CHAPTER 8

Conclusions and Future Work

This Chapter presents the main conclusions of this Thesis and the future work. In this Thesis, we have demonstrated that, to dynamically measure and take into account the performance of parallel applications to decide the processor allocation improves the system performance. Another important point is the coordination between levels. It is shown that coordinating the different scheduling levels, the system performance and the system utilization are also improved.

As a future work we plan to export the ideas presented in this Thesis to other execution environments, with different architectures and applications. We also plan to evaluate the utilization of these techniques to reduce the power consumption without loss of performance. And finally, we also plan to dynamically measure the utilization of resources that can limit the application scalability, such as the memory bandwidth, and use this information to modify the processor scheduling.

8.1 Goals and contributions of this Thesis

This Thesis had the main goal of improving the execution of workloads of parallel applications in shared-memory multiprocessor architectures. In particular, our research has been focused in the improvement of the processor allocation decisions based on these points:

- The measurement of the application performance at run-time.
- The coordination among the different scheduling levels.
- The imposition of a target efficiency to ensure the efficient use of processors.
- The use of real performance information to decide or to adjust the processor allocation.

To demonstrate the validity of our ideas, we have adopted a practical approach consisting of implementing a complete research execution environment where we have included and evaluated our contributions compared with some previous proposals to evaluate the effectiveness of our work.

We have proposed several scheduling mechanisms and policies that have been implemented and evaluated in this Thesis. The particular contributions and conclusions extracted from their evaluation are presented in the next Section.

8.2 Conclusions of this Thesis

8.2.1 Dynamic performance measurement

The first objective of this Thesis was to demonstrate that the performance of parallel applications could be measured at run-time.

We started from the idea that the system can not rely on users requirements. Based on this premise, we propose that the system considers user requirements as hints, but that it dynamically evaluates the real performance that applications are reaching.

We have proposed a run-time library, SelfAnalyzer, that measures the speedup of OpenMP applications at run-time. SelfAnalyzer exploits the iterative behavior of most of the scientific applications to predict the application behavior based on the speedup achieved by few iterations. We assume that the behavior of several iterations can be extrapolated to the behavior of the complete application. The speedup is measured as the ratio between the average execution time of several iterations with a baseline number of processors and the average execution time with the available number of processors.

Results show that the real application speedup actually has the same behavior than the iteration speedup. We have demonstrated that we can measure the speedup of OpenMP applications and that the overhead introduced is acceptable. We have shown that using a baseline of four processors we can measure the application speedup with a small overhead. Of course, both the overhead introduced and the speedup measurement

precision depends on the application characteristics. However, in all the evaluated policies the shape of the speedup curve was correctly detected and the overhead was acceptable.

8.2.2 Coordination between levels

In the initial planning of this Thesis, we planned to demonstrate the convenience of coordinating the different scheduling levels. The aim was to avoid such situations where decisions taken by a scheduling level negativelly affect the system performance because they are not compatible with decisions taken at the other levels.

In this Thesis we propose that the coordination must be extended to all the scheduling levels to achieve a good overall performance. Coordination means exchanging information between different scheduling levels in such a way that this information will be used to take future scheduling decisions. We have shown with the different policies and methodologies proposed that it is relatively easy to coordinate the different levels.

We propose to include a multiprogramming level policy in the processor scheduler to decide the number of applications that can be concurrently running in the system. With this simple modification, the number of applications can be adapted to the workload characteristics.

The coordination between levels has also shown us that some techniques that in the literature were accepted as quite effective, are not necessary if the scheduling levels are coordinated. This is the case of the two_minute_warning technique.

Moreover, once finished this work, we believe that the coordination should be extended to all the parts of the system. Our experience has shown us that one of the problems related to the coordination is the amount of information that the different levels must manage.

8.2.3 Imposing a target efficiency to ensure the efficient use of resources

In this Thesis, we wanted to demonstrate the benefits in both the application and the system performance of considering the real performance of parallel applications could introduce.

With this aim we have designed and implemented a processor allocation policy that establishes its processor allocation decisions on the speedup achieved by applications, Performance-Driven Processor Allocation policy.

Results show us that PDPA is able to dynamically detect the optimal processor allocation of running applications based on a main criterion: to allocate the maximum number of processors that reaches a given target efficiency. PDPA maintains the processor allocation of those applications that reach an acceptable performance with the number of processors requested, and adjusts the allocation of those applications that do not reach it.

In the worst cases, PDPA introduces a maximum slowdown around 10% in the execution time of some workload respect to best execution time achieved by the rest of policies, with an Equipartition. On the other hand, PDPA speedups the rest of evaluated policies in up to a 400% in extreme cases.

