

A SERVICE LEVEL AGREEMENT PROTOCOL AND TABU SEARCH FOR SCHEDULING ON COMPUTATIONAL GRIDS

¹D. Ouelhadj, ²J. M. Garibaldi, ³R. Sakellariou

^{1,2}School of Computer Science and IT, University of Nottingham, Jubilee Campus, Nottingham, NG8 1BB, UK, {¹dxs, ²jmg}@cs.nott.ac.uk

³University of Manchester, Oxford Road, Manchester, M13 9PL, UK, ³rizos@man.ac.uk

Abstract The efficient scheduling of independent jobs on distributed Grid resources is one of the most challenging issues in Grid computing. It is a complex problem since resources are geographically distributed having different usage policies and may exhibit highly non-uniform performance characteristics, heterogeneous in nature, and have varying loads and availability.

A recent funded project aims to establish a fundamental new infrastructure for efficient job scheduling on the Grid based on Service Level Agreements (SLAs) and using advanced scheduling techniques. For each job submitted, an SLA is negotiated between the user and the scheduler which contains information on the jobs to execute. In this paper, we propose a negotiation protocol based on the Contract Net Protocol to schedule the SLAs at the global level of the Grid, and the use of tabu search for an efficient scheduling of the SLAs at the local level of the Grid resources. Global scheduling is responsible for resource advertisement, resource discovery, and resource selection. Local scheduling is responsible for determining the order in which the SLAs are executed on the Grid resource.

Key words: Grid computing, Grid scheduling, Service Level Agreement, tabu search, batch scheduling heuristics, multi-agent systems, Contract Net Protocol, etc.

1. INTRODUCTION

Grid computing has emerged as a new paradigm for next generation distributed computing. A computational Grid is an infrastructure which enables the interconnection of substantial distributed computational resources in order to solve large scale problems in science, engineering, and commerce that cannot be solved on a single system [6]. Scheduling jobs on distributed resources is one of the most challenging issues in Grid computing. It is a complex problem as resources are geographically distributed having different usage policies and may exhibit highly non-uniform performance characteristics, heterogeneous in nature, and have varying loads and availability.

Scheduling jobs on heterogeneous compute resources while trying to minimise some cost function is an NP-complete problem [9]. Most of the Grid scheduling systems [11] such as Nimrod, Nimrod-G, Netsolve, AppLes, Ninf, Condor, Condor-G, LSF, Globus, Legion use simple batch scheduling heuristics, and the common scheduling objective is minimisation of the overall execution time or makespan. Various batch scheduling heuristics have been used [16, 8] including: FCFS, backfilling, list scheduling, smart, Min-min, Max-min, sufferage [2], Xsufferage [4], Qsufferage [15], etc. While many of the Grid scheduling systems use batch-based heuristics for scheduling, these systems do not, however, necessarily aim to find the best scheduling solution. There is little literature on the use of meta-heuristics to generate good quality schedules in Grid computing such as genetic algorithms [1, 3], simulated annealing [1, 17], and tabu search [1]. Jarvis et al. [10] compared FCFS and genetic algorithms to minimise the makespan, and it was found that batch

queuing might be regarded as a worst-case solution. Braun et al. [2] compared the performance of batch queuing heuristics, tabu search, genetic algorithms, and simulated annealing to minimise the makespan. The results showed that genetic algorithms achieved the best results, and the batch queuing heuristics achieved the worst results.

A recent funded project aims to establish a fundamental new infrastructure for efficient job scheduling on the Grid based on Service Level Agreements (SLAs) [12, 13] and using advanced scheduling techniques. For each job submitted, an SLA is negotiated between the user and the scheduler which contains information on the jobs to execute. In this paper, we propose a negotiation protocol to schedule the SLAs at the global level of the Grid, and the use of tabu search for an efficient scheduling of the SLAs at the local level of the Grid resources. Global scheduling is responsible for resource advertisement, resource discovery, and resource selection. Local scheduling is responsible for determining the order in which the SLAs are executed on the Grid resource. Section 2 presents the SLA based Grid scheduling multi-agent infrastructure. Section 3 describes the SLA negotiation protocol for scheduling SLAs at the global level of the Grid, and the scheduling model and the tabu search developed to schedule the SLAs at the local level of the Grid resource. Conclusions are presented in the section 4.

2. SLA BASED GRID SCHEDULING MULTI-AGENT INFRASTRUCTURE

The multi-agent infrastructure proposed [13] is organised as a population of cognitive, autonomous, and heterogeneous agents, for integrating a range of scheduling objectives related to Grid compute resources. Figure 1 shows the multi-agent infrastructure for SLA based Grid scheduling. The infrastructure involves three types of agents: User agents, Local Scheduler (LS) Agents, and Super Scheduler (SS) Agents. Three databases are also used in this architecture to store information on SLAs, LSA agents, and resources.

