Proceedings of the Third International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2016) Sofia, Bulgaria

Jesus Carretero, Javier Garcia Blas, Svetozar Margenov
(Editors)

October, 6-7, 2016

Resource Management Optimization in Multi-Processor Platforms

Atanas Hristov

University of Information Science and Technology, Ohrid, Macedonia
atanas.hristov@uist.edu.mk

Iva Nikolova, Georgi Zapryanov

Technical University of Sofia, Bulgaria
inni@tu-sofia.bg, gszap@tu-sofia.bg

Dragi Kimovski

University of Innsbruck, Austria
dragi@dps.uibk.ac.at

Vesna Kumbaroska

University of Information Science and Technology, Ohrid, Macedonia
vesna.gega@uist.edu.mk

Abstract

The modern high-performance computing systems (HPCS) are composed of hundreds of thousand computational nodes. An effective resource allocation in HPCS is a subject for many scientific research investigations. Many programming models for effective resources allocation have been proposed. The main purpose of those models is to increase the parallel performance of the HPCS. This paper investigates the efficiency of parallel algorithm for resource management optimization based on Artificial Bee Colony (ABC) metaheuristic while solving a package of NP-complete problems on multi-processor platform.In order to achieve minimal parallelization overhead in each cluster node, a multi-level hybrid programming model is proposed that combines coarse-grain and fine-grain parallelism. Coarse-grain parallelism is achieved through domain decomposition by message passing among computational nodes using Message Passing Interface (MPI) and fine-grain parallelism is obtained by loop-level parallelism inside each computation node by compiler-based thread parallelization via Intel TBB. Parallel communications profiling is made and parallel performance parameters are evaluated on the basis of experimental results.

Keywords High-Performance Computing, Parallel Programming Model, Parallel Performance, Parallel Algorithm

I. Introduction

There are many open research problems in the field of high-performance computing systems (HPCS) studied extensively in many scientific research investigations. These systems are composed of hundreds or thousands of computational nodes and combine several technologies - hardware, software, networking and programming to solve advanced problems and performing experimental research work.

Most often the HPCS are used for high-throughput computing in time-sharing mode as well as for running complex parallel applications in space-sharing mode. One of the main challenges in HPC is to achieve highest possible system performance for a given application at optimal load balance and utilization of the available computational resources on the HPC platform. This causes the problem of effective resource management.

The resource management system is responsible for allocation of computing resources for extraordinary use and also to determine an optimal job or task scheduling for a given system topology. An effective resource allocation and scheduling in HPCS is a subject of many scientific research investigations. The problem is well known as NP-complete [1], [2] and a number of approaches to different aspects of this problem can be found in the research literature. Also, many programming models have been proposed during the years. The main purpose of these models is to increase the parallel performance of HPCS. Currently, most of the HPC systems are based on conventional sequential programming languages as C, C++, FORTRAN. In order to achieve better parallel performance, the flat parallel programing model with message passing in distributed memory systems, supported by the MPI standard [3] and parallel programming model with multithreading in shared memory systems using the OpenMP programming interface [4] have been included as template libraries. The main disadvantages of the parallel programing based on conventional programming language are: process synchronization, deadlocks, workload balancing, and thread concurrency. In order to achieve better parallel performance, a parallel programming model must combine the distributed memory parallelization on the node interconnect with the shared memory parallelization inside of each node.

In order to improve this situation, Intel provides a range of tools specifically designed to help developers in parallelizing their applications. Three sets of complementary models for multithreading programming in shared memory systems are supported by Intel: Intel Cilk Plus, Intel Threading Building Blocks (Intel TBB) and Intel Array Building Blocks (Intel ArBB). The main purpose of those models is to increase the reliability, portability, scalability and the parallel performance of the application during the multithreading execution [5], [6].

The complexity class of decision problems NP-complete can be used as a pattern for benchmarking and parallel performance evaluation of multi-core and multi-machine architectures. Parallel versions of several NP-complete problems, such as N-Queens Problem, Travelling Salesman Problem, Sam-Loyd Puzzle etc., will be proposed in order to determinate the overall parallel performance of the system.

