Optimizing the execution time of the SLA-based workflow in the Grid with parallel processing technology

Dang Minh Quan & Jörn Altmann
International University in Germany
School of Information Technology
Bruchsal 76646, Germany
quandm@upb.de & jorn.altmann@acm.org

Laurence T. Yang
St. Francis Xavier University
Department of Computer Science
Antigonish, NS, B2G 2W5, Canada
ltyang@stfx.ca

Abstract

Service Level Agreements (SLAs) is currently one of the major research topics in Grid Computing. Among many system components for the supporting of SLA-aware Grid-based workflow, the SLA mapping module receives important positions. Optimizing execution time is an important task of the mapping module as it helps in finding out a feasible solution when the Grid is busy or by eliminating the negative effects of the error. With the previously proposed algorithm for optimizing the execution time of the workflow, the mapping module may, when many requests come in a short period of time, become the bottleneck of the system. This paper presents a parallel mapping algorithm for optimizing the execution time of the workflow, which can cope with the problem. Performance measurements deliver evaluation results on the quality of the method.

1. Introduction

In the Grid Computing environment, many users need the results of their calculations within a specific period of time. Examples of those users are weather forecasters running weather forecasting workflows or automobile producers running dynamic fluid simulation workflow [1]. Those users are willing to pay for getting their work completed on time. However, this requirement must be agreed on by both, the users and the Grid provider before the application is executed. This agreement is kept in the Service Level Agreement (SLA). SLAs specify the a-priori negotiated resource requirements, the quality of service (QoS), and costs. The application of such an SLA represents a legally binding contract and is a mandatory prerequisite for the Next Generation Grids.

To free users from the complicated task of managing the workflow execution, it is necessary to introduce a broker handling the task for the user. We proposed a business model [3] for the system as depicted in Figure 1.

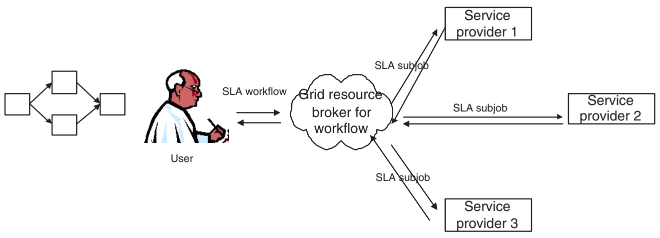

Figure 1. Stakeholders and their business relationship

The SLA workflow broker represents the user as specified in the SLA with the user and controls the workflow execution. This includes the mapping of sub-jobs to resources, signing SLAs with the services providers, monitoring, and error recovery. When the workflow execution has finished, it settles the accounts. It pays the service providers and charges the end-user. The profit of the broker is the difference. The value-added that the broker provides is the handling of all the tasks for the end-user.

We presented a prototype system supporting SLAs for the Grid-based workflow in [2, 4, 3]. In the system handling the SLA-based workflow, the mapping module receives an important position. Mapping a Grid-based workflow within the SLA context usually deals with both optimizing the cost and satisfying the deadline [2]. However, the requirement of optimizing the runtime emerges in the error recovery phase [4] or in finding a feasible solution in a busy Grid.

In the previous work [4], we proposed the algorithm w-Tabu to optimize the execution time of the workflow in the Grid environment. The extensive experiment result shows that the runtime of the w-Tabu algorithm is from 1 to 15 seconds, depending on the Grid resource and the size of the workflow. Thus, when the Grid is busy and there are many users waiting for service from the SLA broker, some may have to wait several minutes to know whether their workflow can be run on time or not. With a large and crowded Grid, this situation may occur frequently and result in a negative effect for the SLA broker. Thus, reducing the runtime of the mapping algorithm while maintaining the quality of the mapping solution is an essential requirement. Moreover, reducing the runtime of the mapping algorithm can also reduce the error recovery reaction time and thus, make the recovery mechanism more efficient.

In this paper, we propose a parallel mapping algorithm based on the w-Tabu algorithm to cope with this problem. The parallel algorithm, called the pw-Tabu, will reduce the time for optimizing the execution time of the workflow to Grid resources without decreasing the quality of the solution.

