Mapping heavy 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 jobs, the SLA mapping module holds an important position and the capability of the mapping module depends on the runtime of the mapping algorithm. With the previously proposed mapping algorithm, the mapping module may develop into the bottleneck of the system if many requests come in during a short period of time. This paper presents a parallel mapping algorithm to map heavy communication SLA-based workflow onto Grid resources which can cope with the problem. Performance measurements thereby deliver evaluation results showing 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 are weather forecasters running weather forecasting workflow 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) [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 have already 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 and 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 incorrect results and skew 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 can be very large. 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

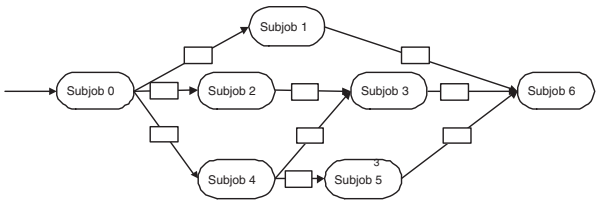

Figure 1. A sample workflow

each HPCC are managed by a software called local Resource Management System (RMS)[1]. Each RMS has its own unique resource configuration 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 advance resource reservation such as CCS [11]. In our model, we reserve three main types of resources: the CPU, the storage, and the 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 assumption can be made 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, thereby requiring the reservation of bandwidth [7].

1.3 Business model

To free users from the complicated task of managing the workflow execution, it is necessary to introduce a broker to handle the task for the user. We proposed a business model [6] for the system as depicted in Figure 2.

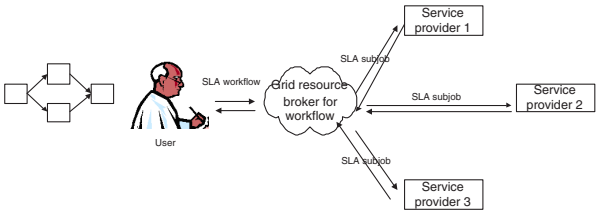

Figure 2. Stakeholders and their business relationship

The SLA workflow broker represents the user as specified in the SLA with the user and controls the workflow execution. This includes 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.

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

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 resources and deadline requirements.
- Let E be the set of data transfer in the workflow, which expresses 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 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 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:

- **Criterion 1:** All $K_i \neq \emptyset$. There is at least one RMS in the candidate set of each sub-job.
- **Criterion 2:** The total runtime period of the workflow must be within the expected period given by the user.
- **Criterion 3:** The dependencies of the sub-jobs are resolved and the execution order remains unchanged.
- **Criterion 4:** Each RMS provides a profile of currently available resources and can run many sub-jobs of a single flow both sequentially and in 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.

- **Criterion 5:** The data transmission task e_{ki} from sub-job s_k to sub-job s_i must not overlap other reserved data transmission tasks on the link between RMS running sub-job s_k to RMS running sub-job s_i. $e_{ki} \in E$.

In the next phase the feasible solution with the lowest cost is sought. The cost C of a Grid workflow is defined in formula 3, 4, 5. It is a sum of four factors: money for using the CPU, money for using the storage, cost of using the experts knowledge and finally money for transferring data between the involved resources.

$$C_1 = \sum_{i=1}^{n} s_i.r_t * (s_i.n_c * r_j.p_c + s_i.n_s * r_j.p_s + s_i.n_e * r_j.p_e) \tag{3}$$

$$C_2 = \sum e_{ki}.n_d * r_j.p_d \tag{4}$$

$$C = C_1 + C_2 \tag{5}$$

with $s_i.r_t, s_i.n_c, s_i.n_s, s_i.n_e$ are the runtime, number CPU, number storage, number expert of sub-job s_i respectively. $r_j.p_c, r_j.p_s, r_j.p_e, r_j.p_d$ are the price of using CPU, storage, expert, data transmission of RMS r_j respectively. $e_{ki}.n_d$ is the number of data to be transferred from sub-job s_k to sub-job s_i.

If two sequential sub-jobs run on the same RMS, the cost of transferring data from the previous sub-job to the later sub-job is neglected. It can be easily shown that the optimal mapping of the workflow to Grid RMS with cost optimizing is an NP hard problem.