The main conclusion that we extract from PDPA is that the performance of parallel applications can be and must be considered to decide the processor allocation. Results also have shown that PDPA is a robust policy that is neither affected by incorrect user requests nor by incorrect system decisions (multiprogramming level in this case).

The point that has been shown as main in the consideration of performance information for processor scheduling is the imposition of a target efficiency. The use if this target efficiency ensures the efficient use of resources.

8.2.4 Using performance information in multiprocessor scheduling

We have shown that the consideration of the speedup is not exclusive from other criteria. Based on that, we have proposed a methodology that modifies processor scheduling policies to include feedback based on performance information. This methodology, Performance-Driven Multiprogramming Level, proposes to use the performance information to adjust the processor allocation decisions taken by the original policy to which PDML is applied. PDML also proposes to define what the system considers a stable allocation, and in this situation modify the multiprogramming level. PDML has been applied to the Equipartition and the Equal_efficiency resulting in the Equip++ and the Equal_eff++. Results show that with simple modifications, the feedback based on performance information, and the multiprogramming level policy, can be included in the processor scheduling policy. In the evaluation we have shown that Equip++ and Equal_eff++ improve the original policies in those cases where they fail in their processor allocation decisions or where the workload execution was affected by the fixed multiprogramming level used. Results also shown that the overhead introduced in those cases where the original policies perform well is not significant (around the 5%).

We have also observed that it is important to take into account the processor scheduling characteristics to decide parameters such as the default multiprogramming level. For instance, if the policy considers the default multiprogramming level as a minimum, it is better for the system performance to set it to a small value and let the policy to increase it when necessary.

We have also evaluated the ideas of this Thesis in a gang scheduling execution environment. We have tried to improve this kind of policies in two ways: by applying PDML to a baseline gang scheduling policy, developping the Performance-Driven Gang Scheduling, and by a new re-packing algorithm that reduces the number of slots based on the performance achieved by running applications. Results show that both approaches can be included in gang scheduling policies and that improve the execution time of

workloads. We have observed that in this kind of policies a critical point can be the time-sharing quantum used because it can result in a degradation on both the applications and system performance.

Another important point that we have observed in the evaluation of gang scheduling contributions is that, under this kind of policies, the pressure that applications introduce in the system resources is greater than under space-sharing policies because of the resource sharing. In this case, the number of applications concurrently loaded (not necessarily running) in the system is very high. We have observed that even receiving the same number of processors, applications do not perform equal if they are running concurrently with a small number of applications (2..6) than when they are running with a high number of applications (20..25).

Related with the use of real performance information, we want to remark that the use of extrapolated values must be done "*with care*" because some times this formulation can result in incorrect values. We believe that these values must be always considered as temporal and verified with real values. Another important point is the stability of the processor scheduling policy. We have detected that one of the main problems of the Equal_efficiency is that small variations in the performance of one application can result in a global processor distribution. If this situation is very frequent, to completely re-distribute processors is a bad approach for performance.

8.2.5 General remarks

The development of this Thesis has given us a valuable experience in the execution of parallel applications in multiprocessor multiprogrammed execution environments. This experience goes from the parallelization of applications, to the operating system design. We want to include some remarks based on this experience.

The first point that we want to remark is the convenience of using a programming model that provides malleability to applications. Malleability is a characteristic that gives flexibility to the system decisions, and to the applications. To the system because its decisions can be dynamically modified and not determined by the user requirements. And to the applications because the number of chances to be executed and to receive processors increases if the allocation of applications is not limited to a fixed number of processors. Malleability is basic to improve the system performance and we have shown that it can be exploited with any kind of policy: space-sharing and gang scheduling. All the applications used in this Thesis use the OpenMP programming model that gives applications this characteristic.

The second and third points are claims for Operating System vendors and compiler developers. We believe that the O.S. should provide mechanisms to have an easy and efficient access to the hardware counters that modern architectures provide. By the moment, information provided by the hardware is very limited, in one hand, and on the other hand, the cost to access to this information is very high (it is through operating system calls). If this information was easily accessible, researchers will have a way to

develop and evaluate their ideas based on this information. The third point is a claim for compilers developers. It will be very interesting that the compiler provides the run-time with information about the application structure such as the iterative structure and the total number of iterations the application executes. We have found that there are several information related with the sequential structure of the application that is very interesting for the run-time parallel libraries.

Finally, we also want to remark the convenience to define what is considered a workload of scientific application, with a well defined workload trace file. It is also important to provide the applications binary code. The problem is that the available workload trace files represent several months of the load of a real system, and are only valid to use in simulations. If we want to be able to reproduce the same load, we need reduced workload representative from real ones and also to have the application binary codes.