The paper is organized as follows. Sections 2 and 3 describe the problem and the related works respectively. Section 4 presents the algorithm. The experiment about the quality and the applicability of the proposed algorithm is discussed in section 5. Section 6 concludes the paper with a short summary.

2 Problem statement

2.1 Grid-based workflow model

In our system, a Grid-based workflow concentrates on intensive computation and data analyzing. Like many popular systems handling Grid-based workflows [5, 1], our system is of the Directed Acyclic Graph (DAG) form. The user specifies the required resources needed to run each sub-job, the data transfer between sub-jobs, the estimated runtime of each sub-job, and the expected runtime of the whole workflow. It is noted that the data to be transferred between subjobs can be very large. Figure 2 presents a concrete example of a Grid workflow.

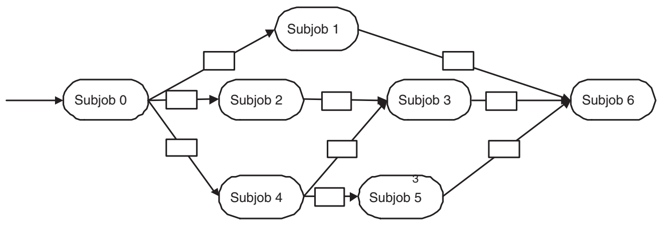

Figure 2. A sample workflow

2.2 Grid service model

The computational Grid includes many High Performance Computing Centers (HPCCs). The resources of each HPCC are managed by a software called local Resource Management System (RMS)[1]. Each RMS has its own unique resource configuration comprising the number of CPUs, the amount of memory, the storage capacity, the software, the number of experts, and the service price. To ensure that the sub-job can be executed within a dedicated time period, the RMS must support advanced resource reservation such as CCS [6]. In our model, we reserve three main types of resources: CPU, storage, and expert. The addition of further resources is straightforward.

If two output-input-dependent sub-jobs are executed on the same RMS, it is assumed that the time required for the data transfer equals zero. This can be assumed since all computing nodes in a cluster usually use a shared storage system such as NFS or DFS. In all other cases, it is assumed that a specific amount of data will be transferred within a specific period of time, and thus requiring the reservation of bandwidth [4].

2.3 Problem specification

The formal specification of the described problem includes following elements:

- Let R be the set of Grid RMSs.
- Let S be the set of sub-jobs in a given workflow.
- Let E be the set of edges in the workflow.
- Let K_i be the set of resource candidates of sub-job s_i. This set includes all RMSs, which can run sub-job s_i, $K_i \subset R$.

Based on the given input, a feasible and possibly optimal solution is sought allowing the most efficient mapping of the workflow in a Grid environment with respect to the given global deadline. The required solution is a set defined in Formula 1. Sub-job s_i run on RMS r_j during the period $start, stop$.

$$M = \{(s_i, r_j, start, stop) | s_i \in S, r_j \in K_i\} \quad (1)$$

If the solution does not have start_slot for each s_i, it becomes a configuration as defined in Formula 2.

$$a = \{(s_i, r_j) | s_i \in S, r_j \in K_i\} \quad (2)$$

A feasible solution must satisfy following conditions:

- **Criterion 1:** All $K_i \neq \emptyset$. There is at least one RMS in the candidate set of each sub-job.

[1]In this paper, RMS is used to represent the cluster/super computer as well as the Grid service provided by the HPCC.

- **Criterion 2:** The dependencies of the sub-jobs are resolved and the execution order remains unchanged.
- **Criterion 3:** The capacity of an RMS must be equal to or be greater than the requirement at any time slot.
- **Criterion 4:** The data transmission task e_{ki} from sub-job s_k to sub-job s_i must take place in a dedication time slots on the link between the RMS running sub-job s_k to the RMS running sub-job s_i. $e_{ki} \in E$.

