Local Search for Grid Scheduling

Dalibor Klusáček
Luděk Matyska
Hana Rudová
Faculty of Informatics, Masaryk University
Botanická 68a, Brno 602 00
Czech Republic
{xklusac, ludek, hanka}@fi.muni.cz

Ranieri Baraglia
Gabriele Capannini
ISTI,CNR
Via Moruzzi, 1
Pisa, Italy
{ranieri.baraglia, gabriele.capannini}@isti.cnr.it

Abstract

This work introduces local search based algorithms as a new technique for the Grid scheduling problem. Specific algorithms based on dispatching rules and local search were proposed and implemented to generate schedule for dynamically arriving jobs. Algorithm performance was compared with typical queue-based algorithms on the basis of objective function optimisation and time required to generate scheduling solutions. Grid environment was simulated by Alea Simulator which is based on modified and extended GridSim toolkit. The results showed that local search based algorithms may be promising technique with good overall performance, providing better results than queue-based approaches while still fast enough to provide solution in a reasonable time.

Introduction

Grid scheduling is a very complex problem where application of advanced scheduling techniques is often not easy. Current scheduling production systems such as Condor (Thain, Tannenbaum, & Livny 2005), LSF (Xu 2001) or PBS (Feng, Misra, & Rubenstein 2007) are usually *queue-based* systems using scheduling policies. Also the more complex tools and systems such as Grid Service Broker (Venugopal, Buyya, & Winton 2004) or GridWay (Huedo, Montero, & Llorente 2005) emphasize scheduling policies, i.e., job and resource prioritization policies.

Our intent is to study the *schedule-based* approach, where a schedule is constructed and optimised. We introduce advanced scheduling algorithms based on local search and dispatching rules (Glover & Kochenberger 2003; Pinedo 2005) to achieve this goal. In comparison with queue-based systems, schedule-based approach maintains information not only about resources but also about the run time of all jobs (Keller & Reinefeld 2001). We concentrate on a dynamic scenario when jobs arrive over the time (He, Sun, & von Laszewski 2003) and disappear from the scheduling process at their completion time. Here application of local search algorithms seems to be very suitable but according to our knowledge it has not been applied to dynamic problems yet. The existing approaches applying local search to the static problems (Armentano & Yamashita 2000; Baraglia, Ferrini, & Ritrovato 2005) are based on the assumption that all the jobs and resources are known in advance and the local search for all jobs must be performed at once. This is potentially very time consuming, and we propose an incremental approach, when the current existing local search solution is modified after arrival of new job(s) through simple moves of jobs within the schedule.

This paper describes application of the tabu search algorithm (Glover & Kochenberger 2003) used to optimise initial schedule computed with the help of dispatching rules (Pinedo 2005) in the dynamic Grid environment. Quality and time requirements of these algorithms were compared with typical queue-based algorithms using our Grid scheduling simulation environment—the *Alea Simulator*. Performed experiments show interesting improvement in the value of the objective function using the schedule-based approach compared to the queue-based systems.

Applied Approaches

In this work the goal of the scheduler was to minimize the number of jobs that did not meet their deadline (Capannini *et al.* 2007). The smaller this number was the higher QoS was provided to the users, i.e., job owners. We use the following setup: A job may require one or more CPUs, machines may have different number of CPUs with a different CPU speed.

The initial schedule was generated using composed dispatching rule *Minimum Tardiness Earliest Deadline First (MTEDF)*. It works in two steps—first the machine with the smallest expected tardiness of jobs assigned to it is selected. Then the newly arrived job is placed into this machine's schedule using *Earliest Deadline First (EDF)* strategy (Pinedo 2005). MTEDF is applied every time a new job arrives and it constructs new schedule using previously computed schedule. This saves the time since the schedule does not have to be computed from scratch. Currently schedule is represented by simplified data structure. For each machine a list of jobs planned on this machine is maintained. The job order inside the list defines the order of job execution on the corresponding machine. These lists together represent the schedule.

Tabu Search (TS) optimisation is then applied to this schedule. The job with the highest expected delay wrt. its

deadline is moved. To minimize the size of the neighborhood we only allow moves onto machines with smaller expected tardiness of assigned jobs. Once a machine is selected the job is placed into it's schedule using EDF strategy and the value of the objective function is computed. If the value is better, i.e., the number of jobs not meeting their deadline is smaller, the move is accepted. Once the job was moved it is placed into the tabu list to prevent cyclic moves.

As a comparison to these techniques we used *First Come First Served (FCFS)*, *Earliest Deadline First (EDF)* and *Easy Backfilling (BF)* (Lifka 1995; Techiouba *et al.* 2007) algorithms as a typical queue-based scheduling policies.