8.3 Future work

Once finished this Thesis, there are several issues that remain opened related with the use of performance information in multiprocessor scheduling, and new points that have appeared as a result of the experience achieved doing this Thesis.

New architectures and programming models

This Thesis has been based on shared-memory multiprocessor architectures and OpenMP applications. However, we believe that the ideas defended in this Thesis can be extrapolated to other environments and to other programming models.

We have planned to analyze the main differences between shared-memory architectures and clusters of SMP's when executing OpenMP applications. As we commented in Chapter 2, clusters of SMP's have the characteristic that they have hardware shared-memory inside each SMP node, and distributed memory between nodes. However, there are some proposals to implement Software Distributed Shared-Memory in clusters of SMP's.

We want to analyze the impact of modifying the allocation of OpenMP applications in this kind of architecture. Our policies and mechanisms already contemplate a step as parameter. This parameter must be extensively tuned in these architectures based on the size of the SMP node. We also have to evaluate the cost to expand the parallelism from one node to more than one node. Probably, the benefit provided by the fact of increasing the parallelism in this way will be lower than in a CC-NUMA architecture, and we have to find solutions to that.

Another point is to complete the SelfAnalyzer in an environment such as a cluster where the cost of modifying the allocation of an application could be significant. An indirect approach could consist of providing the system with a different measure rather than the speedup, and consider the currently number of available processors as the

baseline. A possible approach could be to measure the idleness of the application as a first hint for the system, and only measure the relationship between two execution time measures if the system decides to change the processor allocation of the application.

Computational power balancing based on run-time measurements

We have also planned to use some run-time measured information rather than the speedup to improve the load balancing in non-OpenMP applications. In particular, we want to evaluate the effectiveness of this approach in MPI+OpenMP applications, and a possible useful information coud be the amount of load unbalance between MPI processes.

The number of MPI processes is fixed because this programming model does not consider to dynamically modify the application parallelism. However, if MPI is combined with OpenMP, we can exploit the malleability of OpenMP applications to balance the computational power of each MPI process.

Resource usage distribution

One of the points that we also plan to evaluate is the dynamic measurement of other types of resources such as the memory bandwidth. If two applications that consume a lot of memory bandwidth are concurrently executed, both applications will result negativelly affected. If the memory bandwidth consumed is constant, maybe the best choice will be execute them in serial. However, if these applications have bursts of high memory bandwidth combined with bursts of low memory bandwidth, a good approach will be to phase out their execution and distribute them in the time the moments where each application accesses to memory.

We plan to dynamically measure the memory bandwidth consumed, and to propose an automatic mechanism to detect the behavior of the application respect to the usage of this resource. The goal is to modify the processor scheduling policy to take into account this information and to solve this problem avoiding the system saturation.

Reduction in the power consumption by limiting the use of processors

Proposals made in this Thesis have been oriented to improve the system utilization. The idea was to ensure the efficient use of resources. Using PDPA or applying PDML we can ensure that applications are efficiently using their processors. If there are queued applications, free processors are filled with new applications.

We have planned to evaluate the impact of our proposals in the power consumed by the architecture. The idea is to stop those processors that are not used or even to stop not efficiently used processors. We want to evaluate the impact of modifying the target efficiency in the power consumed by the architecture and the performance achieved by the system and the applications.

Bibliography

[1] G. M. Amdahl, “Validity of the single processor approach to achieving large-scale computing capabilities”, in Proc. AFIPS, vol. 30, pp. 483-485, 1967.

[2] T. E. Anderson, B. N. Bershad, E. D. LAzowska, and H. M. Levy, "Scheduler activations: effective kernel support for the user-level management of parallelism", ACM Trans. Comput. Syst. 10(1), pp. 53-79, Feb 1992.

[3] W.C. Athas and C.L. Seitz, "Multicomputers: messae-passing concurrent computers". Computer 21(8), pp. 9-24, Aug 1988.

[4] J.E. Bahr, S.B. Levenstein, L.A. McMahon, T.J. Mullins, and A.H. Wottreng, "Architecture, design, and performance of Application System/400 (AS/400) multiprocessors". IBM J. Res. Dev. 36(6), pp.1001-1014, Nov 1992.

[5] V. Bala and S. Kipnis. Process groups: a mechanism for the coordination of and communication among processes in the Venus collective communication library. Technical report, IBM T. J. Watson Research Center, October 1992.

[6] V. Bala, S. Kipnis, L. Rudolph, and Marc Snir. "Designing efficient, scalable, and portable collective communication libraries". Technical report, IBM T. J. Watson Research Center, October 1992.

[7] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, and V. Sunderam. Visualization and debugging in a heterogeneous environment. IEEE Computer, 26(6):88--95, June 1993.