The goal here is minimizing the execution time of the workflow. We define the execution time of a workflow as the period from the user's preferred start time to the finished time. Supposing that the Grid system has m RMSs, which can satisfy the requirement of n sub-jobs in a workflow. As an RMS can run several sub-jobs at a time, finding out the optimal solution needs (m^n) loops. It can be easily shown that the optimal mapping of the workflow to the Grid RMS as described above is an NP hard problem.

3. Related works

Mapping algorithms for Grid workflow receives a lot of attention from the scientific society. In the literature, there are many methods for mapping a Grid workflow to Grid resource within different contexts. Among those, the old but well-known algorithm Condor-DAGMan from the work of [7] is still used in some present Grid systems. This algorithm makes local decisions about which job to send to which resource and considers only jobs which are ready to run at any given instance. Also using dynamic scheduling approach, [8] and [9] apply many techniques to frequently rearrange the workflow and reschedule the workflow in order to reduce the runtime of the workflow. Those methods are not suitable for the context of resource reservation because whenever a reservation is canceled, a fee has to be paid. Thus, frequent rescheduling may lead to a much higher running workflow cost.

[5] presented an algorithm, which maps Grid workflows onto Grid resources based on existing planning technology. This work focuses on coding the problem to be compatible with the input format of specific planning systems and thus transferring the mapping problem to a planning problem. Although this is a flexible way to gain different destinations, which include some SLA criteria, significant disadvantages regarding the time-intensive computation, long response times and the missing consideration of Grid-specific constraints appeared.

In recent works, [10] proposed the x-DCP algorithm. [11] proposed minmin, maxmin and suffer algorithms. [12] proposed the GRASP algorithm. All of those algorithms concentrate on scheduling the workflow with parameter sweep tasks on Grid resources. The common destination of those algorithms is optimizing the makespan. Subtasks in this kind of workflow can be grouped in layers and there is no dependency among subtasks in the same layer. All proposed algorithms assume each task as a sequence program and each resource as a compute node. By using several heuristics, all those algorithms do the mapping very fast. Our workflow with the DAG form can also be transformed to the workflow with parameter sweep tasks type. We applied all those algorithms to our problem. The experiment results show that the quality of solutions found by those algorithms are not sufficient [4].

In the previous work [4], we proposed the w-Tabu algorithm. The overview of w-Tabu algorithm is presented in Algorithm 1.

Algorithm 1 w-Tabu algorithm

```
1: Determine assignning sequence for all sub-jobs of the
   workflow
2: Generate reference solution set
3: for all solutions in reference set do
4:     Improve the solution as far as possible with the mod-
       ified Tabu search
5: end for
6: Pick the solution with best result
```

The assigning sequence is based on the latest start_time of the sub-job. Sub-jobs having a smaller latest start time will be assigned earlier. Each solution in the reference solutions set can be thought of as the starting point for the local search so it should be spread as widely as possible in the searching space. To satisfy the space spread requirement, the number of similar map $sub-job : RMS$ between two solutions, must be as small as possible. The improvement procedure based on the Tabu search has some specific techniques to reduce the computation time. More information about the w-Tabu algorithm can be seen in [4].

4. pw-Tabu algorithm

We describe here a parallel algorithm based on the w-Tabu algorithm called pw-Tabu. The architecture of this algorithm framework is presented in Figure 3.

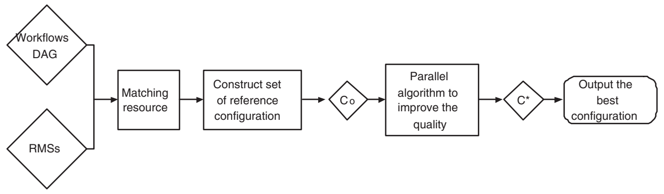
Workflows DAG
RMSs
Matching resource
Construct set of reference configuration
Co
Parallel algorithm to improve the quality
C*
Output the best configuration

Figure 3. The architecture of the algorithm framework

The inputs of the algorithm are the workflow and the Grid resources. After building the configuration space by matching the sub-job's resource requirement and the RMS's resource configuration, the set C_o of initial solutions is created. Then, a parallel algorithm will improve the quality of each initial solution as much as possible. The best solution is the output.