Alea Simulator

As a part of our work we have developed *Alea Simulator* [1], a GridSim (Buyya & Murshed 2002) based Grid simulator. It extends the original Java based toolkit with various new features such as parallel job support, dynamic job arrivals etc. Alea provides newly developed extensible centralized scheduler which allows to implement various scheduling algorithms and compare their performance easily. Both the queue-based as well as schedule-based algorithms can be simulated. Different criterion such as makespan, average flow time, or total tardiness may be used and then optimised by the scheduler. The whole simulator is implemented as a modular environment which admits to extend the simulator with new features such as new scheduling algorithms or job parameters and to model different types of problems like sequential or parallel jobs, etc. All experiments presented in this work were performed in the Alea Simulator.

Experimental Results

The experiments were done for 1500 dynamically arriving jobs with average execution time 1600 s. Jobs may be either sequential or parallel (in average job needs 4.5 CPUs). 70% jobs have deadline parameter which denotes maximal completion time of this job as was requested by the user before the job submission. The Grid consists of 150 machines with 2-16 CPUs, different machines may have different speed. The objective criterion was to minimize number of jobs that did not meet their deadline. The comparison of the queue-based and schedule-based algorithms was based on this objective criterion and the time required to generate scheduling decision for one job. Experimental data sets were generated synthetically using exponential distribution with the generator from (Capannini *et al.* 2007; Techiouba *et al.* 2007). We tested seven different loads of the system. In the first case jobs were arriving frequently so the number of jobs waiting for their execution was very high. Remaining cases had stepwise higher average job inter-arrival time so the load of the system was lower.

Figure 1 shows that the schedule-based approach outperforms the queue-based algorithms in all situations. The Figure 2 shows the average scheduling time for one job. The time complexity depends on the number of jobs present in the queue/schedule. Tabu search time requirements are quite

[1] See `http://www.fi.muni.cz/~xklusac/alea/` for the distribution and source code.

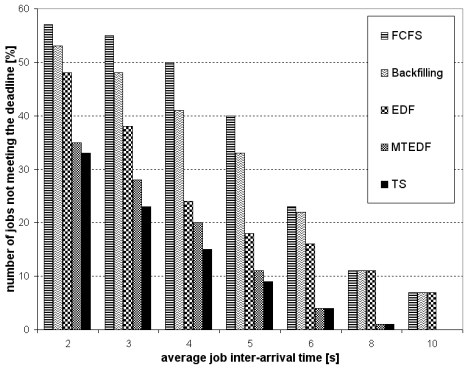

Figure 1: Number of jobs not meeting their deadline.

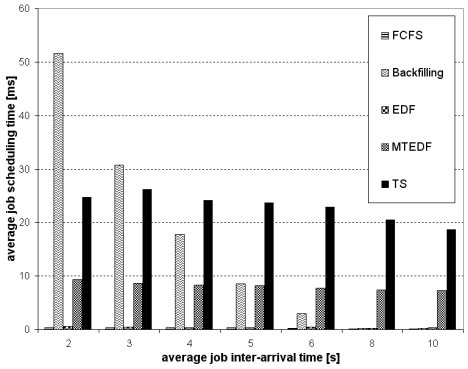

Figure 2: Average job scheduling time in milliseconds.

stable because it is limited with the number of iterations while in case of backfilling the time strongly depends on the number of jobs waiting in the queue. If the job inter-arrival time was high then both the queue or the schedule was almost empty most of the time so the benefit of EDF, backfilling or tabu search did not appear. When comparing the average job scheduling time for particular methods it is also important to realize that one job is scheduled in milliseconds but inter-arrival times are 2 seconds in the worst case. So, the time efficiency of algorithms is not so critical in comparison with the optimisation quality.

Discussion

The schedule-based approach is a promising direction in the area of the Grid scheduling as it was shown for example by the CCS Planning Manager (Keller & Reinefeld 2001). However, there is no work done in the area of local search based methods applied to the dynamic environment. Preliminary results showed certain improvement even when the simplified algorithms and data structures representing the schedule were used. Local search is often considered to be very time consuming algorithm. Therefore, we propose an incremental approach based on a previously computed schedule that has very reasonable runtime requirements. Another advantage is the stability of this approach. Application of a dispatching rule is rather fast and it always generates acceptable schedule, so we can stop local search optimisation

at any time if prompt decisions are required (Bent & Hentenryck 2004).

We can see that schedule-based approaches outperform queue-based approaches. However, improvements of local search over dispatching rules is currently more significant in some cases only and more experiments are needed to fully exploit the local search algorithms potential.

