

Mapping light communication SLA-based workflows onto Grid resources 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) are currently one of the major research topics in Grid Computing. Among many system components for supporting of SLA-aware Grid-based workflows, the SLA mapping module receives an important position. Mapping light communication workflows is one main part of the mapping module. With the previously proposed mapping algorithm, the mapping module may become the bottleneck of the system when many requests come in a short period of time. This paper presents a parallel mapping algorithm for light communication SLA-based workflows, 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 forecaster running weather forecasting workflows, automobile producer 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) [2]. In general, SLAs are defined as an explicit statement of expectations and obligations in a business relationship between service providers and customers. 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. This is a mandatory prerequisite for the Next Generation Grids. We presented a prototype system supporting SLAs for the Grid-based workflow in [3, 7, 6].

1.1 Grid-based workflow model

In our system, a Grid-based workflow concentrates on intensive computation and data analyzing. A Grid-based workflow is characterized by the following features [8]:

- A Grid-based workflow usually includes many sub-jobs (i.e. applications), which perform data analysis tasks. However, those sub-jobs are not executed freely but in a strict sequence.
- A sub-job in a Grid-based workflow depends tightly on the output data from previous sub-jobs. With incorrect input data, a sub-job will produce wrong results and damage the result of the whole workflow.
- Sub-jobs in the Grid-based workflow are usually computationally intensive. They can be sequential or parallel programs and require a long runtime.
- Grid-based workflows usually require powerful computing facilities (e.g. super-computers or clusters) to run on.

Like many popular systems handling Grid-based workflows [9, 10, 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 sub-jobs is very small. Figure 1 presents a concrete example Grid workflow.

1.2 Grid service model

The computational Grid includes many High Performance Computing Centers (HPCCs). The resources of

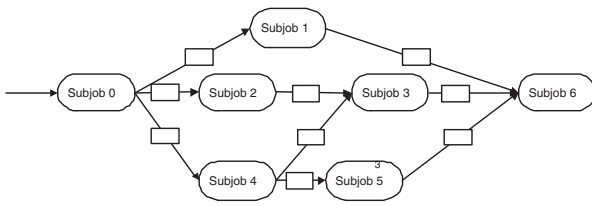


Figure 1. A sample workflow

each HPCC are managed by a software called local Resource Management System (RMS)¹. Each RMS has its own unique resource configuration. A resource configuration comprises 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 advance resource reservation such as CCS [11]. 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 compute nodes in a cluster usually use a shared storage system like 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, requiring the reservation of bandwidth [7].

1.3 Business model

In the case of Grid-based workflow, letting users work directly with resource providers has two main disadvantages:

- The user has to have sophisticated resource discovery and mapping tools in order to find the appropriate resource providers.
- The user has to manage the workflow, ranging from monitoring the running process to handling error events.

To free users from this kind of work, it is necessary to introduce a broker handling the workflow execution for the user. We proposed a business model [6] for the system as depicted in Figure 2. There are three main entities: the end-user, the SLA broker and the service provider:

The end-user wants to run a workflow within a specific period of time. The user asks the broker to execute the workflow for him and pays the broker for the workflow execution service. The user does not need to know in detail

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

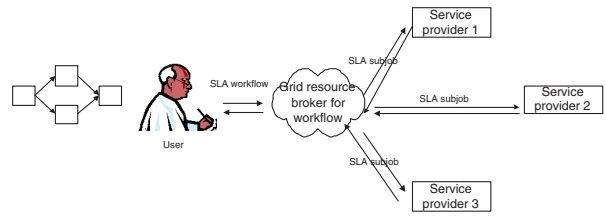


Figure 2. Stakeholders and their business relationship

how much he has to pay to each service provider. He only needs to know the total amount. This amount depends on the urgency of the workflow and the budget of the user. If there is a SLA violation, for example the runtime deadline has not been met, the user will ask the broker for compensation. This compensation is clearly defined in the Service Level Objectives (SLOs) of the SLA.

The SLA workflow broker represents the user as specified in the SLA with the user. It controls the workflow execution. This includes 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-add that the broker provides is the handling of all the tasks for the end-user.

The service providers execute the sub-jobs of the workflow. In our business model, we assume that each service provider fixes the price for its resources at the time of the SLA negotiation. As the resources of a HPCC usually have the same configuration and quality, each service provider has a fixed policy for compensation if its resources fail. For example, such a policy could be that $n\%$ of the cost will be compensated if the sub-job is delayed one time slot.