The procedures of matching resource and forming the set of reference configurations are similar to the w-Tabu algorithm. In particular, the matching between the sub-job's resource requirement and the RMS's resource configuration is done by several logic-checking conditions in the WHERE clause of the SQL SELECT command. The matched RMSs for sub-jobs form a search space. The configuration in the reference set should be spread as widely as possible in the searching space. To satisfy the space spread requirement, the number of similar map sub-jobs: RMS between two solutions, must be as small as possible. A detailed description about the procedures can be seen in [4].

4.1 Parallel algorithm to improve the quality

The task of improving the quality of the initial configuration set is divided into many subtasks. Those subtasks are processed in parallel. The algorithm follows the conventional master-slave model as depicted in Figure 4.

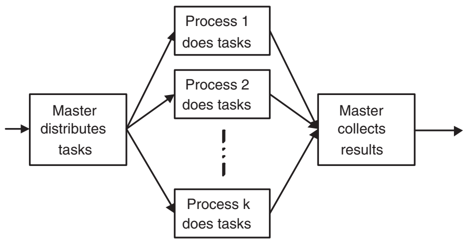

Figure 4. Working model of the improving quality algorithm

To improve the quality of solutions in the initial set C_o, the master process evenly distributes configurations in the set C_o to the slave processes. The data sending from the master process to the slave process is presented in Figure 5. The first field denotes the number of configurations in the message. Each configuration has its number of sub-jobs and list of RMSs for sub-jobs in the workflow.

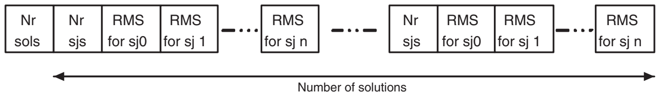

Figure 5. Data format of the improving solutions command

Each slave process improves the quality of each initial configuration by using local search. The procedure tries to replace the present RMS with other RMSs in the candidate list to find the best improvement. To reduce the computing time, we focus on the critical path of the workflow. The process continues until the solution cannot be improved any more. A detailed description about the procedure is presented in [4].

When the improvement is finished, each slave process sends the master only the best found solution with the message presented in Figure 6. It is similar to the one in Figure 5 except that each solution has its cost field.

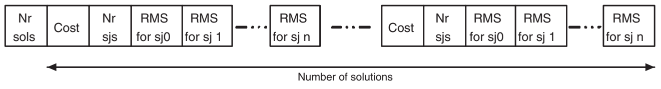

Figure 6. Data format of the replying messages

The master picks the best solution and output.

4.2 Algorithm implementation

The implementation of the master and slave process is presented in Algorithm 2 and Algorithm 3.

Algorithm 2 Master process

```
1: Get information of workflow and Grid resources
2: Create solution space
3: Create the reference set of configurations
4: Distribute the task of improving initial configurations
   to slave processes
5: Collect the improved solutions
6: Pick the best solution
7: Send kill signal to slave process
```

Algorithm 3 Slave process

```
1: Get information of workflow and Grid resources
2: Create solution space
3: When receiving the task of improving the initial solu-
   tion then do it and send back the result to the master
4: When receiving the kill signal, exit
```

We can see that all master and slave processes have the full information about the workflow and the resources. Thus, the data transfer among processes is reduced. The algorithm is implemented using MPI. From the described algorithm architecture and implementation, the following observation can be made. Firstly, the main strategy of the pw-Tabu algorithm still remains as the w-Tabu algorithm. Thus, the quality of the algorithm is kept. Only the computation intensive parts are parallelized to improve the execution time of the mapping module. Secondly, as the size of the reference solution set is limited, the scalability of the pw-Tabu algorithm is also limited. In particular, the maximum effective number of computing nodes equals the size of the reference solution set.

5. Performance experiment and applicability

As the quality of the algorithm is unchanged, the performance experiment is done with simulation to check for the runtime of the mapping algorithms. The software used in the experiments is rather standard and simple (Linux Ubuntu 7.0, MySQL). The whole simulation program is implemented in C/C++. The hardware for the experiment is a cluster including 8 computing nodes 3,0 Ghz, 1GB memory. 8 computing nodes are connected through switch 100 Mbps.

