economic algorithm.

Price-setting on the other hand is usually addressed by the use of some form of auction to extract the market price directly from users' bids. Examples

of auctions are the English auction, a Dutch auction, a double auction or a combinatorial auction [12][13][14][15].

Regulation in computational economies

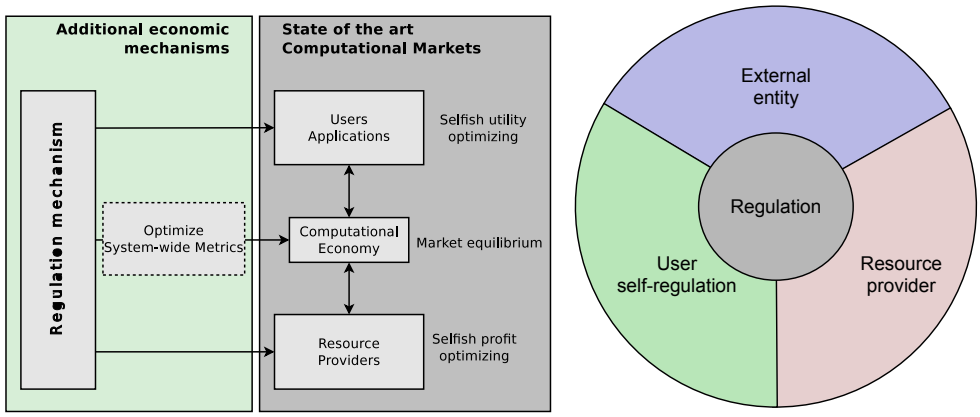
As in real life however, markets may fail to address specific problems not directly sensed by users or resource providers of such markets called *externalities*. An externality in economic terms is defined as the cost which results from an activity and which affects an otherwise uninvolved party who did not choose to incur that cost.

An example of an externality, presented in more detail in chapter 6, is resource congestion in contributive systems like the experimental testbed PlanetLab. Different organizations provide a minimum of two computing servers to the common testbed as a prerequisite to access the open testbed infrastructure. From them on, users of this organization are free to use the whole set of public servers. However, only a subset of them is heavily used –because of reputation, availability, or any other reason– while the rest remain lightly used. This situation leads to an imbalance of resource consumption and contribution which makes some of the organizations support most of the experiments. The ideal situation would be that resource consumption and contribution is equalized to some extent so all participants of the infrastructure contribute the same way to the overall testbed.

The solution used in real economies to cope with externalities is to use some form of *regulation*, a process of the promulgation, monitoring, and enforcement of rules, established usually by the government.

If we turn back our attention to computational economies, the ideal scenario would be to use mechanism design to overcome such externalities and internalize them into the mechanism, so participants of the mechanism are aware of such costs.

However, once a mechanism is designed without considering such costs, it could be extremely complex to redesign it from scratch to account for these externalities. Thus, a potential solution for such systems would be to introduce a new regulation mechanism on top of it that introduce restrictions or limitations harnessing the benefits of the existing economic mechanism.



(a) (b)

Figure 2.1: (a) Research gaps with respect to state of the art and (b) Regulation entities considered

To situate our contributions with respect the state of the art detailed later, figure 2.1 shows the high level entities involved in this thesis. On the right side of figure 2.1(a), users or applications –both terms are interchangeable– obtain resources from resource providers through a market which acts as a scheduler or resource allocator.

Users are assumed to be selfish in the game theoretic sense and always try to maximize the utility obtained. Resource providers are assumed to be selfish as well and try to maximize profit from their resources. As an intermediary, the market-based resource allocation mechanism is in charge of matching resource demand and supply and set the corresponding equilibrium price.

The contributions made in this thesis sit on left side of figure 2.1(a). The regulation mechanisms proposed in this thesis optimize system-wide metrics which, for some reason, are not considered in the initial design of current computational markets.

To this end, we structure our contributions according to the entity responsible for the regulation, or in other words, who is in charge of modifying, limiting or influencing the outcome of the market as depicted in figure 2.1(b):

- *External entity* or third-party regulator. In cases where the computational economy is established, incorporating new metrics to consider into it may not be plausible. Thus, an external entity or regulator would be in charge of monitoring the infrastructure and applying the necessary corrective measures. This would be the role of the government in a real life scenario. We refer the reader to chapter 6 for more details on the subject.

- *Resource provider*. In cases where the presence of a market-based resource allocation mechanism is ruled by the resource provider, it has the authority to modify the behavior of users by, for example, scaling up or down the supply of resources as to modify the price and potentially reducing energy costs. Chapter 7 deepen our work on this matter.

- *User self-regulation*. In this case, we take the mechanism design path to, given a set of external properties that should be accounted by our resource allocator, design a mechanism in such a way that users are given an incentive to care about metrics –energy efficiency– that otherwise are not considered. This contribution is extensively presented on chapter 8.

Chapter 3

State of the art

Market-based algorithms for resource scheduling, or computational economies in general, have been used as a mechanism to allocate resources more efficiently guided by user priorities as far back as in 1968. The work by Sutherland [3] and Nielsen [4] started to consider computational markets as a solution to the resource allocation problem. They already stated similar design concerns –user-driven priorities, funding policies, virtual currencies, price setting, etc.– as systems being build nowadays. However, being the seventies and looking at the technology back then, they did not consider the problems found in current infrastructures due to the huge paradigm shift introduced by parallel computation and how users access and use resources.

GridBus project

Buyya, et al. proposed an economy driven resource management architecture for global computational grids in [16] and more recently in [17, 18]. It consists of a generic framework, called GRACE (Grid Architecture for Computational Economy) for trading resources dynamically, in conjunction with existing grid components such as local or meta-schedulers.

Their aim is to enable supply and demand driven pricing of resources to regulate and control access to computational resources in a grid. The architectural model of resource management is dependent by the way the economic scheduler is structured which may be: (i) centralized scheduling for managing single or multiple resources located either in a single or multiple sites with uniform policies. This solution is not suitable for grid domains as each one is expected

to have different resource management policies; (ii) decentralized scheduling which interact among themselves in order to decide which resource should be allocated to the jobs willing to be executed; and (iii) hierarchical scheduling as an hybrid model (combination of centralized and decentralized) which seems suitable for the federated nature of grid environments.

They propose Nimrod-G [19] as an economic scheduler which may be implemented within this architecture. It serves as a resource broker and supports deadline and budget constraints algorithms for scheduling task-intensive applications. It allows users to lease and aggregate resources depending on their availability, capability, performance, cost and user's QoS constraints. Nevertheless, their algorithms only focus on efficient local allocations among competing users to resources, lacking any kind of coordination between schedulers. Thus, they do not consider the global behaviour of the system taking into account such local decisions made by the economic scheduler.

Therefore, the focus of this work is to provide a framework for computational economies relying on the meta-scheduler model, on which users submits jobs –using a portal– to an economic scheduler which takes the allocation decisions based on the user supplied information –e.g budget and time constraints. Thus, they do not provide any insights on how a large-scale system may behave using this model.

Libra

Libra [20] is a computational economy-based scheduling system for clusters that focuses on improving the utility, and consequently the quality of service delivered to users. Libra is intended to be implemented in the resource management and scheduling (RMS) logic of cluster computing systems, such as PBS [21]. An extension of the previous system [22] focus on dynamically pricing cluster resources by considering the current workload to accommodate supply and demand, aiming for market equilibrium. In addition, Shin Yeo et al. [23] incorporates Service Level Agreements (SLAs) by considering the penalty of not meeting a SLA into the admission control and scheduling decisions in a cluster.

Therefore, the flow of a job in an utility-driven cluster begins with its submission to the centralized gateway, which decides through an Economy-based

Admission Control mechanism whether the job should be accepted or rejected based on its specification. Thereafter, the job is allocated to a certain computing node and monitored.

Although Libra's approach allow more jobs to be completed by their deadline –and consequently less jobs rejected– it critically depends on the quality of the runtime estimation to schedule jobs and resources effectively. Besides, their solution assumes only one centralized gateway to accept submitted jobs compromising its scalability, one of the main objectives of large-scale environments such as the Grid.

Shirako

Shirako [24] is a toolkit for building components of a utility service architecture based on previous work from the same authors called *Cluster on Demand* (COD) [25]. Utility services enable dynamic on-demand sharing of networked resources through programmatic interfaces.

Resource provider sites might export a variety of resources including servers or clusters in data centers, virtual application servers, network storage objects, network attached sensors, or even bandwidth provisioned paths or virtualised routers within the network itself.

Shirako is based on a common, extensible resource leasing abstraction [26] which combines elements of both lifetime management and mutual exclusion. Each offered logical resource unit is held by at most one lease at any given time although providers may choose to overbook their physical resources locally. If the lease holder fails or disconnects, the resource can be allocated to another guest.

This use of leases has three distinguishing characteristics: (i) Shirako leases apply to the resources that host the guest, and not to the guest itself; the resource provider does not concern itself with lifetime management of guest services or objects. (ii) The lease quantifies the resources allocated to the guest; thus, leases are a mechanism for service quality assurance and adaptation. (iii) Each lease represents an explicit promise for resource usage to the lease holder for the duration of the lease. Leases in Shirako are also similar to soft-state advance reservations [6], which have long been a topic of study for real-time network applications.

Although Shirako is focused on flexible mechanisms to trade resources among clients through leases, they lack any coordination mechanism among resource brokers leading to a possible over-provisioning of such resources –as resources are traded through leases– meaning that negotiated SLAs might be broken.

Bellagio

Bellagio [27] is a market-based resource allocation system for federated distributed computing infrastructures. Users specify resource of interest in the form of combinatorial auction bids [28]. Thereafter, a centralized auctioneer allocates resources and decides payments for users. The Bellagio architecture consists of a resource discovery subsystem called SWORD [29] and resource market subsystem.

Regarding the resource market, it uses a centralized auction system, in which users express resource preferences using a bidding language, and a periodic auction allocates resources to users. A bid for a resource includes sets of resources desired, processing duration and the amount of virtual currency which a user is willing to spend. The centralized auctioneer clears the bid every 8 hours. The amount of virtual currency owned by a site is directly determined by the site's overall resource contribution to the federated system.

As a resource allocation algorithm, they employ SHARE [30] which clears a combinatorial auction –known to be a NP-complete problem– by using approximation algorithms for winner determination. SHARE uses a *threshold rule* [31] which forces users to reveal their true value for goods when bidding. Nevertheless, their study focuses on the allocation of resources in a federated environment using a strategy-proof auction using centralized mechanisms.

Although the allocation achieved is proportional to the bids advertised by users, this solution is not really well suited for large-scale systems as its scalability is compromised. Besides, they do not consider the global effect in the system of such centralized mechanism.

Tycoon

Tycoon [32] is a distributed market-based resource allocation system. Application scheduling and resource allocation in Tycoon are based on decentralized

isolated auctions. Every resource owner in the system runs its own auction for his local resources. Although made popular at the beginning of this century, this system was largely based on one of the first successful implementations of a distributed computational market called Spawm [33], which was the first to propose a proportional share auction for this kind of infrastructures.

Auctioneers implement a proportional share-based auction in which users receive a certain amount of a single resource (CPU and memory) proportional to the bid they advertise with respect to other bids placed by other users in the same resource.

They use virtualisation techniques such as Xen [34] to share a single resource. They also implement a centralized banking service to manage the virtual currency in the system. Users are endowed with a fixed amount of currency which they are able to spend over time to allocate their tasks.

Tycoon research interest is focused on the proportional allocation of local resources in a flexible way allowing users to trade off their preferences for low latency –e.g a web server–, high utilization –e.g batch processing–, etc. Nevertheless, they do not take into account coordination mechanisms of independent markets –one per resource– neither consider the effect of their mechanisms in the macroeconomic behaviour of the system as a whole.

A game theoretic analysis on the aforementioned proportional share mechanism is presented in [35, 36] where they provide an algorithm to find the *best response*[1] of an agent to the system. Given a fixed budget X and a pool R of divisible resources, their algorithm finds the distribution of bids across resources that yields the highest utility for an individual player i. This work is very interesting as it provides bounds on the efficiency of the proportional share resource market and provides an algorithm for automated agents which optimises user's utility allowing the abstraction of the market mechanisms from end users.

Other works

Popovici and Wilkes [37] examine profit-based scheduling and admission control algorithms called First Profit and First Opportunity that consider a scenario

[1] In game theory, the best response is the strategy (or strategies) which produces the most favourable outcome for a player, taking other players' strategies as given.

where service providers rent resources from resource providers who then run and administer them. Clients have jobs that need processing with price values (specifically, a utility function) associated with them. The service provider rents resources from the resource provider at a cost, and the price differential is the job's profit that goes to the service provider. However, resources may be over-provisioned considering this scenario and the uncertainty in resource availability. If the service provider promises resources they cannot deliver, the clients' QoS targets will not be met and the price they will pay will decline, as defined by the client utility function. It is assumed that service providers have some domain expertise and can reasonably predict running times of jobs in advance.

G-commerce [38] is a resource allocation system based on the commodity market model where providers decide the selling price after considering long-term profit and past performance. It is argued and shown in simulations that this model is able to achieve better price predictability than auctions. The main issue of their proposal is the necessity for long-term previous study of their system behaviour to decide optimal price settings. Besides, they lack the ability to quickly self-adjust prices to sudden changes in market conditions –e.g high imbalance between supply and demand.

In the context of completely decentralized systems, PeerMart [39] implements distributed auctions by associating a broker for each resource or service type. Brokers are implemented as peer-sets on a structured P2P overlay. Synchronization of peers in the set and detection of malicious behaviour is necessary to avoid peers' collusion when modifying user's accounts, although it introduces significant message overheads. Furthermore, auctions are clearly influenced by market forces and contention against a single resource as similar resources may end up with very different prices in different auctions. Therefore, this segmentation of markets puts the onus on the clients to choose between equivalent resources although this framework does not provide any means to decide which resource fits better its necessity. Finally, they propose the existence of a virtual currency although do not provide any insights on how it should be managed.

Filling the gaps

Traditional schedulers offered by computing farm jobs of HPC clusters –e.g. FIFO, SJB, different queueing mechanisms, etc. – focus on optimizing throughput, usage or response time as general proof of goodness of the performance delivered.

Instead, the work presented in previous sections focus on the QoE (Quality of Experience) offered to the users. As a result, the design decisions to solve the resource allocation problem focused on user-centric priorities are a first order metric to evaluate the goodness of the proposed system. Thus, markets are a good abstraction to capture user priorities for their jobs and a good fit to solve the resource allocation problem. Their key characteristic is the abstraction provided to users to optimize their efficiency by aggregating the information of multiple objectives –resource selection or job priority– to a single metric, the price.

However, while the micro-economic issues applied to resource allocation have been widely explored in current and previous work, other system-wide metrics like energy efficiency or load balancing remain out of the scope of those approaches because of their design principles and the inability of the proposed computational markets to incorporate system-wide metrics. As a result, users are not able to sense these problems when making their economic decisions.

Thus, the research gaps identified with respect to previous works, which sets us apart from the state of the art as well, can be summarized as follows:

- There is a need to include a larger range of system-wide metrics into the computational economy to optimize multiple system objectives when traditional market-based mechanisms fail to do so –load balancing and energy efficiency as examples– without detracting from a key characteristic of markets, e.g. optimize user efficiency in the presence of selfish agents.
- There is a need to explore the feasibility of regulation mechanisms from different perspectives, be it an external entity or providing users tools to self-regulate themselves.

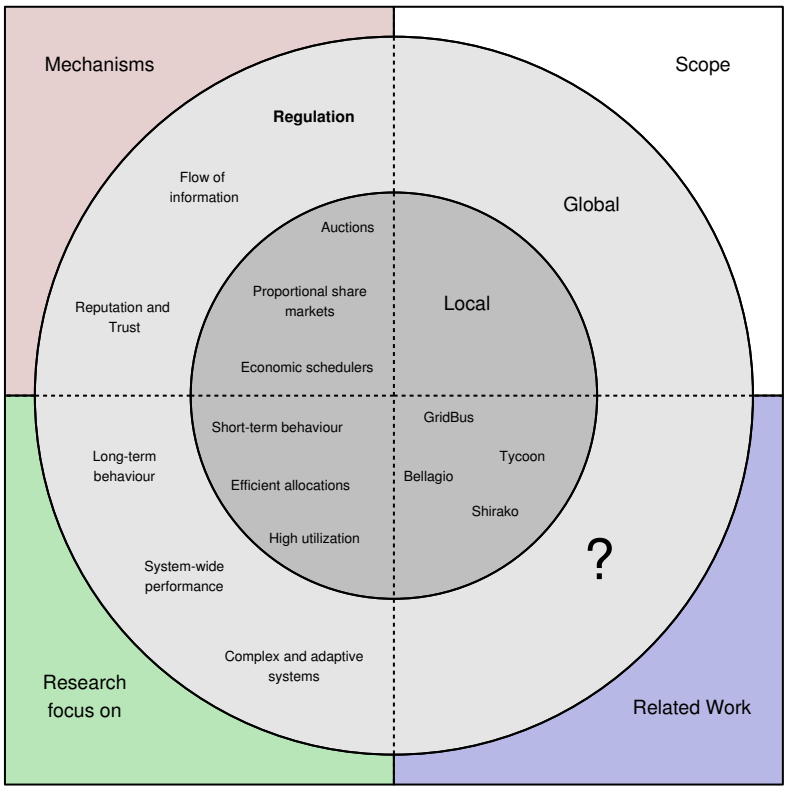


Figure 3.1: Research gaps with respect to state of the art

Chapter 4

Summary of contributions

In this chapter, the main research questions are presented. Each set of questions is then discussed and the related contributions are summarized according to the approach we chose. The specific contributions of this thesis are presented in detail in chapters 5, 6, 7 and 8.

Modeling resource usage in shared infrastructures

Q1: What phenomena can we find in large scale infrastructures when resources are shared and finite? What is the usage pattern of resources? What are the root causes for such behaviours? Can we characterize the underlying nature of interactions among distributed applications in a shared infrastructure?

In order to successfully capture, model and present the existing problems in the area of resource allocation in large-scale infrastructures, we decided to study in depth one of the largest open operational infrastructures nowadays. The selection of PlanetLab as such was mainly driven for its publicly available set of traces which span hundreds of nodes and thousands of active users.

We believe that first answering these questions is crucial in the design of resource allocation and regulation policies. Previous analysis in the scientific literature [40][41][42] focused on studying issues from the node point of view like resource contention or failure characterization. Instead, little attention has been paid to the processes creating these behaviours –i.e. users or applications using these networked nodes – and the study of their root causes. Thus, while most of the literature focus on *what* problems can we see on a shared

infrastructure, we focus on *why* these problems occur and how economic regulation policies may solve some of the known issues.

In our detailed study presented in chapter 5 and journal paper [P1] as well, we identify several behaviours emerging from an infrastructure with finite resources which are shared by different users.

First, we find that the Pareto principle, also known as the 80/20 rule [43], holds in this kind of infrastructures. Few applications running on the infrastructure –usually long-running services like content distribution networks– monopolize almost all the resource consumption.

Besides, this phenomenon jointly with the imbalance between resource consumption and resource contribution to the public and open infrastructure –the resource consumption made by roughly 30% of the participating entities accounts for the contribution made by the other 70%– is a clear evidence of the well-known problem of the *tragedy of the commons* or *free-riding*: the benefits of exploitation are received only by individuals (applications) whereas the costs are distributed between all those who provide access to the resources (entities contributing resources).

This situation is a clear consequence of the lack of two characteristics: the lack of incentives for resource providers to supply better resources in terms of quantity, quality and availability; and the lack of proper regulation policies to control and limit the access to resources in an open infrastructure.

Another interesting finding was that users' choice of resources is mainly driven by the reputation of the entity which provides the nodes and past interactions with those nodes. Thus, characteristics like locality or current load are dismissed by users when deploying new applications. This situation appears due to the lack of more complete information about resource characteristics in current resource discovery tools.

Regulation through taxing resource pricing

Q2: Can third-party regulative entities moderate the behaviour of large scale systems with economic mechanisms? Which regulation mechanisms might be necessary?

As we have introduced, the combination of resource-intensive applications and user preferences can lead to levels of demand that saturate the infrastructure, negatively affect other users or compromise the overall stability of the system. In addition, resource congestion usually affects only a subset of the infrastructure, known as hot spots, whereas the load remains low over the rest of the system.

While policies and algorithms based on micro-economic principles have proven to be a decentralized, scalable and efficient way of allocating resources according to user preferences, they can also lead to system-wide performance penalties and they usually neglect the economic externalities associated to such allocations.

In this case, the externality we focus on is the resource congestion suffered by a subset of the infrastructure which may be present even with the introduction of micro-economic algorithms mainly as a result of correlated preferences for resources. As in the real economy, where free-market mechanisms sometimes fail to address externalities, central governments impose restrictions (regulations) on the system that are designed to act as incentives to influence users to behave in a certain way.

In this case, given a computational market which suffers from resource congestion, we propose a third-party regulative entity which oversees and monitors the system and applies a tax mechanism to increase the price on hot spots and encourages users to bid for alternative less used resources.

In chapter 6 and the journal paper [P2], we show that this mechanism is able to influence user decisions to move their workload from highly-used nodes to other less congested nodes, effectively internalizing the problem of resource congestion.

Although this work is mainly focused on resource congestion, other high level decisions by the regulative entity could be achieved. For example, the availability of certain nodes can be promoted by imposing taxes on those resources with low availability. A reduction in the overall energy consumption of a system can be induced by imposing taxes to those resources with high energy costs effectively discouraging users of its use and encouraging resource providers to deploy more energy-friendly hardware.

Resource providers as regulators

Q3: Can resource providers provide a resource allocation that fulfil an objective (e.g. energy-efficiency) without limiting the choice of resources by introducing restrictions or limitations? Which information is valuable to resource providers to perform this regulation?

We have seen how third-party entities to the computational market may introduce regulation mechanisms to overcome certain limitation of free-market economies. However, it is not always feasible to rely on external entities to perform such regulation due to architecture limitations, lack of control of resources or absence of information to perform such regulations.

Thus, we focus our attention on how resource providers can regulate the access to resources without limiting the freedom to choose specific resources according to user preferences using micro-economic algorithms. In this case, the objective of the regulation mechanism is to reduce energy consumption and cut down costs which is a key concern in networked computing systems like current data centers. Following that line of research, our study presented in chapter 7 and publications [P3, P4] contains two parts.

In the first part, we derive an energy model for computing elements (servers) using real data and we observe that the energy consumption when servers are idle is not negligible compared to the energy consumption which is proportional to the load. According to our model, resource providers have an incentive to keep unused resources shut down if they are not necessary to meet users requirements.

In the second part, we model the interplay between the need for resources from users and the goal to minimize power consumption using a game theoretic model based on the Stackelberg game. This classic game is structured at two levels. The first level is the infrastructure operator acting as the leader and determining the resources to keep switched on and off. And the second level is the set of strategic users buying resources as followers.

Following this model, we derive a regulation algorithm that allows resource providers act as a regulative entity. This algorithm computes the minimum amount of resources needed by applications to meet their objectives. Thus, the resource provider is able to shut down exceeding resources without incurring

in penalties for not satisfying the service level agreements of participating users.

In this work, we show that resource providers need to have specific knowledge about users' requirements in terms of parallelism and resource capacity to produce optimal energy savings. However, one limitation of this approach is that this information may not be available to the resource provider or may not be completely accurate which makes the allocation decision incur in the risk of being sub-optimal.

Economic incentives for the self-regulation of users

Q4: Can we reach an optimal resource allocation (finish jobs on time with minimum resource usage) with the collaboration of users if they can exchange unused resources at any time? Is this incentive enough to help users self-regulate the usage of resources?

Throughout this chapter we show how regulation can be carried out by third-party entities external to the computational system and by the resource providers themselves. We have seen that for resource providers to make smarter allocation, they need to have information about the resource requirements of applications, but with the drawback that this information may be not available or may no be truthfully reported by users trying to game the mechanism.

Thus, we turn our attention to self-regulation: on the design of an incentive mechanism in which users are provided an incentive to truthfully report their resource requirements to the resource provider.

In chapter 8 and paper [P5], we show in detail how our incentive mechanism pursues two different goals from the regulation point of view. From the perspective of the user, we show that our mechanism effectively encourages users to report their true capacity requirement for a given job. Our mechanism is based on providing a shares market in which users engage to dynamically scale up (buy) or down (sell) the allocated time slots of a job without intervention of the infrastructure operator. We also show that it is in the best interest of the user to participate in such a market. This way, we promote the regulation of resource allocations made by the users themselves.

On the other side, the proposed dynamic shares market obtains valuable private information from users which allows the resource provider to shut down a portion of the infrastructure to reduce energy-related costs without reducing the quality of service provided to users.

Our main result from the incentive mechanism analysis and evaluation is that users are able to find an optimal resource allocation which allows them to meet their deadlines with minimum resource usage. We also show how the incentive mechanism we propose is incentive compatible in the sense that it is always on the users' best interest to report their true resource needs, effectively promoting the self-regulation of resource access. Our evaluation shows that our shares market provides better user efficiency with a lower resource consumption compared to well known and widely used schedulers, at the cost of slower completion times, clearly indicating the trade-off between job execution throughput and quality of experience.

Chapter 5

Modeling Resource Usage in Shared Infrastructures: PlanetLab's Case Study

Abstract

Understanding how different applications with different resource requirements interact in a shared infrastructure like Internet is crucial in the design of resource allocation and regulation policies. In this chapter[a], our focus is on analyzing a well known planetary scale experimental facility as a case study –PlanetLab. Previous analysis focused on the workload characterization from the node point of view, while little attention has been paid to the processes leading to such workload –i.e. distributed applications. In particular, our aim is two-fold: i) characterize the underlying nature of interactions among distributed applications and understand the interference or mutual influence of one slice on other slices. And ii) characterize important aspects of applications' resource usage, the short-term distribution and temporal dynamics of CPU, memory and network usage made by experimental applications. Based on the analysis of publicly available traces, we find that the distribution of resource usage is highly skewed –with a few applications producing most of the resource usage. We use our findings to develop a model that produces good matches in the metrics studied, allowing us to generate a workload with similar statistical characteristics.

[a]This chapter is based on the paper [P1]

5.1 Introduction

During the last few years, there has been a huge interest in creating large-scale distributed computational and communication infrastructures capable of mimicking real conditions found in the current Internet. These test beds are intended to help in the development, deployment and evaluation for the next generation of large-scale services –content distribution networks, peer-to-peer file sharing or network measurement. Examples of current large-scale experimental infrastructures are the US-initiated PlanetLab[6] and GENI [7], OneLab2 [8] and Federica in Europe, and AKARI [9] in Japan –known in the European context as Future Internet Research and Experimentation facilities (FIRE).

These infrastructures are growing to significant size, as well as geographic and administrative diversity. As a result, they are working towards a global environment through federation agreements much like Autonomous Systems (AS) in Internet, which are expected to evolve in the following years.