In the previous work [5], we proposed the algorithm H-Map to map heavy communication workflow to the Grid resources. The result of the extensive experiment shows that the runtime of the H-Map algorithm is between 1 to 14 seconds depending on the Grid resource and the size of the workflow. Thus, in a critical case, the SLA workflow broker could serve only 4 users' SLA requests per minute. With a large and crowded Grid, this capability is obviously insufficient and the SLA workflow broker may wind up being the bottleneck of the system. Thus, reducing the runtime of the mapping algorithm while maintaining the quality of the mapping solution is an essential requirement.

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

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

2. Related works

In two separate works [13, 12], Zeng, et al and Iwona, et al built systems to support QoS features for a Grid-based workflow. In their work, a workflow includes many sub-jobs, which are sequential 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 the Integer Programming method. But applying Integer Programming to our problem creates many difficulties. The first is the flexibility in runtime of the data transfer task. The time to complete this depends on the bandwidth and the reservation profile of the link and this varies from link to link. The variety in completion time of the data transfer task makes the constraints presentation very complicated. The second is that an RMS can handle many parallel programs at a time. Thus, presenting the constraints of profile resource requirement and profile of resource availability in Integer Programming is very difficult to perform.

Using the same resource reservation and workflow model, in [4], we proposed an algorithm which mapped a light communication workflow to Grid resources. The proposed algorithm uses Tabu search to find the best possible assignment of sub-jobs to resources. In order to shorten the computation time caused by the high number of resource profiles to be analyzed and by the flexibility while determining start and end times for the sub-jobs, several techniques for reducing the search space are introduced. However, these techniques cannot be applied to solve the problem presented in this paper because the bandwidth reservation model is not considered [4].

Metaheuristics such as GA, Simulated Annealing [15], etc., were proved to be very effective in mapping, and 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 difficult to accomplish. If the problem is enormous with a highly flexible variable, the classic searching algorithms need a very long time to find a good and correct solution [5].

In [5], we presented an algorithm called H-Map for mapping heavy communication workflows onto the Grid resources. 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 locate the best solution. To form the set of initial solutions, the candidate RMSs of each sub-job are sorted by the order of cost.

Then 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. The framework of the algorithm is described in Figure 3.

```
Sort the solution space according to the computation cost
Clear the initial set of solutions
While not enough solutions {
   Form new configuration by combining 2 cost levels
   Compute timetable to check the feasible
   If feasible, put to the initial set of solutions
}
For each solution in the set {
   Do local search with the cost function
}
Pick the best solution
```

Figure 3. Framework of the H-Map algorithm

3. pH-Map algorithm

We describe here a parallel algorithm based on the H-Map algorithm called pH-Map. The architecture of the algorithm framework is presented in Figure 4.

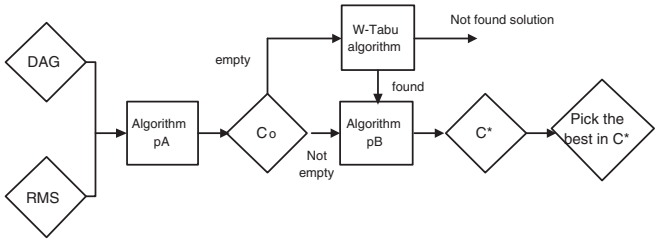

Figure 4. The architecture of the algorithm frame work

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 parallel algorithm pA is invoked to build the set C_o of initial solutions. If C_o is empty, the w-Tabu algorithm [7] to find the optimal workflow execution time is invoked. Otherwise, the parallel algorithm pB will improve the quality of each initial solution as much as possible. The best solution becomes the output.

3.1 Matching resource and building the configuration space

Each sub-job has different resource requirements about the type of RMS, the type of CPU and so on. There are a lot of RMSs with different resource configurations. Thus, the matching between the sub-job's resource requirement and the RMS's resource configuration is performed by several logic checking conditions in the WHERE clause of the SQL SELECT command. This satisfies Criterion 1. For present purposes, we use a workflow which includes 7 sub-jobs as presented in Figure 1 and with each sub-job having 3RMSs in the candidate list.

With each sub-job s_i, we sort the RMSs in the candidate set K_i according to the cost needed to run s_i. The cost is computed according to Formula 5. The configuration space of the sample now can be presented in Figure 5 and Table 1. In Figure 5, the RMSs lying along the axis of each sub-job have increasing cost moving from inside to outside. The line connecting each point in every sub-job axis forms a configuration. Figure 5, for example, presents 3 configurations with an increasing index from inside to outside. Figure 5 also presents the cost distribution of the configuration space according to Formula 5. The configuration in the outer layer has a greater cost than the configuration in the inner layer. The cost of the configuration lying between two layers is greater than the cost of the inner layer and smaller than the cost of the outer layer.