1.4 Problem statement

The formal specification of the described problem includes following elements:

- Let R be the set of Grid RMSs. This set includes a finite number of RMSs, which provide static information about controlled resources and the current reservations/assignments.
- Let S be the set of sub-jobs in a given workflow including all sub-jobs with the current resource and deadline requirements.
- Let E be the set of data transfer in the workflow, which express the dependency between the sub-jobs and the necessity for data transfers between the sub-jobs.

- 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, which allows 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 as Formula 1.

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

If the solution does not have start_slot for each s_i , it become 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:

- **Criterion1:** All $K_i \neq \emptyset$. There is at least one RMS in the candidate set of each sub-job.
- **Criterion2:** The total runtime period of the workflow must be within the expected period given by user.
- **Criterion3:** The dependencies of the sub-jobs are resolved and the execution order remains unchanged.
- **Criterion4:** Each RMS provides a profile of currently available resources and can run many sub-jobs of a single flow both sequentially and parallel. Those sub-jobs, which run on the same RMS, form a profile of resource requirement. With each RMS r_j running sub-jobs of the Grid workflow, with each time slot in the profile of available resources and profile of resource requirements, the number of available resources must be larger than the resource requirement.

Because the number of data transfer among sub-jobs of the workflow is very small, we can omit the cost of data transfer. It is also not necessary to reserve the bandwidth. Thus, the cost of running a workflow is a sum of three factors: money for using CPU, money for using storage, cost of using experts knowledge as in Formula 1.

$$C = \sum_{i=1}^n s_i \cdot r_t * (s_i \cdot n_c * r_j \cdot p_c + s_i \cdot n_s * r_j \cdot p_s + s_i \cdot n_e * r_j \cdot p_e) \quad (3)$$

with $s_i \cdot r_t$, $s_i \cdot n_c$, $s_i \cdot n_s$, $s_i \cdot n_e$ are the runtime, number CPU, number storage, number expert of sub-job s_i respectively. $r_j \cdot p_c$, $r_j \cdot p_s$, $r_j \cdot p_e$ are the price of using CPU, storage, expert of RMS r_j respectively.

It can be shown easily that the optimal mapping of the workflow to Grid RMS with cost optimizing is a NP hard problem.

In the previous work [4], we proposed the algorithm L-Map to map light communication workflow to the Grid resources. The extensive experiment result shows that the runtime of the L-Map algorithm is from 1 to 10 seconds depending on the Grid resource and the size of the workflow. Thus, in the critical case, the SLA workflow broker could serve only 6 users' SLA requests per minute. With a large and crowd Grid, this capability is obviously insufficient and the SLA workflow broker may become the bottleneck of the system. Thus, reducing the runtime of the mapping algorithm while remaining the quality of the mapping solution is an essential requirement.

In this paper, we propose a parallel mapping algorithm based on L-Map algorithm to increase the capability of the SLA workflow broker. The parallel algorithm called pL-Map will reduce the time for mapping light communication workflow to Grid resources without decreasing the quality of the solution.

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

2. Related works

In two separated works [13, 12], Zeng et al and Iwona et al built systems to support QoS features for Grid-based workflow. In their work, a workflow includes many sub-jobs, which are sequent programs, and a Grid service has ability to handle one sub-job at a time. To map the workflow on to the Grid services, they used Integer Programming method. Applying Integer Programming to our problem faces many difficulties. The first is the flexibility in runtime of the data transfer task. The time to complete data transfer task depends on the bandwidth and the reservation profile of the link, which varies from link to link. The variety in completion time of data transfer task makes the constraints presentation very complicated. The second is that a RMS can handle many parallel programs at a time. Thus, presenting the constraints of profile resource requirement and profile of resource available in Integer Programming is very difficult to perform.

With the same resource reservation and workflow model, we proposed the H-Map algorithm which mapping a heavy communication workflow to Grid resources in [5]. The main idea of the H-Map algorithm is that a set of initial solutions distributed over the search space according to cost factor will be further refined to find the best solution. To form the set of initial solutions, candidate RMSs of each sub-job are sorted in cost order. A configuration is formed by selecting an RMS at a ranking level. Each configuration is then checked for feasibility and improved by using a local

search. Because of not considering the characteristics of a light communication workflow, the performance of H-Map algorithm is not sufficient when applied to this problem [4].

Metaheuristics such as GA, Simulated Annealing [15], etc were proved to be very effective in mapping, scheduling problems. McGough et al also use them in their system [14]. However, in our problem, with the appearance of resource profiles, the evaluation at each step of the search is very hard. If the problem is big with highly flexible variable, the classical searching algorithms need very long time to find a good solution [5].