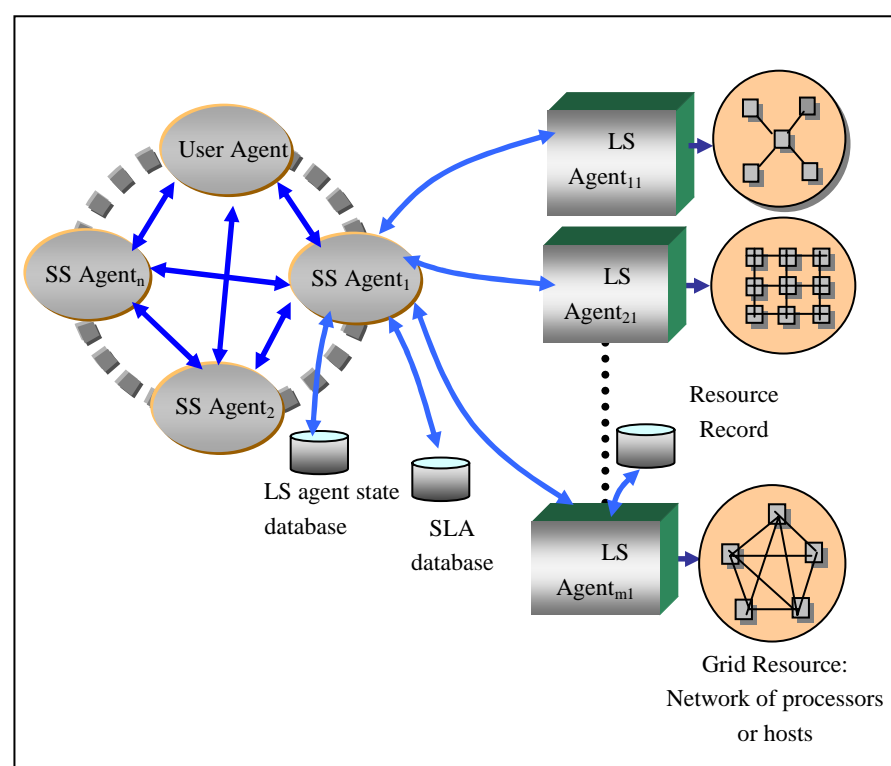


Figure 1. SLA based Grid scheduling system infrastructure

User Agent: The user agent requests the execution of SLAs on the Grid and negotiates SLAs with the SS agents. For each job submitted by the user, a SLA is initiated using information on the job supplied by the user and may be refined or additional information may be added in the final agreed SLA at the end of negotiation. The initial SLA submitted by the user describes usually a very high level description of a resource metric containing resource details such as machine information, the range of processors, the expected processing time, etc.

Local Scheduler Agent: Each Grid resource within each administrative domain is assigned to an LS agent. The set of Grid resources and their corresponding LS agents constitute a cluster. The LS agents are responsible for scheduling SLAs, usually assigned by the SS agents, at the local level of the Grid resources. However in practice, jobs are unlikely to be arriving from the SS agents, but they can be injected from other means such as PC which is locally connected to the machine that hosts the LS agent.

Super Scheduler Agent: SS agents are distributed in the Grid having at most one at every domain. They act as mediators between the user agent and the LS agents. Each cluster of LS agents of the same domain is coordinated by an SS agent. The user agent usually submits SLAs to the SS agents. SS agents negotiate SLAs with the user agent and the LS agents to discover and select a collection of Grid resources that can satisfy SLAs requirements.

Databases: The databases used in the architecture are the following:

- *Resource record:* holds information about the compute resources on which the jobs are executed by an LS agent.
- *SLA database:* stores the description of the SLAs.
- *LS agent state database:* provides information about the current availability and capability of Grid resources. This information is used to locate resources, to identify properties of the resources, and for numerous other purposes during the process of translating high-level resource specifications into requests to specific LS agents.

3. SLA SCHEDULING

Scheduling SLAs on the Grid involves two stages: global scheduling and local scheduling. Global scheduling discovers and selects the Grid resource to send the SLAs to. Once an SLA is assigned to run at a particular Grid resource, it is then managed by the LS agent at that Grid resource. The local scheduling decides how to schedule the SLAs on the local Grid resource.

3.1 GLOBAL SCHEDULING

Global scheduling is concerned with the discovery and selection of available Grid resources that provide the minimum tardiness and the SLA's compute resource requirements. The global scheduling is performed at the level of the SS agent using a negotiation protocol based on the Contract Net Protocol. The Contract Net Protocol is a high level protocol for achieving efficient cooperation in agent-based systems [5] introduced by Smith [14] based on a market-like protocol. In this protocol, agents exchange task announcements, bids, and awards for distributed task allocation. The SLA negotiation protocol allows the SS agents to negotiate with the user agent and the LS agents the schedule of SLAs on the distributed Grid resources.