The paper investigates the efficiency of parallel algorithm for resource management optimization. It is proposed a metaheuristic approach based on swarm optimization with Artificial Bee Colony (ABC) [7] to solve the resource allocation problem for multi-core platform. The experimental work is based on the efficiency analysis of the proposed resource allocation scheme when solving of a package of three well known NP-complete problems - N-Queen, Travelling Salesman and Sam-Loyd Puzzle on homogeneous multi-processor platforms. Programmatically, the proposed scheme is implemented on the basis of multi-level hybrid parallel computational model using Intel TBB [8] and MPI [3] libraries. This model combines coarse-grain and fine-grain parallelism. Coarse-grain parallelism is achieved through domain decomposition by message passing among computational nodes using Message Passing Interface (MPI) and fine-grain parallelism is obtained by loop-level parallelism inside each computation node by compiler-based thread parallelization via Intel TBB.

The rest of this paper is organized as follows. An overview of the resource scheduling problem is presented in Section II with discussing some related works. In Section III, the proposed resource allocation scheme, based on ABC metaheuristics and its parallel implementation is presented. An experimental results and summary are offered in Section IV.

II. Related work

The resource management optimization problem has been studied extensively in the parallel and distributed computing literature for more than two decades. Many studies have been done in the field in order to effectively utilize the costly high performance computing platforms. Most of the advanced resource management systems are vendor-specific, but often they do not comply with specific features of a particular computing platform. Thus, they are not well optimized to provide efficient management to reach the required for a given parallel application performance of the implementation.

A variety of policies, strategies, schemes and algorithms have been proposed, developed, analyzed and implemented in a number of studies. These works investigate the problem in terms of diverse target HPC platforms. The most common researches are done in the field of high performance distributed computing with cluster, grid and cloud computing systems. Regardless of the conceptual closeness of these systems, the strategies for an optimal reserving of computing resources, effective load balance and resource utilization are different. Also, the resources that each parallel application for distributed processing requires can be very different from one to other and this raised the problem of finding an optimal job and tasks schedule for a given set of parallel resources. Taking into account the specific architectural, system and communication characteristics of a given parallel computer platform, finding an optimal solution of resource management task is further complicated.

The parallel resource scheduling problem is known to be NP-hard [1], [2]. It is usually solved by various heuristic and meta-heuristics algorithmic schemes [9], [10] depending on the homogeneity of the parallel system and the scheduling method applied - static (off-line) or dynamic (on-line) one. Also, there are several exact algorithms with the goal to solve small to medium size problems to optimality [11], [12].

In the schemes with static scheduling it assumes that the total number of parallel executing tasks, as well the duration of each task is known in advance. The decision concerning computational resource allocation and task assignment is made at the start of the job execution. Because the execution time for a task is dependent on the input data, static scheduling carries some degree of uncertainty. This leads to unbalanced load, which leads to longer parallel execution time and low system resource utilization. With dynamic scheduling, the number of computational resources allocated to a job may vary during the execution. Also, the task assignment to the allocated resources takes place during the execution of a job. As pointed out in [13], dynamic scheduling policies are complementary to static policies in both their advantages and drawbacks. Because they are implemented at the execution time, dynamic policies usually incur in high run-time overhead, which may lead to a degradation of performance. Since decisions are made during job execution, scheduling should be based on simple and constant time heuristics. On the other hand, dynamic scheduling mechanisms exhibit an adaptive behavior, which leads to a high degree of load balancing.

In [14] the resource scheduling problem is explored in terms of real-time jobs executing on heterogeneous clusters. Heterogeneity in the parallel systems introduces an additional degree of complexity because, in addition to the problem of deciding when and how many computing resources to allocate, scheduling policies have also to deal with the choice among processor nodes of different speeds and also with reliability issues and tasks independency in parallel jobs. The proposed heuristic dynamic scheduling scheme (reliability-driven algorithm (DRCD)) for a various cluster sizes (between 4 and 18 machines) has been experimentally tested on a real world application DSP [15] as well as synthetic workloads, based on binary trees [16], lattices [17] and random graphs [18].