In [4], we presented an algorithm called L-Map to map light communication workflows onto the Grid resources. The main idea of the L-Map algorithm is that a high quality and feasible solution is created. From this solution, we will reduce the solution space. The reduced solution space will be searched carefully to find out the best solution by using local search. The skeleton of the algorithm is described in Algorithm 1.

Algorithm 1 L-Map algorithm

- 1: Create and sort the configuration space
 {Find the initial feasible solution}
 - 2: Create the initial configuration
 - 3: **repeat**
 - 4: Compute earliest-latest timetable of the configuration
 - 5: Build reservation profiles of the related RMSs
 - 6: **if** having conflicts in reservation profiles **then**
 - 7: Adjust sub-jobs of the conflict period or move sub-jobs to other RMSs
 - 8: **end if**
 - 9: **until** There are no conflicts in reservation profiles
 - 10: Limit the configuration space
 - 11: Create the set of initial configurations
 - 12: **for all** configuration in the set **do**
 - 13: Improve the solution with local search
 - 14: **end for**
 - 15: Pick the solution with best result
-

3. pL-Map algorithm

We describe here a parallel algorithm based on the L-Map algorithm called pL-Map. After measuring the runtime of the L-Map algorithm, we noticed that the phase of finding the initial feasible solution does not need much time. Thus, we do not parallelize this phase. The architecture of the algorithm framework is presented in Figure 3.

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, we find the initial feasible solutions.

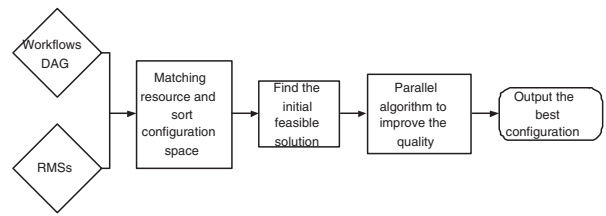


Figure 3. The architecture of the algorithm framework

Then, a parallel algorithm will improve the quality the initial solution as far as possible. The best solution is the output.

The procedures of matching resource and forming the initial feasible solution are similar to the L-Map 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 initial configuration is created by assigning each sub-job to the lowest cost RMS in the candidate list. The initial configuration is adjusted to become the feasible solution by resolving the conflicts in RMSs’ resource reservation profile. Detail description about the procedures can be seen in [4].

3.1 Parallel algorithm to improve the quality

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

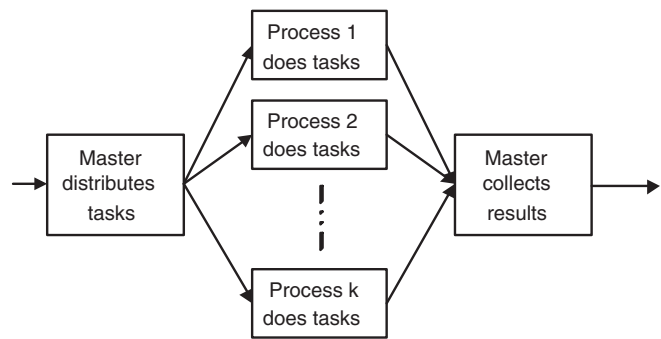


Figure 4. Working model of the improving quality algorithm

3.1.1 Master module

After having the initial feasible solution, the master module does following steps:

Step 1: Limiting the configuration space. Suppose that each sub-job has m candidate RMSs. Suppose that in the initial feasible solution, the RMS has the highest ranking at k . Thus, with each sub-job, we remove all its candidate RMSs having a rank greater than k . We limit the solution space in this way because the higher ranking RMSs only create higher cost solution than the initial feasible solution.

Step 2: Creating the initial configurations set. The set of initial configurations should be distributed over the solution space as widely as possible. Therefore, we create the new configuration by shifting onward to the first feasible solution. Assume each sub-job of having k candidate RMSs, we will shift $(k - 1)$ times to create $(k - 1)$ configurations. Thus, there are k configurations in the initial set including the found feasible solution.

Step 3: Distributing initial configurations to slave modules. To improve the quality of the initial configuration set, the master process distributes evenly configurations in the set to slave processes. The data sending from the master process to the slave process is presented in Figure 5.



Figure 5. Data format of the improving solutions command

The first field denotes the number of configuration in the message. Each configuration has its number of sub-jobs and list of RMSs for sub-jobs in the workflow. The first configuration is the initial feasible solution. We have to include the initial feasible solution to all messages because it is used to limit the search space in each slave process.