The proposed SLA negotiation protocol is a hierarchical bidding mechanism involving two negotiation levels: Meta-SLA negotiation and sub-SLA negotiation.

1. Meta-SLA negotiation level: In this level the SS agents and the user agents negotiate meta-SLAs. A meta-SLA contains high-level description of jobs supplied by the user and may be refined or additional information may be added in the final agreed SLA at the end of negotiation. The negotiation steps are the following:

- a₁. Meta-SLA-request:** The user agent submits a SLA to the nearest SS agent to request the execution of a meta-SLA. The meta-SLA at this stage describes baseline user requirements such as required resources over a time period, expected processing time, job start and end time, etc.
- a₂. Meta-SLA-announcement:** Upon receiving the meta-SLA-request from the user agent, the SS agent advertises the meta-SLA-announcement to the neighbouring SS agents in the net.
- a₃. Meta-SLA-bidding:** In response to the meta-SLA-announcement, each SS agent queries the LS agent state database to check the availability of required resources for the job response time to identify suitable set of resources. The SS agents select the best resources which satisfy the earliest execution time and the SLA compute resource requirements. Each SS agent submits a meta-SLA-bid to the user agent. The meta-SLA-bid describes the new estimated response time and the compute resources required.
- a₄. Meta-SLA-award:** The user agent, upon receiving the meta-SLA-bids, selects the SS agent with the best meta-SLA-bid with the earliest execution time and sends a notification agreement to it. The selected SS agent stores the agreed SLA in the SLA database. The user has now an agreement with the Grid provider to use its resources.

2. Sub-SLA negotiation level: in this level the SS agents negotiate sub-SLA with the LS agents. The SS agents decompose the meta-SLA into its low level resource attributes, sub-SLAs which contain low level raw resource description such as processes, memory processors, architecture, memory, disk size, CPU, network, etc. The sub-SLA negotiation steps are the following:

- b₁. Sub-SLA-announcement:** The SS agent sends the sub-SLA-announcements to the identified suitable LS agents within the cluster under its control.
- b₂. Sub-SLA-bidding:** The LS agents, upon receiving the sub-SLA-announcements, query the resource record for information on CPU speed, memory size, etc. to identify suitable resources and submit bids to the SS agent.
- b₃. Sub-SLA-award:** The SS agent evaluates the best sub-SLA in accordance to execution time and resource/requirement match and sends a notification agreement to the LS agent with the best bid. The selected LS agents update the resource record and reserve the resources to execute the job.

3.2 LOCAL SCHEDULING USING TABU SEARCH

The local scheduling is performed at the level of the LS agent which schedules the SLAs on the local Grid resource. A Grid resource is considered to be a set of processors or a cluster of workstations.

Consider a Grid resource with N hosts (processors). At a certain time, there is a set of SLAs denoted by S that wait to be scheduled to the suitable hosts. For each SLA in S , we assume an estimation of the execution time is provided by the user.

The scheduling problem is defined by the allocation of a set of independent SLAs, $S = \{SLA_1, \dots, SLA_s\}$ to a network of hosts $H = \{H_1, \dots, H_n\}$.

The expected execution time τ_{ij} of SLA_i on host H_j is defined as the amount of time taken by H_j to execute SLA_i .

The earliest possible start time ST_i of SLA_i .

The latest finish time FT_i of SLA_i .

The expected completion time C_i of SLA_i on host H_j is defined as the time at which H_j completes SLA_i .

The completion time of SLA_i is defined by $C_i = ST_i + \tau_{ij}$

The lateness of SLA_i is defined by $L_i = \max \{0, C_i - FT_i\}$

The maximum lateness is defined as:

$$L_{max} = \max_{1 < i < S} (L_i)$$

The objective function is to minimise the maximum lateness:

$$\min L_{max}$$

The Grid scheduling problem is an NP-hard optimisation problem. To solve this combinatorial optimisation-scheduling problem, we propose to use a tabu search method [7] with short-term memory. To generate the initial solution, we propose to use the following batch queuing heuristics: FCFS, Min-min, Max-min, sufferage, and backfilling. Then the solution is improved by using two moves: SLA-swap and SLA-transfer moves. The SLA-swap move swaps two SLAs performed by different processors. SLA-transfer move shifts the SLA to another processor. The search evaluates all the possible moves, and then selects the best move to apply, even if the move yields a worse objective value than the current one until the stopping criterion is reached. Tabu search makes use of a tabu list, where it stores all applied moves in order to avoid cycles, by forbidding a move which reverses a recent one. In practice, each tabu move is initially given a maximum count number, which is the chosen number of forbidden iterations. Then, each time the list is updated the count number of each tabu move is decreased by 1, and on reaching zero, the move is removed from the tabu list, thus becoming valid. When the best move is selected, the tabu list is checked to find out if the move exists. If the best move is not tabu, it is applied to the current solution and then added to the list. If the best move is tabu, then it cannot be applied unless the aspiration criterion holds when the resulting objective value is higher than that of the best solution so far seen. If the criterion fails, then the best move is discarded, the next best move is considered, and the process is repeated.