[8] F. Bellosa, "Locality-information-based scheduling in shared-memory multiprocessors". In Job Scheduling Strategies for Parallel Processing, D.G. feitelson and L.Rudolph (eds.), pp. 271-289, Springer-Verlag, 1996. Lecture Notes in Computer Science Vol 1162.

[9] M. Berry, D. Chen, P.Koss, D.Kuck, S.Lo, Y.Pang,L.Pointer, R. Roloff, A. Sameh, E. Clementi, S.Chin, D. Schneider, G. Fox. P. Messina, D. Walker, C. Hsiung, J. Scharzmeier, K. Lue, S. Orszag, F. Seidl, O. Johnson, R. Goodrum and J, Martin. “The PERFECT Club Benchmarks: Effective Performance Evaluation of Supercomputers”. *The International Journal of Supercomputer Applications*, 3(3):5-40,1989.

[10] G.E. Bier and M.K. Vernon, "Measurement and prediction of contention in multiprocessor operating systems with scientific application workloads". In Intl. Conf. Supercomputing, pp. 9-15, Jul 1988.

[11] D.L. Black, "Processors, priority, and policy: Mach scheduling for new environments". In Proc. Winter USENIX Technical Conf., pp. 1-12, Jan 1991.

[12] D.L. Black, "Scheduling support for concurrency and parallelism in the Mach operating system". Computer 23(5), pp. 35-43, May 1990.

[13] T. B. Brecht, K. Guha. ”Using Parallel Program characteristics in dynamic processor allocation”, Performance Evaluation, 27&28, pp. 519-539, 1996.

[14] J. Corbalan, X. Martorell, J. Labarta. "Performance-Driven Processor Allocation". In Symposium on Operating Systems Design and Implementation (OSDI 2000), pp. 59-71, October 2000.

[15] J. Corbalán and J. Labarta. "Improving Processor Allocation through Run-Time Measured Efficiency". In 15th International Parallel and Distributed Processing Symposium (IPDPS'2001), pp. 74-80, April 2001.

[16] J. Corbalán, X. Martorell and J. Labarta. "Improving Gang Scheduling through Job Performance Analysis and Malleability". In 15th ACM International Conference on Supercomputing, pp. 303-311, June 2001.

[17] J. Corbalán, J. Labarta, "Dynamic Speedup Calculation through Self-Analysis", Technical Report number UPC-DAC-1999-43, Dep. d'Arquitectura de Computadors, UPC, 1999.

[18] J. Corbalán, X. Martorell, J. Labarta, "A Processor Scheduler: The CpuManager ", Technical Report UPC-DAC-1999-69 Dep. d'Arquitectura de Computadors, UPC, 1999.

[19] Cray T3E. `http://www.psc.edu/machines/cray/t3e/t3e.html`

[20] S.-H. Chiang, R. K. Mansharamani, M. K. Vernon. "Use of Application Characteristics and Limited Preemption for Run-To-Completion Parallel Processor Scheduling Policies", In Proc. of the ACM SIGMETRICS Conference, pp. 33-44, May 1994.

[21] H. Chu, . Nahrstedt, 'A Soft Real-Time Scheduling Server in UNIX Operating System", University of Illinois at Urbana Champaign, UIUCDS-R-97-1990.

[22] D. De Paoli, A. Gonscinski, M. Hobbs, and P. Joyce, "Performance comparison of process migration with remote process creation mechanism in RHODOS". In 16th Intl. Conf. Distributed Comput. Syst., pp. 554-561, May 1996.

[23] M. Devarakonda and A. Mukherjee, "Issues in implementation of cache-affinity scheduling", In Proc. Winter USENIX technical Conf., pp. 345-357, Jan 1992.

[24] J. Dongarra, A. Geist, R. Manchek, and V. Sunderam. Integrated PVM framework supports heterogeneous network computing. Computers in Physics, 7(2):166--75, April1993.

[25] L. Dowdy. "On the Partitioning of Multiprocessor Systems". Technical Report, Vanderbilt University, June 1988.

[26] A.B. Downey, "Using queue time predictions for processor allocation", In Job Scheduling Strategies for Parallel Processing, D.G. Feitelson and L.Rudolph (eds), pp. 35-57, Springer Verlag, 1997. Lectures Notes in Computer Science Vo. 1291.

[27] D. L. Eager, R. B. Bunt, "Characterization of programs for scheduling in multiprogrammed parallel systems". *Performance evaluation* 13 (1991) pp. 109-130.

[28] D.L. Eager, E.D. Lazowska, and J. Zahorjan. "Adaptive load sharing in homogeneous distributed systems". IEEE Trans. Softw. Eng. SE-12(5), pp. 662-675, May 1986.