The goal of the experiment is to measure the time needed for the computation. To do the experiment, 18 workflows with different topologies, number of sub-jobs, sub-job specifications and amount of data transferring were generated as workload. The Grid resources includes 20 RMSs with different resource configurations and different resource reservation contexts. Detailed information about the workload information and resource information can be seen in [4].

In the first experiment, we study the runtime of the algorithm for 18 single workflows with different number of computing nodes. Each computing node run one slave process. With the case of one computing node, we use the w-Tabu algorithm. With more than one computing nodes, we use the pw-Tabu algorithm. As the time entity in our experiment is second, the smallest runtime of the algorithm is one second. The result is presented in Table 1. The first column presents the number of sub-jobs in a workflow. Other columns present the runtime of the mapping algorithm according to the different number of computing nodes.

From the data in Table 1, we can see that the increase in performance of the pw-Tabu algorithm with small workflows is not as clear as with large workflows. One reason for this is that the 1 second resolution is not fine enough for small workflows which already need a small runtime of the w-Tabu algorithm. Another reason is that the rate between the overhead and the main computing part of the algorithm with small workflows is bigger than with large workflows. Thus, the parallel part applying to small workflows brings less effect.

With the large workflow as in the advance level experiment, the character of the pw-Tabu algorithm can be seen more clearly. The runtime of the algorithm is not reduced linearly with the increasing computing nodes. This is because of the overhead and the communication of the parallel processes. When the number of parallel process increases, the overhead and communication increase. Thus, they reduce the speedup of the algorithm.

Sjs	1 CPU	2	3	4	5	6	7	8
Simple level experiment								
7	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1
10	2	2	1	1	1	1	1	1
11	2	2	2	2	2	1	1	1
12	3	2	2	2	2	1	1	1
13	2	2	2	2	2	1	1	1
Intermediate level experiment								
14	2	2	2	2	2	2	2	1
15	3	3	3	2	1	2	2	1
16	4	4	2	2	2	2	2	1
17	4	4	3	3	2	2	2	1
18	5	5	4	4	2	2	2	1
19	6	6	4	4	2	2	2	2
Advance level experiment								
20	5	5	3	4	3	2	2	2
21	5	5	4	4	3	2	2	2
25	10	9	7	7	5	4	4	3
28	12	11	9	8	6	5	4	3
32	15	14	12	10	9	7	5	4

Table 1. Performance result of mapping 18 single workflows

To study more carefully the performance of the pw-Tabu algorithm, we do the second experiment with mixed workload. To do the experiment, we generate 100 requests and each request is selected randomly from 18 workflows. Then, we continuously map the 100 requests with different number of computing nodes and record the necessary time to finish the mapping. In the case of one computing node, we use the w-Tabu algorithm. With more than one computing nodes, we use the pw-Tabu algorithm. The result is presented in Figure 7.

From Figure 7, we can see the same trend as the above experiment that the speedup of the algorithm is reduced when the number of computing nodes increases. We can also see that the runtime of the algorithm has minor changes when the increase in number of computing nodes is not enough. This is because the workload of the heaviest computing nodes is unchanged. For example, in the case of 5 and 6 CPUs in the experiment, the total number of initial solutions is 25 and thus, the heaviest process in both cases must handle 5 initial solutions.

As can be seen in Figure 7, the broker needs an average of 6 and 2 seconds for a request with 1 CPU and 8 CPUs respectively. This means that the brokers can serve the users 300% faster with 8 CPUs compared to 1 CPU. Beside that, with the business Grid, the broker could easily have flexible computing power. He could hire many computing nodes