4. DISCUSSION

In this paper we have described an SLA negotiation protocol for the scheduling of the SLAs at the global level of the Grid. For an efficient schedule of the SLAs to minimise the lateness of SLAs at the local level of the Grid resource, we propose to use tabu search. The proposed novel scheduling methodology based on SLA negotiation protocol and tabu search offer advanced degree of efficiency and flexibility in the negotiation of the SLAs and their scheduling to satisfy user requests. Experiments will be conducted to compare the performance of the schedules generated using simple batch queuing heuristics and the scheduling model using the SLA negotiation protocol and tabu search.

ACKNOWLEDGEMENTS

This work is funded by the EPSRC Fundamental Computer Science for e-Science initiative (Grants GR/S67654/01 and GR/S67661/01), whose support we are pleased to acknowledge.

5. REFERENCES

1. Abraham, A., Buyya, R., Nath, B. (2000) Nature's heuristics for scheduling jobs on computational Grids. In the 8th International Conference on Advanced Computing and Communications, Cochin, India.
2. Braun, T., Siegel, H. J., Beck, N., Hensgen, D., Freund, R. F. (2001) A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, 61, 810-837.
3. Cao, J., Jarvis, S. (2002) ARMS: An agent-based resource management system for Grid computing. *Scientific Programming*. 10, 135-148.
4. Casanova, H., Obertelli, G. (2000) The AppLes parameter sweep template: user level middleware for the Grid. In the Proceedings of the High Performance Networking and Computing of the 2000 ACM/IEEE conference on Supercomputing, Dallas, Texas, United States.
5. Ferber, J. (ed.): *Multi-agent systems: An introduction to Distributed Artificial Intelligence*. Addison-Wesley, London (1999).
6. Foster, I. and Kesselman, C. (eds.) (1998) *The Grid: Blueprint for a new computing infrastructure*. Morgan Kaufman Publishers (1998).
7. Glover, F. and Laguna, M. (1997) *Tabu search*. Kluwer Academic Publishers, Boston.
8. Hamscher, V., Schwiigelshohn, U., Streit, A., Yahyapour, R. (2000) Evaluation of job scheduling strategies for Grid computing. *Lecture Notes in Computer Science*, pp. 191-202.
9. Ibarra, O. H. and Kim, C. E. (1977) Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of ACM*, 24 (2), 280-289.
10. Jarvis, S. A., Spooner, D. P., Lim Choi Keung, H. N., Nudd, G. R., Cao, J., Saini, S. (2003) Performance Prediction and its use in parallel and distributed computing systems. In the Proceedings of the IEEE/ACM International Workshop on Performance Modelling, Evaluation and Optimization of Parallel and Distributed Systems, Nice, France.
11. Krauter, K., Buyya, R., Maheswaran, M. (2002) A taxonomy and survey of Grid resource management systems. 32(2), 135-164.
12. MacLaren, J., Sakellariou, R., Garibaldi, J., Ouelhadj, D. (2004) Towards service level agreement based scheduling on the Grid. In the Proceedings of the Workshop on Planning and Scheduling for Web and Grid Services, in the 14th International Conference on Automated Planning & Scheduling, Whistler, Canada, pp. 100-102.
13. Ouelhadj, D., Garibaldi, J., MacLaren, J., Sakellariou, R., Krishnakumar, K. (2005) A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in Grid Computing. Accepted for Publication in the Proceedings of the European Grid Conference, *Lecture Notes in Computer Science*, Springer-Verlag.
14. Smith, R. (1980) The contract net protocol: high level communication and control in distributed problem solver. *IEEE Transactions on Computers*, 29, 1104-1113.
15. Weng, C. and Lu, X. (2003) Heuristic scheduling for bag-of-tasks applications in combination with QoS in the computational Grid. *Future Generation Computer Systems*, 21(2), 271-280.
16. Yahyapour, R. (2003) Design and evaluation of job scheduling strategies for Grid Computing, PhD thesis.
17. Yarkhan, A. Dongarra, J. J. (2002) Experiments with scheduling using simulated annealing in a Grid environment. *Lecture Notes In Computer Science, Proceedings of the Third International Workshop on Grid Computing*, pp. 232 – 242.