As the size and scope of federated infrastructure grows, important challenges need to be addressed, including resource discovery, scheduling and resource allocation policies, reliability among mutually distrustful users and organizations or the global government, and sustainability of the infrastructure. Clearly, the appropriate mechanisms and policies highly depend on the usage characteristics of applications running on the systems under consideration.

Therefore, understanding how different distributed applications with different resource requirements interact in an unregulated shared infrastructure is a key challenge for analyzing and developing appropriate mechanisms for resource allocation, scheduling and regulation policies for the overall infrastructure. The results presented in this chapter might be interpreted as a potential evolution of current unregulated systems like experimental facilities, which do not exercise any explicit control over resource consumption.

Previous works, discussed in the related work section, have focused on a per-node study of resource usage to understand the workload supported by the infrastructure –i.e. the *what*. However, they completely leave out the underlying characteristics of applications and, therefore, the information and modeling of the processes generating such workloads –i.e. the *how*. In this

work, we complement the current knowledge of large-scale infrastructures by developing a model of the applications' resource usage.

This work is based on a large collection of data obtained from PlanetLab, the largest and perhaps most used of current experimental infrastructures. Our approach starts with an investigation of global phenomena related to the behavior of infrastructure users that may have a potential effect on its performance. Furthermore, we conduct an in-depth analysis of current applications running on the infrastructure at different times and different aggregation scales. Our goal is to construct a theoretical framework to capture the distribution and dynamics of the workload generated by applications.

The motivation behind this work is two-fold: i) the novel metrics presented and described in Section 5.3 related to the interactions between applications and the underlying resource infrastructure –i.e. consumption versus contribution, the reciprocity metric and resource selection goodness– could be used by other works to compare different mechanisms related to resource selection and allocation. In addition, these results could be used by the developers to anticipate the problems of a new shared infrastructure if no actions are taken regarding the problems presented here; and ii) the models presented in Section 5.4 could be used by other researchers developing new allocation and selection mechanisms in the initial steps of their implementation through simulation, allowing them to perform speculative analysis of their solutions by tweaking model parameters.

Thus, our major contributions can be summarized as follows:

- We identify and study many global phenomena inherent to open shared experimental infrastructures that may produce an effect on the usage of computer and network resources and the quality of service offered by nodes to applications. Specifically, we look at the balance between the contribution and consumption of resources, at the reciprocity in resource exchanges among sites, at the mechanisms to promote or limit resource usage in beneficial ways, and the effects of resource discovery and selection. Results extracted from PlanetLab should not be extrapolated to the current and future Internet. However, they show several interesting problems which could arise in the future if no proper regulation mechanisms are provided to control resource usage.

- We model applications' resource requirements at different scales: first, we analyze the distribution of resource usage among applications; second, the resource usage of each application among its selected nodes; third, the temporal dynamics of applications; and fourth, the periods of activity of the experiments. Taken together, these models allow us to simulate synthetic traces of different types of applications with similar statistical properties as the workload found in a real environment and do a what-if analysis by tweaking model parameters, something that was not possible with previous models.

The rest of the chapter is organized as follows. Section 5.2 provides an overview of PlanetLab and describes the data set used in this study. Section 5.3 presents our analysis on global phenomena related to resource allocation. Section 5.4 describes our analysis of the distribution and time dynamics of resource usage as well as our stochastic model that captures such behaviors. We review related work in Section 5.5 and conclude in Section 5.6.

5.2 Overview of PlanetLab as a Planetary-Scale Shared Infrastructure

The main purpose of planetary-scale experimental facilities is to support the deployment and evaluation of large-scale applications and network services in a geographically distributed overlay of nodes.

Figure 5.1 shows a simplified view of the slice-based facility architecture of current experimental facilities, in which a set of nodes owned by different sites (universities, research centers or companies) are shared among a basic abstraction called a *slice*[6]. A *slice* is a set of nodes on which the applications (networked experiments or services) receives a fraction of each node's resources through a virtual machine called a *sliver*.

From the researchers' point of view, this distributed virtualization architecture allows services to run in an isolated environment of PlanetLab's global resources. Under the hood, each slice competes with the other slices in every node to acquire a share of a node's resources. Currently, scheduling in a node is done through a *fair share* scheduling policy in which each slice has the right to use

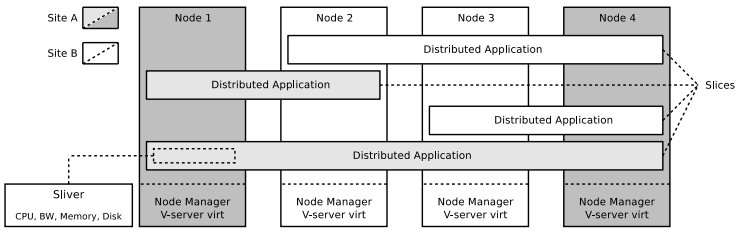


Figure 5.1: Simplified view of PlanetLab

an equal proportion of a resource –i.e. CPU or bandwidth– no matter the number of processes running within a sliver.

From the sites' point of view, each site must provide at least two networked nodes to join the consortium and use the infrastructure. Researchers are then able to manually choose which nodes will belong to their slice. The information provided to researchers about node characteristics is just technical and static information about CPU clock rate, memory available, bandwidth limits, etc. However, dynamic information like availability and load are not currently provided through the selection interface and researchers need to use other tools like CoMon to learn these parameters. Once the nodes are manually selected and initialized, research teams from a site may create a slice without restrictions on the number of slivers and deploy, test and run their applications on any number of PlanetLab nodes.

Thus, we can view PlanetLab as a set of *sites* contributing a set of *nodes* and consuming resources from the global infrastructure through their registered *slices*.

5.2.1 Data set description

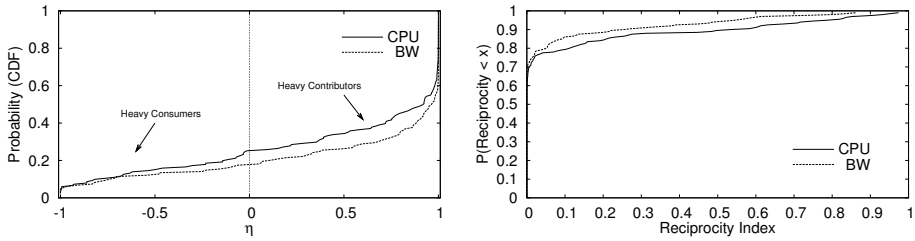
The raw data we use in this study was obtained from CoMon [44], a large-scale monitoring infrastructure already deployed in PlanetLab. CoMon queries each node every five minutes to maintain a historic repository of two different datasets: i) the *monall* dataset which contains node-centric information (load average, I/O performance, network usage, memory, among others) and, ii) the *topall* dataset which contains slice-centric information (CPU, network and memory usage).

In this study, we focus on the slice-centric dataset (topall) to obtain the activity of each slice regarding its resource usage (CPU, memory and bandwidth) for each of the nodes in which a slice is registered. Unfortunately, data regarding disk usage (I/O performance) is not available in any dataset although it might be of interest for increasingly popular data intensive applications –e.g map-reduce based applications.

Thus, for each slice and for each sample (with a granularity of 5 minutes) we maintain a list of nodes in which the slice had activity. Specifically, we keep track of the resource usage about network (in Kbps), CPU (in percentage) and memory (in MB). Additionally, we maintain the list of nodes belonging to PlanetLab with their location –i.e. latitude and longitude– and the site they belong to. To provide an overview of the volume of information analyzed, our measurements show an average of over 134 sites managing 917 nodes spread over the world with an average of 276 active slices executing at any point in time.

However, there are a few caveats with the quality of the data in this study. There are nine days missing or with erroneous information in the datasets due to problems with the monitoring infrastructure because of updates in PlanetLab's underlying infrastructure. For this work, we have removed these days to avoid misinterpretation of results that should have a low effect on the overall measures, as the erased days are small compared to the whole trace. However, we consider this fact when choosing the set of days we use for our temporal analysis, and chose an interval of continuous data so as not to destroy temporal dependencies in our analysis.

Furthermore, a node may not reply to CoMon's queries although it is up and running. A plausible reason for this is that a node may suffer from sudden increases in load at certain intervals, leading to a slow response to queries (triggering a timeout). However, a failing node should have a low effect on the overall measures, as we are dealing with large-scale applications of up to 600 slivers. Finally, we do not have detailed information about slice activity. In other words, we may know looking at the traces that a slice is using a certain amount of a resource but we have no information about the nature of such activity such as whether this activity corresponds to an initial deployment and testing, to an experiment, or to a production service activity.



(a) Consumption vs contribution

(b) Reciprocity

Figure 5.2: Global phenomena evaluation. Subfigure 5.2(a) shows the ratio of consumption vs. contribution of sites participating in PlanetLab. Subfigure 5.2(b) shows the degree of reciprocity among sites. The length of the trace used in these analyses is 8 months (the whole trace available).

We have collected and analyzed the aforementioned data for a period of 8 months (from April to November) in 2009. However, we use different subsets of these traces of different lengths to simplify computation in some cases.

5.3 Measurement of Global Phenomena

Through this section, we focus our attention on global phenomena of interest related to resource allocation in large-scale infrastructures like the balance between the contribution and consumption of resources, the reciprocity in resource exchanges among sites, the mechanisms to promote or limit resource usage in beneficial ways, and the effect of resource discovery and selection.

5.3.1 Im/balance Between Resource Consumption and Contribution

Collaborative experimental facilities rely on the individual contributions of resources (networked computers) from participants (sites) to create a distributed infrastructure powerful enough to allow them to run large-scale experiments and services.

A key characteristic of this kind of shared infrastructures is its peer-to-peer nature, as participants are both resource consumers (through the execution

of distributed applications in a slice) and resource providers (through the provision of nodes associated with a site) acting on their own interest.

Thus, we measure the degree of resource consumption and contribution in such environments with the normalized ratio shown in Equation 5.1. For each site i, we aggregate the total amount of resources consumed by its slices (c_i) and the total amount of resources contributed by its nodes (p_i) during the whole measurement period considering CPU[1] and bandwidth[2].

$$\eta_i = \frac{p_i - c_i}{p_i + c_i} \tag{5.1}$$

This normalized ratio allows the relative comparison among sites with different volumes of resource usage. Notice that sites consuming a certain amount of resources from their own nodes are not accounted for in the final result as both terms (p_i and c_i) are canceled in the numerator. Values close to 0 reflect balance between resource usage and contribution, whereas imbalance is represented by values close to 1 (excess of contribution) or -1 (excess of consumption).

For example, a site executing an application in 10 nodes that consumes 20% of CPU ($c_i = 10 * 20/100 = 2$ CPU powers) and contributing three nodes which CPU usage is 20% ($p_i = 3 * 20/100 = 0.6$ CPU powers) will have a $\eta_i = -0.54$ which indicates that this site has consumed more resources than contributed.

As shown in Figure 5.2(a), around 30% of sites have a negative contribution metric. This means that resource consumption made by this 30% of sites accounts for the contribution made by the other 70% of sites. This is a clear example of the well-known problem of the *tragedy of the commons* or *free-riding*: the benefits of exploitation are received by individuals (sites) whereas the costs are distributed between all those who have access to the resource. With the expected increase of usage in the coming years, the overall stability of the infrastructure will be compromised if no proper incentive mechanisms to

[1]The unit of measure is CPU powers which is the accumulated percentage of usage of nodes divided by 100 to obtain the equivalent number of CPUs used. For example, a slice using 5 nodes at 20% of usage each, will have a consumption of 1 CPU power equivalent to 1 fully used node.

[2]The unit of measure is the total bytes transmitted and received.

contribute resources are provided as, obviously, it is much easier to consume than contribute resources considering operational costs.

This current imbalance of resource consumption and contribution is the result of the policies currently present at PlanetLab in which, for example, sites are required to provide only two networked nodes in order to use the whole infrastructure. This policy leads to a minimal contribution of resources in contrast with a potential unlimited consumption.

To solve this problem of unlimited consumption, the current design and implementation in PlanetLab includes limits on resource consumption, but this is only a mechanism to stop intended or unintended clearly abusive behavior from one slice or sliver when limits are reached. For example, PlanetLab's policy is to allow the creation of up to 10 slices per site, with an average bandwidth of 1.5 Mbps per slice. Moreover, the application consuming most of the memory available on a node is killed when memory reaches 90% of utilization.

In contrast to hard limits, incentive mechanisms can act on applications by sending signals or establishing dynamic limits proportional to the actual resource contribution of a site that promote or limit the behavior of applications. These mechanisms should encourage application developers and operators to optimize applications to be more "environmentally friendly" by reducing their consumption of resources and increasing the availability of the contributed resources. Some of these mechanisms to control resource consumption by providing incentives might be market-based [45][46] in which resource owners earn money (either real or virtual) by contributing resources and spend that money using available resources. As a promising way to control resource consumption and thus improve the imbalance between consumption and contribution, we are currently studying different incentive policies suitable for open and shared infrastructures.

5.3.2 Degree of Reciprocity in Resource Exchanges

We are also interested in the relationship between sites in terms of resource exchanges to introduce proper incentive mechanisms adequate to these exchanges.

Thus, we turn our attention to the degree of reciprocal exchange among sites. This measure is motivated by the fact that common decentralized peer-to-peer incentive mechanisms are based on some kind of bargaining protocol in which reciprocal exchanges are necessary.

To evaluate the degree of reciprocity in resource exchanges, we compute a matrix C where $C_{i,j}$ is the consumption of every slice belonging to site i executing in any node of site j. Thus, the reciprocity metric $\overline{r_i}$ is computed for each site i following Equation 5.2, where S is the set of sites.

$$\begin{aligned} r_{i,j} &= \frac{\min(C_{i,j}, C_{j,i})}{\max(C_{i,j}, C_{j,i})} \\ \overline{r_i} &= \frac{1}{|S|} \sum_{\forall j \in S \setminus \{i\}} r_{i,j} \end{aligned} \tag{5.2}$$

Values closer to 1 indicate a higher degree of reciprocal exchange. Notice that we do not consider the reciprocal exchange for the site i itself because it would be always equal to one. As shown in Figure 5.2(b), 80% of sites have a reciprocity value very close to 0. This result is not intrinsically related to the PlanetLab resource allocation scheme but a consequence of the resource usage and node selection made by researchers –which are currently not provided with this reciprocity information.

This result shows that incentive mechanisms based on bargaining protocols like Bittorrent's tit-for-tat [47] might not be applicable in these kind of environments because there is no stable bilateral exchange between sites. Thus, mechanisms based on transferable rights [26][30][45] might be more adequate.

5.3.3 Resource Selection Impact

Finally, one of the main tasks researchers have to do before conducting their trials is the resource discovery and selection process. An inadequate selection of nodes may distort results and misguide conclusions from experiments.

Currently, node selection is done manually based on static criteria such as location or bandwidth limits. However, results may be highly influenced by

dynamic node attributes like load or availability. This information is currently not provided to users in the selection interface –although there are currently efforts to integrate such information through the MySlice tool– and they need to learn these characteristics by other means. Usually, researchers use the CoMon tool although it only presents current load and, therefore, does not provide historical information like availability and past load.

To assess the quality of choices made by researchers, we computed the average node load and availability from the dataset[3] during one month (April 2009) to obtain enough long-term coverage of such attributes[4]. This way, we evaluate if better resource choices could be made with such available information. Thereafter, we obtain the list of nodes used by each slice during the subsequent day (May 1st, 2009). From the list of nodes of each slice, we consider a researcher has made a *bad choice* selecting a node A if there is another node B nearby[5] which satisfies the following conditions: i) node B is less loaded than A; ii) node B provides higher availability than node A; iii) both of the previous conditions exist at the same time.

We consider the geographical proximity of nodes because of the lack of more complete information in the dataset –e.g. inter-node latency. In the context of PlanetLab, geographical proximity is a reasonable measure considering an academic network infrastructures like universities (nearby node location) because of their highly hierarchical structure[6].

Thus, the quality of choices b_i made by a slice i is defined by Equation 5.3, where N_{bad} is the number of *bad choices* and N is the number of nodes used by a slice.

$$b_i = \frac{N_{\text{bad}}}{N} \tag{5.3}$$

[3]The raw dataset indicates, at a given sampling time, whether a given node is available by providing its usage information or the node is down by omitting such information.

[4]At the time of the analysis of this metric, we only had one month of the raw dataset available, enough to compute a fairly accurate measure of historical load and availability.

[5]Using the latitude and longitude of each node, we can measure its distance considering Earth's curvature. We consider a nearby node if it is located at less than 100km from the original node.

[6]A nearby node in PlanetLab is likely to be owned by the same operator with similar network characteristics in terms of bandwidth and inter-node latency

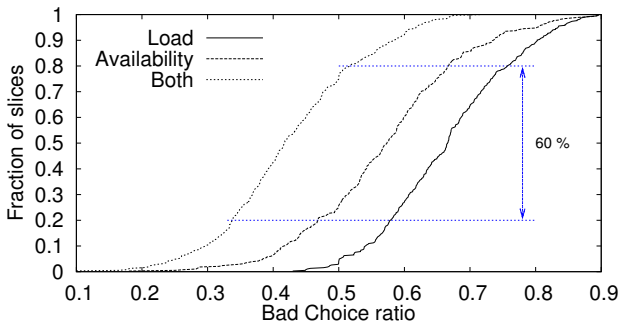


Figure 5.3: It shows a measure of how good node selection is made by slices. The length of the trace analyzed is one month (April 2009).

Figure 5.3 shows the empirical cumulative distribution function (CDF) of the bad choice ratio for each of the three conditions presented above. It shows how 60% of slices have a percentage of bad choices between 35% and 75% of the overall selected nodes. These percentages would be reduced by using current selection services deployed in PlanetLab like SWORD [48] or those already planned like Raven [49]. Moreover, resource discovery in the context of long-term experiments should not be understood as a one-shot process but rather as a continuous process with live migration capabilities in which resources are continuously monitored and application components redeployed to enable the adaptation of applications to changes in resource supply.

5.4 Resource Usage of Slices

So far, we have defined and presented interesting metrics relevant to resource allocation in networked experimental infrastructures. Throughout this section, we evaluate and present models for applications' resource usage responsible for the observed behavior. In this case, we use standard statistical techniques to build a model useful in performing synthetic but realistic workload simulations.

We first examine the long term distribution of resource usage (over the entire measured period) of all slices. Figure 5.4 shows the CDF of resource usage ranked by CPU and bandwidth consumption. We observe in both cases a highly skewed distribution; the top 35% of slices are responsible for 99.9% of the resource usage in this period. Thus, to simplify computation without loss

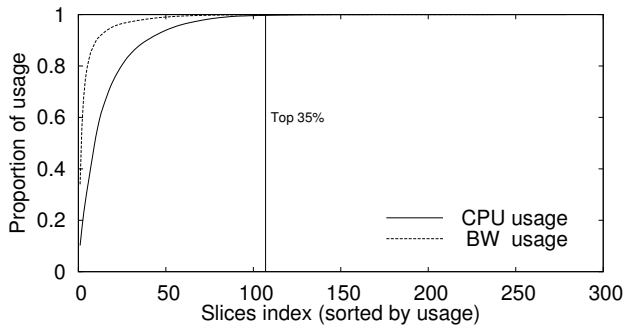


Figure 5.4: Most of the resource usage is made by a small number of applications or slices. We used the whole trace to compute the proportion of usage made by slices (8 months).

of generality, we focus only on these high consuming slices for the rest of the chapter.

5.4.1 Short-term Distribution of Resource Usage

We next focus on the short term distribution of resource usage with respect to the attributes we are studying –i.e. CPU, network, memory and number of nodes. Thus, through this section, we examine two different aspects: i) short-term distribution of resource usage among individual slices (*inter-slice*) and ii) short-term distribution of resource usage of slivers within a single slice (*intra-slice*).

Inter-slice distribution: First of all, we examine the inter-slice distribution of resource usage at different points in time across all slices. For each slice, we compute its *characteristic resource usage* at a given point in time as the arithmetic mean among all slivers belonging to a slice. Interestingly, we find the distribution of each type of resource nearly invariant over different points in time considering a single day. Figure 5.5 illustrates the CDF of the characteristic resource usage of a slice in terms of CPU, bandwidth, memory and the number of slivers using 1 hour aggregation granularity at different points in time (12-1 AM, 5-6 AM, 12-1 PM and 5-6 PM in the Eastern Daylight Time (EDT) which is the time zone used by the traces). We find the curves very close to each other, meaning that independently of the time of the day, it

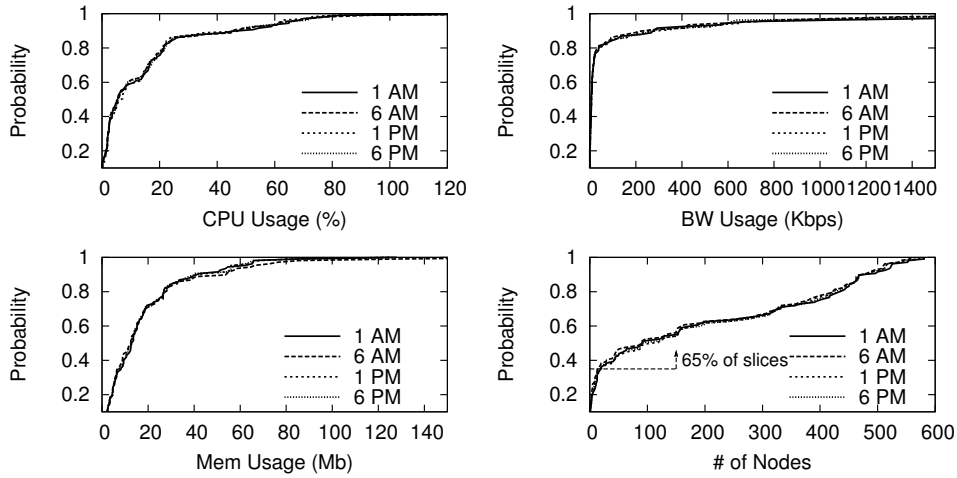
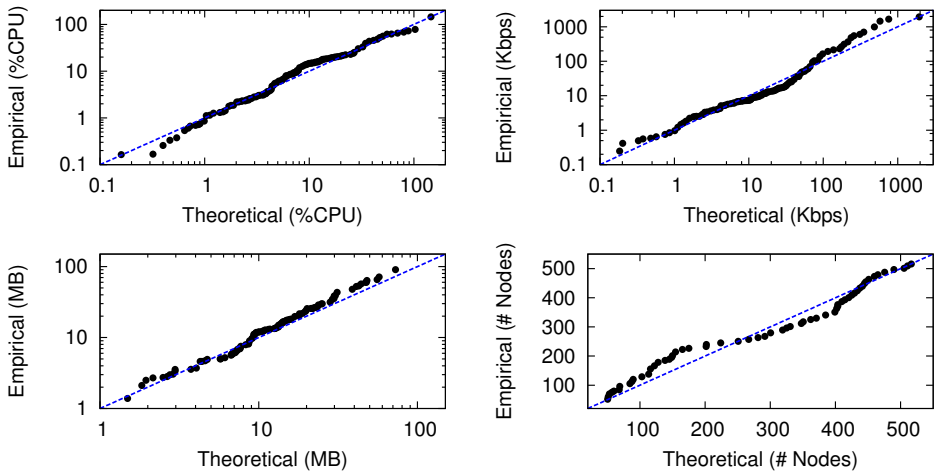


Figure 5.5: Hourly distribution inter-slice. We used one day of the dataset for this analysis (April 1st, 2009).

follows the same distribution. We stress that the nearly invariant distribution function does not necessarily imply that the resource usage for a given slice is stable during the whole period as some of them present strong daily patterns. We will turn to the temporal dynamics in Section 5.4.2.

The distributions shown are highly skewed and with a long tail for CPU, bandwidth and memory, giving us a hint that those distributions might be represented by a heavy-tailed distribution. To determine the model parameters that best describe the data trace analyzed, we apply a data fitting procedure against well-known heavy-tailed distributions –i.e. lognormal, Weibull and Pareto distributions– for each one of the attributes based on maximum likelihood estimation (MLE) [50]. From the fitted distributions, we select those with a lower standard error. For CPU, bandwidth and memory, we find that the best fitted model is the lognormal distribution. In the case of the number of slivers belonging to a slice, none of the heavy-tailed distributions presented a good match. Thus, we chose a uniform distribution considering only the 65% of slices with higher numbers of nodes, as they are responsible for most of the resource usage.



(a) Hourly distribution qq-plots

Attributes	**Distribution**	**K-S test**
CPU	$\ln \mathcal{N} \sim(\mu = 1.812, \sigma = 1.504)$	0.636
BW	$\ln \mathcal{N} \sim(\mu = 2.636, \sigma = 1.931)$	0.337
Mem	$\ln \mathcal{N} \sim(\mu = 2.476, \sigma = 1.046)$	0.976
# Slivers	$\mathcal{U} \sim(\min = 20, \max = 516)$	0.135

(b) Hourly distribution parameters

	Bandwidth	**CPU**	**Memory**	**#Slivers**
Bandwidth	*	0.082	0.040	0.060
CPU	-0.082	*	0.052	0.089
Memory	-0.040	-0.052	*	0.059
#Slivers	-0.060	-0.089	-0.059	*

(c) Pair-wise Pearson correlation coefficients

Figure 5.6: Hourly distribution QQ plots and model parameters. We used one day of the dataset (April 1st, 2009).

Table 5.6(b) shows the parameters of our model that best describe our empirical results. Additionally, it shows the probability value of the non-parametric one-sample Kolmogorov-Smirnov test, which quantifies the distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution and is sensitive to differences in both location and shape of the distributions. All tests report probability values higher than the 0.1 confidence interval, which means that we cannot reject the null hypothesis of distributions coming from the same distribution. These tests are confirmed by the QQ (quantile-quantile) plots in Figure 5.6(a), which demonstrate good matches between our models and real traces collected.

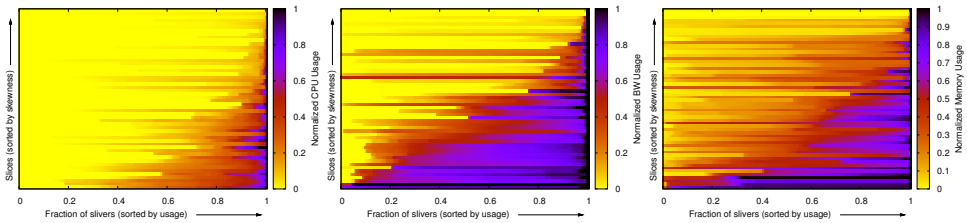
To capture any existing inter-dependency between slice's number of slivers, CPU utilization and bandwidth consumption, we first perform a pair-wise correlation analysis between these variables. Thus, we compute the Pearson correlation coefficient for each pair of variables. The correlation results presented in Table 5.6(c) show that each pair of variables exhibit a very low correlation coefficient. Thus, each pair of distributions is not correlated and we consider this a fair signal of their independence.