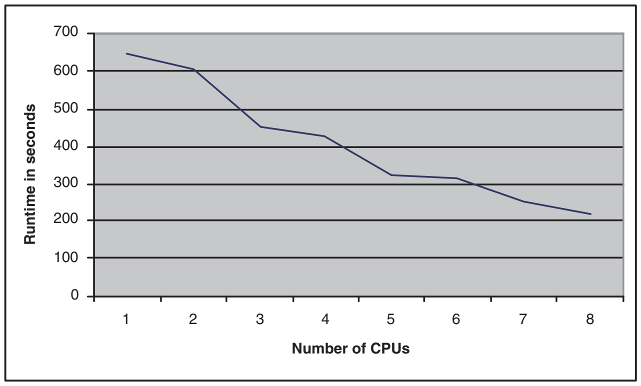

Figure 7. Performance result of mapping with mixed workload

in the critical period and return them when the Grid is not crowded. Compared to the income of the broker, the cost of hiring more computers for mapping is very small. For example, with Amazon pricing schema (13-3-2008), a computing node costs 0,10 $ per hour. Thus, hiring 8 CPUs in 1 hour costs only 0,8 $. This means that the applicability of the approach is very high. By applying parallel processing technology, the broker can increase significantly its ability to serve users with low cost.

6. Conclusion

This paper has presented a method, which reduces the necessary time to optimize the execution time of SLA-based workflows in the Grid environment. In particular, we proposed a parallel algorithm pw-Tabu which is based on the w-Tabu algorithm. The main strategy of the w-Tabu algorithm is still maintained while the computing intensive parts are parallelized. Thus, the quality of the algorithm is kept while the runtime is reduced significantly. The performance evaluation showed that the algorithm is very effective, especially with large sized workflows which require great computation power. On average, the algorithm can speed up to 300% with 8 CPUs. With the low cost of hiring computing resources, the method can be applied without difficulty in real environments.

References

[1] Lovas, R., Dzsa, G., Kacsuk, P., Podhorszki, N., Drtos, D. (2004) 'Workflow Support for Complex Grid Applications: Integrated and Portal Solutions', *Proceedings of 2nd European Across Grids Conference*, pp.129-138.

[2] Quan, D.M., Kao, O. (2005) 'On Architecture for an SLA-aware Job Flows in Grid Environments', *Journal of Interconnection Networks*, Vol. 6, No. 3, pp.245-264.

[3] Quan, D.M., Altmann, J. (2007) 'Business Model and the Policy of Mapping Light Communication Grid-Based Workflow Within the SLA Context', *Proceedings of the International Conference of High Performance Computing and Communication (HPCC07)*, pp.285-295.

[4] Quan, D.M. (2007) 'Error recovery mechanism for grid-based workflow within SLA context', *Int. J. High Performance Computing and Networking*, Vol. 5, No. 1/2, pp.110-121.

[5] Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M., Vahi, K. and Livny, M. (2004) 'Pegasus : Mapping Scientific Workflows onto the Grid', *Proceedings of the 2nd European Across Grids Conference*, pp.11-20.

[6] Hovestadt, M. (2003) 'Scheduling in HPC Resource Management Systems:Queuing vs. Planning', *Proceedings of the 9th Workshop on JSSPP at GGF8*, LNCS, pp.1-20.

[7] CondorVersion 6.4.7 Manual. www.cs.wisc.edu/condor/manual/v6.4 [10 December 2004].

[8] Duan, R., Prodan, R., Fahringer, T. (2006) 'Run-time Optimization for Grid Workflow Applications', *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing (Grid'06)*, pp. 33-40.

[9] Ayyub, S. and Abramson, D. (2007) 'GridRod - A Service Oriented Dynamic Runtime Scheduler for Grid Workflows'. *Proceedings of the 21st ACM International Conference on Supercomputing*, pp. 43-52.

[10] Ma, T. and Buyya, R. 2005 'Critical-Path and Priority based Algorithms for Scheduling Workflows with Parameter Sweep Tasks on Global Grids', *Proceeding of the 17th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2005), IEEE CS Press*, pp.251-258.

[11] Casanova, H., Legrand, A., Zagorodnov, D. and Berman, F. 2000 'Heuristics for Scheduling Parameter Sweep applications in Grid environments', *Proceeding of the 9th HeterogeneousComputing workshop (HCW'2000)*, pp.292–300.

[12] Blythe *et al*. 2005 'Task Scheduling Strategies for Workflow-based Applications in Grids', *Proceeding of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005)*, pp.759-767.