The problem of resource management in large many-core systems is addressed in [19], where a novel resource-management scheme that supports so-called malleable applications is proposed. These applications can adopt their level of parallelism to the assigned resources. [19] design a scalable decentralized scheme that copes with the computational complexity by focusing on local decision-making. The proposed algorithm is tested via simulation experiments on different system sizes ranging from 5x5 to 32x32 cores and synthetically generated workload consisting of 16, 32 and 64 parallel applications that is generated using the widely used Downey model [20].

Many papers have been published to address the problem of resource allocation in Grid computing environments. Some of the proposed algorithms are modifications or extensions to the traditional distributed systems resource allocation algorithms. A survey of job scheduling and resource management algorithms in Grid computing can be found in [21] where various algorithms are compared on various parameters like distributed, hierarchical, centralized, response time, load balancing, and resource utilization. The experiments were conducted via simulations with help of GridSim for number of jobs varied from 50 to 300.

In [22] resource-aware hybrid scheduling algorithm for different type of application: batch jobs and workflows are proposed. The performance tests are conducted in a realistic setting of CloudSim tool [23] with respect to load-balancing, cost savings, dependency assurance for workflows and computational efficiency. Multimedia applications that consist in both independent tasks and tasks with dependencies (workflows), and are both CPU intensive (they process a large amount of data) and I/O intensive (they access remote data) are used as test case scenarios. The experiments are performed with a 1000 tasks, 1000 Processing Elements and 10 Virtual Machines.

III. Parallel Implementation of Resource Management Optimization Algorithm

An effective resource utilization of the modern high performance computing (HPC) platforms is a subject for many scientific research investigations. The resource management optimization for those platforms is an essential part for optimal resource allocation while solving NP hard problems. An effective resource management algorithm strongly determines the overall parallel performance of the high-performance computing system. The proposed algorithm for resource management optimization in multi-core and multi-machine platforms is based on Artificial Bee Colony (ABC) metaheuristic. The ABC simulates the collective behavior of the honeybees in nature. The basic approach during implementation process is building a computer model which will simulate the collective behavior of the bees while collecting nectar.

In the proposed algorithm, the bees are divided in two beehives, beehive of the scout bees (beehive 1) and beehive of the onlooker and worker bees (beehive 2). When the algorithm is started, beehive 1 generates N number of scout bees, where N represents the number of processors in the system. Each scout bee checks whether a processor is free or busy by execution of specific task on it. If a free resource

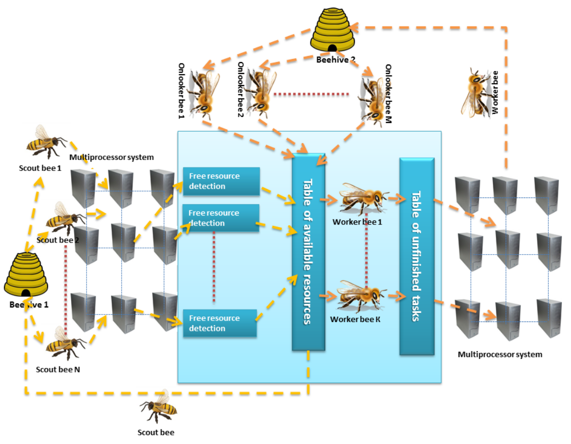

Figure 1: Parallel computing model for resource management optimization based on Artificial Bee Colony metaheuristic

is found the scout bee record the ID of the processor into the table of available resources and returns to the beehive 1, where it is terminates. The main purpose of the beehive 2 is to generate M number of onlooker bees, where M is the optimal number of parallel threads. After generation, the onlooker bees search in to table of available resources. If the onlooker bee finds a free resource, it takes the ID of the processor and removes it from the table. If the onlooker bee do not find a free resource in the table, the bee will return to the beehive 2 and will be terminate. Once the onlooker bee takes the available resource it starts to behave as a worker bee. Thus obtained K number of worker bees initially turned to the table of outstanding tasks where they taking certain sub-problem, remove it from the table and submit it for the performance by the processor which ID has been taken from the table of available resources. After the processor solves a sub-problem, it provides the solution to a worker bee. The worker bee with the current solution returns to beehive 2, where it is terminated.