Step 4: Selecting the best solution and output. The master process collects all results from the slave processes and outputs the best solution.

3.1.2 Slave module

Each slave module also uses the initial feasible solution to reduce the configuration space as the master did.

After that, 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 feasible improvement. The process continues until the solution cannot be improved any more. Detail description about the procedure is presented in [7].

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 the solution has its cost field. It is noted that the slave process may not find out a feasible solution. In this case, the field "Nr sols" has value zero and the master will ignore the message.



Figure 6. Data format of the replying messages

3.2 Algorithm implementation

The implementation of 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 sorted configuration space
 - 3: Create the initial feasible solution
 - 4: Limit the configuration space
 - 5: Create a set of initial configuration
 - 6: Distribute the task of improving initial configurations to slave processes
 - 7: Collect the improved solutions
 - 8: Pick the best solution
 - 9: Send kill signal to slave process
-

Algorithm 3 Slave process

- 1: Get information of workflow and Grid resources
 - 2: Create sorted configuration space
 - 3: **if** Receive the task of improving initial configuration **then**
 - 4: Reduce the configuration space
 - 5: **for all** Received configurations **do**
 - 6: Improve the quality with local search
 - 7: **end for**
 - 8: Send back the master the best feasible solution
 - 9: **end if**
 - 10: **if** Receive the kill signal **then**
 - 11: Exit
 - 12: **end if**
-

We can see that all master and slave processes have 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, we have following comments.

- The main strategy of the pL-Map algorithm is still remained as the L-Map 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.
- As the size of the reference solution set is limited, the scalability of the pL-Map algorithm is also limited. In particular, the maximum effective number of computing nodes equals to the size of the initial configurations set.

4 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, 20 workflows with different topologies, number of sub-jobs, sub-job specifications, amount of data transferring were generated as workload. The Grid resources includes 20 RMSs with different resource configuration and different resource reservation context. The detail 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 20 single workflows with different number of computing nodes. Each computing nodes run one slave process. With the case of one computing node, we use L-Map algorithm. With more than one computing nodes, we use pL-Map algorithm. The runtime is measured using clock time. We record the start and stop time of the mapping algorithm in each running instance to calculate the runtime. 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 id of the workflow. Workflow having bigger id has bigger number of sub-jobs. Other columns present the runtime of the mapping algorithm according to different number of computing nodes.

From the data in Table 1, we can see that the runtime of the algorithms for each workflow is different. The runtime of the algorithms depends mainly on the time to calculate the workflow timetable and the number of that calculation.

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

Table 1. Performance evaluation result 1

- If the number of sub-jobs in the workflow increases, the time to calculate the workflow timetable increases and the number of that calculation also increases.
- If the size of the reduced configuration space increases, the number of timetable calculation also increases.

Due to the difference in those above parameters, the algorithm needs different amount of time for computation.

From the data in Table 1, we can see that the increasing in performance of the pL-Map algorithm with small workflows is not as clear as with big workflows. One reason is that the 1 second resolution is not fine enough for small workflows which has already need small runtime of the L-Map 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 big workflows. Thus, the parallel part applying to small workflows brings less effect.

With the big workflow as in the advance level experiment, the character of the pL-Map algorithm can be seen more clearly. The runtime of the algorithm is not reduced linearly with the increasing computing nodes. It is caused by the overhead and communication of parallel processes. When the number of parallel process increases, the overhead and communication increase. Thus, they reduce the

speedup of the algorithm.

To study more carefully the performance of the pL-Map algorithm, we do the second experiment with the mixed workload. To do the experiment, we generate 100 requests. Each request is selected randomly from 20 workflows. Then, we map continuously 100 requests with different number of computing nodes and record the necessary time to finish the mapping. With the case of one computing node, we use the L-Map algorithm. With more than one computing nodes, we use the pL-Map algorithm. The result is presented in Figure 7.