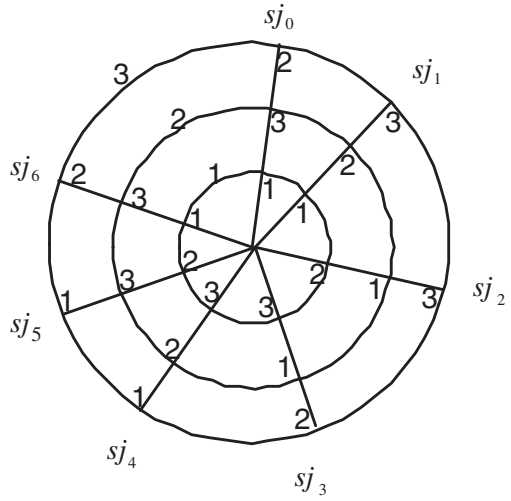

Figure 5. The configuration space according to cost distribution

3.2 Algorithm pA - Constructing initial solutions

The purpose of algorithm pA is to create a set of initial solutions which is distributed widely over the search space. A configuration can be created in two ways.

- By employing each layer of the sorted configuration space as the configuration.

Sj_ID	RMS	RMS	RMS
sj0	R1	R3	R2
sj1	R1	R2	R3
sj2	R2	R1	R3
sj3	R3	R1	R2
sj4	R3	R2	R1
sj5	R2	R3	R1
sj6	R1	R3	R2

Table 1. RMS candidate for each sub-job in cost order

- By merging two neighborhood layers of the sorted configuration space to form the configuration.

Assume that each sub-job has m candidate RMSs. This means we have m layers in the sorted configuration space and have in total 2*(m-1) configurations. With this method we can ensure that the set of initial solutions is distributed over the search space according to cost criteria.

The task of finding the initial solution set is divided into many subtasks which are processed in parallel. The algorithm follows the conventional master-slave model as depicted in Figure 6.

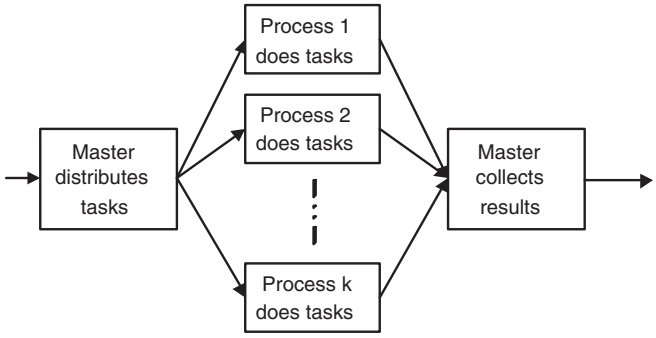

Figure 6. Working model of the pA and pB algorithms

3.2.1 The master process

Instead of directly creating the configurations, the master process evenly sends the information about the sorted layers to each slave process so that it can start finding the initial solutions. The format of the data sending from the master process to the slave processes is presented in Figure 7.

The first field Cmd is the command from the master to the slaves. The second field, Nr of entities, stores the total number of entities contained in the message. Each entity includes 2 fields. The Nr of layers presents the amount of layers needed to create the configuration. The Layer ID is the index of the layer in the sorted configuration space.

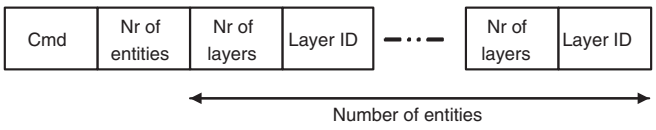

Figure 7. Data format of the finding initial solution command

3.2.2 The slave process

If a slave process receives a message with Nr of layer equal to 1, it will create the configuration by employing the layer having its index denoted in the Layer ID.

If a slave process receives a message with Nr of layer equal to 2, it will create the configuration by merging the layer having its index denoted in the field Layer ID and the next layer. For example, if Layer ID equals 1, layer 1 and layer 2 will be merged to produce a new configuration. To do this, we take the p first elements from the first layer and then the p second elements from the second layer and repeat until having enough n elements to form the completed configuration. n is the number of sub-jobs of the workflow. Thus, we get half the number of elements from the first layer and the other half number of elements from the second one. Combining in this way will ensure the target configuration having maximal difference in cost according to Formula 5 comparing to the source configurations.