In Figure 1, parallel computing model for resource management optimization based on artificial bee colony metaheuristic is presented. Parallel implementation of the algorithm was realized by using MPICH-2 message passing model and Intel TBB programming model built in Intel Parallel Studio 2010. For virtualization of resources a virtual machine of Intel ArBB, built-in Intel Parallel Studio 2010 was used.

IV. Experimental Evaluation

The experimental results were conducted by using multi-processor platform. The platform is represented by a homogenous cluster composed of twelve Blade servers, HS21, Xeon Quad Core E405 80w 2.00GHz/1333MHz/12MB L2 and hard disk drive subsystems IBM 750GB Dual Port HS SATA HDD and Windows Server 2008 operating system.

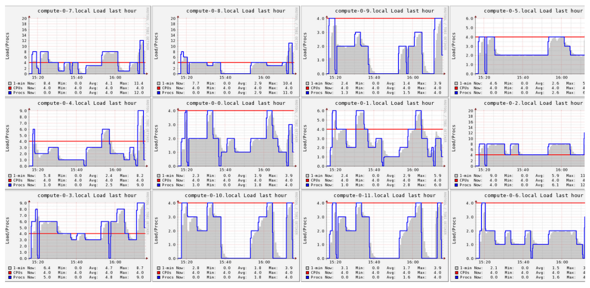

Figure 2: CPU load while solving package of three NP-Complete problems without the algorithm for resource management optimization

The load of the computational resources while solving a package of three NP-Complete problems - Traveling Salesman Problem, the N-queens problem, and the Sam-Loyd puzzle are presented in Figure 2. The package is started without algorithm for resource management optimization. From the charts shown on the figure, it is clear that the load of the processors is not well balanced, because only at a certain point of the time the processors have good load balance i.e. there are only few processors where the number of cores corresponds with the number of active processes. On the other hand, during the most of the time some of the processors have less number of active processes than cores, while some processors are overloaded i.e. the number of active processes exceeds number of cores in the processor.

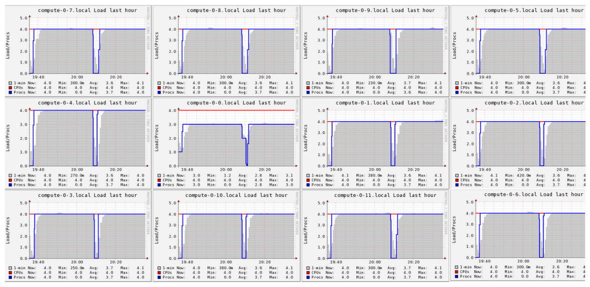

Figure 3: CPU load while solving package of three NP-Complete problems by using the algorithm for resource management optimization

In order to improve this situation, the proposed algorithm for resource management optimization was implemented on the target platform. In Figure 3, the load of the processors while solving a package of three NP-Complete problems by using the algorithm for resource management optimization based on ABC metaheuristic is shown.

According to the figure above, it is clear that after the implementation of the optimization algorithm, the load of the processors is almost optimal as the overloading of the processors is avoided i.e. starting a bigger number of threads than cores on single processor, while dissatisfied load is shown only during the timeslots reserved of implementation of the algorithm for resource planning.

Figures 4 and 5 presents the load of the cluster during the execution of the tested package.