Intra-slice distribution: Also interesting is to look at the degree of usage within each slice –i.e. the workload introduced by slices to each one of the instantiated slivers.

To measure this, we show in Figure 5.7 a heat map where each line represents a distribution of usage within a single slice where the x-axis is the fraction of nodes sorted by usage and the color gradient represents the degree of usage.

Our first intuition was to see that nodes might be uniformly used as each slice is considered to run a large-scale experiment with homogeneous software. However, we can appreciate that the distribution of usage is highly heterogeneous among slices. Notice that most of the slices have predominantly low values (light color) for a large fraction of their slivers. This is especially true in the case of CPU usage. For bandwidth and memory usage we can appreciate a large portion of nodes with low-medium values and only slices with a low positive skew show a uniform increase of usage across nodes.

In this work, we do not provide a specific model because of the highly heterogeneous behavior across slices, and it remains for future work to establish how to represent such behavior. However, this result contradicts the assumption



(a) CPU (b) BW (c) Mem

Figure 5.7: Hourly distribution intra-slice. We used one day of the dataset (April 1st, 2009).

of uniform usage in a distributed application made in different theoretical analyses of resource allocation [36][35].

It remains unclear for us which distribution model would better fit the observed behavior. However, based on our intuition, we performed some simulations with the family of lognormal distributions with variable mean and standard deviations –i.e. a simulated distribution per slice– and we observed a reasonable match between the empirical heat maps and the simulated ones. Despite the promising initial results, we leave this part of the model as an interesting part of our future work.

5.4.2 Temporal dynamics of resource usage

The short-term distribution of resource usage among slices was described in detail, and we now turn our attention to the time dynamics for individual slices. We undertook a time series analysis by looking at slices' stationarity and autocorrelation properties and their frequency domain structure.

We only consider the first 12 days of April 2009, one of the most loaded periods analyzed, and limit our study to CPU and bandwidth usage. Our aim with this analysis is to extract information on short-term patterns –e.g. daily or weekly patterns– and we found that such a short period was a good trade-off

between efficient computation of parameters[7] and clarity of the presented results.

Besides, given the experimental nature of applications shown in Section 5.4.3 –different alternate periods of activity and inactivity– we only consider those time frames in which applications were active –i.e. an application using a non-trivial amount of CPU or bandwidth. We decided to leave out of this study the dynamic behavior of memory, because memory reallocation made by the operating system is not instantaneous and loosely related to the application dynamics but rather based on the current memory usage and load on a single machine. In other words, the memory allocation is more static than CPU and bandwidth, and does not provide a good picture of the current memory size used by an application.

Stationary and autocorrelation properties: To test the stationarity of resource usage time series, we apply the parametric *KPSS (Kwiatkowski-Phillips-Schmidt-Shin) test* [51] which tests for the null hypothesis that an observable time series is stationary around a deterministic trend. The core of the test consists of expressing the time series as the sum of a deterministic trend, a random walk and a stationary error. Thus, the null hypothesis corresponds to the hypothesis that the variance of the random walk equals zero meaning that the series follows a stationary distribution around a deterministic trend.

At the 0.05 confidence interval, we find that 93% of CPU time series and 84% of network usage time series pass the stationary test. The remaining 7% of CPU time series show aberrant behavior mostly due to bursty experimentation with the slice and are discarded for the rest of the temporal dynamic analysis to simplify our model. In the case of time series for bandwidth consumption, the remaining 16% show either aberrant behavior similar to the CPU case or strong diurnal patterns, which makes the series non-stationary. However, we decided not to discard some of these time-series as they belong to heavily used public services and discarding them would produce misleading simulations. The presence of diurnal patterns was surprising at first because PlanetLab is used worldwide by a wide range of users. However, we assume that such

[7]The analysis of 300 slices with traces of 8 months and 4 variables –i.e. hundreds of Gigabytes of raw data– was too computationally expensive to extract results iteratively in the development phase of the analysis framework with R.

patterns are present because most of the users of such public services are located in the United States time zones.

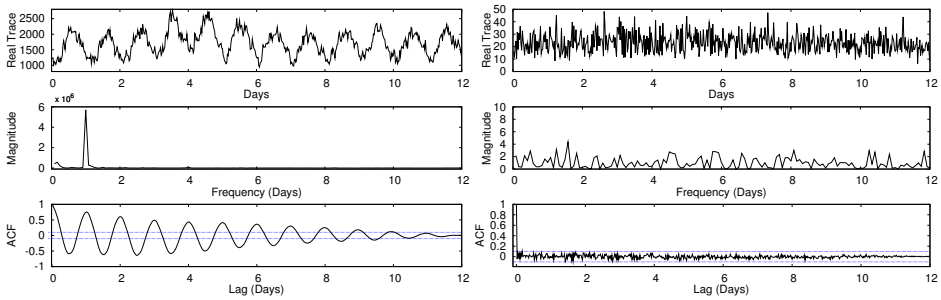
To detect diurnal patterns, we look the frequency domain of the time series (see Figure 5.8(a)). We find that 5% of slices exhibit diurnal patterns (high frequency components at 1 day) corresponding to public services available using the PlanetLab infrastructure.

If we look at the autocorrelation function (ACF) of the example stationary time series (see Figure 5.8(b)), all correlations are basically zero independently of the lag difference, and a flat power spectrum, a typical behavior observed in purely random processes. To assess such evidence, we apply the non-parametric *runs test* for randomness [52]. Given a time series $X(t)$, the runs test computes the median value of X_i over all the time series and marks the ones below the median as "−" and the rest as "+". Then, a consecutive sequence of equal elements (either "−" or "+") is considered a *run* and it compares the distribution of consecutive runs to known run-count distributions of random data –e.g. a normal distribution. At the 0.01 confidence interval, all the application traces, which previously showed a stationary behavior, pass the runs-test confirming the random nature of such processes.

By considering only the processes that do comply with the stationary hypothesis or the presence of strong diurnal patterns, we account for 98% of the resource usage both in CPU and bandwidth usage, which is a good trade-off between simplicity of model and representativeness. Including the whole population of applications might over-complicate our model without offering extra information from the resource utilization point of view.

Stationary and non-stationary models: Given the evidence presented so far –i.e. temporal correlations in ACFs, spectral density in the frequency domain and the stationary behavior– we choose two different approaches for modeling the observed time series depending whether they are pure random processes or non-stationary with diurnal patterns.

(a) Random processes. The independent distribution of runs in a time series, its stationary property, the absence of autocorrelation in the time domain and the flat power spectrum, are clear evidence of the well-known *white Gaussian noise* (WGN). It is characterized by Equation 5.4 where μ is the long-term



(a) Real network usage trace (seasonal) (b) Real CPU usage trace (stationary)

Figure 5.8: Real traces. We used the first 12 days of April 2009 for this analysis.

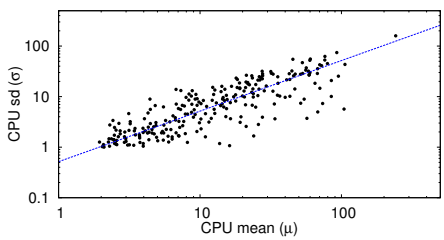
mean of the series and σ its standard deviation.

$$X_t = \mu + \mathcal{N} \sim (0, \sigma) \tag{5.4}$$

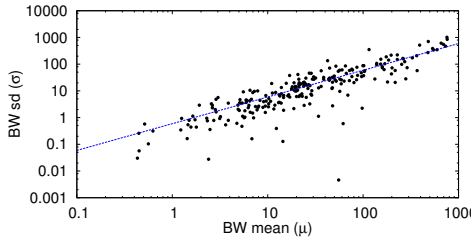
To simplify the number of parameters needed to simulate such processes, we look at the correlation between the long-term mean μ and its standard deviation σ. Figures 5.9(a) and 5.9(b) show the scatter plot of the explanatory variable μ in the x-axis against the response variable σ in the y-axis –notice the log-log plot. We find that points follow a linear relationship confirmed by the Pearson's correlation coefficient in Table 5.9(e). Thus, we approximate this correlation using the ordinary least squares (OLS) method to the straight lines superimposed, with the resulting slopes presented also in Table 5.9(e). Therefore, we estimate the standard deviation σ of the stochastic process following Equation 5.5, where α are the slopes previously obtained depending on if we are estimating the variance of a CPU or bandwidth process.

$$\sigma = \mu^{\alpha} \tag{5.5}$$

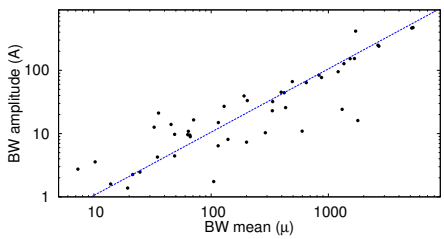
(b) Non-stationary processes with diurnal patterns. To model an approximation of those processes with strong diurnal patterns –high frequencies at one day– we decompose the time series in their structural components –i.e. seasonal, trend and remainder– following the *locally weighted scatter plot smoothing* (LOESS) method, which combines much of the simplicity of OLS with the flexibility



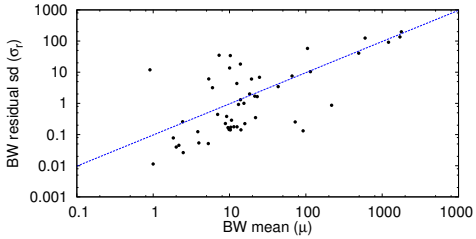
(a) Correlation CPU μ - CPU σ



(b) Correlation BW μ - BW σ



(c) Correlation Bw μ - BW A



(d) Correlation Bw μ - BW σ_r

	Attributes	**Pearson's corr.**	**Linear slope**
Stationary process	CPU (μ vs σ)	0.8249	0.5165
	BW (μ vs σ)	0.8156	0.5913
Seasonal process	BW (μ vs A)	0.6061	0.1084
	BW (μ vs σ_r)	0.9189	0.1284

(e) Correlation parameters

Figure 5.9: Correlation between mean μ as explanatory variable and standard deviation σ, amplitude of the diurnal pattern A and standard deviation of the residual component σ_r as response variables. We used the first 12 days of April 2009 for this analysis.

of nonlinear regressions. It does this by fitting simple models to localized subsets of the data to build a function that describes the deterministic part of the variation in the data, point by point. The resulting seasonal component is then fitted through OLS by a cosine function with a fixed frequency of one day, representing the diurnal pattern. This way, we find the amplitude of the diurnal pattern as the coefficient of the fitted cosine function. Our model is then constructed by the sum of two independent process as shown in Equation 5.6: i) a cosine function representing the deterministic diurnal pattern and ii) a white Gaussian noise representing the stochastic component of the process where σ_r is the standard deviation of the remainder component.

$$X_t = A\cos(2\pi t) + \mathcal{N} \sim (0, \sigma_r) \tag{5.6}$$

Again, to simplify the number of parameters needed to simulate such processes, we look at the correlation between the long-term mean μ and the amplitude of the fitted cosine function A. We also look at the correlation between the long-term mean μ and the standard deviation σ_r of the residual component (modeled as WGN). Figures 5.9(c) and 5.9(d) show a slight linear relation confirmed also by the Pearson's correlation coefficient. Thus, we estimate the response variables A and σ_r in a similar fashion as the stationary σ parameter (see Table 5.9(e) for the slopes of the linear relations). Although the number of processes with diurnal patterns is too low to make a strong statistical conclusion on such correlation, it will serve our purposes for our simplified model.

Model validation. We validate our model against measurement data by simulating every trace with the estimated parameters and quantifying the similarity (or dissimilarity) of the resulting processes using the *cosine similarity* metric, commonly used in the data-mining field [53]. It measures the similarity between two vectors of n dimensions (where n is the length of the measurement period) by finding the cosine value of the angle between them. Values closer to 1 indicate higher similarity.

$$similarity(A, B) = cos(\theta) = \frac{A \cdot B}{||A||\ ||B||} \tag{5.7}$$

Figure 5.10(c) shows that 10% of slices have a value less than 0.9. Those simulated traces that showed a low cosine similarity measure are those by which our parameter estimation is not effective and thus produce quite different patterns. However, most of the simulated traces produced quite similar patterns as shown by the remaining 90% of slices with a similarity metric very close to 1. Figures 5.10(a) and 5.10(b) show the simulated traces with their autocorrelation function and spectral analysis of the frequency domain of the same traces presented in Figures 5.8(a) and 5.8(b) respectively. Both sets of figures contain a similar structural behavior, a diurnal pattern with a high frequency at one day for the seasonal traces and a flat frequency spectrum with autocorrelation almost zero independently of the lag for the random processes.

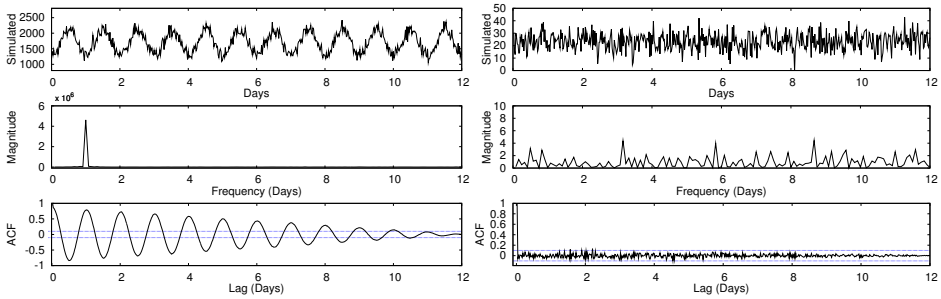
This validation states that our model is able to reproduce existing resource usage performed by applications. We consider this validation sufficient as our objective is not to forecast future workloads but to reproduce current conditions in a synthetic manner through simulations. This way, we can perform what-if simulations following a model based on real measurements.

5.4.3 Duration of Experiments

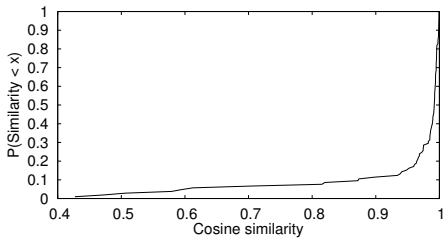
Finally, we look at the intermittent behavior in the activity pattern of slices. As experimental applications, the duration of such active periods may be highly variable because of potential rounds of testing, deployment and evaluation.

Thus, we consider the whole period of our traces and measure the time when a slice shows some kind of activity. A slice is considered active if it shows non-trivial CPU usage or has used at least 1 Kbps of network bandwidth in a given sample. We then compute the mean time of activity (MToA) and the mean time of inactivity (MToI), an analogous measure to the mean time to failure (MTTF) and mean time to recover (MTTR) in resource reliability analysis [54].

Figure 5.11(a) shows the relation between MToA and MToI. There is a high heterogeneity and no correlation between them at first sight. However, we can distinguish between three types of experiments: i) long-running services with high availability –i.e. short periods of inactivity of less than 1 hour– corresponding to public services currently deployed in PlanetLab –e.g. the



(a) Simulated network usage trace (seasonal)(b) Simulated CPU usage trace (stationary)

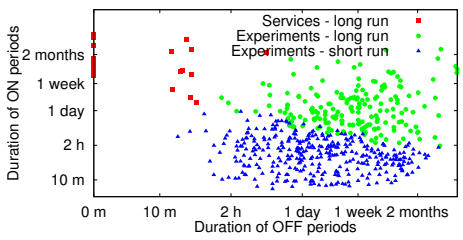


(c) Cosine similarity CDF

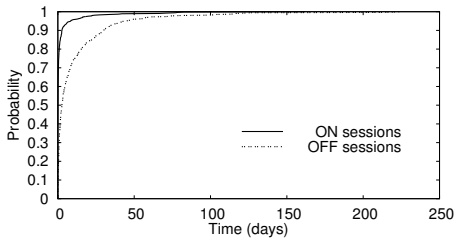
Figure 5.10: Simulated traces and cosine similarity CDF

content distribution networks (CDN) of the *CoDeeN* family [55] or Oasis [56], an overlay network stack; ii) long-run experiments with MToA between 1 day and several weeks with high MToI variability and iii) short-run experiments with a very low MToA (ranging from several minutes to a few hours) and high variability of their inactivity periods.

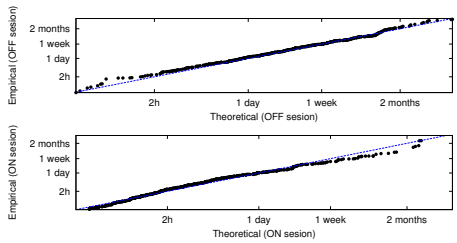
It is important to notice that the MToA and MToI distributions are highly skewed (see Figure 5.11(b)), meaning that almost all applications belong to the third type while the mass of the resource usage is caused by the long-running services and experimental applications. We fit the empirical data to a lognormal distribution using MLE (see parameters in Table 5.11(d)). Looking at the QQ plots in Figure 5.11(c), it shows a good fit between the empirical distribution and the estimated distribution. Besides, both distributions pass the Kolmogorov-Smirnov test at the 0.05 confidence interval.



(a) Clustering of slices



(b) CDF of ON-OFF sessions



(c) QQ plot of ON-OFF goodness of fit

Attributes	**Distribution**	**K-S test**
ON session	$\ln \mathcal{N} \sim(\mu = 5.364,\ \sigma = 2.144)$	0.209
OFF session	$\ln \mathcal{N} \sim(\mu = 8.173,\ \sigma = 1.969)$	0.791

(d) ON-OFF distribution parameters

Figure 5.11: Results of ON-OFF session analyses. We used the whole data set for these analysis (8 months of available data).

To model such an active-inactive pattern, we decide to model each application's activity as an on-off process, setting the rate of transition according to Equation 5.8 using the MToA and MToI extracted from the distribution parameters in Table 5.11(d).

$$\lambda = \frac{1}{MToA} \qquad \mu = \frac{1}{MToI} \tag{5.8}$$

5.5 Related Work

Traditionally, workload characterization of computational infrastructures has been widely studied in the literature. However, most of them focused on the workload supported by resources –e.g. networked nodes– instead on focusing on the structural behavior of applications that leads to such workload.

Chun et al. [40] made an extensive study of the workload in PlanetLab by analyzing node load and node reliability as well as some characteristics of slice behavior. However, their work was carried out in the initial stages of the experimental infrastructure when a small fraction compared to today's resources were available. Moreover, they did not present a specific model to generate the workload they described. Finally, the objectives, metrics and scope of our work are different than Chun et al. work and, thus, results are not directly comparable. However, we consider an interesting future work to be the comparison of similar metrics of previous work with the current usage present in PlanetLab to see the evolution of usage characteristics in this infrastructure.

Oppenheimer et al. [41] also performed an analysis of PlanetLab traces from the service placement point of view. Their work focused on assessing whether a discovery and selection service might be useful for the infrastructure considering the current usage of resources. Their analysis led to the development of SWORD[29], a resource discovery tool deployed in PlanetLab. However, its lack of usability and integration hinders their current applicability, confirmed by our findings in Section 5.3. Spring et al. [42] used measurements of PlanetLab's nodes to dispel various *myths* about the infrastructure to do experimental research focusing on node performance and *tips and tricks* to deploy and successfully execute experimental applications.

In the wider context, not specifically related to shared experimental infrastructures but to computational grids and data centers, Verma et al. [57] studied the temporal dynamics of a workload in a private company's data center to design new allocation policies to minimize power consumption by consolidating applications. Although they also focus on the structural behavior of applications, their work is based on a small-scale sample (4 applications) of the workload giving a biased estimation of their workload. Finally, Foster et al. [58] describes resource utilization on Grid3 from the utilization perspective to describe which aspects of their grid would be important to improve without giving a detailed model of such usage .

5.6 Conclusions

In this work we analyze usage characteristics of PlanetLab as a case study. Our results may be extrapolated to other types of experimental infrastructures that have similar goals –experimentally driven research– and are driven by the same architectural principles –resources spread around the globe, slice-based design to share resources, collaborative contribution of resources to the infrastructure, hard limits on single resource consumption but no incentives to cooperate, etc.

We present an analysis and a model of resource usage of applications to understand the structural behavior of its participants (slices), in contrast to previous work which focused on modeling the workload itself (nodes).

Considering the current state of the infrastructure, we find that the distribution of usage among slices is highly skewed with few slices (long-running services) monopolizing almost all resources. This phenomenon jointly with the highly skewed distribution of resource contribution presented in previous works [45], where almost 50% of nodes were persistently overloaded while the others are underloaded, leads to a highly unbalanced ratio of contribution and consumption, clear evidence of free-riding –i.e. there are only minimum resource requirements but no incentives to contribute more resources in proportion to the usage of the infrastructure.

Although this study is focused on experimental infrastructures and our results may not be directly extrapolated to real applications in the current and future Internet, our findings may be useful to understanding the potential impact of unlimited and unrestricted usage made by applications in real infrastructures

–e.g. resource congestion, lack of incentives to behave, poor resource selection, etc.

Given the short-term distribution of resource usage and temporal dynamic model presented in this work, our plan for future work is to refine our model to capture the global phenomena presented in Section 5.3.

With the expected growth in the near future of these kind of infrastructures –and generalizing towards the Future Internet– proper mechanisms for resource discovery, selection and allocation, and global mechanisms for the overall regulation and governance of the infrastructure seem necessary for its sustainability. Besides, the increased scale of the system may aggravate the effect of the detected imbalances.

Chapter 6

Using economic regulation to prevent resource congestion in large-scale shared infrastructures

Abstract

In this chapter[a] we study the problem of large-scale resource congestion from the control and regulation point of view. Applications and services running in large-scale shared infrastructures like Grids or PlanetLab have different resource usage profiles and different resource consumption strategies according to their specific requirements. However, users of these types of infrastructure tend to prefer a subset of available nodes to execute their tasks. As a result, this pattern of user behaviour usually leads to an unfair distribution of work between nodes. We find that most current research focuses on short-term and per-resource scheduling, and the issue of efficient resource allocation in the long-term and system-wide is not yet appropriately studied. Our main contribution is the introduction of a novel macro-scheduling (long-term and system-wide) mechanism for resource capacity self-regulation in which virtual currency is used as a tool to govern resource usage in massively distributed settings, which are otherwise hard to control. We show by simulation that our approach successfully redistributes the load in a fair and economically efficient manner.

[a]This chapter is based on the paper [P2]

6.1 Introduction

The current and future Internet, as an open shared network and service infrastructure, is used by more people and more diverse applications every day. However, the growth in usage, capacity and diversity makes it increasingly difficult to provide users with sustainable, high-quality services.

In addition, shared computing infrastructures rely on the individual contributions of participants to create an infrastructure with enough power to run large-scale applications and services. A key characteristic of this type of shared infrastructures is its peer-to-peer nature, in which participants are both consumers and resource providers acting in their own interests. Examples include scientific collaboration grid networks [59][60] or network testbeds such as PlanetLab [61] or EmuLab [62].

To solve the resource allocation problem, traditional schedulers optimise usage, throughput or response time at a cost of centralising components and compromising scalability. An opposite approach is to implement an economic solution in which computational markets give users control over the service levels they require in large-scale resource sharing. These economics-inspired systems have been shown to be a decentralised, scalable and efficient way of allocating resources according to user preferences.

However, the combination of resource-intensive applications and user preferences can lead to levels of demand that saturate the infrastructure, affect negatively other users or compromise the overall stability of the system. In addition, saturation usually affects only a subset of the infrastructure resources, known as hot spots, whereas the load remains low over the rest of the system.

In this chapter, we present a set of mechanisms for self-regulation of resource and service exchange to ensure that work is distributed in a fair and stable way. Besides, current economics-inspired models focus only on short-term (i.e. micro-economic[1]) interactions between participants, and the effects of long-term, system-wide (i.e macro-economic[2]) interactions have not yet been

[1] A branch of economics that focuses on the ways in which individuals, households and firms determine how to allocate limited resources, typically in markets where goods or services are being bought and sold.

[2] A branch of economics that focuses on the behaviour of an economy at the aggregate level and the effects of government actions (such as laws or taxation levels).

analysed in depth. Although economic scheduling is efficient regarding the social welfare of a system, it can also lead to system-wide performance penalties.

We stress the importance of introducing regulatory mechanisms for controlling and limiting user demand for shared resources. Consequently, our main **contribution** is the *introduction of a novel mechanism for self-regulation of resource capacity based on virtual currency management*. Besides, we show that our macro-economic mechanism is a powerful tool that (i) provides users with incentives to distribute their tasks to prevent the emergence of hot-spots in large-scale infrastructures and (ii) enables the redistribution of wealth to improve the social fairness of the system. Additionally, the money-based infrastructure introduces an economic incentive for enforcing regulatory standards to improve the overall governability and sustainability of the network.

The rest of this chapter is organised as follows: in Section 6.2 we present the motivation and problem statement; in Section 6.3 we present related work on resource allocation mechanisms; in Sections 6.4 and 6.5 we describe the system model and present our regulatory mechanism; in Section 6.6 we analyse the simulation results; in Sections 6.7 and 6.8 we conclude the study by discussing the applicability of our solution and proposing areas for future work.

6.2 Motivation and problem statement

Free access and unrestricted demand for finite resources ultimately leads to over-exploitation and degrades quality of service (QoS). This problem arises because the benefits of exploitation are received by individuals –each of whom is determined to maximise their use of the resource– whereas the costs of the exploitation are distributed between all of those who have access to the resource. This, in turn, increases demand for the resource to such an extend that the resource is exhausted and becomes useless. As explained in chapter 2, this is a clear example of the well-known problem *tragedy of the commons* [63].

As a motivation, in this section we analyse a reference system like PlanetLab using data from CoMon, a monitoring infrastructure for PlanetLab [44]. We

measured PlanetLab usage over a one-month period[3], restricting our study to available nodes (i.e nodes to which users actually had access) and discarding those nodes that were inoperative due to maintenance work, network connectivity problems, or for other reasons.

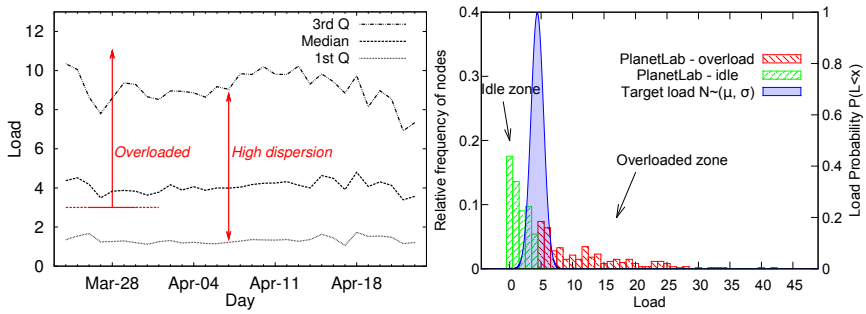
Figure 6.1(a) shows the load quartiles (1st, 2nd and 3rd quartiles) during the measurement period for all working nodes in PlanetLab. Despite daily variations, the nodes appear to be very highly loaded: the mean for each day is over 3, which is considered to be overloaded [64]. Although this is not necessarily a problem, since it is a sign of usefulness of the infrastructure for their users. However, Figure 6.1(b) shows that almost 50% of nodes are persistently overloaded whereas the others are underloaded. The overloaded nodes provide lower QoS to applications, whereas the other nodes are mainly idle. In Table 6.1(c), we can see that the distribution has a positive skew, which indicates that the mass of the distribution is unbalanced.