After receiving a configuration, the slave process checks the Criterion 2 of the configurations. To verify Criterion 2, we have to determine the timetable for all sub-jobs of the workflow. Details about the procedure to perform this task are described in [5]. Here, the procedure is only presented briefly. First, we have to determine the assigning sequence of sub-jobs in the workflow. Then, we determine the most suitable time schedule for each sub-job and data transfer following the assigning sequence.

With entities having Nr of layer equal to 2, if the created solution does not satisfy the Criterion 2 requirement, we construct more to have enough 2*m-1 configurations. To do the construction, we change the value of p parameter in the range from 1 to (n/2) in order to create the new configuration.

When the slave process finishes computing, it sends the result back to the master. The data sent from the slave process to the master process is presented in Figure 8.

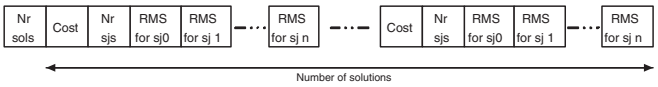

Figure 8. Data format of the reply from slave

The first field denotes the number of solution in the message. Each solution has its cost, number of sub-jobs and list

of RMSs for sub-jobs in the workflow.

3.3 Algorithm pB - Improve the quality of solutions

To improve the quality of solutions in the initial set C_o, the master process evenly distributes solutions in the set C_o to slave processes. We have to collect the initial solutions before redistributing them to the slave processes because each slave process may reveal a different number of solutions. Thus, without the redistribution, the workload in each slave process is not equal.

The data sending from the master process to the slave process is presented in Figure 9. It is similar to the one presented in Figure 8 except that there is one extra field to denote the command type and each solution does not have the cost field.

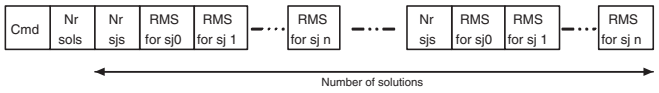

Figure 9. Data format of the improving solutions command

Each slave process improves the quality of each initial solution by using local search. This procedure tries to replace the present RMS with other RMSs in the candidate list to find the best improvement. The process continues until the solution cannot be improved any more. A detailed description about the procedure is presented in [5].

When the improvement is finished, each slave process sends the master only the best found solution with a message similar to the one in Figure 8. The master then picks the best solution and outputs it.

3.4 Algorithm implementation

The implementation of master and slave process is presented in Algorithm 1 and Algorithm 2.

Algorithm 1 Master process

1: Get information of the workflow and Grid resources
2: Create a sorted solution space
3: Distribute the task of finding the initial solution to the slave processes
4: Collect initial solutions from slaves
5: Distribute the task of improving initial solutions to slave processes
6: Collect the improved solutions
7: Pick the best solution
8: Send the kill signal to the slave processes

Algorithm 2 Slave process

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

We can see that all master and slave processes have complete 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 the following comments.

- The main strategy of the pH-Map algorithm still remains as the H-Map algorithm. Thus, the quality of the algorithm is kept. Only the computation intensive parts are parallelized in order to improve the execution time of the mapping module.

- As the size of the initial solution set is limited, the scalability of the pH-Map algorithm is also limited. In particular, the maximum number of initial solutions is 2*m-1. Thus, the maximum effective number of computing nodes is 2*m-1.

4. Performance experiment and applicability

As the quality of the algorithm is unchanged, the performance experiment is simulated 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 this, 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. The details information about the workload information and resource information can be seen in [5].

In the first experiment, we studied the runtime of the algorithm for 18 single workflows with different number of computing nodes. Each computing nodes run one slave process. In the case of 1 computing node, we used H-Map algorithm. With more than 1 computing nodes, we

used pH-Map algorithm. As the time entity in our experiment is in second, the smallest runtime of the algorithm is 1 second. The result is presented in Table 2. 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.