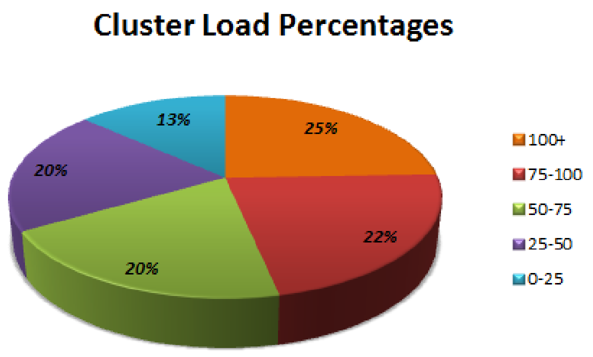

Figure 4: Cluster load during the execution of a package with three NP-Complete problems without the optimization algorithm

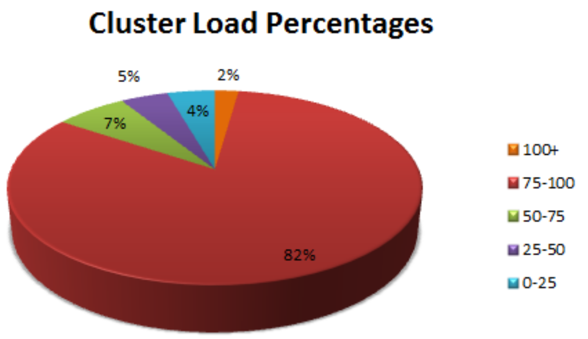

Figure 5: Cluster load during the execution of a package with three NP-Complete problems by using the proposed optimization algorithm

During the execution of package with three NP-Complete problems without the proposed algorithm for resource management optimization, only 22,22% of the resources of the cluster have optimal load balancing with 75-100%, while the remaining resources are overloaded with 100%+ or not good loaded i.e. below 75%. On the other hand, during the execution of the package by using the algorithm for resource management optimization, 82.22% of the resources of the cluster have optimal load balancing and only 17.77% of the resources have poor load balance. These 17% of the resources with poor load balance appears mainly due to the time required for implementation of the algorithm for resource planning as well as other system costs of the platform.

V. Conclusion and future work

An effective resource utilization of the modern high performance computing (HPC) systems is a subject for many scientific research investigations. The resource management for those platforms is an essential part for optimal resource allocation while solving NP complete problems. An effective resource management algorithm strongly determines the overall parallel performance of the high-performance computing system.

This paper suggests an innovative algorithm for effective resource management in multi-processor platforms based on parallel metaheuristic "Artificial Bee Colony" (ABC) optimization. The efficiency of the proposed algorithm for resource management in multi-processor platforms was evaluated on the basis of the software tools of Intel Array Building Blocks build-in Intel Parallel Studio.

Moreover, parallel programming implementations of three NP-Complete problems: the N-Queens problem, the Sam-Loyd puzzle, and the Traveling Salesman Problem (TSP) have been proposed in order to evaluate the overall parallel performance of the platform. The proposed parallel implementations were developed on the basis of Message Passing Interface (MPI) and Intel Threading Building Blocks (TBB) programming models.

Finally, we applied the proposed algorithm a homogenous cluster composed of twelve Blade servers HS21. This allows us to observe the behavior of the cluster while simultaneously is started a package of three NP-Complete problems. From the experimental results we conclude that in the cases when the proposed algorithm is run on the target platform, the cluster has very good load balance, which leads to increasing of the overall parallel performance of the system.

Future objectives of this research include implementation of our algorithm on very large-scale systems and on the new generation of ExaScale machines.

Acknowledgment

This work is partially supported by EU under the COST Program Action IC1305: Network for Sustainable Ultrascale Computing (NESUS).

References

[1] Ullman, J.D., 1975. NP-complete scheduling problems. Journal of Computer and System sciences, 10(3), pp.384-393.

[2] Hall, N.G. and Sriskandarajah, C., 1996. A survey of machine scheduling problems with blocking and no-wait in process. Operations research, 44(3), pp.510-525.