If we consider a reasonable scenario in which tasks are distributed uniformly among resources, the load distribution (i.e. the number of running tasks on a given resource) should follow a normal distribution, with low variance and a skew value close to zero, which indicates that most of the nodes support the same workload. We assume that our target load distribution should be similar to a normal distribution because if tasks are distributed randomly among resources following an unbiased and uniform distribution, all resources are treated equal. Therefore, according to the law of large numbers, if there are a large number of resources, the sum of independent identical variables (the sum of tasks on a resource equals to the load) constitutes a normal distribution.

The presence of overloaded nodes is a result of correlated user preferences (i.e. users tend to prefer similar nodes in terms of reputation or technical characteristics) and the lack of incentives to behave considerately in these types of collaborative environments.

Even if micro-economic schedulers are used, the correlated preferences could lead to overloading certain nodes with higher preference weights. However, the higher revenue generated by these overloaded nodes leads to an unfair distribution of wealth among participants.

[3]Observed from 24 March to 23 April 2009, with samples of each node taken every 5 minutes. Traces are publicly available following the instructions at `http://comon.cs.princeton.edu`



(a) Load evolution

(b) Load distribution

	Real Load	**Target Load**
Mean	4.393	$\simeq$ 4.393
StDev	6.565	min
Median	5.935	$\simeq$ 4.393
Skewness	2.118	$\simeq$ 0.000

(c) Load (uptime)

Figure 6.1: Measures of system overload for all working nodes in PlanetLab from 24 March to 23 April 2009. Figure 6.1(a) represents the evolution of load quartiles. Figure 6.1(b) is the frequency histogram of the load average for each node and the load CDF for the same period. The green zone represents idle resources and the red zone represents overloaded resources. Table 6.1(c) contains load distribution statistics.

This scenario is very similar in real-world economies, where wealth is distributed following a Pareto distribution (i.e. "long tail" distribution). Although a Pareto wealth distribution is not inherently unhealthy, a fair distribution of wealth (in open testbed infrastructures like PlanetLab) would be one that gives researchers the same chance to test their proposals in similar conditions, regardless their actual incomes.

To address the unbalanced load between nodes and the unfair wealth distribution between users, we propose a macro-economic approach (i.e. long-term, system-wide strategy) based on regulation through virtual currency management, which can be viewed as capacity management mechanism, that gives resource providers and consumers incentives to redistribute the load in large-scale shared infrastructures to prevent congestion on hot-spots nodes and to distribute wealth equitably –i.e. accommodate user demand in a more reasonable scenario (see Figure 6.1(b)). Our solution is based on automatically detecting and forcing the redistribution of tasks by taxing resource prices until we reach a reasonable distribution of work.

6.3 Related work

In this section we present related resource allocation studies in which both economic and non-economic mechanisms are used.

Studies which do not consider economic concepts analyse the resource allocation as a scheduling problem (see the survey by Pinedo [65]). These proposals range from simple, centralised scheduling algorithms like First Come First Serve (FCFS), to Shortest Job First (SJB) which provides efficient allocations but does not take into account the different values of the users' tasks.

On the other hand, extensive research has also been carried out into the application of economic models to resource allocation in large-scale shared infrastructures. Computational markets have been shown to allocate resources efficiently in a decentralised way in the presence of selfish utility-optimising resource consumers and selfish profit-optimising resource providers. Shirako [24] is a toolkit for building utility services for dynamic on-demand sharing of networked resources through programmatic interfaces. Shirako is based on a common, extensible resource leasing abstraction [26] similar to those used in the allocation of airline seats. It combines elements of lifetime management

and mutual exclusion. Although Shirako mainly uses flexible mechanisms for trading resources between clients through a series of leases, there is no regulation mechanism between resource brokers, which can lead to over-provisioning of resources. Consequently, negotiated SLAs might be broken and the infrastructure, or a subset of it, may suffer congestion.

Bellagio [27] is a market-based resource allocation system for federated distributed computing infrastructures like PlanetLab. The Bellagio architecture is based on a centralised auctioneer which allocates resources periodically and determines the corresponding user payments. Users specify resources of interest by bidding in a combinatorial auction [28]. The amount of virtual currency owned by a site is determined directly by the central authority, which establishes the share of virtual currency assigned to each site. Although the final allocation is proportional to the bids advertised by users, Bellagio does not provide software agents that can act on behalf of their users to maximise their utility; consequently, the system can be sub-optimal, because human users may not behave in an economically rational way under certain circumstances.

Buyya et al. developed Nimrod-G [19], a resource broker that supports deadline and budget constrained scheduling algorithms [66] for task-intensive applications in clusters. However, their algorithms are only designed to make efficient local resource allocations between competing users, and do not establish a coordination between schedulers or fully analyse the long-term effect of their algorithms on the overall infrastructure. Also, studies of double auctions (see the survey by Friedman [13]) or combinatorial auctions (see the survey by De Vries [14]) allocate users to resources on the basis of short-term economic efficiency and do not take into account infrastructure-wide metrics such as the distribution of work among resources.

Finally, Tycoon [32] is a distributed market-based resource allocation system in which every node in the system runs an independent auction for its local resources. Auctioneers conduct a proportional share-based auction in which users receive a proportional amount of a single resource (virtualised CPU and memory) determined by the size of the bids made by all users for the same resource. Users are assigned a fixed amount of currency to spend over time to allocate their tasks. Although this model is similar to the one presented in Section 6.4, it does not consider system-wide metrics associated with correlated resource preferences –e.g. the uneven distribution of work among nodes. In

addition, the Tycoon model considers symmetric systems in which all users have the same budget, whereas our model takes into account the behaviour of the system in the presence of variable budget constraints between participants.

We believe that most current research focuses on the short-term allocation and maximisation of user utility and does not address the behaviour and health of the system as a whole or system-wide metrics like the distribution of work load, the proportionality between consumption and contribution, or the impact of different budget constraints on user utility.

6.4 System Model

Our solution is designed for a system consisting of an arbitrary large set of nodes (physical or virtual machines) at diverse locations which communicate via message passing over a network such as the Internet. The system is dynamic in the sense that nodes and networks can be added or removed and can degrade (overload) or fail at any time. The nodes are resources owned by different organisations and, although there are common protocols and rules, there is no need for a central executive authority to carry out the day-to-day management of the system. Each organisation can freely determine the number of resources it contributes to (or shares with) the system beyond a specified minimum. Participants in the system are human users (or software agents participating on their behalf) who usually belong to a single organisation and execute their tasks across a subset of available nodes (see Figure 6.2(b)).

In view of the above scenario and the benefits of market-based resource allocation in decentralising resource scheduling in a end-to-end way, the system requires a short-term micro-economic resource allocation foundation to enable users to express their preferences as prices. Extensive research has been carried out in this field. The most popular approach is to use some form of auction to extract the market price directly from the users' bids, for example an English auction, a Dutch auction, a double auction or combinatorial auction [12][13][14][15] . The main drawbacks of auction-based systems are that the response time is slow (bidders have to wait for auction clearing) and they are unsuitable for divisible resources because of the complexity involved in determining the most efficient way to divide a resource.

6.4.1 Proportional share allocation

The simplest and most appealing mechanism for shared divisible resources is to use proportional share auctions, which have already been proposed for OS process scheduling [67], I/O disk scheduling [68] or task scheduling in grid environments [69]. In this case, allocations are proportional to the consumer's weight (or preference for a resource) and inversely proportional to the sum of all other users' weights for the same resource. Therefore, we base the mode for our system on the price-anticipating mechanism proposed in [35], in which each user submits a bid for resources and the price of the machine is determined by the total bids submitted. More formally, the price of resource j is set to $Y_j = \sum_{i=1}^{k} x_{ij}$, where k is the number of bids on resource j and x_{ij} is a non-negative user's i bid for resource j. Following the proportional share allocation mechanism, user i receives a fraction $r_{ij} = \frac{x_{ij}}{Y_j}$ of resource j.

A game theory analysis of the aforementioned price-anticipating mechanism can also be found in [35]. Feldman et al. propose an algorithm for finding the *best response*[4] of an agent to the system. Given a fixed budget X and a pool R of divisible resources, the algorithm finds the distribution of bids across resources that yields the highest utility for an individual player i by solving the following optimisation problem (6.1):

$$\begin{aligned} &\text{maximize } U_i(\forall_{j \in R} \frac{x_{ij}}{Y_j}) \text{ subject to} \\ &\sum_{j=1}^{m} x_{ij} = X_i \text{ and } x_{ij} \geq 0 \end{aligned} \tag{6.1}$$

The computational cost of the optimisation algorithm is $\theta(n \log n)$, which is acceptable considering the computational power of current hardware and the input size of the problem. Finally, Feldman et al. show that there is always a Nash equilibrium when the players' utility functions are strongly competitive, i.e. when there are at least two users competing for each resource, which is a reasonable assumption in systems like PlanetLab. They also show that the Nash equilibrium resulting from the best response dynamics is efficient and fair.

[4] In game theory, the best response is the strategy (or strategies) that produces the most favourable outcome for a player, taking other players' strategies as given.

6.4.2 Metrics

The *utility* of each user i is represented by a function U_i of the shares obtained by the user from each machine. An important issue in representing utility is the notion of preference for resources, since each user could have a different preference for the same machine. Consequently, we consider a linear utility function $U_i(r_{i1}, \ldots, r_{in}) = w_{i1}r_{i1} + \ldots + w_{in}r_{in}$, where w_{ij} is the private preference of user i for resource j. This utility function is suitable for heterogeneous environments in which resources are valued differently by each participant.

To study the behaviour of our proposed mechanism, we consider the following metrics:

- **Load uniformity and dispersion**: The load distribution is an interesting metric for measuring the overall health of the system in terms of congestion. In this model we assume that if users are willing to spend virtual money on a resource, they will eventually execute a process. Therefore, we consider the load L on a resource to be the number of positive bids on that resource. To measure the distribution of load among nodes, we define two different metrics. Firstly, the *load uniformity* is represented by $\frac{\min L_i}{\max L_i}$, which is the ratio between the minimum and maximum loads. The higher the ratio, the greater the distribution of the load among resources, since each resource supports a similar workload. Secondly, we measure the dispersion of the load as the standard deviation $\sigma(L)$ of the node loads.

- **Efficiency (price of anarchy)**: For an allocation scheme ω at equilibrium, the efficiency is computed as $\pi(\omega) = \frac{U(\omega)}{U^*}$, where $U(\omega) = \sum_{i=0}^{n} U_i(\omega_i)$ and $U^* = \max(U(\omega))\forall\omega$. In our case, it is easy to compute the social optimum U^* because it is achieved when we allocate a whole node to the user with the highest preference weight on that node. Thus, it represents the loss in efficiency as a result of user's selfishness and decentralisation.

- **Fairness (uniformity, envy-freeness)**: To represent the fairness of our system, we consider two different metrics: utility uniformity and envy-freeness. *Utility uniformity* is represented by $\upsilon(\omega) = \frac{\min U_i(\omega_i)}{\max U_i(\omega_i)}$, which is the ratio between the minimum and maximum utilities. The

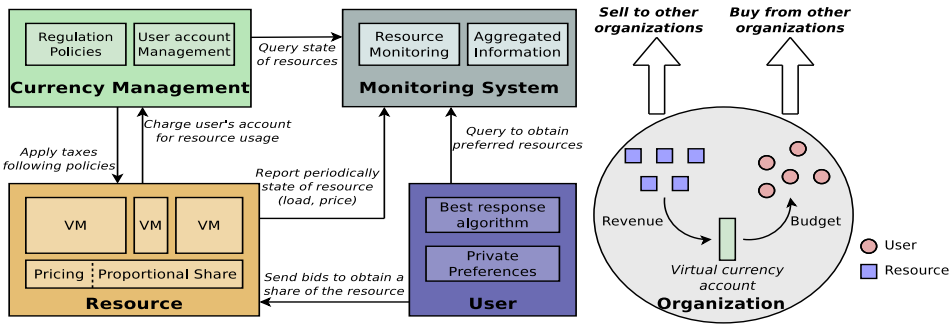
higher the ratio, the fairer the mechanisms, since users obtain similar utility from the system. Another way to measure the fairness of an allocation in Economics is to determine the *envy-freeness* [70], which is represented by $\rho(\omega) = \min(\min_{ij} \frac{U_i(\omega_i)}{U_i(\omega_j)}, 1)$, where $U_i(\omega_i)$ is the utility of user i and $U_i(\omega_j)$ is the utility that user i would have if it was allocated the resource shares of user j. In other words, envy is related to user i's perception of its own allocation with respect to those received by the other users.

An economically healthy resource allocation scheme should enforce a Nash equilibrium with high efficiency and high fairness. We also aim to guarantee high load uniformity and low dispersion, to distribute the load evenly while maintaining the high efficiency and fairness produced by the proportional share model.

6.5 Currency Management System: an economic-inspired self-regulation mechanism

As explained in Section 6.2, the main problem is the unfair distribution of tasks among nodes in heavily-loaded systems. As in the real economy, free-market mechanisms sometimes fail to address such problems and central governments impose restrictions (regulations) on the system that are designed to act as incentives to behave in a certain way. The existence of a central authority with a degree of global and aggregated knowledge of the system does not necessarily restrict its scalability, as discussed in Section 6.7.

We propose a self-managed regulatory body (i.e a virtual bank or *Currency Management System (CMS)*) which manages through simple policies the virtual currency used by participants to bid for resources. The CMS has a two-fold aim: i) to limit the amount of currency each user can spend on resources, thereby restricting their long-term purchasing power; and ii) to introduce long-term, system wide macro-economic policies that act as a self-regulation mechanism by taxing resource prices according specified policies and redistributing wealth (virtual currency) among participants to improve the overall fairness of the system.



(a) System architecture (b) Virtual organisation model

Figure 6.2: System architecture and system model overview. Figure 6.2(a) shows the interactions between components. Figure 6.2(b) is a schematic representation of a virtual organization based on contributed resources that earn virtual currency and users who consume resource from other organisations. Resources and users from the same organisation share the same virtual account.

Resource taxation (following specified policies) increases prices and encourages users to bid for alternative resources. For example, one policy would be to *improve availability* by imposing taxes on those resources with low availability; consequently, resources with low availability would generate less revenue because users would tend to change their bids to resources with lower taxes, which would, in turn, provide an incentive to improve the availability of resources. Similarly, another policy would be to tax overloaded nodes to attenuate hot-spots.

It is important to note the difference between: i) the resource **price**, which represents the cost required to gain possession of a resource considering user's preferences and competition; and (ii) the **tax** price, determined dynamically by our mechanism as a means for solving the uneven distribution of work, similar to other taxes on consumption such as *value added taxes* (VAT). The former is a micro-economic mechanism for the regulation of access to resources and the latter is a macro-economic mechanism which introduces a correcting factor based on an observed effect seen at macroscopic level (eneven distribution of work among congested resources).

The architecture of our proposal is shown in Figure 6.2(a). The CMS gathers system-wide and aggregated statistics from the *monitoring infrastructure*, such as average load and its dispersion, the effective contribution and consumption of users, etc. After a certain period of time, or *epoch*, the CMS uses predefined policies to determine the appropriate taxes on each resource (a factor to be applied on the price). During the epoch, users can freely evaluate their needs and spend their budgets according to their own strategies, without any other external restriction. Once the taxes have been determined, the price of resource j is computed according to Equation 6.2, where k is the number of bids on resource j, b_i is the bid of consumer i and $\text{tax}_j \in [-1, \infty) \subset \mathbb{R}$ is the tax applied by CMS to resource j.

$$Y_j = \left(\sum_{i=1}^{k} b_i \right) * (1 + \text{tax}_j) \tag{6.2}$$

Since the aim of our proposal is to redistribute the load as evenly as possible, we impose a higher tax on resources with higher loads, which increases the price and encourages participants to use spare resource instead (those with a lower tax and, consequently, a lower price).

The CMS does not know the users' preferences in advance, so it cannot anticipate user behaviour in response to a specific set of taxes. We therefore use a heuristic based on the ratio between the load and the target load to move towards the set of target taxes under which the load is distributed equally among all nodes (e.g. the load dispersion is minimal).

Algorithm 6.1 shows the procedure for determining the tax to apply during the next epoch. The tax does not jump straight to the target value but instead moves towards it at a rate determined by a learning rule, which prevents oscillations and produces a smooth approximation to the target value. The learning rule used is the Widrow-Hoff rule, which is a well-known learning mechanism used for back propagation in neural networks [71] and used to move towards the target price in different economic agents [72]. It contains a parameter β (learning rate) that represents the speed with which the adjustment takes place.

Therefore, once the CMS has gathered information about the load and the current tax for each resource, it computes the uniformity metric to determine the current load dispersion in the system and adjusts the learning rate of the algorithm: a lower learning rate is used at higher uniformity to produce a smooth approximation to the target load. The CMS then applies the heuristic described above to the current tax to determine the new tax to apply to each resource.

Algorithm 6.1 Pricing tax regulation algorithm

Require: $\tau \leftarrow$ target load
Require: $S \leftarrow \{\forall j \in R(load_j, tax_j)\}$ ▷ Set of resource info
  uniformity $\leftarrow \dfrac{\min load_j}{\max load_j}$
  $\beta \leftarrow 1.0 - \text{uniformity}$
  for all $(load_j, tax_j) \in S$ **do**
    $\Delta_{tax,j} \leftarrow \left(\dfrac{load_j}{\tau} - 1\right) * \beta$
    $tax_j^{t+1} \leftarrow tax_j^t + \Delta_{tax,j}$
  end for

6.6 Performance Analysis

We use simulations to evaluate the long-term impact of our system. To determine the effectiveness of our regulatory mechanism in improving the load distribution in comparison with the free-market (unregulated) scenario, we compare the best-response dynamics from the game theory analysis of the price-anticipating model (Section 6.4) with the best response dynamics under our regulatory mechanism (Section 6.5).

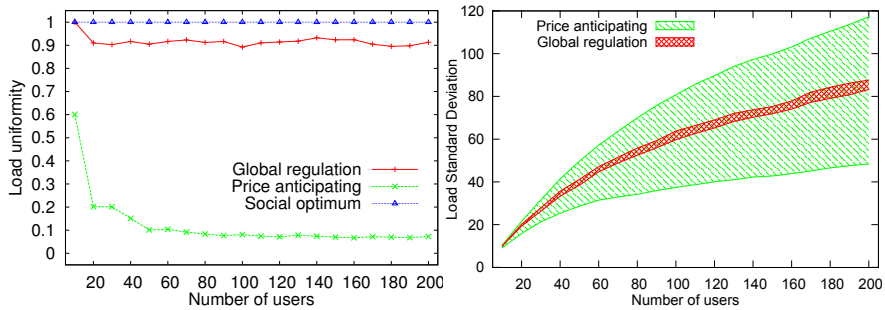
Method. The set-up of the simulations consists in fixing the number of resources m to 100 and varying the number of users n (from 10 to 200) to asses the scalability of the solution as more users (and, therefore, more load) are added to the system. We do not present the results for a variable number of resources because the simulations showed that different executions with the same ratio of resources to users produce similar results. The best-response algorithm is updated after each time step (1 simulated minute) and the epoch (at the end of which the tax regulation algorithm is executed) is defined as 60 time steps (1 simulated hour).

User preferences. Some nodes are persistently more loaded than others (see Figure 6.1(b)) due to correlations of user preferences. To capture these correlations, we experiment with the following user preference model. For each user, we create a list of weights that are independently and identically distributed according to a uniform distribution $U \sim (0, 1)$. Next, we arrange the list in descending order to create a user's preference weight on resources so that $p_i = (p_{i1}, ..., p_{in}), p_{ik}$ represents user i's weight on resource j and $p_{ik} > p_{i(k+1)}$. We then normalise the expression so that $\sum_{j=0}^{m} w_j = 1$. Consequently, we expect to have a high load on the first resources and a lower load as the weights decrease.

Note that we only consider positive weights on resources, so every user obtains a certain positive utility from each resource. If we had included resources with weights equal to zero, those users following the best-response algorithm would not have bid on these resources because the utility provided is also zero. In practical terms, this means that those resources with a weight equal to zero are not available. Therefore, and with no loss of generality, we only consider available nodes in our simulations.

Convergence criteria and results. The convergence time is a measure of the speed with which the system reaches an equilibrium. As in [35], the price-anticipating system converges to a Nash equilibrium when the difference in the best-response utility between two time-steps is less than ε (0.001 in our experiments). However, when we apply our regulatory mechanism, we change the environment (i.e. the prices of resources) at the end of each epoch, and each epoch evolves iteratively to a different Nash equilibrium. Therefore, we consider that our tax regulation mechanism has converged when the value of $\Delta_{tax,j}$ (see Algorithm 6.1) is less than δ (0.1 in our simulations). The results presented in these sections are taken when the system has converged.

The simulation results show that the best-response dynamics converge after 5 iterations (5 simulated minutes), as in [35], whereas our regulatory mechanism converges in a range of 3-5 epochs (3-5 simulated hours). This shows that, although our mechanism is designed to be executed in a long-term time window, it is also able to converge in few iterations. Therefore, this mechanism can be executed frequently (i.e. using short epochs) when the load conditions are dynamic but the epoch can be longer when the load conditions are in a steady state.



(a) (b)

Figure 6.3: (a) Load uniformity and (b) Load dispersion

Load uniformity and dispersion. The effectiveness of our proposal is based on distributing the load effectively among resources. We measure the load uniformity and its dispersion for the price-anticipating model and the regulatory mechanism, once they have converged. As shown in Figure 6.3(a), the price-anticipating mechanism produces low load uniformity because users tend to bid for their preferred resources, which creates a large difference between the maximum and minimum loads. However, under our tax regulation mechanism, when the CMS detects that there is a subset of heavily loaded nodes and it begins to increase the corresponding tax (and, therefore, the price), users following the best-response algorithm tend to distribute their bids to cheaper (less loaded) resources to maintain as high their utility as possible. Our results show that the system encourages users to redistribute the load more evenly (similar load on nodes) regardless of the number of users.

Similarly, Figure 6.3(b) shows the dispersion of the statistical variable load L ($\mu(L) \pm \sigma(L)$) in the system. As the number of users increases, the load on the system also increases. However, without tax regulation the dispersion is higher and increases with the number of users. Conversely, when taxes are applied the dispersion is maintained at similar values, which demonstrates that our proposal is scalable independently of the number of users .

Figure 6.4(a) shows the empirical CDF of the load distribution for a simulation with 100 resources and 100 users. We can see that load is highly dispersed without regulation; approximately 50% of nodes are highly loaded, at levels

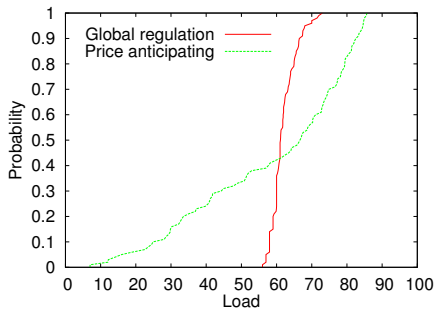
above the mean load (> 60 users bidding on them). However, when our regulatory mechanism is applied, the load distribution is centered at the target load (the mean load in our simulations) and the CDF shows that the variance is very low, because the load values of each resource fall within a small range (between 55 and 65).

Specifically, Table 6.4(b) shows that the standard deviation is very high when no regulations are enforced. However, under regulation, the average load remains similar but the standard deviation and skewness decrease. The Kolmogorov-Smirnov normality test for the global regulation case shows a significance value of 0.232 given the assumed significance level of 0.05, so normality cannot be ruled out. The normality assumption is also supported by the Normal Q-Q plot in Figure 6.4(c). The results show that our mechanism provides users with an incentive to redistribute their workloads evenly and ensures a reasonable distribution of work among resources.

Importantly, the behaviour of the price-anticipating algorithm (without regulation) is similar to the high load variance illustrated in Figure 6.1(b) and Table 6.1(c) in Section 6.2, which shows the relationship between our simulations and real observed results from PlanetLab. These results clearly demonstrate that our regulatory mechanism redistributes the load among nodes and achieves a similar distribution to our target distribution, where the load is centered at the mean and shows low dispersion.

Efficiency (price of anarchy). Figure 6.4(d) shows efficiency as a function of the number of users. The efficiency achieved by the price-anticipating algorithm is very high (approximately 0.95) and the tax regulation mechanism does not lower the efficiency, irrespective of the number of users in the system; in other words, the system provides users with the same level of efficiency but the load is effectively redistributed to prevent hot-spots.

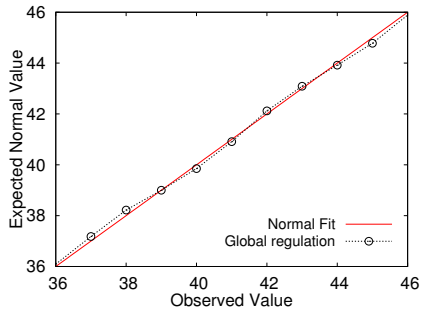
Fairness (uniformity, envy-freeness). Figure 6.5(a) shows the utility uniformity as a function of users for the correlated preferences presented above. Our regulatory mechanism achieves high utility uniformity (> 0.8, all users obtain similar utility from the system), although with a small amount of uniformity lost in comparison with the price-anticipating simulation (approximately 15%, taking the highest difference). This is because, once our regulatory mechanism has been applied, those users bidding on the nodes with the highest preference weights obtain higher utility than those bidding



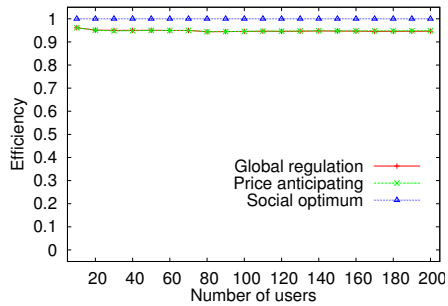
(a)

	Price Anticipating	Global Regulation
Mean	58.930	61.546
StDev	22.113	2.841
Median	66.823	61.017
Skewness	-1.735	0.473
K-S test	0.000	0.232

(b)



(c)

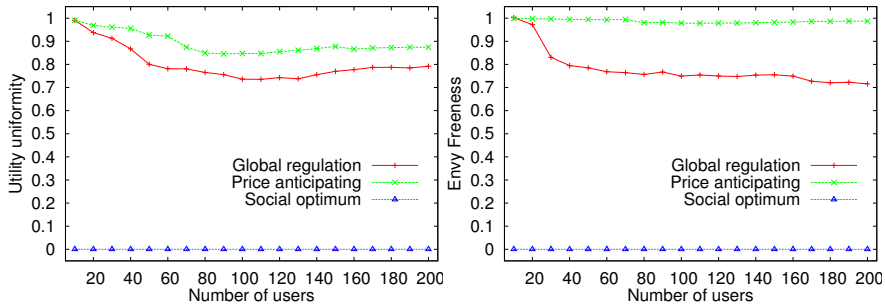


(d)

Figure 6.4: (a) CDF of load distribution, (b) Load distribution statistics, (c) Normal Q-Q plot of the load for the global regulation case and (d) Efficiency of the system.

on the less preferred nodes, as the load is similar for every node. However, user's perception is no longer envy-free, because agents who eventually bid on the nodes with the lowest preference weights (due to the increase in price on loaded nodes) would be *happier* with the allocation obtained by the users who bid on the more preferred nodes. Nevertheless, the envy-freeness index is still very high (> 0.7) compared to the social optimum, which illustrates the trade-off between maintaining a highly efficient system, redistributing the load among nodes, and maintaining a high level of fairness.