[29] D.L. Eager, E.D. Lazowska, and J. Zahorjan. " The limited performance benefits of migrating active processes for load sharing". In SIGMETRICS Conf. Measurement & Modeling of Comput. Syst., pp. 63-72, May 1988.

[30] D. L. Eager, J. Zahorjan, E. D. Lawoska. "Speedup Versus Efficiency in Parallel Systems", IEEE Trans. on Computers, Vol. 38,(3), pp. 408-423, March 1989.

[31] D. G. Feitelson, B. Nitzberg. "Job Characteristics of a Production Parallel Scientific Workload on the NASA Ames iPSC/860", in JSSPP Springer-Verlag, Lectures Notes in Computer Science, vol. 949, pp. 337-360, 1995.

[32] D. G. Feitelson. “Job Scheduling in Multiprogrammed Parallel Systems”. IBM Research Report RC 19790 (87657), October 1994, rev. 2 1997.

[33] D. G. Feitelson, “Packing Schemes for Gang Scheduling”, Job Scheduling Strategies for Parallel Processing, pp. 89-110, Springer-Verlag, 1996. Lectures Notes in Computer Science, vol. 1162.

[34] D. G. Feitelson, M. A. Jette, “ Improved Utilization and Responsiveness with Gang Scheduling”, Job Scheduling Strategies for Parallel Processing, pp. 238-261. Springer-Verlag, 1997. Lectures Notes in Computer Science vol. 1291.

[35] D. G. Feitelson, L. Rudolph, “Distributed Hierarchical Control for Parallel Processing”,

[36] D. G. Feitelson, L. Rudolph, “Toward Convergence in Job Schedulers for Parallel Supercomputers”, Job Scheduling Strategies for Parallel Processing, pp. 1-26. Springer-Verlag 1996. Lectures Notes in Computer Science vol. 1162.

[37] D. G. Feitelson, L. Rudolph, “Evaluation of Design Choices for Gang Scheduling Using Distributed Hierarchical Control”, journal of Parallel and Distributed Computing 35, pp. 18-34 (1996)

[38] D. G. Feitelson, A. M. Weil, “Utilization and predictability in scheduling the IBM SP2 with backfilling”, In 12th International Parallel Processing Symposium, pp. 542-546, April 1998.

[39] F. Freitag, J. Corbalán and J. Labarta. "A Dynamic Periodicity Detector: Application to Speedup Computation". In 15th International Parallel and Distributed Processing Symposium (IPDPS’2001), pp. 2-8, April 2001.

[40] G. Ghare, S. Leutenegger, “The effect of Correlating Quantum Allocation and Job Size for Gang Scheduling”, Job Scheduling Strategies for Parallel Processing 1999.

[41] R. Gibbons, "A historical profiler for use by parallel schedulers". In Job Scheduling Strategies for Parallel Processing, D.G. Feitelson and L. Rudolph (eds.), pp. 58-77, Springer Verlpag, 1997. Lecture Notes in Computer Science Vol. 1291.

[42] B. Hamidzadeh, D. J. Lilja, ”Self-Adjusting Scheduling: An On-Line Optimization Technique for Locality Management and Load Balancing”, Int. Conf. on Parallel Processing, vol II, pp. 39-46, 1994.

[43] D. P. Helmbold, Ch. E. McDowell, “Modeling Speedup (n) greater than n”, IEEE Transactions Parallel and Distributed Systems 1(2) pp. 250-256, April 1990.

[44] M. Herlihy, B-H. Lim, and N. Shavit, "Low contention load balancing on large-scale multiprocessors", In 4th Symp. Parallel Algorithms & Architectures, pp. 219-227, Jun 1992.

[45] International Business Machines Corporation. AIX V4.3.3 Workload Manager. Technical White Paper. February 2000.

[46] IRIX 6.5 - Man pages.

[47] H. Jin, M. Frumkin, J. Yan. “The OpenMP Implementation of NAS Parallel Benchmarks and Its Performance”. Technical Report: NAS-99-011, 1999.

[48] D. Klappholz and H-C. Park, "Parallelized process scheduling for a tightly-coupled MIMD machine", In Intl. Conf. Parallel Processing, pp. 315-321, Aug 1984.

[49] S. Krakoviac, "Principles of Operating Systems". MIT Press, 1988.

[50] P. Krueger, T-H. Lai, and V.A. Radiya, "Processor allocation vs. job scheduling on hypercube computers". In 11th Intl. Conf. Distributed Comput. Syst., pp. 394-401, May 1991.

[51] P. krueger and M. Livny, "A comparison of preemptive and non-preemptive load distributing", In 8th Intl. Conf. Distributed Comput. Syst., pp. 123-130, Jun 1988.