[3] Gropp, W., Lusk, E., Doss, N. and Skjellum, A., 1996. A high-performance, portable implementation of the MPI message passing interface standard. Parallel computing, 22(6), pp.789-828.

[4] Sato, M., 2002, October. OpenMP: parallel programming API for shared memory multiprocessors and on-chip multiprocessors. In Proceedings of the 15th international symposium on System Synthesis (pp. 109-111). ACM.

[5] Wooyoung Kim, Voss M., 2011. Multicore Desktop Programming with Intel Threading Building Blocks. IEEE Software journal, Page(s): 23 âĂŞ 31.

[6] Newburn C.J., Byoungro So, Zhenying Liu, McCool M., Ghuloum A., Toit S.D., 2011. Intel's Array Building Blocks: A retargetable, dynamic compiler and embedded language. In Proceedings of the 9th Annual IEEE/ACM International Symposium on Code Generation and Optimization, Page(s): 224 âĂŞ 235.

[7] Karaboga, D. and Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization, 39(3), pp.459-471.

[8] Reinders, J., 2007. Intel threading building blocks: outfitting C++ for multi-core processor parallelism. " O'Reilly Media, Inc."

[9] Haouari, M., Gharbi, A. and Jemmali, M., 2006. Tight bounds for the identical parallel machine scheduling problem. International Transactions in Operational Research, 13(6), pp.529-548.

[10] Talbi, E.G., 2009. Metaheuristics: from design to implementation (Vol. 74). John Wiley & Sons.

[11] Darbha, S. and Agrawal, D.P., 1998. Optimal scheduling algorithm for distributed-memory machines. IEEE transactions on parallel and distributed systems, 9(1), pp.87-95.

[12] Dell'Amico, M., Iori, M., Martello, S. and Monaci, M., 2008. Heuristic and exact algorithms for the identical parallel machine scheduling problem. INFORMS Journal on Computing, 20(3), pp.333-344.

[13] Saha, D., Menasce, D. and Porto, S., 1995. Static and dynamic processor scheduling disciplines in heterogeneous parallel architectures. Journal of Parallel and Distributed Computing, 28(1), pp.1-18.

[14] Qin, X. and Jiang, H., 2005. A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters. Journal of Parallel and Distributed Computing, 65(8), pp.885-900.

[15] Woodside, C.M. and Monforton, G.G., 1993. Fast allocation of processes in distributed and parallel systems. IEEE Transactions on parallel and distributed systems, 4(2), pp.164-174.

[16] Srinivasan, S. and Jha, N.K., 1999. Safety and reliability driven task allocation in distributed systems. IEEE Transactions on Parallel and Distributed Systems, 10(3), pp.238-251.

[17] Qin, X. and Jiang, H., 2001, September. Dynamic, reliability-driven scheduling of parallel real-time jobs in heterogeneous systems. In Parallel Processing, 2001. International Conference on (pp. 113-122). IEEE.

[18] Ahmad, I. and Kwok, Y.K., 1999. On parallelizing the multiprocessor scheduling problem. IEEE Transactions on Parallel and Distributed systems, 10(4), pp.414-431.

[19] Kobbe, S., Bauer, L., Lohmann, D., SchrÃűder-Preikschat, W. and Henkel, J., 2011, October. DistRM: distributed resource management for on-chip many-core systems. In Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis (pp. 119-128).

[20] Downey, A.B., 1997. A model for speedup of parallel programs. University of California, Berkeley, Computer Science Division.

[21] Buyya, R. and Murshed, M., 2002. Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency and computation: practice and experience, 14(13âĂŘ15), pp.1175-1220.

[22] Vasile, M.A., Pop, F., Tutueanu, R.I. and Cristea, V., 2013, December. HySARC2: hybrid scheduling algorithm based on resource clustering in cloud environments. In International Conference on Algorithms and Architectures for Parallel Processing (pp. 416-425). Springer International Publishing.

[23] Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A. and Buyya, R., 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience, 41(1), pp.23-50.