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

Table 2. Performance evaluation result 1

From the data in Table 2, we can see that the increase in performance of the pH-Map algorithm with small workflows is not as clear as with large workflows. One reason is that the 1 second resolution is not fine enough for small workflows which already needs short runtime of the H-Map algorithm. Another reason is that the rate between the overhead and the main computing part of the algorithm with small workflows is larger than with the large workflows. Thus, the parallel part applying to small workflows results in a lesser effect.

With the large workflow such as in the advance level experiment, the character of the pH-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 acceleration of the algorithm. We can also see that the runtime of the algorithm is not changed when the increasing in number of computing nodes is not enough. This is because the workload of the heaviest computing nodes is unchanged. For example, with the case of 4 and 5 CPUs in the experiment of the workflow which includes 25 sub-jobs, the total number of initial solutions is 16 and thus, the heaviest process in both cases must handle 4 initial solutions.

To study more carefully the performance of the pH-Map algorithm, we performed the second experiment with the mixed workload. For this, we generated 100 requests. Each request was selected randomly from 18 workflows. Then, we continuously mapped the 100 requests with a different number of computing nodes and recorded the needed time to finish the mapping. For the case of 1 computing node, we used the H-Map algorithm. With more than 1 computing nodes, we used the pH-Map algorithm. The result is presented in Figure 10.

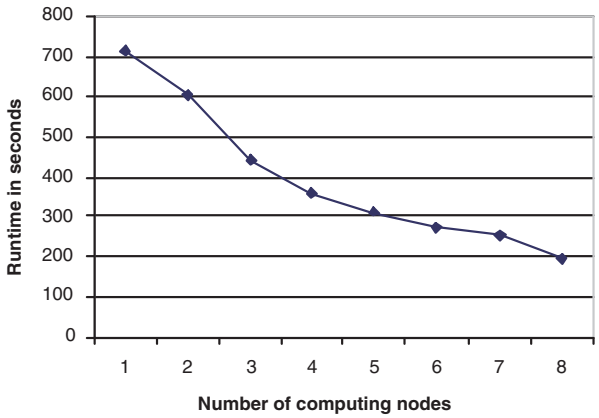

Figure 10. Performance evaluation result 2

From Figure 10, we can see the same trend as the above experiment that the acceleration of the algorithm is reduced when the number of computing nodes increases.

As can be seen in Figure 10, the broker needs an average of 7 and 2 seconds for a request with 1 CPU and 8 CPUs respectively. This means that the capability and income of the brokers increases by 350% with 8 CPUs compared to 1 CPU. More over, with the business Grid, the broker could easily have more flexible computing power. He could hire many computing nodes during 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 the Amazon pricing scheme (13-3-2008), a computing node costs 0,10 $ per hour. Thus, hiring 8 CPUs for an 1 hour would cost only 0,8 $. This means that the applicability of the approach is very high. By applying parallel processing technology, the broker can significantly increase his capability with low cost.

5. Conclusion

This paper has presented a method, which reduces the necessary time to map a heavy communication workflow onto Grid resources with respect to SLAs defined deadlines and cost optimization. In particular, we proposed a parallel algorithm pH-Map based on the H-Map algorithm. The main strategy of the H-Map algorithm still remains while the computing intensive parts are parallelized. Thus, the quality of the algorithm is kept while the runtime is significantly reduced. The performance evaluation showed that the algorithm is very effective especially with large size workflows requiring great computation power. On average, the algorithm can accelerate up to 350% with 8 CPUs. With low cost of hiring computing resources, the method can be applied to real environments without difficulty.

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*, Springer LNCS, 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*, IEEE CS press, 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*, World Scientific press, Vol. 6, No. 3, 2005, pp.245-264.

[4] D.M. Quan, O. Kao, "Mapping Grid job flows to Grid resources within SLA context", *Proceedings of the European Grid Conference,(EGC 2005)*, Springer LNCS press, 2005, pp.1107-1116.

[5] D.M. Quan, "Mapping heavy communication workflows onto grid resources within SLA context", *Proceedings of the International Conference of High Performance Computing and Communication (HPCC06)*, Springer LNCS press, 2006, pp. 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)*, Springer LNCS press, 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*, Inderscience press, 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*, Springer LNCS press, 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*, IEEE CS press, 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*, Springer LNCS press, 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*, IEEE CS press, 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*, IEEE CS press, 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*, Oxford press, Vol. 48, No. 3, (2005) pp.358-368.

[15] P. Brucker, *Scheduling Algorithm*, Third edition, Springer Verlag, 2004.