[52] J. Labarta, S.Girona, V. Pillet, T. Cortes, L.Gregoris. "DiP : A Parallel Program Development Environment". 2ond Intl. EuroPar Conf. (EuroPar 96), Lyon (France), August 1996.

[53] J. Laudon and D. Lenoski, “The SGI Origin: A ccNUMA Highly Scalable Server”. Proc. 24th Int. Symp. on Computer Architecture, pp. 241-251, 1997.

[54] S.T. Leutenegger and M.K. Vernon, "Multiprogrammed Multiprocessor Scheduling Issues", Research report RC 17642 (#77699), IBM T.J. Watsin Research Center, Nov 1992.

[55] S. T. Leutenegger and M. K. Vernon. “The Performance of Multiprogrammed Multiprocessor Scheduling Policies”, In Proc. of the ACM SIGMETRICS Conference, pp. 226-236, May 1990.

[56] D. Lifka, "The ANL/IBM SP scheduling system". In Job Scheduling Strategies for Parallel Processing", D.G. feitelson and L. Rudolph (eds.), pp. 295-303, Springer-Verlag, 1995. Lectures Nostes in Computer Science Vol. 949.

[57] F.C.H. Lin and R.M. Keller, "The gradient model load balancing method". IEEE TRans. Softw. Eng. SE-13(1), pp. 32-38, Jan 1987.

[58] M. Madhukar, J. D. Padhye, L. W. Dowdy, ”Dynamically Partitioned Multiprocessor Systems”, Computer Science Department, Vanderbilt University, TN 37235, 1995.

[59] S. Majumdar, D. L. Eager, R. B. Bunt, ”Characterisation of programs for scheduling in multiprogrammed parallel systems”, Performance Evaluation 13, pp. 109-130, 1991.

[60] S. Majumdar, D. L. Eager, R. B. Bunt, "Scheduling in multiprogrammed parallel systems". In SIGMETRICS Conf. Measurement & Modeling of Comput. Syst., pp. 104-113, May 1988.

[61] B. D. Marsh, T. J. LeBlanc, M. L. Scott, E. P. Markatos, “First-Class User-Level Threads”. In 13th Symp. Operating Systems Principles, pp. 110-121, Oct. 1991.

[62] X. Martorell, "Dynamic Scheduling of Parallel Applications on Shared-memory Multiprocessors". PhD Thesis, Universitat Politècnica de Catalunya (UPC), July 1999.

[63] X. Martorell, J. Labarta, N. Navarro and E. Ayguade, “Nano-Threads Library Design, Implementation and Evaluation”. Dept. d’Arquitectura de Computadors - UPC, Technical Report: UPC-DAC-1995-33, September 1995.

[64] X. Martorell, J. Labarta, N. Navarro and E. Ayguade, “A Library Implementation of the Nano-Threads Programming Model”. Proc. of the Second Int. Euro-Par Conf., vol. 2, pp. 644-649, Lyon, France, August 1996.

[65] C. McCann, R. Vaswani, J. Zahorjan. “A Dynamic Processor Allocation Policy for Multiprogrammed Shared-Memory Multiprocessors”. ACM Trans. on Computer Systems, 11(2), pp. 146-178, May 1993.

[66] Message Passing Interface Forum. http://www.mpi-forum.org

[67] A. W. Mu’alem, D. G. Feitelson, "Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling". Tech. Report 2000-33, July 2000.

[68] "MIPS R10000 Microprocessor", http://www.sgi.com/processors/r10k/

[69] NANOS Consortium, “Nano-Threads Compiler”, ESPRIT Project No 21907 (NANOS), Deliverable M3D1. Also available at http://www.ac.upc.es/NANOS, July 1999.

[70] T.M. Ni, C-W. Xu, and T.B. Gendreau. " A distributed drafting algorithm for load balancing". IEEE Trans. Softw. Eng. SE-11(10), pp. 1153-1161, Oct 1985.

[71] J. Nieh, M. Lam, “The Design, Implementation and Evaluation of SMART: A Scheduler for Multimedia Applications”, Proc. of the 16th ACM Symposium on Operating Systems Principles, St. Malo, France, October, 1997.

[72] J. Nieh, M. Lam, “SMART UNIX SVR4 Support for Multimedia Applications”, Proc. of the IEEE Int. Conf. on Multimedia Computing and Systems, Ottawa, Canada, June 1997.

[73] D. S. Nikolopoulos, T. S. Papatheodorou. "A Quantitative Architectural Evaluation of Synchronization Algorithms and Disciplines on ccNUMA Systems: The Case of the SGI Origin 2000". Proc. of the 13th Intl. Conf. on Supercomputing, Rhodes (Greece), June 1999.