Impact of users' preferences on utility. Finally, we compare the behaviour of our system without regulation –Figures 6.6(a)(c)– and the system



(a) (b)

Figure 6.5: (a) Utility uniformity and (b) envy-freeness

with regulation –Figures 6.6(b)(d). Specifically, we compare the ratio of fitness[5] to revenue from the point of view of the resource provider –Figures 6.6(a)(b)– and the ratio of budget available to utility obtained from the point of view of the user –Figures 6.6(c)(d)[6].

Fitness and revenue show an almost linear relationship in the unregulated system. Consequently, the higher the preference weight of a resource, the more revenue it will generate, although it will also be affected by higher load (a resource obtains more virtual currency as more users bid on it). However, when regulations are enforced, the revenue generated by approximately 80% of resource providers is very similar (between 0.4 and 0.6 of normalised revenue), which means that wealth is distributed more evenly among organisations following application of regulatory taxes. This percentage represents those users "protected" by the control mechanism. However, those resources with higher fitness will still generate higher revenue because they attract higher user preference. This proves that our regulation system prevents the emergence of strong organisations (monopoly) that could dominate the resource market. As in real *welfare states*, those people with higher incomes are somehow "penalised"

[5]The *fitness* of a resource is defined as the sum of weights of all users on that resource $\varpi(j) = \sum_{i=0}^{k} w_{ij}$ where k is the number of users and w_{ij} is the weight of user i on resource j.

[6]All values are normalised in the range $[0, 1]$ considering maximum and minimum values obtained from the *price-anticipating* simulations.

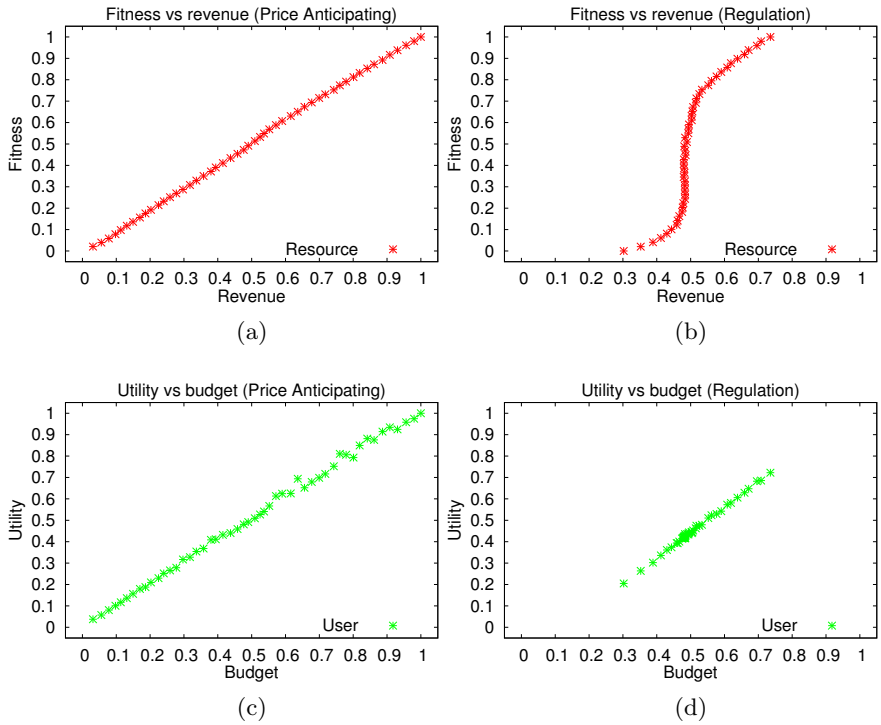


Figure 6.6: System behaviour considering variable revenue and budget among participants and assuming that each resource has different fitness. Figures (a)(c) represent the behaviour of the system without regulation (Price Anticipating). Figures (b)(d) represent the behaviour when regulatory policies are enforced (Currency Management System, CMS).

with higher taxes to increase the overall social welfare and the fairness of the state.

Because our model is decentralised (the revenue generated by a resource is spent by users belonging to the corresponding organisation to obtain resources outside the organisation), we can see that the revenue generated by resources (i.e. the budget from the user's point of view) is translated into higher utility as more budget becomes available. This is true for the simulations with and without regulation; the only difference is that wealth is distributed more equitably when regulations are enforced and, therefore, there is a cluster of users with similar utilities (i.e. utilities are less dispersed when regulatory policies are applied).

6.7 Discussion

In this section, we discuss how our proposal could be integrated into a planetary-scale distributed system like PlanetLab to solve the problems explained in Section 6.2. Firstly, our regulatory mechanism (CMS) must be scalable to cope with large numbers of resources and users. Instead of a centralised solution like GridBank [73] or Tycoon Bank [32], which are susceptible to scalability problems, we have designed a prototype [74] on top of a structured peer-to-peer overlay similar to Chord [75], which enforces and manages a virtual currency account for each user in a scalable and efficient way.

In our proposal, the load derived from managing the users' virtual accounts is distributed among the set of nodes that make up the DHT. In addition, our algorithm for calculating taxes can be executed independently by each node because it only needs information about the target load –which is not expected to change very frequently– and the load of the resources the node is responsible for managing (i.e. the resource's account is mapped to that node).

The system-wide information that the CMS uses to apply its policies –e.g. average load, effective consumption and contribution, availability, etc.– can easily be computed from the data already provided by PlanetLab's monitoring infrastructure CoMon [44], so no additional overhead is generated. Other information systems with higher scalability based on structured overlays, such as MIS [76], or discovery systems like SWORD [29], could also be used. In addition, we do not require instant information, which could be impossible to

obtain in planetary-scale systems, but instead use aggregated information for relatively long time windows, in the order of hours, days or weeks depending on the system dynamics.

Finally, we discuss how our model could be integrated into the PlanetLab environment. PlanetLab's architecture and design allows the existence of resource allocation and brokering services through resource loans. Thus, a privileged slice (i.e. a service executing across several nodes with a specific share of reserved resources) might lend a subset of a node (e.g a percentage of CPU or bandwidth) to other slices in exchange for a reward, in this case virtual money. The privileged slice would be responsible for executing and managing the proportional share mechanism explained in this chapter, thereby allowing PlanetLab users to bid for a specific resource. It would also be responsible for applying the appropriate taxes to the resources price. Next, the privileged slice would transfer the actual payment from the user's account to the resource owner's account through our CMS. The specific details and exact implementation of the protocol are beyond the scope of this work.

6.8 Conclusions

Large-scale shared and open infrastructures are based on resource contribution by organisations and are suitable for deploying planetary-scale public services and applications. Due to their diverse aims, these applications may have different, complex and often conflicting strategies. Without scalable and decentralised resource allocation, incentives and self-regulation mechanisms, these types of systems can suffer from resource overloading and congestion, which reduce the QoS offered to users.

In this chapter we presented a mechanism in which virtual currency is used to give resource providers an incentive to work collaboratively and contribute to the efficient operation of the shared infrastructure. Currency management is also an incentive for consumers to use resources efficiently and a regulatory mechanism to ensure that they behave under certain rules designed to guarantee fairer access to the resources. Currency management limits the resource capacity to which each user is entitled and taxes resource prices according to a series of specific rules. Specifically, we present in this chapter a regulatory policy for distributing work evenly between resources and preventing hot-spots.

Simulations showed that our system provides a high level of efficiency in the presence of self-interested participants and decreases the load dispersion between nodes. In addition, the system offers a fair volume of resources (utility) according to the virtual currency owned by each organisation, even under high-demand conditions.

Finally, we discussed how our proposal could be applied to a large-scale experimental facility such as PlanetLab at low cost and using existing tools and services. We considered the architecture and implementation of the PlanetLab testbed and explained how our solution could be successfully integrated.

Chapter 7

Stackelberg Game to Derive the Limits of Energy Saving for the Allocation of Data Center Resources

Abstract

Energy related costs are becoming one of the largest contributors to the overall cost of operating a data center, whereas the degree of data center utilization continues to be very low. An energy-aware dynamic provision of resources based on the consolidation of existing application instances can simultaneously address the under-utilization of servers while greatly reducing energy costs. The economics behind energy costs cannot be treated separately from resource provision and allocation. However, current scheduling techniques based on market mechanisms do not specifically deal with such a scenario. In this chapter[a], we model the problem of minimizing energy consumption when allocating resources to networked applications as a Stackelberg leadership game, to establish the upper bound of energy saving, . The model is applied to a proportional-share mechanism where resource providers can maximize profit by minimizing energy costs; while users can select resources that ensure that their minimum requirements are satisfied. We show that our mechanism can determine the optimal set of resources on and off, while maintaining user service level agreements (SLA) – even in realistic conditions considering incomplete information.

[a]This chapter is based on the papers [P3, P4]

7.1 Introduction

Energy consumption is a key concern in networked computing systems, including service overlays, content distribution systems and many other distributed systems. These systems require a collection of networked computing resources from one or many data center providers around the world.

All data centers or cloud computing providers face the problems of changing resource demand by applications and high energy costs that discourage low server utilization. For instance, Hoelzle et al. [77] looked at the average CPU utilization of 5,000 Google servers during a six-month period. It was shown that, on average, servers spend relatively little aggregate time at high load levels, and spend most of the time at the 10–50% CPU utilization range, where server efficiency in terms of energy is the lowest.

Two popular methods for effectively matching power consumption to workload requirements in a data center are via *processor speed scaling* and *powering down* nodes.

The goal of this work is to develop a theoretical framework for analyzing the limits of energy saving and strategies for determining the right set of computing nodes on and off in order to minimize energy consumption – while keeping the right level of service for networked applications using these computational resources. We consider several characteristics of modern applications and resources – e.g. proportional share of resources, granularity of application processes, level of parallelism, and migration and consolidation of virtual machines.

The focus and contributions of this chapter are as follows: i) we derive an energy model for computing elements based on SPEC benchmarks using real data; ii) we model the problem of energy minimization in resource allocation as a Stackelberg competition game to derive an algorithm to compute the upper bound on energy saving that optimizes both energy consumption and the resource requirements of applications; iii) although the model assumes complete information, we also explore the effect of having incomplete information, a key aspect in large-scale systems; iv) we also conduct a comprehensive evaluation by simulation of the influence of user budgets, their diversity, profit maximization and energy minimization.

The chapter is organized as follows. In Section 7.2 we present the energy consumption model for computing elements in a data center, and the allocation model of shared resources to multiple networked applications. Sections 7.3 and 7.4 establish a competition model, with complete and incomplete information, between a leader determining the resources to keep on and off, and the users buying resources. In Section 7.5 we provide the experimental results and findings based on simulation. Section 7.6 presents the related work and finally, in Section 7.7, we summarize the chapter and discuss future work.

7.2 Model

7.2.1 Energy consumption model

In this section we introduce a model for energy consumption of computing elements present in a data center. The energy curve of a server can be characterized in terms of the CPU load by

$$f(s) = \sigma + \mu * s^{\alpha}$$

σ represents the fixed cost of maintaining a server powered on and ready to perform work. μ, α are parameters of the device used to characterize its dynamic energy consumption taking into account the load in percentage $s \in [0, 100]$.

To understand the trend in power consumption over recent years, we examined a comprehensive set of 500 server profiles made publicly available through the SPEC (Standard Performance Evaluation Corporation) website [78] using the `SPECpower_ssj2008` benchmark. This benchmark is an industry-standard that evaluates the power and performance characteristics of volume server class and multi-node class computers.

We performed a fitting procedure using maximum likelihood estimation (MLE) [50] to the above energy function and obtained resource parameters with a low relative error of 3.7%, thus validating our model. The results (in Watts if not otherwise stated) are presented in Table 7.1.

The value of $\mu * s^{\alpha}$ (where $s = 100\%$) should be understood as the maximum amount of energy consumed by a device at full capacity without considering

Parameter	2008	2009	2010	Mean
σ	133.60	120.14	96.32	126.46
μ	1.39	4.10	5.02	3.50
α	0.92	0.81	0.76	0.83
$f(s) - \sigma = \mu * s^{\alpha}$	96.17	170.92	166.22	159.98
$\frac{\sigma}{f(s)}$ (%)	58.15	41.28	36.69	44.15

Table 7.1: Estimate of server parameters (in Watts) and considering full load where applicable ($s = 100$).

the energy consumed when the server is idle. We can see that the trend is to decrease the *idle* consumption while the *dynamic* consumption increases. This trend is due to the increasing interest from companies in achieving *energy proportionality*, which refers to the goal by which the amount of energy consumed by a device is proportional to the workload supported.

However, we also observe that the values of the idle consumption σ are still of the same magnitude as $\mu * s^{\alpha}$ in the worst case. If we look at the ratio between idle and maximum consumption, σ represents a fraction from 58% to 37% of the total consumption in the worst case –when $s = 100\%$ and the node is fully used.

From this fact, we can conclude that the idle consumption is not negligible compared to the term proportional to the load in the energy curve. If this were not the case and energy consumption was proportional to the load, then dynamic power scaling of nodes could be beneficial because moving load to other servers could enable system administrators to adapt server power consumption to the current load. Moreover, dynamic scheduling according to power consumption of nodes could also be beneficial depending on the shape (parameters μ, α) of the energy curve.

However, energy proportionality is yet a goal to achieve. Therefore, the idle consumption is a non-negligible operational cost that cannot be removed without powering down nodes.

We argue that higher savings in large-scale infrastructures can be achieved if we consider the problem of powering up or down nodes instead of focusing on scheduling decisions based on dynamic power scaling of nodes.

7.2.2 Resource allocation model

Sharing of computing elements (resources) can significantly increase system utilization by combining in a single resource different types of workloads with complementary resource requirements. Although our results are quite independent of the specific resource allocation model, we have decided to employ a market-based approach to allocate computing resources in contrast to traditional centralized schedulers.

The rationale behind this decision is the complexity of efficiently allocating applications to resources given that each user has different requirements (i.e. private information that users are unwilling to disclose) and, therefore, centralized schedulers that optimize resource allocation have a high cost in terms of information needed and computational complexity. The use of market-based mechanisms enables such data to be reduced to a single piece of information, namely the price, which enables the system to scale up.

The most popular approach is to use some form of auction to extract the market price directly from user bids using, for example, Vickrey auctions, double auctions or combinatorial auctions. Their main drawback is the slow response time (bidders have to wait for auction clearing) and the fact that auctions are unsuitable for divisible resources because of the complexity involved in determining the most efficient way to divide a resource.

However, the simplest and most appealing mechanism for shared divisible resources is the *proportional share allocation mechanism*, in which each user submits bids for the different resources and receives a fraction proportional to the user bid and inversely proportional to the sum of all other user bids for the same resource. More formally, the price of the resource is $Y_j = \sum_{i=1}^{k} x_{ij}$, where k is the number of bids for resource j and x_{ij} is a non-negative bid of user i for resource j. Following the proportional share allocation mechanism, user i receives a fraction $r_{ij} = \frac{x_{ij}}{Y_j}$ of resource j.

One of the benefits of this mechanism is that its straightforward formulation enables users to understand what will be the response of the system to a

specific bid. Current operating systems and virtual machine systems already provide proportional share mechanisms that further simplify implementation from the technological point of view [67][68][79][80].

The use of proportional share also enables dynamic pricing based on the load supported by the system by giving an incentive to users to schedule the execution of their applications based on private information – e.g. if a service has a high priority, a user will be willing to pay more when the load in the system is high although it will be reluctant to do so with low priority tasks.

Moreover, this dynamic and fine grained partition of a single resource in virtual machines might potentially improve the consolidation of applications compared to current consolidation techniques that consider fixed-sized virtual machines.

As mentioned previously, our aim is to minimize energy consumption by considering energy costs. We extend this mechanism to consider the state of a resource – i.e. *on* or *off* state. Therefore, the fraction of a resource received by user i is actually $r_{ij} = q_j \frac{x_{ij}}{Y_j}$, where $q_j \in \{0, 1\}$ is a binary variable representing the state of resource j – i.e. 0 if the resource is shutdown and 1 when powered up.

7.3 Stackelberg competition model

One of the contributions is a novel model to characterize the competition between the resource provider and its clients. We develop a model of the data center economy concept considering energy costs as a Stackelberg game [81] on two levels. The first level is the infrastructure operator acting as the leader and determining the resources to keep on and off. The second level is the set of strategic users buying resources as followers.

We then present the respective payoff functions of users and the infrastructure operator, and the efficient algorithms to compute the best response of both roles when considering each others' actions. This model will enable us to provide lower and upper bounds for metrics of interest to both users and providers.

7.3.1 User model

To model user [1] behavior, we consider the utility function presented in [36], a linear payoff function $U_i(r_{i1}, \ldots, r_{im}) = q_{i1}w_{i1}r_{i1} + \ldots + q_{im}w_{im}r_{im}$, where r_{ik} is the resource share obtained from resource k, $w_{ik} \geq 0$ is user i private preference for resource k, and $q_{ik} \in \{0, 1\}$ represents whether the resource is off or on respectively. However, following the trends of resource providers of providing homogeneous resources to users in the form of virtual machines, these weights or preferences for actual resources no longer apply and thus $\forall_{j,k} w_{ij} = w_{ik}$.

Typically we also assume that users are selfish and strategic – i.e. they all act to maximize their own utility as defined by their payoff functions. The best response of an agent given a set of resource prices is to find the set of bids on resources that maximizes its utility function. More formally, the best response is the solution to the following optimization problem (7.1):

$$\max U_i(x_1, \ldots, x_m) = \sum_{j=1}^{m} q_{ij} \frac{x_{ij}}{x_{ij} + y_j} \tag{7.1}$$

$$\text{subject to } \sum_{j=1}^{m} x_{ij} \leq X_i$$

$$\frac{x_{ij}}{x_{ij} + y_j} \geq \phi_i \in (0, 1]$$

$$|S| \geq \tau_i$$

7.3.2 Restrictions

We assume several restrictions for the optimal solution of each agent. The first restriction is that each user i has a budget (or money) constraint X_i and its total positive bids must match its budget. This constraint represents the fact that each user does not have an infinite budget, as opposed to standard assumptions in traditional economic models.

The following restrictions can be seen as the minimal SLA a user is willing to accept in order to be satisfied with the resource provider. The second restriction

[1] We may use the word *user* and *application* indistinctly through the chapter to describe an entity which consumes resources from the infrastructure.

assumes a certain application granularity. An application i needs at least a minimum share at each node specified by ϕ_i. Therefore, the optimization problem should select those resources in which this constraint is satisfied. Notice that this restriction may be reformulated in terms of the minimum bid to state: $x_{ij} \geq \frac{y_j \phi_i}{1-\phi_i}$.

The third restriction is on the level of distribution[2]. Let $S = \{r_{ij} : r_{ij} \geq \phi_i\}$ be the set of nodes in which the second restriction is satisfied. We assume that an application has a minimum level of parallelism (defined as the cardinality of S) of up to τ_i nodes. If this restriction is not satisfied, the user has no incentive to participate in the resource market as its requirements will not be met. To simplify the notation through the rest of the chapter, we denote this restriction as a boolean named *satisfaction*.

The second and third restrictions improve the representation of real distributed applications. Previous models did not consider that an optimal allocation for the above problem could contain a share so small that it is impossible (or impractical) to make an allocation in a real computing resource.

Although this model is more realistic in terms of application requirements, we still do not consider the cost in time of moving virtual machines between nodes. However, recent results for live migration in several virtual machine monitors show migration latencies at the millisecond level, supporting our decision to disregard this cost [82][83][84].

7.3.3 User best response algorithm

Algorithm 7.1 summarizes the algorithm for computing the best response – i.e. the bidding vector $x^* = (x_1^*, \ldots, x_m^*)$ that maximizes the payoff function U_i – considering the above optimization problem (for a complete proof of correctness refer to [36]). In short, it determines the maximum number of bids x_j on a resource j such that $x_j \geq \frac{y_j \phi_i}{1-\phi_i}$.

The computational complexity of this algorithm is $O(m \log m)$, dominated by the initial sorting. This algorithm is distributed in the sense that each user finds the set of bids that maximizes its own utility considering its private information

[2]Distribution is understood as the need for several instances of a virtual machine in separate nodes. This need would arise when dealing with fault-tolerance, load-balancing, etc.

Algorithm 7.1 UserBestResponse(ϕ_i, X_i) – User i's best response algorithm

Require: ϕ_i ▷ user i's minimum share
Require: τ_i ▷ user i's minimum number of nodes
Require: X_i ▷ user i's budget
Require: $\{y_1, \ldots, y_m\}$ ▷ list of resource prices
Require: $\{q_1, \ldots, q_m\}$ ▷ list of resource states (on/off)
$M = \{y_j : q_j = 1\}$ ▷ list of prices of *on* machines
Sort the set M by y_j in increasing order
Compute largest k such that

$$\frac{\sqrt{y_k}}{\sum_{i=1}^{k}\sqrt{y_i}}(X + \sum_{i=1}^{k} y_i) - y_k \geq \frac{y_j\phi}{1-\phi}$$

Set $x_j = 0$ for $j > k$, and for $1 \leq j \leq k$, set:

$$x_j = \frac{\sqrt{y_j}}{\sum_{i=1}^{k}\sqrt{y_i}}(X + \sum_{i=1}^{k} y_i) - y_j$$

return k, $(x_1, \ldots, x_m)$ ▷ k is the number of nodes with positive bids

(ϕ, τ and X). Following an iterative process of updating the bidding vector, the utility and resource prices converge to an efficient equilibrium in a few iterations (2-3 iterations in our experiments).

7.3.4 Provider model

The payoff function of the resource provider is that of maximizing its profit when considering income (price paid by users for each resource) and the cost of maintaining the infrastructure (price paid by the provider for its *on* resources).

As the provider is in a privileged position to decide in advance which resources to keep on-line and which ones to shutdown, we can model the interaction between users and provider according to a Stackelberg game, in which the provider acts as the leader (deciding the values of q_j) and users act as followers who decide their optimal bidding vector once the values of q_j are announced.

More formally, the best response is the solution to the following optimization problem (7.2):

$$\begin{aligned} &\max && P(q_1,\ldots,q_m) = \sum_{j=1}^{m}(q_j \sum_{i=1}^{n} x_{ij}) - \sum_{j=1}^{m} q_j c_j \\ &\text{subject to} && \forall_i \, |S_i| \geq \tau_i \end{aligned} \tag{7.2}$$

7.3.5 Restrictions

We assume that the provider has no incentive to perform admission control on users given that there are enough resources to accommodate the aggregate demand from users. In other words, the provider's restriction is to satisfy every user –i.e. minimum resource requirements are met – providing there are enough resources. An interesting line of future work would be to increase profit by considering admission control on users and keeping those users who provide a higher income with a minimum associated cost. However, in this chapter we consider that admission control is made beforehand through a contract between a user and the provider. Therefore, our goal is to satisfy all contracts previously accepted.

7.3.6 Provider best response algorithm

The Stackelberg model can be solved by finding the subgame perfect Nash equilibrium (SPNE) –i.e. the strategy profile that best serves each player, given the strategies of the other player. This approach entails every player playing in a Nash equilibrium.

A common method for determining subgame perfect equilibrium in the case of a finite game is backward induction. To calculate the SPNE, the best response functions of the follower must first be calculated. As we already provide an efficient algorithm to compute the user best responses, we now present an algorithm to compute the provider best response –the leader in our game.

To compute the best response, we first sort machines by their associated energy cost in increasing order. Without loss of generality, suppose that $c_1 \leq c_2 \leq \ldots c_m$. Suppose that $q^* = (q_1^*, \ldots, q_m^*)$ is the optimal solution (values of 0 and 1 representing *off* and *on* machines).

Claim. If $q_i^* = 0$, then it must hold that for any $j > i$, $q_j^* = 0$.

PROOF. By the optimality constraint in Equation 7.2, we have that $\forall_i |S_i| \geq \tau_i$, and therefore, each user participating in the market spends all of its budget in the optimal subset of *on* resources (recall that *off* resources have a price of 0). Then:

$$\sum_{j=1}^{m}(q_j^* \sum_{i=1}^{n} x_{ij}) = \sum_{i}^{n} X_i$$

Substituting the provider's payoff function we have that:

$$P(q_1^*, \ldots, q_m^*) = \sum_{i=1}^{n} X_i - \sum_{j=1}^{m} q_j^* c_j$$

As $\sum_{i=1}^{n} X_i$ is a constant value, we need to minimize $\sum_{j=1}^{m} q_j^* c_j$ to maximize $P(q_1^*, \ldots, q_m^*)$.

Now suppose that *Claim 1* is untrue. There is then a q' permutation of q^* that is different to q^* with the same number of *on* machines. Recall that q^* already contains the minimum amount of *on* nodes to support the user workload following the hypothesis of q^* being optimal. The sum of the *on* machines in q' is then less than the sum of the costs of the *on* machines in q^*,

$$\sum_{j=1}^{m} q'_j c_j < \sum_{j=1}^{m} q_j^* c_j$$

We construct q' by swapping two elements k, l of q^* where $k < i \leq l$ in such a way that $q'_k = q_l^* = 0$ and $q'_l = q_k^* = 1$. Although we only present the proof for two different elements for the sake of clarity, the proof for more than two different elements follows the same principle presented here.Then,

$$\sum_{j=1}^{k-1} c_j + \sum_{j=k+1}^{i-1} c_j + c_l < \sum_{j=1}^{k-1} c_j + c_k + \sum_{j=k+1}^{i-1} c_j$$

Canceling terms, we have that

$$c_l < c_k$$

But, by construction, $k < i \leq l \rightarrow k < l$. As machines are ordered in increasing order of their energy costs, we have that $c_k \leq c_l$ and we conclude with a contradiction. □

Intuitively, as q^* already contains the nodes with the lowest associated energy cost, there is no other arrangement of q^* that yields a lower aggregated cost.