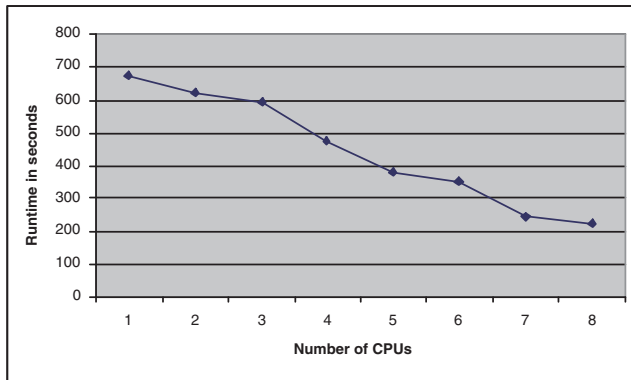


Figure 7. Performance evaluation result 2

From Figure 7, we can see that the runtime of the algorithm has minor change when the increasing in number of computing nodes is not enough. It is because the workload of the heaviest computing nodes is unchanged. For example, with the case of 5 and 6 CPUs in the experiment, the total number of initial solutions is 13 and thus, the heaviest process in both cases must handle 3 initial configurations.

As can be seen in Figure 7, the broker needs in average 6,7 and 2,2 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 comparing to 1 CPU. Beside that, with the business Grid, the broker could easily have flexible computing power. He could hire many computing nodes in the critical period and return them when the Grid is not crowd. Comparing 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.

5 Conclusion

This paper has presented a method, which reduces the necessary time to optimize the cost of running light communication SLA-based workflows on the Grid environment. In particular, we proposed a parallel algorithm pL-Map which is based on the L-Map algorithm. The main strategy of the L-Map algorithm is still remained 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 size workflows which require great computation power. In average, the algorithm can speed up to 300% with 8 CPUs. With low cost of hiring computing resources, the method can be applied without difficulty in real environments.

References

- [1] R. Lovas, G. Dzsa, P. Kacsuk, N. Podhorszki, D. Drtos. Workflow Support for Complex Grid Applications: Integrated and Portal Solutions, *Proceedings of 2nd European Across Grids Conference*, (2004) pp.129-138.
- [2] A. Sahai, S. Graupner, V. Machiraju, and A. Moorsel. Specifying and Monitoring Guarantees in Commercial Grids through SLA, *Proceeding of the 3rd IEEE/ACM CCGrid2003*, (2003) pp.292-300.
- [3] D.M. Quan, O. Kao. On Architecture for an SLA-aware Job Flows in Grid Environments, *Journal of Interconnection Networks*, Vol. 6, No. 3, (2005) pp.245-264.
- [4] D.M. Quan, J. Altmann, Resource allocation algorithm for light communication Grid-based workflows within an SLA context, Accepted by the International Journal of Parallel, Emergent and Distributed Systems, 2008
- [5] D.M. Quan. Mapping heavy communication workflows onto grid resources within sla context, *Proc. International Conference of High Performance Computing and Communication (HPCC06)*, (2006) 727-736.
- [6] D.M. Quan, J. Altmann. 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)*, (2007) pp.285-295.
- [7] D.M. Quan, Error recovery mechanism for grid-based workflow within SLA context, *Int. J. High Performance Computing and Networking*, Vol. 5, No. 1/2, (2007) pp.110-121.

- [8] M. P. Singh, and M. A. Vouk. (1997) *Scientific Workflows: Scientific Computing Meets Transactional Workflows*, <http://www.csc.ncsu.edu/faculty/mpsingh/papers/databases/workflows/sciworkflows.html>
- [9] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. Su, K. Vahi, and M. Livny. Pegasus : Mapping Scientific Workflows onto the Grid, *Proceedings of the 2nd European Across Grids Conference*, (2004) pp.11-20.
- [10] D. P. Spooner, S. A. Jarvis, J. Cao, S. Saini and G. R. Nudd. Local Grid Scheduling Techniques Using Performance Prediction, *IEEE Proceedings - Computers and Digital Techniques*, (2003) pp.87-96.
- [11] M. Hovestadt. Scheduling in HPC Resource Management Systems:Queuing vs. Planning, *Proceedings of the 9th Workshop on JSSPP at GGF8, LNCS*, (2003) pp.1-20.
- [12] I. Brandic, S. Benkner, G. Engelbrecht, and R. Schmidt. QoS Support for Time-Critical Grid Workflow Applications, *Proceedings of the first International Conference on e-Science and Grid Computing*, (2005) pp.108-115.
- [13] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, H. Chang. QoS-Aware Middleware for Web Services Composition, *IEEE Transactions on Software Engineering*, Vol. 30, No. 5, (2004) pp.311-327.
- [14] S. McGough, A. Afzal, J. Darlington, N. Furmento, A. Mayer, and L. Young. Making the Grid Predictable through Reservations and Performance Modelling, *The Computer Journal*, Vol. 48, No. 3, (2005) pp.358-368.
- [15] P. Brucker, (2004), *Scheduling Algorithm*, Third edition, Springer Verlag.