[74] T.D. Nguyen,J. Zahorjan, R. Vaswani, “Maximizing Speedup through Self-Tuning of Processor Allocation”. *IPPS 96*, Technical report UW-CSE-95-09-02. University of Washington

[75] T.D. Nguyen, J. Zahorjan, R.Vaswani, “Parallel Application Characterization for multiprocessor Scheduling Policy Design”. JSSPP, vol.1162 of Lectures Notes in Computer Science. Springer-Verlag, 1996.

[76] T. D. Nguyen, J. Zahorjan, R. Vaswani, “Using Runtime Measured Workload Characteristics in Parallel Processors Scheduling”, in JSSPP volume 1162 of Lectures Notes in Computer Science. Springer-Verlag, 1996.

[77] OpenMP Organization. “OpenMP Fortran Application Interface”, v. 2.0 http://www.openmp.org, June 2000.

[78] J. K. Ousterhout, “Scheduling Techniques for Concurrent Systems”, In Third International Conference on Distributed Computing Systems, pp. 22-30, 1982.

[79] K-H. Park and L.W. Dowdy, "Dynamic partitioning of multiprocessor systems". Intl. J. Parallel Programming 18(2), pp. 91-120, Apr 1989.

[80] E. W. Parsons, K. C. Sevcik. “Benefits of speedup knowledge in memory-constrained multiprocessor scheduling”, Performance Evaluation 27&28, pp.253-272, 1996.

[81] E. W. Parsons, K. C. Sevcik. “Implementing multiprocessor scheduling disciplines". In Job Scheduling Strategies for Parallel Processing, D.G. Feitelson and L.Rudolph (eds.), pp. 166-192, Springer-Verlag 1997. Lecture Notes un Computer Science Vol. 1291.

[82] P. Pierce. The NX/2 operating system. In Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications, pages 384--390. ACM Press, 1988.

[83] C.D. Polychronopoulos, "Multiprocessing versus multiprogramming". In Intl. Conf. Parallel Processing, vol. II, pp. 223-230, Aug 1989.

[84] E. D. Polychronopoulos, X. Martorell, D. S. Nikolopoulos, J. Labarta, T. S. Papatheodorou and N. Navarro, “Kernel-level Scheduling for the Nano-Threads Programming Model” Proc. of the 12th *ACM International Conference on Supercomputing*, pp. 337-344, Melbourne, Australia, July 1998.

[85] E. D. Polychronopoulos, D. S. Nikolopoulos, T. S. Papatheodorou, X. Martorell, J. Labarta, N. Navarro. "An Efficient Kernel-Level Scheduling Methodology for Multiprogrammed Shared Memory Multiprocessors". Proc. of the 12th Intl. Conf. on Parallel and Distributed Computing Systems. Fort Lauderdale, Florida, August 1999.

[86] E. Rosti, E. Smirni, L.W. Dowdy, G. Serazzi, and B.M.Carlson, "Robust partitioning schemes of multiprocessor systems". Performance Evaluation 19 (2-3), pp. 141-165, Mar 1994.

[87] A. Serra, N. Navarro, T. Cortes, “DITools: Application-level Support for Dynamic Extension and Flexible Composition”, in Proceedings of the USENIX Annual Technical Conference, pp. 225-238, June 2000.

[88] S. K. Setia, “Trace-driven Analysis of Migration -based Gang Scheduling Policies for Parallel Computers”, ICPP97, August 1997.

[89] K. C. Sevcik, ”Application Scheduling and Processor Allocation in Multiprogrammed Parallel Processing Systems”. Performance Evaluation 19 (1/3), pp. 107-140, Mar 1994.

[90] K. C. Sevcik. “Characterization of Parallelism in Applications and their Use in Scheduling”. In Proc. of the ACM SIGMETRICS Conference, pp. 171-180, May 1989.

[91] C. Severance, r. Enbody, and P. Petersen. "Managing the overall balance of operating system threads on a multiprocessor using automatic self-allocating threads (ASAT)". J. Parallel & Distributed Comput. 37(1), pp. 106-112, Aug 1996.

[92] SGI Techpubs library, `http://techpubs.sgi.com/`

[93] Silicon Graphics Inc. Origin2000 and Onyx2 Performance Tuning and Optimization Guide. `http://techpubs.sgi.com`, Document Number 007-3430-002, 1998.

[94] F. A. B. Silva, I. D. Scherson, “Improving Throughput and Utilization in Parallel Machines Through Concurrent Gang”, International Parallel and Distributed Processing Symposium 2000.