We see the major advantage of the local search based algorithms in the possibility to directly and extensively manipulate with the explicit schedule. Using it, we plan to be able to predict the resource load at any specified time. We will use this information to move jobs to the time where the capacity of resources is not fully occupied by the currently planned jobs. This feature can be compared with the capability of backfilling algorithms where unused capacity of resources is to be filled by other waiting jobs. Even more local search moves may include a swap of two jobs which can also improve the quality of the schedule (e.g., late jobs is swapped with the job having a later deadline). We believe that all of that will induce further improvement in the performance of the local search algorithms.

Conclusion and Future Work

Proposed solution based on the global schedule produced significantly better optimisation results than queue-based solutions with acceptable run time of both the MTEDF dispatching rule and the tabu search. The algorithm demonstrates promising direction for new efficient local search based algorithms applicable for Grid scheduling. Also the Alea Simulator was developed and allowed us to implement, simulate and compare different algorithms among each other.

In the future we would like to optimise the local search algorithm to improve its performance and effectiveness. Another direction is to study large scale problems with many parameters, where the complexity of the problem will become very significant. Also, we would like to introduce network simulation, failure tolerance, preemptivity and job migration together with the use of larger and real workloads in our simulations to verify algorithm's performance. Moreover we plan to include proposed algorithms into the PBS scheduler (Feng, Misra, & Rubenstein 2007) and test them in the real Grid environment.

Acknowledgments

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research intent No. 0021622419, by the Grant Agency of the Czech Republic with grant No. 201/07/0205, and by the EU CoreGRID NoE (FP6-004265).

References

Armentano, V. A., and Yamashita, D. S. 2000. Tabu search for scheduling on identical parallel machines to minimize mean tardiness. *Journal of Intelligent Manufacturing* 11:453–460.

Baraglia, R.; Ferrini, R.; and Ritrovato, P. 2005. A static mapping heuristics to map parallel applications to heterogeneous computing systems: Research articles. *Concurrency and Computation: Practice and Experience* 17(13):1579–1605.

Bent, R., and Hentenryck, P. V. 2004. Online stochastic and robust optimization. In *Proceedings of the 9th Asian Computing Science Conference (ASIAN'04)*, volume 3321 of *Lecture Notes in Computer Science*, 286–300. Springer.

Buyya, R., and Murshed, M. 2002. GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. *The Journal of Concurrency and Computation: Practice and Experience (CCPE)* 14:1175–1220.

Capannini, G.; Baraglia, R.; Puppin, D.; Ricci, L.; and Pasquali, M. 2007. A job scheduling framework for large computing farms. In *SC07 International Conference for High Performance Computing, Networking, Storage and Analysis*. To appear.

Feng, H.; Misra, V.; and Rubenstein, D. 2007. PBS: a unified priority-based scheduler. *SIGMETRICS Performance Evaluation Review* 35(1):203–214.

Glover, F. W., and Kochenberger, G. A., eds. 2003. *Handbook of Metaheuristics*. Kluwer.

He, X.; Sun, X.; and von Laszewski, G. 2003. QoS guided min-min heuristic for Grid task scheduling. *Journal of Computer Science and Technology* 18(4):442–451.

Huedo, E.; Montero, R.; and Llorente, I. 2005. The GridWay framework for adaptive scheduling and execution on Grids. *Scalable Computing: Practice and Experience* 6(3):1–8.

Keller, A., and Reinefeld, A. 2001. Anatomy of a resource management system for HPC clusters. *Annual Review of Scalable Computing* 3.

Lifka, D. A. 1995. The ANL/IBM SP Scheduling System. In *IPPS '95: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, 295–303. London, UK: Springer-Verlag.

Pinedo, M. 2005. *Planning and Scheduling in Manufacturing and Services*. Springer.

Techiouba, A. D.; Capannini, G.; Baraglia, R.; Puppin, D.; and Pasquali, M. 2007. Backfilling strategies for scheduling streams of jobs on computational farms. CoreGRID Workshop on Grid Programming Model, Grid and P2P Systems Architecture, Grid Systems, Tools and Environments, Greece.

Thain, D.; Tannenbaum, T.; and Livny, M. 2005. Distributed computing in practice: the Condor experience. *Concurrency - Practice and Experience* 17(2-4):323–356.

Venugopal, S.; Buyya, R.; and Winton, L. 2004. A Grid Service Broker for scheduling distributed data-oriented applications on global Grids. In *MGC '04: Proceedings of the 2nd workshop on Middleware for grid computing*, 75–80. New York, NY, USA: ACM Press.

Xu, M. Q. 2001. Effective metacomputing using LSF multicluster. In *CCGRID '01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid*, 100. Washington, DC, USA: IEEE Computer Society.