The remaining question is to determine the number of *on* nodes. This is the minimum number of nodes such that each user is satisfied with their allocation. Recall that the satisfaction of each user is determined by their minimum requirements in terms of share in a node ϕ and a τ number of minimum nodes. Thus, we obtain the following algorithm to compute the best response of a provider in Algorithm 7.2.

Algorithm 7.2 ProviderBestResponse – Provider best response algorithm

Require: $\{\phi_1, \ldots, \phi_n\}$ ▷ users minimum share
Require: $\{\tau_1, \ldots, \tau_n\}$ ▷ users minimum number of nodes
Require: $\{X_1, \ldots, X_n\}$ ▷ users budget
Require: $M = \{c_1, \ldots, c_m\}$ ▷ list of resource costs
  Sort the set M by c_j in increasing order
  k=0
  repeat
    $k \leftarrow k + 1$
    Set $q_j = 1$ for $j \leq k$
    Set $q_j = 0$ for $j > k$
    repeat
      for all user i **do**
        S_i, $(x_1^i, \ldots, x_m^i) \leftarrow$ UserBestResponse(ϕ_i, τ_i, X_i)
      end for
    until convergence
  until $\forall_i S_i \geq \tau_i$ OR $k = m$
  return $(q_1, \ldots, q_m)$

Intuitively, it checks if the outcome of the SPNE complies with the *satisfaction* constraint for all users by iteratively adding a single machine into the set of *on* machines. This algorithm stops when every user is satisfied with its allocation or the provider needs to keep every machine on. The computational complexity of this algorithm is $O(m^2 n \log m)$ dominated by the external loop with a cost of $O(m)$ and the internal loop that executes for each user its best response with a cost of $O(nm \log m)$ (as mentioned earlier, the rate of convergence is a small constant in the order of units). We could improve the computational cost to $O(mn \log^2 m)$ by replacing the external linear search loop with a binary search with a cost $O(\log m)$. However, we present this linear algorithm for the sake of clarity.

7.4 Strategies with incomplete information

The algorithm presented above is useful for understanding the upper and lower bounds on metrics of interest for the provider. However, the main problem with the above model is the complete information required by the provider in order to compute the SPNE. Thus, we present two strategies that use only aggregated information (either economic or technical) available to the provider –i.e. without knowing the private information of each user.

Proportional cost threshold strategy. In this case, the provider will decide to shutdown those machines that do not cover their own energy cost. More formally, it will shutdown the set of machines $OFF = \{j : \forall c_j > \sum_i^n x_{ij}\}$. The intuitive idea behind this is to shutdown those nodes that do not report a positive profit for the provider because of their high energy costs.

Demand aware strategy. In this case, instead of relying on the energy cost of each resource, the provider will make use of the aggregate demand present in its resource pool. In this way, it will compute a rough number of nodes that could potentially carry the current load. More formally, we compute the number of nodes k to maintain on-line, considering a mean resource usage of $p \in [0, 1]$ and a confidence parameter $\alpha \in [0, \infty]$, as $k = (1 + \alpha) * p * m$.

As in the complete information case, we then sort machines by cost in increasing order and set $q_j = 1$ for $j \leq k$ and $q_j = 0$ for $j > k$. Intuitively, we are *estimating* the optimal number of machines necessary to satisfy every user. The parameter α is related to the confidence on the estimation of the mean

resource usage p or aggregate demand. In addition, α represents the trade-off between user satisfaction and the reduction in energy costs. A higher α will lead to an *overestimation* of the necessary resources to keep users satisfied with the cost of maintaining resources on, while a lower α will be more aggressive and accept the risk of user's insatisfaction at the benefit of lower costs.

7.5 Experimental results

In this section, we provide detailed experimental findings. In our simulations we fix the number m of nodes at 1000 and we assign an individual cost of c_j in Watts to each node drawn from a uniform distribution $U \sim (96.32, 133.60)$ extracted from real measurements and summarized in Table 7.1. We assume that the economic cost of each Watt is one monetary unit.

We compare the behavior of the strategies presented in previous sections by varying the number of users, budget, and minimum requirements ϕ and τ.

7.5.1 Metrics

Profit is the outcome of the providers utility function defined in Equation 7.2. That is, it is the total income provided by users bidding on available machines minus the energy cost of those available machines. As we are not interested in the actual profit but the ratio between the profit and the money available in the system –i.e. the money owned by users– we normalize the resulting profit by $\sum_i^n X_i$. Formally,

$$\text{Profit} = \frac{\sum_j^m (q_j \sum_i^n x_{ij}) - \sum_j^m q_j c_j}{\sum_i^n X_i}$$

Energy saving is the ratio between the cost saving achieved by shutting down nodes and the total cost if all nodes were available. Formally,

$$\text{Energy savings} = 1 - \frac{\sum_j^m q_j c_j}{\sum_j^m c_j}$$

Server usage is the mean server usage of the whole infrastructure. In our simulations, users consume up to ϕ of a single resource. For example, if a

user's share of resource is $r_{ij} = 0.7$ and its resource requirements $\phi = 0.3$, then 40% of that resource will be unused –i.e. resource reservation minus actual usage. Recall that $r_{ij} \geq \phi$ is always true because of the optimality restrictions of the user payoff function presented in Equation 7.1. Instead of a percentage, we present a ratio in the range $[0, 1]$ for comparison purposes.

Satisfaction shows the efficiency of our strategies in terms of user satisfaction. It is the ratio between the number k of satisfied users (those fulfilling their satisfaction criteria) and the total number n of users.

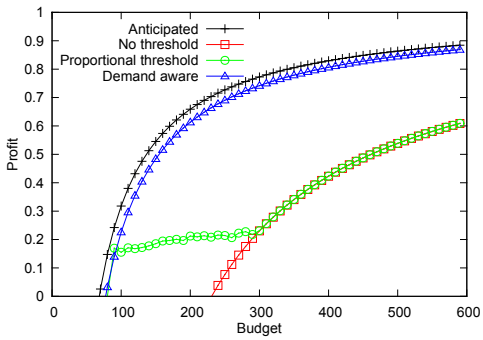
User efficiency or price of anarchy (PoA) is defined as the ratio between the social welfare $U(\omega)$ considering a current allocation ω and social welfare at the optimal allocation U^* where $U^* = \max(U(\omega))\ \forall\omega$. We compute social welfare as the $\sum_i^n U_i(r_1, \ldots, r_m)$ or, intuitively, the sum of the utilities obtained by each user considering their individual allocations. For an explanation on how to compute the optimal social welfare U^* we refer the reader to [36].

For the sake of comparison, we introduce another simple strategy which is to maintain all nodes always available (*No cost* in the figures). This strategy will enable us to compare the behavior of different strategies and provide a lower bound on how the system would behave regardless of the strategy followed by the provider.

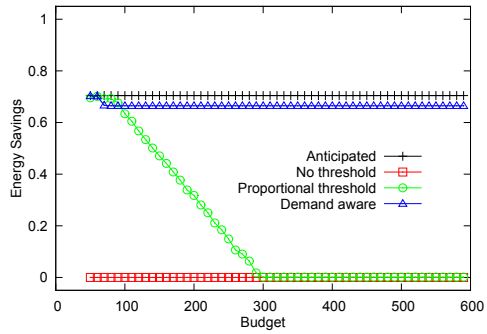
7.5.2 Impact of user budgets

We first experiment with a fixed number n of homogeneous (running equivalent applications) users of 250 –meaning a ratio of 1 distributed application per 4 machines, a reasonable scenario. For each user i we fix its minimum requirements $\phi_i = 0.1$ and a minimum number of virtual machines $\tau_i = 5$. In this way, we introduce a constant light workload that could be supported by a subset of the whole infrastructure. Through subsequent iterations, we increase the budget of each user in the range $[60, 600]$ with steps of 10 monetary units. The expected behavior is that the greater the budget available to users, the more resources they buy to increase their own utility.

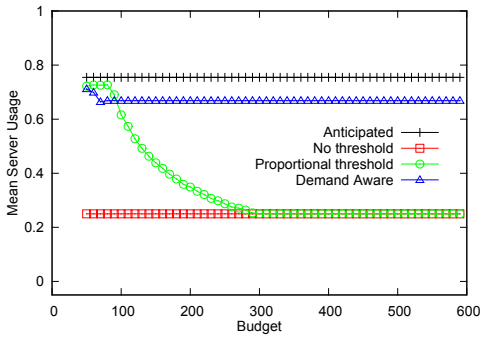
The results show that for the simplest strategy of maintaining the whole infrastructure available (*No threshold* in Figures 7.1(a)-(e)) the provider has no incentive to participate in the market –i.e. its profit is less than zero– until the level of income is at least equal to the cost of maintaining the infrastructure



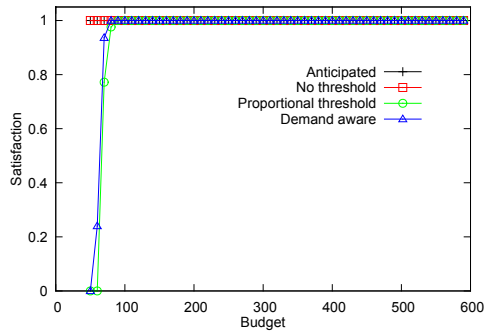
(a) Profit



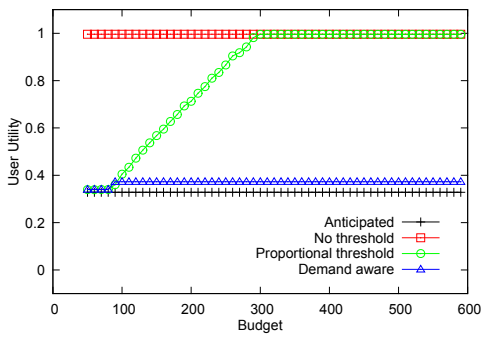
(b) Energy Savings



(c) Server Usage



(d) Satisfaction



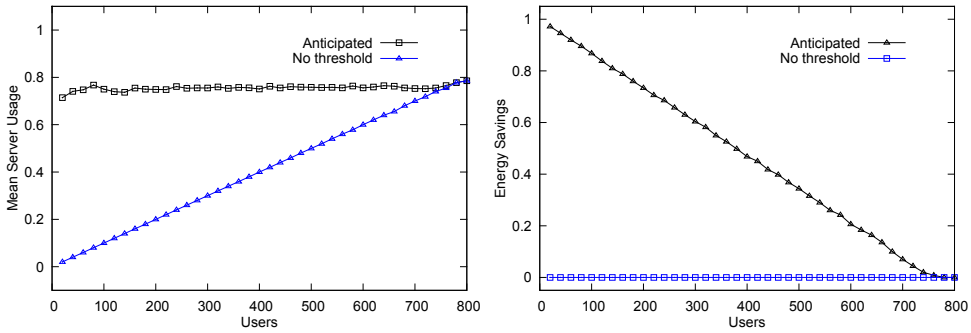
(e) User Efficiency

Figure 7.1: Results with variable budget per user

(around 220 monetary units per user). Moreover, the energy savings and server mean usage are the lowest possible given that the whole infrastructure is active and the workload is constant. However, in this case, user satisfaction and efficiency are the highest possible as there are always enough resources to meet each user requirement. This scenario is interesting as it provides lower bounds for energy savings and profit while providing upper bounds for user satisfaction and user utility.

In contrast, let's look at the strategy derived from the Stackelberg game (*Anticipated* in Figures 7.1(a)-(e)). This has proven to be the best response of the Stackelberg game and in which the provider has a privileged position over its followers. Figure 7.1(a) shows that the greater the budget available to users the more profit is obtained by the provider because it can adjust the number of available machines for a given workload (which is fixed in these simulations). At the same time, the mean server utilization (Figure 7.1(c)) is higher than in the simplest strategy because of the consolidation of applications into a subset of the infrastructure. More importantly, this adjustment of available resources to a given workload implies a drastic reduction in energy costs and so makes the infrastructure more environmentally and economically efficient (Figure 7.1(b)). However, this privileged position gives the provider a great advantage over users as shown in Figure 7.1(e). In this case, the provider's profit is maximized whereas the efficiency considering user utility is decreased to the minimum considering the minimum requirements of *satisfaction*. This scenario, although unrealistic because of the need for complete information, is interesting as it provides an upper bound on energy savings, profit, and mean server usage; and a lower bound on user efficiency.

Any other strategy should lie between these two scenarios. The *proportional threshold* strategy behaves similarly to the *anticipated* strategy when there is little money in the system, and its performance decreases as more money is introduced. For example, Figure 7.1(a) shows that profit slowly increases as more budget is introduced by users in contrast with the rapid increase of the *anticipated* strategy. This is because users are able to pay for the specific costs of each machine as they have more money – independently of the workload. We can see a point of *saturation* (around a budget of 300 monetary units) where users have enough money to awaken all the resources of the infrastructure and then behave as in the *no cost* strategy.



(a) Server Usage (b) Energy savings

Figure 7.2: Results with variable demand in the number of users

Interestingly, Figures 7.1(a)–7.1(e) also show the behavior of the *demand aware* strategy in which the infrastructure considers the aggregate demand to estimate the number of nodes to shutdown with an $\alpha = 0.25$. The value of α represents a trade-off between provider satisfaction (increasing revenue) and user satisfaction. A lower value of α would lead to shutting down more nodes and potentially decreasing user satisfaction; yet increasing energy savings and therefore reducing costs and increasing provider profit. Conversely, the behavior of this strategy would move slowly towards the *no cost* behavior if α were slowly increased. This scenario is also interesting as tuning this parameter would enable system administrators to decide – using only aggregated information – at which point of the solution space to operate between the *no cost* (lower bound) and *anticipated* (upper bound) strategies. This information is commonly available in currently deployed monitoring infrastructures.

7.5.3 Variable number of users

In these experiments, we wanted to understand the behavior of our strategies when the amount of money in the system grows to become excessive. That is, each user has enough money to buy the whole infrastructure. We also modify the aggregate workload by changing the number of users in the system. We only show the *anticipated* and the *no cost* strategies as we have seen that any other strategy would behave in between. Figure 7.1(c) shows the mean server usage for each strategy. As we increase the number of users, the

anticipated strategy calculates the resources necessary to cope with such a workload and maintain a high overall utilization independently of the workload. This result comes from the fact that this strategy provides the strict minimum number of nodes necessary and so, maintains a high degree of utilization – while increasing the energy costs as more nodes become available to users (see Figure 7.2(b)). However, the *no cost* strategy always provides the whole infrastructure – i.e. without energy savings – and the degree of utilization increases linearly as more users are added.

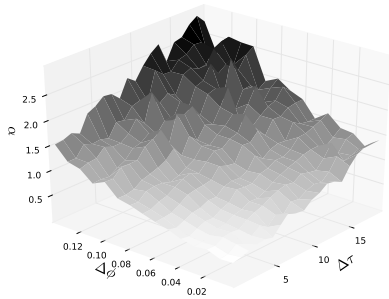
7.5.4 Impact of users heterogeneity

Our last set of results shows the impact on system performance considering heterogeneous users (with different minimum consumption ϕ and minimum parallelism τ) and the behavior of the *demand aware* strategy with a homogeneous budget among users (Figure 7.3) and proportional budget among users with respect to the private demand parameters minimum consumption ϕ and minimum parallelism τ (Figure 7.4).

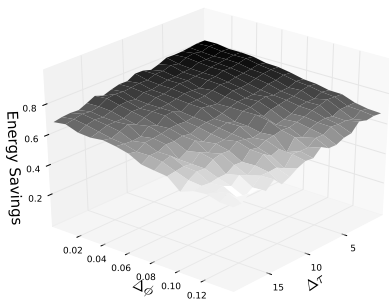
In both cases, we select for each user a consumption ϕ drawn from a uniform random variable $U \sim (0.15-\Lambda_\phi, 0.15+\Lambda_\phi)$ where $\Lambda_\phi \in [0, 0.15]$ changing along the X-axis, and a τ drawn from a uniform random variable $U \sim (20-\Lambda_\tau, 20+\Lambda_\tau)$ where $\Lambda_\tau \in [0, 20]$ changing along the Y-axis. The Z-axis represents the variable we are studying – as a visual aid, the darker the surface the higher the value. With this setup, we aim to study the effect of the degree of heterogeneity on certain variables.

For a given demand, we compute the *anticipated* strategy to calculate the optimal allocation using the Stackelberg game. For Figures 7.3 and 7.4, the surface represents the optimal value of the studied variable. Values above the surface represents an over-provision of resources (waste of energy) and values below the surface represent an under-provision of resources (unsatisfied users).

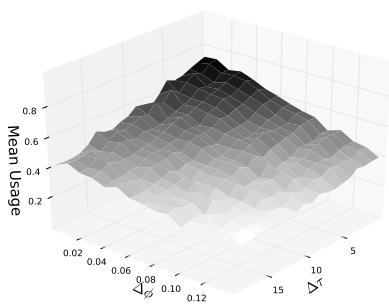
Figure 7.3(a) shows what would be the optimal confidence value α in the *demand aware* strategy. It shows that when there is no diversity among applications –i.e. $\Lambda_\phi = 0$ and $\Lambda_\tau = 0$ and, therefore, users are homogeneous with the same minimum requirements– the estimation of the number of nodes through the aggregate demand works quite well since the α value is low. In other words, the correction that should be made is minimal. However, as we



(a) Confidence value α

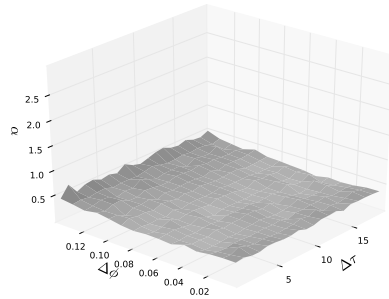


(b) Energy Savings

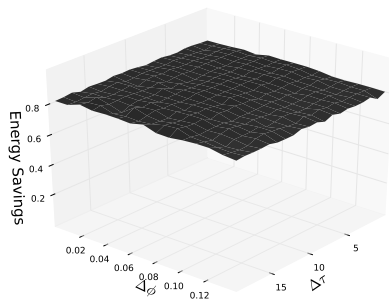


(c) Mean server usage

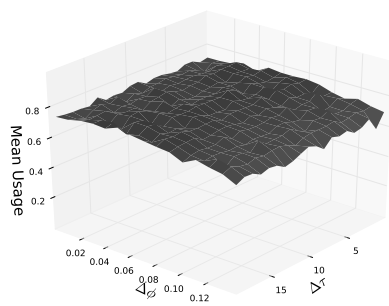
Figure 7.3: Behavior of α, energy savings and mean server usage with respect to the degree of freedom in user's requirements with homogeneous budget



(a) Confidence value α



(b) Energy Savings



(c) Mean server usage

Figure 7.4: Behavior of α, energy savings and mean server usage with respect to the degree of freedom in user requirements with proportional budget for user workload.

increase diversity among users –i.e. increasing Λ_ϕ and Λ_τ– the value of α should increase because of the loss of information caused by using aggregate load values instead of complete information.

We also observe that a greater diversity implies a loss in server utilization and therefore in energy saving (see Figures 7.3(b) and 7.3(c)[3]). The conclusion we can draw from these results is that when heterogeneous users with the same budget compete for a finite set of resources, there is an unfair competition among *over-budgeted* users and *under-budgeted* users. In other words, a user with small resource requirements will pay the same amount of money and will obtain the same share of resources as a user with higher resource requirements. Although these *over-budgeted* users obtain a specific share of a resource, they will only use up to its minimum requirements –e.g. a user may purchase 75% of a resource but only use 25%, which implies that 50% of that resource would be unused.

These findings lead us to the question of what would be the right budget for a user given its private demand. Our intuition was that users should have the right to access resources in a fair manner. Accordingly, we explored the previous scenario but this time considering heterogeneous budgets among users proportional to their workload. In this way, we specify the budget of user i as $X_i = B \times \phi_i \times \tau_i$ where B is an arbitrary large number (enough to buy all resources).

Figures 7.4(a)–7.4(c) show that with a proportional budget among users, the three metrics studied (α, energy savings and mean server usage) describe a flat surface and this means that they are independent of the variability between user requirements. Note that the α value and optimal energy savings drop in Figure 7.3 as we decrease and increase the variability respectively; however, this is not the case in Figure 7.4. The conclusion we can extract is that the influence of user market power (through budgets) on the resource market should not be neglected as we have shown that it has an impact on power consumption. Thus, from the provider perspective, it should offer an incentive for users to adjust their budgets in proportion to their requirements.

[3]To avoid confusion and overlapping of the surface, notice the change on the order of the X-Y-axis

If proper incentives were given to users to adjust their budgets, the infrastructure operator would be able to choose the right amount of resources to be made available without complete information –because α would become a constant that is independent of user heterogeneity. This leaves an interesting open question for future work.

7.6 Related Work

Energy related costs will become the single largest factor in the overall cost of operating a data center, while most data centers continue to have underutilized servers. Most of current work focuses on scheduling techniques and architectures for consolidating virtual machines in fewer servers to reduce energy consumption. Takeda et al. [85] propose an algorithm to migrate virtual machines based on priorities among competing applications. Verma et al. [86] also considers the performance issues associated with the design and implementation of pMapper, a power and cost-aware scheduling algorithm for virtual machine placement. Schröder et al. [87] discuss an interesting scenario of migration among geographically distributed data centers. The primary concern of these works is the design and evaluation of a specific piece of software with specific workloads regardless of the economic issues behind user behavior. Moreover, their scheduling techniques consider complete information on application performance requirements. In contrast, we focus on providing algorithms for computing upper and lower bounds on the number of resources needed to support a workload when considering the economic decisions performed by either heterogeneous and homogeneous applications, and from which upper and lower bounds on provider profit, energy savings and overall data center performance may be extracted.

To the best of our knowledge, our work is the first to model a theoretical economic framework using *Stackelberg games* to understand the trade-offs in user demand, market power, and provider decisions in the context of energy efficiency in computer networks. However, Stackelberg games have been extensively used in the literature for understanding the behavior of a set of users when considering that one (or several) have a specific advantage such as information, market power, decision power, etc. This situation arises quite often in the real world and makes this type of game very appealing to model and analyze new mechanisms. For example, Biczók et al. [88]

use a Stackelberg game to model the problem of users willing to share their bandwidth at home as opposed to the traditional ISP–consumer relationship in the context of wireless community networking. Regarding power control on wireless networks, Wang et al. [89] studied the problem of resource allocation in a decentralized and cooperative network setting by modeling users in a two level game (Stackelberg game) where source nodes play the role of buyers and relay nodes play the role of sellers.

At the network level, different techniques to reduce energy consumption have been proposed. Most of the work focuses on exploring models related to speed scaling and power down methods to match network traffic to available resources. For example, Andrews et al. [90][91] studied energy minimization algorithms at a global wired network level in both models respectively and Nedevschi et al. [92] considered energy minimization when the operational voltage and the frequency of transmission could be scaled. Although these works are not intrinsically related to ours, we believe that energy minimization at both network and server levels should go together and there is an open door for future work in the intersection between these areas.

7.7 Conclusion

This work investigates the limits of energy saving for the allocation of data center resources by finding an algorithm that is efficient in selecting the optimal set of active resources to maximize energy consumption while satisfying the minimum performance requirements from networked applications. The focus is on the choice to power down and power up nodes, as most of the power consumption in energy curves for computing nodes is determined by the fixed cost for being on. We model the interplay between the need for resources from users and the goal to minimize power consumption as a Stackelberg competition, where the market leader determines which resources should be on and off, and the followers select among the resources left on. The game ensures that networked applications, the followers, will have enough resources to comply with their minimum performance levels. Although we focus on a proportional share model for resource allocation in this work, the results can be applied to other resource allocation models.

The results can be extended in many directions. One possibility is to demonstrate the feasibility of energy minimization through an implementation with

real workloads; making comparisons with the upper bounds presented here and evaluating the implementation with other resource allocation models (economic and traditional). Energy-aware admission control of users might help to keep and optimize the most profitable set of users in terms of income and associated cost. The resource model can be extended by looking at nodes with multiple power levels that depend on load instead of just being switched on and off. An interesting line of work is to study and analyze the intersection of energy minimization at the network and server levels. Finally, another direction is to explore competition models with incomplete information on user strategy (Bayesian games).

Chapter 8

Incentives for Dynamic and Energy-aware Capacity Allocation for Multi-tenant Clusters

Abstract

Large scale clusters are now being used in shared, multi-tenant scenarios by heterogeneous applications with completely different requirements. In this scenario, it's interesting to explore the intersection of two different lines of research. On one side, energy efficiency is an important factor to consider in this world with increasing operating costs related to energy consumption. On the other side, heterogeneous applications emphasize the problem of distributing the execution capacity among competitive users in a shared setting. In this chapter[a], we address these two problems by introducing an incentive mechanism to make users report their actual resource requirements, allowing them to dynamically scale-up or down as necessary. In turn, this information is used by the infrastructure operator to shut down resources without reducing the QoS provided to users and effectively reducing energy costs. We show how our mechanism is able to meet the performance requirements of applications without over-provisioning physical resources, which in turn is translated into energy savings.

[a]This chapter is based on the paper [P5]

8.1 Introduction

Energy consumption is a key concern in networked computing systems, including service overlays, content distribution networks, and many other distributed systems. One of the main usages of large scale clusters is data analysis and manipulation using distributed computation models like Map Reduce [93]. In this context, all data centers or cloud computing providers face the problem of a changing resource demand by applications over time and high energy costs that makes low server utilization a luxury.

Hoelzle and Barroso [94] looked at the average CPU utilization of 5000 Google servers during a six-month period. It was shown that, on average, servers spend relatively little aggregate time at high load levels, but that they spend most of the time at the 10-50% CPU utilization range, where server efficiency in terms of energy is the lowest. Given that, energy efficiency should be considered a first-order metric when designing data centers and the software they run.

On a different perspective, these large scale clusters are now being used in shared, multi-user settings in which submitted applications may have completely different requirements in terms of space (number of simultaneous running tasks) and time (duration of the execution), from small almost interactive executions, to very long program that take hours to finish. This situation makes task scheduling, which is the mechanism by which jobs are assigned a set of resources, even more relevant.

For example, Hadoop (the implementation of the Map Reduce model of computation supported by the Apache foundation) provides two different schedulers for multi-tenant scenarios: the *fair share* scheduler and the *capacity* scheduler. The former focus on delivering a similar share of resources to all running applications targeting fairness as their main objective while the latter focus on accommodating heterogeneous applications with different share requirements. However, both approaches are highly static in the sense that shares are granted by high-level policies decided by the infrastructure operator, and end users must negotiate a change in case the allocated shares are not enough. Besides, none of the approaches considers energy efficiency as a metric on the scheduling decisions.