[95] F. A. B. Silva, I. D. Scherson, “Improving Parallel Job Scheduling Using Runtime Measurements”, 6th Workshop on Job Scheduling Strategies for Parallel Processing, 2000.

[96] J.P. Singh, W.D.Weber, and A. Gupta. “SPLASH: Standford Parallel Applications for Shared Memory”. *Computer Architecture News*, 20(1):5-44, 1992.

[97] J. Skovira, W. Chan, H. Zhou, and D. Lifka, "The EASY- LoadLeveler API project". In Job Scheduling Strategies for Parallel Processing, D.G Feitelson and L. Rudolph (eds.), pp. 41-47, Springer-Verlag, 1996. Lectures Notes in Computer Science Vol. 1162.

[98] M.S. Squillante and E.D. Lazowska, "Using processor-cache affinity information in shared-memory multiprocessor scheduling", IEEE Trans. Parallel & Distributed Syst. 4(2), pp.131-143, Feb 1993.

[99] Standard Performance Evaluation Corporation. SPEC CPU95 Benchmarks. Available at http://www.spec.org/osg/cpu95, 1995.

[100] I. Subramanian, C. McCarthy, M. Murphy. “Meeting Performance Goals with the HP-UX Workload Manager”, Proc. of the First Workshop on Industrial Experiences with Systems Software, WIESS 2000, pp. 79-80. October 2000, San Diego, California.

[101] Sun Microsystems. “Solaris Resource Manager [tm] 1.0: Controlling System Resources Effectively”, technical white paper, http://www.sun.com/software/white-papers/wp-srm, 2000.

[102] The Standard Workload Format”, http://www.cs.huji.ac.il/labs/parallel/workload/swf.html

[103] J. Torrellas, A. Gupta, and J. Henessy, "Characterizing the caching and synchronization performance of a multiprocessor operating system". In 5th Intl. Conf. Architect. Support for Prog. Lang. & Operating Syst., pp. 162-174, Oct 1992.

[104] S.K. Tripathi, G. Serazzi, and D. Ghosal. "Processor scheduling in multiprocessor systems". In Parallel Computation, H.P. Zima (ed), pp. 208-255, Springer-Verlag, 1992. Lectures Notes in Computer Science Vol 591.

[105] A. Tucker, A. Gupta, “Process control and scheduling issues for multiprogrammed shared-memory multiprocessors”. In 12th Symposium Operating Systems Principles. pp. 159-166, December 1989.

[106] R. Vaswani and J. Zahorjan, "The implications of cache affinity on processor scheduling for multiprogrammed, shared-memory multiprocessors". In 13th Symp. Operating Systems Principles, pp. 26-40, Oct 1991.

[107] M. J. Voss, R. Eigenmann, ”Reducing Parallel Overheads Through Dynamic Serialization”, Proc. of the 13th Int. Parallel Processing Symposium, pp. 88-92, 1999.

[108] B. Weissman, “Performance Counters and State Sharing Annotations: A Unified Approach to Thread Locality”, Proc. of the 8th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, pp. 127 - 138, 1998.

[109] Workload logs, http://www.ac.upc.es/homes/juli

[110] K. C. Yeager, “The MIPS R10000 Superscalar Microprocessor”. IEEE Micro vol. 16, 2 pp. 28-40, 1996.

[111] I-L. Yen and F.B. Bastani, "Robust parallel resource management in shared memory multiprocessor systems". In 9th Intl. Parallel Processing Symp., pp. 458-465, apr 1995.

[112] K.K. Yue and D.J. Lilja, "Loop-level process control: an effective processor allocation policy for multiprogrammed shared-memory multiprocessors". In Job Scheduling strategies for Parallel Processing, D.G. Feitelson and L. Rudolph (eds.), pp. 182-199, Springer-Verlag, 1995. Lectures Notes in Computer Science Vol. 949.

[113] J. Zahorjan and C. McCann, "Processor scheduling in shared memory multiprocessors". In SIGMETRICS conf. Measurement & Modeling of Comput. Syst., pp. 214-255, May 1990.

[114] Y. Zhu and M. Ahuja, "On job scheduling on a hypercube", IEEE TRans. Parallel & Distributed Syst. 4(1), pp. 62-69, Jan 1993.

[115] B. B. Zhou, D. Walsh, R. P. Brent, “Resource Allocation Schemes for Gang Scheduling”, Job Scheduling Strategies for Parallel Processing, pp. 74-86, Springer-Verlag 2000, Lectures Notes in Computer Science vol. 1911.

[116] Y. Zhang, H. Franke, J. E. Moreira, A. Sivasubramaniam, “ Improving Parallel Job Scheduling by Combining Gang Scheduling and Backfilling Techniques”, IPDPS 2000, Cancun.