To better understand the rationale behind our design, we start with a set of desirable properties that any resource scheduling policy should satisfy among others: *social efficiency*, the allocation should maximize the utility or satisfaction (quality of experience, QoE) perceived by users; *capacity differentiation*, it should be capable of provisioning different capacity to different applications depending on their needs; *fairness*, each application should have a chance to obtain resources proportional to its assigned capacity; *elasticity*, or high utilization, a job should not be delayed or not executed if there are free or spare resources in the infrastructure which may occur because of an imbalance of execution capacity among users; *dynamically adaptive*, application capacity should be allowed to change or adapt dynamically depending on application's needs without (or minimal) intervention of administrators to increase responsiveness; and *energy efficient*, in today's world where energy operating costs are an important share of the total costs, the scheduling algorithm should consider energy costs as a first-order metric to optimize the energy consumption related to operating the infrastructure.

Current schedulers already consider capacity differentiation, fairness and elasticity as important elements on its design. Thus, in this chapter we focus and explore the intersection of two additional lines of research: i) energy efficiency of big data clusters through resource scheduling policies and ii) enabling dynamic capacity allocation on shared multi-tenant clusters.

From the energy efficiency perspective, it's risky for resource provides to simply power down a portion of the infrastructure without breaking service level agreements (SLAs) usually described as deadlines, given the heterogeneity of running applications and their changing requirements over time. Thus, we propose an incentive mechanism to maximize progress at an acceptable rate minimizing power consumption or unnecessary resource usage by *promoting* users to report their actual requirements in terms of resources instead of deadlines.

From the scheduling point of view, our goal is to move from operator-oriented static allocation policies to a user-oriented dynamic allocation to provide guarantees to users depending on their global or instant needs, without the intervention of the infrastructure operator, instead of relying on fixed policies to allocate resources.

This chapter is organized as follows. Section 8.2 introduces some technical concepts related to map-reduce to situate the context of our mechanism. Section 8.3 and section 8.4 present our incentive mechanism and associated algorithms respectively. Section 8.5 presents the simulation results that support the contribution of our work. Finally, section 8.6 shows the related work and we conclude the chapter in section 8.7.

8.2 Background

The Map Reduce [93] model of computation was originally designed by Google to exploit large clusters to perform parallel computations on extremely large sets of data. It basically consists on the implementation of two functions by a programmer: a *map* function, which processes fragments of input data to produce intermediate results, usually in the form of key-value pairs. This intermediate data then feed a *reduce* function to combine the intermediate results to create the final output.

All nodes in the cluster execute these functions on different subsets of data. The Map Reduce runtime divides and distributes the data across nodes and collects the results once nodes finish their calculations.

Although there are different implementations of this model for different purposes and architectures [95][96], we focus on Hadoop [97], one of the most widely used frameworks by companies like Yahoo!, Facebook or Amazon.

One of the core components of Hadoop is the job scheduler which allocates resources to jobs. Currently, there are three different scheduling policies implemented on Hadoop. The *FIFO* scheduler which pulls jobs from the work queue, oldest being the first. This scheduling policy has no concept of the size or resource requirement of a job, but the approach was simple enough to implement at first. The *fair share* scheduler –developed by Facebook– assigns resources to jobs in a way that on average, each job obtains a similar share of the available resources over time. This scheduler is able to interleave low consuming jobs with short time spans with jobs that require more resources and more time to complete, providing a more responsive system and avoiding starvation of small jobs in favor of larger ones. Finally, the *capacity* scheduler –developed by Yahoo!– shares some of the principles of the fair share scheduler in the sense that a certain amount of shares can be assigned to different

users or applications. However, it was defined for large clusters with multiple, independent users and target applications, providing greater control over the capacity guarantees among users.

An important factor about the operation efficiency of Hadoop is how the data is spread among nodes. It is usually replicated over different nodes as to improve data availability in case some of the replicas are busy or just failed. Without digging into details, the aforementioned schedulers already deal with such issues related to data replication. Current research is dealing with data placement policies to overcome the problem of shutting down nodes to save energy costs as well [98][99][100].

As we will see, enhancements over data placement policies are orthogonal to our work since since our mechanism is in fact an extension of the fair share or capacity schedulers and any improvement on that matter would be applicable to our solution as well. Thus, we will leave out the issues related to the mapping of jobs and data for simplicity and it will allow us to simplify our model. It will remain future work stating the impact of our solution considering data placement as another variable to consider.

8.3 Dynamic allocation based on incentives

In this section we present the design, operation and objectives of the incentive-based allocation mechanism introduced in this chapter. The mechanism consists of two elements: an incentive for users to report their actual resource requirement, and the strategy to allocate available resources.

8.3.1 Design Goal

The goal of the incentive mechanism presented in this chapter is twofold. From the application perspective, users should receive an incentive to report their actual resource requirement in order to allocate only those resources necessary to complete the task within their completion goal, and no more. This way, we may be able to aggregate unused resources that are not necessary to improve the Quality of Service (e.g. SLAs) of users and, therefore, put strategies in place to shut-down such resources to reduce energy costs. In contrast with current mechanisms which use the maximum amount of resources available to complete the task, targeting job runtime as a primary goal, we propose

a more rational way to share resources by making explicit the cost of using such resources and giving users the option to scale up and down their resource allocation.

From the infrastructure operator perspective on the other side, the allocation of resources is usually static because of the complexity of the allocation decision. For example, the Hadoop fair share scheduler always allocate the same amount of resources to each application regardless of application requirements and the Capacity scheduler is able to allocate a specific share of resources to each application but requires the infrastructure operator to manually change such allocations in case of a change on the priority of different applications. Our mechanism simplifies the decision of how many resources allocate to each user – because now users are in charge of such decision – and provides a solution to reduce energy costs by shutting-down unused resources without breaking the QoS contracts signed with users.

Therefore, this may lead us to a win-win situation in which users are able to finish their tasks within a given time goal and infrastructure operators are able to reduce energy costs by shutting-down unused resources and simplify their allocation decisions.

8.3.2 System model

While Map Reduce was originally used for batch data processing, it is now also being used in shared, multi-tenant environments in which submitted jobs may have completely different priorities depending on resource requirements and completion time: from small, almost interactive, executions, to very long programs that take hours to complete. This shift in the initial paradigm makes the associated resource allocation even more relevant. In this multi-tenant scenario, each user has a guaranteed resource capacity according to different strategies implemented by different schedulers. For example, the fair share scheduler allocates 1/n of the available slots to each application and the Capacity scheduler allocates a fixed amount of resources to each application according to high-level policies –e.g. 5% to application A, 25% to application B and 70% to application C. Our mechanism can be considered an hybrid of these two in the sense that an application has a fair share of the resource guaranteed but is free to hand out spare resources to other applications or ask for more resource when necessary – this is the essence of dynamic allocation.

Throughout this document, we will indistinctly refer as user or application a piece of software running in the cluster to perform data intensive computations. These applications are modeled as a Map Reduce job characterized by an upper (r_{max}) and lower (r_{min}) bound on resource requirements in the form of a percentage or share of the cluster capacity. This information is considered in principle unknown to the resource provider and private to the user.

This simple model allow us to specify the minimum share to meet a certain time goal to complete the job. Thus, the utility function of a user will be computed as a function of the share allocated by the scheduler (s_i) and the share requirement (r_{min}) in the following way:

$$U(s_i, r_{min}) = \begin{cases} 1 & \text{if } s_i \geq r_{min} \\ \frac{s_i}{r_{min}} & \text{if } s_i < r_{min} \end{cases} \tag{8.1}$$

In few words, this utility function is in the range $[0, 1]$ and models the fact that users will not obtain more benefit if they obtain more than their minimum requirement, which is the minimum share to meet the deadline of the job. In case the allocated share is less than the share requirement, the utility obtained by the user will decrease linearly.

For problem

8.3.3 Making private information explicit through incentives

The basic principle behind our mechanism is a *use your assigned resources now or better save them for later* approach. Our goal is to provide users an incentive to truthfully declare their actual resource requirements (r_{min}). This way, the infrastructure operator can decide which minimum portion of the cluster is necessary to provide jobs with enough resources to complete their tasks in time.

As we stated in the model above, applications are given a fair share ($1/n^{th}$ where n is the number of concurrent applications) of the time slots available for execution. Sometimes applications will need more than this fair share, and sometimes applications will need less. The incentive we propose is a simple market in which users sell their spare allocations when not needed to the operator to obtain a certain amount of credits, and buy resources with these

credits from the pool of unused resources when they need more than their fair share. In a few words, it's an enhancement over the fair-share scheduler in which users can dynamically decide which is their actual share allocated as a function of their requirements.

The benefits of such mechanism is two-fold: i) applications are able to scale up (buy) and down (sell) their allocations based on their requirements which provides a dynamic environment to adapt to changing conditions according to the workload (*dynamic capacity allocation*); and ii) if at any point in time, the aggregated demand is less than the total capacity, the infrastructure operator can decide to reduce costs by powering down (or switch resources to a low power consumption state) without breaking application's SLAs (*energy efficiency*).

8.4 Algorithms behind the scenes

Throughout this section, we will detail the procedure to allocate resources to applications according to their requested share. The allocation mechanism presented in this work consists of two main components: i) an algorithm to update the capacity allocated to each application which is executed every time a new allocation request is made (Algorithms 8.1 and 8.2) and, ii) a procedure to recompute the credits earned by selling spare resources, or the credits spent (virtual money) by buying extra capacity required to accomplish the QoS needed (Algorithm 8.3).

Dynamic capacity allocation. Every time an application sends a request to obtain a specific set of resources, Algorithm 8.1 is executed. It recomputes the shares assigned to all running jobs taking into account the new request. Basically, we are given a set of share requests submitted by users (ϕ), which are the shares (percentage of the cluster) needed by an application to complete within a specific deadline and a set of budgets or credits for each user.

First, we divide all requests in two groups by looking if the request is lower (set S) or greater (set Q) than its fair share (1/n). In the first case, we directly grant the request (ϕ_i) and add the spare capacity not planned to be used ($1/n - \phi_i$) to the pool of free resources (p). In the second case, we add this extra capacity requested to a counter (r) for later use and include it on the second set (Q), always checking that the application has enough credit to

Algorithm 8.1 UpdateCapacityAllocation – main algorithm to compute capacity allocation

Require: $\phi = \langle \phi_1, \ldots, \phi_n \rangle$ ▷ share request
Require: $C = \langle c_1, \ldots, c_n \rangle$ ▷ credits
  $Q \leftarrow \{\emptyset\}$ ▷ greater than ϕ_i set
  $S \leftarrow \{\emptyset\}$ ▷ lower than ϕ_i share set
  $p \leftarrow 0$ ▷ pool of free resources
  $r \leftarrow 0$ ▷ extra capacity requested
  for all request ϕ_i **do**
    if $\phi_i \leq 1/n$ **then**
      $s_i \leftarrow \phi_i$ ▷ sellers
      $S \leftarrow S \cap \{s_i\}$
      $p \leftarrow p + (\frac{1}{n} - \phi_i)$
    else
      if $c_i > 0$ **then**
        $q_i \leftarrow \phi_i - \frac{1}{n}$ ▷ buyers
        $r \leftarrow r + q_i$
        $Q \leftarrow Q \cap \{q_i\}$
      end if
    end if
  end for
  if $r < p$ **then**
    for all request $q_i \in Q$ **do**
      $s_i \leftarrow \frac{1}{n} + q_i$
      $Q' \leftarrow Q' \cap \{s_i\}$
    end for
  else
    $Q' \leftarrow$ AllocateUnusedCapacity(Q, C, p)
  end if
  return $S \cap Q'$

Algorithm 8.2 AllocateUnusedCapacity – allocate spare capacity among users

Require: $Q = \langle q_1, \ldots, q_m \rangle$ ▷ extra share request
Require: $C = \langle c_1, \ldots, c_m \rangle$ ▷ credits
Require: p ▷ % of free resources
  for all $i \in Q$ **do**
    $X \leftarrow \sum_{i=0}^{m} c_i$
    for all $c_i \in C$ **do**
      $x_i \leftarrow c_i / X$
      $y_i \leftarrow (x_i * p)/q_i$
    end for
    Sort the set Q by y_i in decreasing order ▷ if $y_i > 1 \rightarrow$ request $<$ capacity
    if $y_i > 1$ **then**
      $a_i \leftarrow q_i$
    else
      $a_i \leftarrow (x_i * p)$
    end if
    $s_i \leftarrow 1/n + a_i$
    $p \leftarrow p - a_i$
  end for
  return $S \leftarrow (s_1, \ldots, s_m)$

Algorithm 8.3 updateCredits – update the amount of credits

Require: $S = \langle s_1, \ldots, s_n \rangle$ ▷ actual allocation made by UpdateCapacityAllocation
Require: $C = \langle c_1, \ldots, c_n \rangle$ ▷ current number of credits
Require: t ▷ time since last updateCapacityAllocation
  for all $s_i \in S$ **do**
    $c_i \leftarrow c_i + t * (\frac{1}{n} - s_i)$
  end for
  return $C \leftarrow \{c_1, \ldots, c_n\}$

spend on extra resources. Notice that the set S is the set of applications *selling their spare capacity*, and the set Q is the set of applications *buying extra capacity*.

Thereafter, if the amount of extra capacity required is lower than the pool of free resources or, in economic terms, demand is lower than supply, we allocate this extra capacity to applications buying resources – which will be charged for this extra use and accomplish the *elasticity* criteria presented on section 8.1. On the other hand, when resources are scarce and demand is higher than supply, we allocate these extra requests proportional to the credits earned previously –which accomplish the *fairness* criteria presented also on section 8.1– following Algorithm 8.2 which is a loosely-based proportional share allocation algorithm.

On algorithm 8.2, we take into account the set Q of extra capacity requests and the budget in credits owned by each application. Following our goal of allocating only the necessary (and no more) resources to applications, we first compute for each application the theoretical share of resources proportional to their budget (x_i) and compute the ratio y_i which will be greater than 1 for applications requesting less than their theoretical proportional share and lower than 1 for applications requesting more than their theoretical proportional share. Given that, by sorting the set Q by the ratio y_i in decreasing order, we first allocate the extra requested capacity to applications requesting less than their theoretical proportional share to accumulate their spare capacity to subsequent applications. In case $y_i < 1$, the theoretical proportional share is the upper bound on the amount of resources an application can obtain given the credits. This procedure is repeated until no application requests are left in the set Q.

Accounting of credits. Before the actual allocation is made, algorithm 8.3 is processed to update the credits each application owns based on the previous execution. The current budget of credits for each application is computed as a function of the current allocation provided by algorithm 8.1 at time t-1. Therefore, if the application bought extra resources ($s_i \geq \frac{1}{n}$) on the previous round of the algorithm, the term $t * (\frac{1}{n} - s_i)$ will be negative and the corresponding amount of credits will be discounted. On the other hand, if the application sold spare resources ($s_i < \frac{1}{n}$), the term $t * (\frac{1}{n} - s_i)$ will be positive and the equivalent number of credits will be added to the current budget.

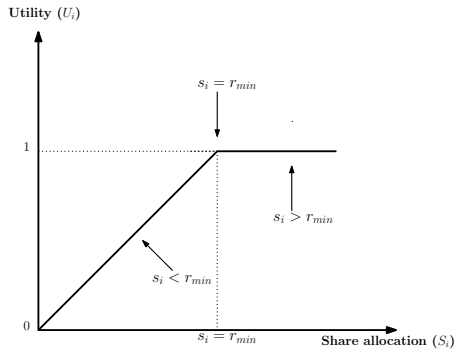


Figure 8.1: User Utility comparing r_{min} and s_i

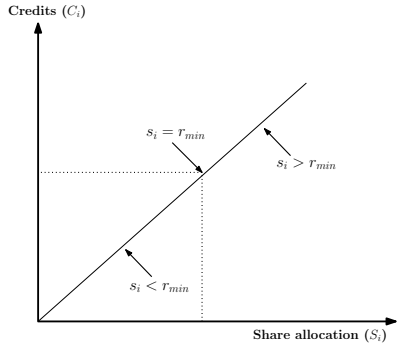


Figure 8.2: Credits earned/spent comparing r_{min} and s_i

Analysis of incentive compatibility. A key factor of our mechanism is its *incentive compatibility*. A mechanism is incentive compatible if all the participants consider their best interest to truthfully reveal any private information inquired by the mechanism.

In our case, the private information that the mechanism asks users is their actual resource requirement in terms of shares to finish the job within a deadline. To perform the analysis, we will consider the case in which a user declare a share request below, above or exactly equal to its actual share requirement and observe how our mechanism reacts to these values and the utility function described in Equation 8.1.

An informal proof of incentive compatibility is as follows. Given a user i, a minimum share requirement for a job r_{min} and the outcome of our mechanism which is an actual share allocation s_i, we can observe these situations considering a user reports r_i:

- **case** $r_i \leq 1/n$. Request share is lower than fair share (potential earnings). In this case, $s_i = r_i$ because all requests less than the granted fair share are accepted without charges. Users have no incentive to report $r_i < r_{min}$ because it will not obtain its minimum share to complete the job ($U_i(s_i, r_i) = \frac{s_i}{r_{min}} < 1$), as shown in Figure 8.1. Reporting a r_i above

its requirement r_{min} will grant the user more shares and maximize utility ($U_i(s_i, r_i) = 1$) but with the drawback of earning less credits to spend in the future without gaining any utility out of it. Therefore, the best response of the user is to report a $r_i = r_{min}$.

- **case** $r_i > 1/n$. Request share is higher than fair share (potential payments).

  - **case** $r_i = r_{min}$. Request share is equal to requirement. This is the ideal situation in which users truthfully report their requirements. In this case, $s_i \leq r_{min} = r_i$. Because the user is requesting more resources than its fair share, it will pay up to c_i credits which is proportional to $s_i - \frac{1}{n}$ or, in the best case, proportional to $r_{min} - \frac{1}{n} = r_i - \frac{1}{n}$. Looking at the utility function $U_i(s_i, r_i) = 1$ because $s_i = r_{min}$ in the best case. In the following cases, we will see that this is the best possible situation and, thus, the users have no incentive to misreport r_i.

  - **case** $r_i > r_{min}$. Request share is higher than requirement. In this case, $r_{min} \leq s_i \leq r_i$. Again, user will pay up to c_i credits which is proportional to $s_i - \frac{1}{n}$ or, in the best case of allocating the whole request, proportional to $r_i - \frac{1}{n}$. The request in this case is greater than the request in case ii-i, so the total amount of credits to pay will be higher ($s_i > r_{min}$ in Figure 8.2). Given that the outcome of our algorithm $U_i(s_i, r_i) = 1$ is the same as the case above because $s_i > r_{min}$, it will end up paying more credits for the same utility.

  - **case** $r_i < r_{min}$. Request share is lower than requirement. In this case, the utility obtained $U_i(s_i, r_i) = \frac{s_i}{r_i} < 1$ because $s_i < r_{min}$ and it will be lower than in the previous two cases because our algorithm allocates a maximum number of shares equal to the request.

Following that, we can conclude that our mechanism always maximizes the utility $U_i(s_i, r_i) = 1$ while minimizing the amount of credits to pay in case of buying resources, and also maximizes the amount of credits to earn in case of selling resources. ■

8.5 Evaluation

We use simulations to evaluate the long-term impact of our system. To determine the effectiveness of our incentive mechanism in reducing the cluster usage without reducing the QoS perceived by users, we compare our algorithm – *cooperative* label in the figures – with the outcome of a widely used scheduler – the fair share scheduler used in Hadoop. We also use for comparison purposes an *optimum* allocation mechanism which always allocates the share requirement r_{min} to jobs effectively maximizing user utility and minimizing resource usage, independently of the market of resources (credits earned and spent). This *optimum* scheduler however has an unbound resource pool so it can always allocate the optimum shares regardless of the load.

The set-up of the simulations consists in varying the number of simultaneous users n (from 2 to 100) to assess the scalability of the solution as more users (and, therefore, more load) are added to the system. In addition, each user is assigned a set of jobs to execute in the cluster. Specifically, each job is represented by a tuple (t, r_{min}, r_{max}) where t is the deadline for completing the job, r_{min} is the minimum share necessary to finish the job before time t and r_{max} is the maximum number of shares the job is able to use taking into account that the level of parallelism is bounded or, in other words, we model the fact that depending on the nature of the job, it cannot use the whole cluster even if a single job is running on it.

Given the lack of public reliable map reduce-like workloads, we simulate a synthetic workloads that try to mimic real world workloads as described by Zaharia et al. [101]. Thus, we assign r_{min} and t drawn from a log-normal distribution $L \sim (1, 1.25)$ which produce a workload distribution in which most of the jobs have a rather short running time and share requirement (small to medium size jobs) and fewer jobs with higher running times and share requirements. r_{max} is derived from r_{min} by adding a variable k drawn from a uniform distribution $U \sim (0, 100)$. We normalize r_{min} and r_{max} in the range $[0, 100]$ as it represents the shares in percentage of a cluster and t is capped in the range $[0, 10000]$ which is an arbitrary number large enough to not misrepresent the random distribution but effective to avoid not real or excessively large jobs.

To study the behavior of our proposed mechanism, we consider the following metrics.

- *Cluster usage.* This is the average usage of the cluster in percentage over the whole simulation period. Although peak usage may differ over time depending on the actual jobs being run on the cluster, this is an indication of the overall usage over time, the time the cluster remains unused and the percentage of the cluster that could be shutdown to reduce power consumption on average.

- *Satisfaction.* Given a specific share request r_i for user i and a specific share allocation s_i, the satisfaction or efficiency is $\phi_i = s_i - r_i$. In other words, it is the difference between the share requirement of user i and the actual allocation made by the algorithm. This is a measure of the QoS the user perceives from the running time on the cluster considering their requests and their actual allocations. Given that this metric is measured as a difference, the lower the better.

- *Mean completion time.* Given a set of jobs for user i and its actual completion time (time elapsed between submitting the job and gathering the results), it is the mean time to complete each of the jobs. This measure is an indication of how well the scheduler is able to scale down the number of shares given to a job to save resources.

All figures are normalized in the range $[0, 1]$ for comparison purposes and because the actual numbers are actually meaningless because of the synthetic nature of our workload. Thus, we are only interested in the relative difference between our proposal and the widely used fair share scheduler.

Figure 8.3 shows the QoS perceived by users. Because satisfaction is measured as the difference between the share request (requirement) and the share allocated, the lower the number the better. We can see that our mechanism (label *cooperative* in the figures) is able to provide better QoS to users compared to the widely used fair share scheduler because it is able to allocate dynamically more resources to those jobs in more need instead of allocating the same fair share to all of them. It is also able to provide a QoS on par with the optimum up to the point where resources become scarce and the gap between share request and share allocation becomes noticeable (20-30 simulated users). However, we

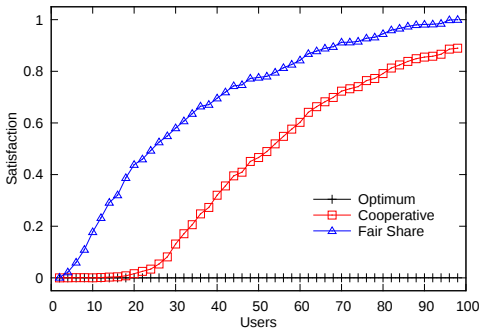


Figure 8.3: User Satisfaction

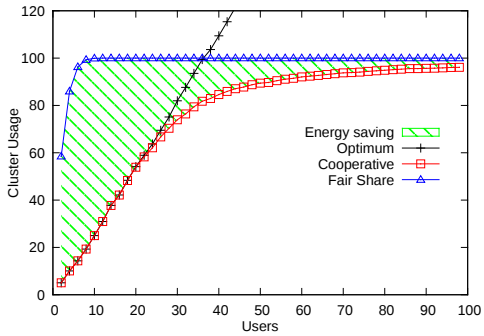


Figure 8.4: Cluster Usage

stress the importance of our incentive mechanism because users are able to buy more resources to improve their QoS even in high contention scenarios (more than 20-30 users).

If we now look at the cluster usage (Figure 8.4), we can observe that our algorithm consumes up to 50% less resources than the fair share scheduler with 20 simultaneous users. This lower resource consumption is possible because users *sell* resources to the infrastructure operator when not strictly necessary to meet the job's deadline. It's important to note that this lower cluster usage may allow the infrastructure operator to scale down the available nodes to reduce power consumption and, thus, increase the revenue obtained given the same workload. In contrast, the fair share scheduler always allocates the maximum amount of shares to a given job regardless of the actual minimum resource requirement. This is because the fair share scheduler is agnostic to the job characteristics – i.e. minimum share requirement and deadline. In contrast, our incentive mechanism is actually able to extract such information from the user and use it to the operator's benefit to maintain the resource usage to a minimum without impacting user efficiency.

However, these benefits (reduced resource consumption, lower energy costs, higher user's satisfaction) comes at a certain cost, namely a higher mean completion time (see Figure 8.5). Because our mechanism influence users to report the minimum share to meet a certain deadline, its easy to see that the mean completion time for the set of jobs a users must run will be higher

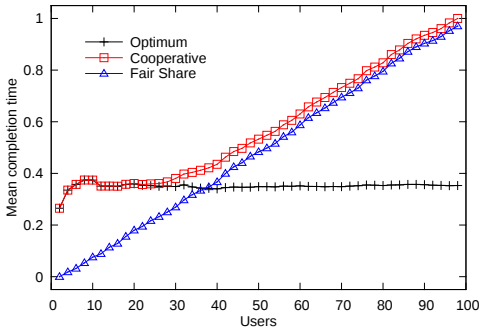


Figure 8.5: Mean Completion Time

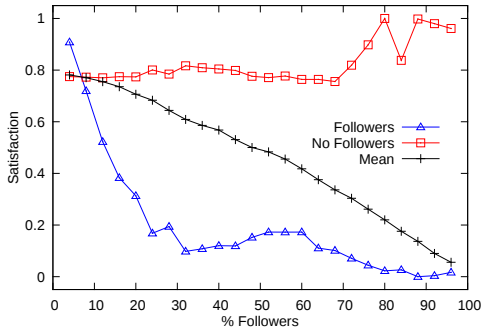


Figure 8.6: Variable % of followers

than in the case of traditional schedulers whose main aim is to finish a job as soon as the cluster capacity allows. However, this minor drawback is bearable considering that almost no deadlines are broken when there are less than 30 users. In fact, below this threshold its performance is comparable to the optimum allocation. As the number of concurrent users increases, our mechanism behaves almost exactly as the other schedulers.

It's important to notice that users are free to follow our incentive mechanism by reporting the minimum share requirement and engaging in buying and selling resources, or reject it and just behave as a regular user which receives its fair share.

To state the effectiveness of our incentive mechanism, we observe the satisfaction of users depending on the opt-in ratio in our mechanism. Figure 8.6 shows the scalability of our mechanism with a variable ratio of opt-in users (label *followers*) and opt-out users (label *no followers*). We can see that the more users opt-in to collaborate through our mechanism the better the satisfaction (recall that a lower number is better by the definition of satisfaction). This is true because there will be more users selling resources which in turn can be used by other users in need of more resources effectively allowing a dynamic allocation of resources.

8.6 Related work

The focus of current schedulers in distributed processing frameworks is on high performance computations targeting job run time as a primary objective. However, the trade off between performance and the effect on data center energy efficiency has not been fully investigated. Our work is a step towards the reconciliation of quality of service allowing dynamic scale up and down of jobs and energy cost efficiency.

There are different strategies to address the problem of energy efficiency, most of them based on data replication strategies to allow shutting down a portion of the cluster safely. Jacob et al. [99] find that running Hadoop in fractional configurations by means of distributing data replicas over a covering set can save energy by shutting down idle fractions of the clusters at a cost of losing performance due to data contention. Kaushik et al. [98] makes use of hardware heterogeneity and different power consumption policies by divide the cluster in two zones: the *hot* zone where highly popular data is placed and where servers run without any power saving policy and the *cold* zone where low spatial or temporal popularity data is placed but with an aggressive policy towards saving energy. Interestingly, their simulations were carried out using a three month worth of traces from Yahoo! and found an impressing 26% savings in energy costs. Finally, Lang et al. [100] proposes to use the entire cluster for a certain period of time to run a specific workload and then powers down the entire cluster until the next running period, certainly consolidating dispersed workload over a shorter period. Although an interesting analysis, the practicality of such mechanism remains to be seen as starting up and shutting down a large-scale cluster is not trivial and would need improvements on the hardware side between transition periods. From the economic standpoint, Sandholm et al. [102, 103] already proposed a market for dynamically assign resources of a shared cluster to multiple Hadoop instances. The priorities are assigned using high level policies like budgets similarly to our solution. However, they had to deal with the complexities of a "real" market like inflation and deflation or forcing the users to understand the mapping between currency and real resources. In our case, we simplify this complexity leveraging the share concept and applying a direct translation between shares, time and credits earned or bought, which help users understand how many resources they can buy or sell for a given share at any point in time.

It is important to note that, as previously stated on this chapter, our mechanism relies on the users' knowledge about their jobs and requirement targets. This knowledge could be acquired over time by learning from previous executions. However, recent works propose a prediction mechanism for Hadoop environment [104] which could be integrated into our mechanism to aid users in their decisions or even implement high level policies to act on behalf of users using such predictions.

Our proposal is orthogonal to such previous works in the sense that our focus is not on data placement strategies nor performance prediction but on scheduling decisions of the time slots allocated to each application and, as such, any improvement over data placement policies is complementary to our solution.

8.7 Conclusions

In this chapter we investigate the possibility of reducing energy costs by providing users an incentive to report their actual needs instead of over reporting the size of their jobs. Our mechanism is based on markets and could be implemented as an extension of the traditional Capacity Scheduler.

Our mechanism pursues two different goals. From the user perspective, we show how our incentive mechanism effectively encourages users to report their true share requirement for a given job. Thus, we are able to provide a *shares market* in which users engage to dynamically scale up (buy) or down (sell) the allocated time slots of a job without the intervention of the infrastructure operator, providing a more elastic and agile infrastructure. We also show that it's in the users' best interest to participate in the market to improve their QoS instead of default.

Furthermore, map-reduce computations are characterized by long, predictable, streaming I/O, massive parallelization, and non interactive performance. These computational services are often used in real-time data processing scenarios [105] and, as such, they can benefit from our mechanism as quality of service become more important than job run time.

From the infrastructure operator point of view, our mechanism is able to extract valuable private information from users (share requirements). In a scenario in which energy related costs is one of the single largest factor in the

overall cost of operating a data center, this information can be used to shut down a portion of the cluster without reducing the quality of service provided to users, which in turn could reduce energy costs.

Looking at the results, we can conclude that our mechanism is a step towards a more rational use of the available resources. It is able to dynamically scale up and down the shares allocated to jobs with the aid of users and at the same time provide valuable information to the resource provider to shutdown spare resources without breaking SLAs or affecting QoS.

To the best of our knowledge, our work is the first to explore the problem emerging from the intersection of ensuring users' quality of service providing dynamic allocation of resources based on user's requests and reducing energy costs together.

Chapter 9

Conclusions

We have investigated how economic regulation can be integrated into large-scale computational infrastructures to not only make economically and computationally efficient allocation decisions, but to include system-wide metrics or externalities into the computational market which are not taken into account by economically rational users otherwise.

The methods proposed were designed with a specific scenario in mind. Our first challenge was to improve load balancing in infrastructures where an economic mechanism is in place but in a decentralized setting where no single authority exists. We proposed the creation of a third-party regulation entity which monitors the health of the overall infrastructure and introduce a tax mechanism to solve the problem of hot spots and the imbalance of resource consumption and contribution.

Our next challenge was to address energy related costs in data centers operated by a single resource provider in charge of the supply of computing elements. We proposed an analytical model to study the problem and derive an algorithm to scale up and down resource supply as a regulation mechanism in such a way that, at any point in time, users are able to meet their resource requirements but using the minimum amount of resources. This regulation mechanism aids the resource provider to reduce energy-related costs by powering down nodes when not strictly necessary.

The last challenge we address in this thesis was the creation of a new mechanism with two principles in mind: it should be incentive compatible so users are

encouraged to truthfully reveal their resource requirements; and the resource provider could use this information to make smarter allocations so users get what they want with the minimum amount of resources. They key difference from the previous contribution is that the mechanism encourages users to self-regulate themselves by giving away resources they don't need in exchange for future rights to use those resources if available.

We believe that computational markets are indeed a good approach to the resource allocation problem. Moving complex decisions from a central scheduler oriented approach towards users in an end-to-end approach may seem moving the problem around but the key insight is that the important scheduling information –be it job priority or resource requirements– resides on the users-end, not on the scheduler. As in real life however, markets sometimes fail to address such problems and additional regulative mechanism must be provided to correct, limit or modify its behaviour.

Future directions

The main remaining challenge for computational markets is to make resource allocation as simple as possible without losing the fine-grained control of service-levels. The simplification of the overall user experience is critical for the success of market-based resource allocation and the artificial intelligence field may play a very important role in that aspect by developing agents to act on behalf of users to aid them in making smarter allocations.

Whether this usability goal is met can only be evaluated in pilot projects with real users in large markets with resource contention. A thorough analysis of the benefits of our proposals and economic allocations in a live system with real users would expose the psychological factors of decision making which are hard to capture accurately in simulations.

Once more experience is gained with real users, the next challenge is to explore more advanced regulative measures if necessary. We have provided insights on how to cope with hot spots and energy efficiency in different scenarios but other types of externalities may be investigated: not only care about user's own allocation but the allocation to others, avoid monopolies on federated scenarios or reward those users whose executions provides a higher utility to the overall community as examples.

The use of regulative measures in computational markets opens the door to a new variety of mechanisms ranging from income taxes on the resource providers to monetary policies and exchange rates in federated environments. However, their usefulness remains to be seen while there are no new problems that require such mechanisms.

Finally, the implementation of resource pools instead of specific resources –e.g. percentage of a cluster instead of specific resources– may help in the simplification process of markets. In this thesis, we have seen that standard support of the notion of pool of resources on parallel programming environments like MapReduce can help to make scheduling decisions. In the advent of cloud computing and network virtualization, the abstraction of pool of resource could be translated into virtualized network resources –switches, links– and include them on the market for scheduling decisions in coordination with a pool of computing resources –CPU, memory, storage.

Bibliography

[1] James Edward Meade and Karin Hjertonsson. *The Theory of Economic Externalities: The Control of Environmental Pollution and Similar Social Costs*, volume 2. Brill Archive, 1973. (page 2)

[2] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007. (page 3)

[3] IE Sutherland. A futures market in computer time. *Communications of the ACM*, 11(6):449–451, 1968. (page 5, 15)

[4] Norman R Nielsen. The allocation of computer resources—is pricing the answer? *Communications of the ACM*, 13(8):467–474, 1970. (page 5, 15)

[5] Ian Foster. What is the grid?-a three point checklist. *GRIDtoday*, 1(6), 2002. (page 6)

[6] Andy Bavier, Mic Bowman, Brent Chun, David Culler, Scott Karlin, Steve Muir, Larry Peterson, Timothy Roscoe, Tammo Spalink, and Mike Wawrzoniak. Operating system support for planetary-scale network services. In *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1*, pages 19–19, Berkeley, CA, USA, 2004. USENIX Association. (page 7, 17, 30, 32)

[7] Global environment for network innovations, 2009. (page 7, 30)

[8] OneLab: Future Internet Test Beds, 2009. (page 7, 30)

[9] AKARI: Architecture Design Project for New Generation Network, 2008. (page 7, 30)

[10] G. Hardin. The Tragedy of the Commons. *Science*, 162(3859):1243–1248, 1968. (page 9)

[11] Martin J Osborne. *An introduction to game theory*, volume 3. Oxford University Press New York, 2004. (page 9)

[12] J.H. Kagel. Auctions: a survey of experimental research. *Internation Library of critical writings in Economics*, 113:601–685, 2000. (page 12, 64)

[13] D. Friedman. The double auction market institution: A survey. *The Double Auction Market: Institutions, Theories, and Evidence*, pages 3–25, 1993. (page 12, 63, 64)

[14] S. De Vries, R.V. Vohra, Center for Mathematical Studies in Economics, and Management Science. Combinatorial auctions: A survey. *INFORMS Journal on Computing*, 15(3):284–309, 2003. (page 12, 63, 64)

[15] Hesam Izakian, Ajith Abraham, and Vaclav Snasel. Comparison of heuristics for scheduling independent tasks on heterogeneous distributed environments. In *Computational Sciences and Optimization, 2009. CSO 2009. International Joint Conference on*, volume 1, pages 8–12. IEEE, 2009. (page 12, 64)

[16] David Abramson Rajkumar Buyya and Jonathan Giddy. An economy driven resource management architecture for global computational power grids. In *PDPTA'00: Proceedings of the 7th International Conference on Parallel and Distributed Processing Techniques and Applications*, Las Vegas, USA, June 2000. (page 15)

[17] R. Buyya and S. Venugopal. The gridbus toolkit for service oriented grid and utility computing: an overview and status report. In *1st IEEE International Workshop on Grid Economics and Business Models (GECON'04)*, pages 19–66, April 2004. (page 15)

[18] R. Buyya, D. Abramson, and S. Venugopal. The grid economy. *Proceedings of the IEEE*, 93(3):698–714, March 2005. (page 15)

[19] David Abramson, Rajkumar Buyya, and Jonathan Giddy. A computational economy for grid computing and its implementation in the nimrod-g resource broker. *Future Generation Computer Systems*, 18(8):1061 – 1074, 2002. (page 16, 63)

[20] Jahanzeb Sherwani, Nosheen Ali, Nausheen Lotia, Zahra Hayat, and Rajkumar Buyya. Libra: a computational economy-based job scheduling system for clusters. *Softw. Pract. Exper.*, 34(6):573–590, 2004. (page 16)

[21] Robert L. Henderson. Job scheduling under the portable batch system. In *IPPS '95: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 279–294, London, UK, 1995. Springer-Verlag. (page 16)

[22] Chee Shin Yeo and Rajkumar Buyya. Pricing for utility-driven resource management and allocation in clusters. *Int. J. High Perform. Comput. Appl.*, 21(4):405–418, 2007. (page 16)

[23] Chee Shin Yeo and R. Buyya. Service level agreement based allocation of cluster resources: Handling penalty to enhance utility. *IEEE International Cluster Computing, 2005*, pages 1–10, Sept. 2005. (page 16)

[24] Laura Grit, David Irwin, Aydan Yumerefendi, and Jeff Chase. Virtual machine hosting for networked clusters: Building the foundations for "autonomic" orchestration. In *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing (VTDC '06)*, page 7, Washington, DC, USA, 2006. IEEE Computer Society. (page 17, 62)

[25] Jeffrey S. Chase, David E. Irwin, Laura E. Grit, Justin D. Moore, and Sara E. Sprenkle. Dynamic virtual clusters in a grid site manager. In *HPDC '03: Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing*, page 90, Washington, DC, USA, 2003. IEEE Computer Society. (page 17)

[26] Yun Fu, Jeffrey Chase, Brent Chun, Stephen Schwab, and Amin Vahdat. Sharp: an architecture for secure resource peering. *SIGOPS Oper. Syst. Rev.*, 37(5):133–148, 2003. (page 17, 38, 62)

[27] A. AuYoung, B.N. Chun, A.C. Snoeren, and A. Vahdat. Resource allocation in federated distributed computing infrastructures. In *Proceedings of the 1st Workshop on Operating System and Architectural Support for the On-demand IT Infrastructure*, 2004. (page 18, 63)

[28] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of the 2nd ACM conference on Electronic commerce (EC '00)*, pages 1–12, New York, NY, USA, 2000. ACM. (page 18, 63)

[29] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Design and implementation tradeoffs for wide-area resource discovery. In *Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing (HPDC '05)*, pages 113–124, Los Alamitos, CA, USA, July 2005. IEEE Computer Society. (page 18, 54, 77)

[30] B.N. Chun, J. Ng, and D.C. Parkes. Computational resource exchanges for distributed resource allocation. Technical report, Intel Research Laboratory, Berkeley, 2004. (page 18, 38)

[31] D.C. Parkes, J. Kalagnanam, and M. Eso. Achieving budget-balance with vickrey-based payment schemes in combinatorial exchanges. Technical report, IBM, 2001. (page 18)

[32] Kevin Lai, Lars Rasmusson, Eytan Adar, Li Zhang, and Bernardo A. Huberman. Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiagent Grid Syst.*, 1(3):169–182, 2005. (page 18, 63, 77)

[33] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. Kephart, and W. Scott Stornetta. Spawn: A distributed computational economy. *Software Engineering, IEEE Transactions on*, 18(2):103–117, 1992. (page 19)

[34] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177. ACM Press New York, NY, USA, 2003. (page 19)

[35] M. Feldman, K. Lai, and L. Zhang. A price-anticipating resource allocation mechanism for distributed shared clusters. In *Proceedings of the 6th ACM conference on Electronic commerce*, pages 127–136. ACM New York, NY, USA, 2005. (page 19, 45, 65, 71)

[36] Michal Feldman, Kevin Lai, and Li Zhang. The Proportional-Share Allocation Market for Computational Resources. *IEEE Transactions on Parallel and Distributed Systems*, 20(8):1075–1088, 2009. (page 19, 45, 87, 88, 95)

[37] F.I. Popovici and J. Wilkes. Profitable services in an uncertain world. *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, pages 36–36, Nov. 2005. (page 19)

[38] R. Wolski, J.S. Plank, J. Brevik, and T. Bryan. G-commerce: Market formulations controlling resource allocation on the computational grid. *Proceedings of the 15th International Parallel & Distributed Processing Symposium table of contents*, 2001. (page 20)

[39] David Hausheer and Burkhard Stiller. Decentralized auction-based pricing with peermart. In *9th IFIP/IEEE International Symposium on Integrated Network Management*, 2005. (page 20)

[40] B. Chun and A. Vahdat. Workload and failure characterization on a large-scale federated testbed. *Intel Research Berkeley, Tech. Rep. IRB-TR-03-040*, 2003. (page 23, 54)

[41] D. Oppenheimer, B. Chun, D. Patterson, A.C. Snoeren, and A. Vahdat. Service placement in shared wide-area platforms. In *Proceedings of the twentieth ACM symposium on Operating systems principles*, page 1. ACM, 2005. (page 23, 54)

[42] Neil Spring, Larry Peterson, Andy Bavier, and Vivek Pai. Using planetlab for network research: myths, realities, and best practices. *SIGOPS Oper. Syst. Rev.*, 40(1):17–24, 2006. (page 23, 54)

[43] Mark EJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005. (page 24)

[44] K.S. Park and V.S. Pai. CoMon: a mostly-scalable monitoring system for planetlab. *ACM SIGOPS Operating Systems Review*, 40(1):65–74, 2006. (page 33, 59, 77)

[45] Xavier León, Tuan Anh Trinh, and Leandro Navarro. Using economic regulation to prevent resource congestion in large-scale shared infrastructures. *Future Generation Computer Systems*, 26(4):599 – 607, 2010. (page 37, 38, 55)

[46] M.J. Freedman, C. Aperjis, and R. Johari. Prices are right: Managing resources and incentives in peer-assisted content distribution. In *IPTPS'08: Proceedings of the 7th Internation Workshop in Peer-to-Peer Systems*, Tampa Bay, Florida, USA, January 2008. (page 37)

[47] B. Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer Systems*, volume 6. Berkeley, CA, USA, 2003. (page 38)

[48] J. Albrecht, D. Oppenheimer, A. Vahdat, and D.A. Patterson. Design and implementation trade-offs for wide-area resource discovery. *ACM Transactions on Internet Technology (TOIT)*, 8(4):1–44, 2008. (page 40)

[49] The Raven Provisioning Service, 2009. Raven Consortium funded by the GENI initiative. (page 40)

[50] W.N. Venables and B.D. Ripley. *Modern applied statistics with S*. Springer verlag, 2002. (page 42, 83)

[51] D. Kwiatkowski, P.C.B. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of econometrics*, 54(1-3):159–178, 1992. (page 46)

[52] J.V. Bradley. *Distribution-free statistical tests*. Prentice-Hall Englewood Cliffs, NJ, 1968. (page 47)

[53] S. Bani-Ahmad, A. Cakmak, G. Ozsoyoglu, and A. Al-Hamdani. Evaluating Publication Similarity Measures. *IEEE Data Eng. Bull*, 28(4):21–28, 2005. (page 50)

[54] D.A. Menasce and V.A.F. Almeida. *Capacity planning for Web services: metrics, models, and methods*. Prentice Hall PTR, 2002. (page 51)

[55] Limin Wang, Kyoung Soo Park, Ruoming Pang, Vivek Pai, and Larry Peterson. Reliability and security in the codeen content distribution network. In *ATEC '04: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 14–14, Berkeley, CA, USA, 2004. USENIX Association. (page 52)

[56] H.V. Madhyastha, A. Venkataramani, A. Krishnamurthy, and T. Anderson. Oasis: An overlay-aware network stack. *ACM SIGOPS Operating Systems Review*, 40(1):48, 2006. (page 52)

[57] A. Verma, G. Dasgupta, T. Nayak, P. De, and R. Kothari. Server workload analysis for power minimization using consolidation. In *Usenix ATC*, 2009. (page 55)

[58] I. Foster et al. The grid2003 production grid: Principles and practice. *High-Performance Distributed Computing, International Symposium on*, 0:236–245, 2004. (page 55)

[59] Tobias Scholl, Bernhard Bauer, Benjamin Gufler, Richard Kuntschke, Angelika Reiser, and Alfons Kemper. Scalable community-driven data sharing in e-science grids. *Future Generation Computer Systems*, 25(3):290 – 300, 2009. (page 58)

[60] Marc Sànchez-Artigas and Pedro García-López. escigrid: A p2p-based e-science grid for scalable and efficient data sharing. *Future Generation Computer Systems*, In Press, Corrected Proof:–, 2009. (page 58)

[61] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003. (page 58)

[62] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. In *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, December 2002. USENIX Association. (page 58)

[63] G. Hardin. The tragedy of the commons. *Science*, 162(3859):1243–1248, 1968. (page 59)

[64] J. Peek, T. O'Reilly, M.K. Loukides, and L. Mui. *UNIX power tools*. O'Reilly, 1997. (page 60)

[65] M.L. Pinedo. *Scheduling: theory, algorithms and systems*. Springer, 2008. (page 62)

[66] Saurabh Kumar Garg, Rajkumar Buyya, and Howard Jay Siegel. Time and cost trade-off management for scheduling parallel applications on utility grids. *Future Generation Computer Systems*, In Press, Corrected Proof, 2009. (page 63)

[67] C.A. Waldspurger. *Lottery and Stride Scheduling: Flexible Proportional-Share Resource Management*. PhD thesis, Massachusetts Institute of Technology, 1995. (page 65, 86)

[68] Young Jin Nam and Chanik Park. Design and evaluation of an efficient proportional-share disk scheduling algorithm. *Future Generation Computer Systems*, 22(5):601 – 610, 2006. (page 65, 86)

[69] Chunlin Li and Layuan Li. Competitive proportional resource allocation policy for computational grid. *Future Generation Computer Systems*, 20(6):1041 – 1054, 2004. Computational science of lattice Boltzmann modelling. (page 65)

[70] Thomas Sandholm. *Statistical Methods for Computational Markets*. Doctoral Thesis ISRN SU-KTH/DSV/R–08/6–SE. Royal Institute of Technology, Stockholm, 2008. (page 67)

[71] C. Caux. Neural networks applied on identification of ship motions. In *Proceedings of the International Conference on Marine Simulation and Ship Manoeuvrability (MARSIM'96), Copenhagen, Denmark, 9-13 September*, page 577. Taylor & Francis, 1996. (page 69)

[72] P. Vytelingum, D. Cliff, and NR Jennings. Strategic bidding in continuous double auctions. *Artificial Intelligence*, 172(14):1700–1729, 2008. (page 69)

[73] A Barmouta and R Buyya. Gridbank: a grid accounting services architecture (gasa) for distributed systems sharing and integration. In *Proceedings of the 17th Parallel and Distributed Processing Symposium (IPDPS'03)*, page 8, April 2003. (page 77)

[74] Xavier León and Leandro Navarro. Currency management system: a distributed banking service for the grid. Technical Report UPC-DAC-RR-XCSD-2007-6, Universitat Politècnica de Catalunya, Spain, July 2007. (page 77)

[75] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM New York, NY, USA, 2001. (page 77)

[76] R. Brunner, F. Freitag, and L. Navarro. Towards the development of a decentralized market information system: Requirements and architecture. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS 08)*, pages 1–7, April 2008. (page 77)

[77] Urs Hoelzle and Luiz Andre Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 2009. (page 82)

[78] Spec power benchmarks, 2010. (page 83)

[79] K. Jeffay, F. D. Smith, A. Moorthy, and J. Anderson. Proportional Share Scheduling of Operating System Services for Real-Time Applications. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS '98)*, page 480, Washington, DC, USA, 1998. IEEE Computer Society. (page 86)

[80] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 164–177, New York, NY, USA, 2003. ACM. (page 86)

[81] D. Fudenberg and J. Tirole. Game Theory MIT Press. *Cambridge, MA*, 1991. (page 86)

[82] Michael Nelson, Beng-Hong Lim, and Greg Hutchins. Fast transparent migration for virtual machines. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 25–25, Berkeley, CA, USA, 2005. USENIX Association. (page 88)

[83] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association. (page 88)

[84] Michael R. Hines and Kartik Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 51–60, New York, NY, USA, 2009. ACM. (page 88)

[85] Shingo Takeda and Toshinori Takemura. A rank-based vm consolidation method for power saving in datacenters. *IPSJ Online Transactions*, 3:88–96, 2010. (page 102)

[86] Akshat Verma, Puneet Ahuja, and Anindya Neogi. pmapper: power and migration cost aware application placement in virtualized systems. In *Middleware '08: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 243–264, New York, NY, USA, 2008. Springer-Verlag New York, Inc. (page 102)

[87] Kiril Schröder, Daniel Schlitt, Marko Hoyer, and Wolfgang Nebel. Power and cost aware distributed load management. In *e-Energy '10: Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pages 123–126, New York, NY, USA, 2010. ACM. (page 102)

[88] Gergely Biczók, László Toka, András Gulyás, Tuan A. Trinh, and Attila Vidács. Incentivizing the global wireless village. *Computer Networks*, 55(2):439 – 456, 2011. ¡ce:title¿Wireless for the Future Internet¡/ce:title¿. (page 102)

[89] Beibei Wang, Zhu Han, and K.J.R. Liu. Distributed relay selection and power control for multiuser cooperative communication networks using buyer/seller game. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 544 –552, may 2007. (page 103)

[90] Matthew Andrews, Antonio Fernández Anta, Lisa Zhang, and Wenbo Zhao. Routing for energy minimization in the speed scaling model. In *INFOCOM'10: Proceedings of the 29th conference on Information communications*, pages 2435–2443, Piscataway, NJ, USA, 2010. IEEE Press. (page 103)

[91] Matthew Andrews, Antonio Fernández Anta, Lisa Zhang, and Wenbo Zhao. Routing and scheduling for energy and delay minimization in the powerdown model. In *INFOCOM'10: Proceedings of the 29th conference on Information communications*, pages 21–25, Piscataway, NJ, USA, 2010. IEEE Press. (page 103)

[92] Sergiu Nedevschi, Lucian Popa, Gianluca Iannaccone, Sylvia Ratnasamy, and David Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 323–336, Berkeley, CA, USA, 2008. USENIX Association. (page 103)

[93] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008. (page 106, 108)

[94] Luiz André Barroso and Urs Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis Lectures on Computer Architecture*, 4(1):1–108, 2009. (page 106)

[95] M Mustafa Rafique, Benjamin Rose, Ali R Butt, and Dimitrios S Nikolopoulos. Cellmr: A framework for supporting mapreduce on asymmetric cell-based clusters. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–12. IEEE, 2009. (page 108)

[96] Bingsheng He, Wenbin Fang, Qiong Luo, Naga K Govindaraju, and Tuyong Wang. Mars: a mapreduce framework on graphics processors. In *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, pages 260–269. ACM, 2008. (page 108)

[97] Tom White. *Hadoop: The definitive guide*. O'Reilly Media, 2012. (page 108)

[98] Rini T Kaushik and Milind Bhandarkar. Greenhdfs: Towards an energy-conserving storage-efficient, hybrid hadoop compute cluster. In *Proceedings of the USENIX Annual Technical Conference*, 2010. (page 109, 122)

[99] Jacob Leverich and Christos Kozyrakis. On the energy (in) efficiency of hadoop clusters. *ACM SIGOPS Operating Systems Review*, 44(1):61–65, 2010. (page 109, 122)

[100] Willis Lang and Jignesh M Patel. Energy management for mapreduce clusters. *Proceedings of the VLDB Endowment*, 3(1-2):129–139, 2010. (page 109, 122)

[101] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica. Job scheduling for multi-user mapreduce clusters. Technical Report UCB/EECS-2009-55, EECS Department, University of California, Berkeley, Apr 2009. (page 118)

[102] Thomas Sandholm and Kevin Lai. Mapreduce optimization using regulated dynamic prioritization. In *11th international conference on Measurement and modeling of computer systems*, SIGMETRICS '09, pages 299–310, New York, NY, USA, 2009. ACM. (page 122)

[103] T. Sandholm and K. Lai. Dynamic proportional share scheduling in Hadoop. In *Job Scheduling Strategies for Parallel Processing*, pages 110–131. Springer, 2010. (page 122)

[104] Jorda Polo, David Carrera, Yolanda Becerra, Jordi Torres, Eduard Ayguadé, Malgorzata Steinder, and Ian Whalley. Performance-driven task co-scheduling for mapreduce environments. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 373–380. IEEE, 2010. (page 123)

[105] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C Hsieh, Deborah A Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4, 2008. (page 123)