

Business model and the policy of mapping light communication Grid-based workflow within the SLA context

Dang Minh Quan¹ and Jörn Altmann¹²

¹ International University of Bruchsal, Campus 3, 76646 Bruchsal, Germany
`dang.minh@i-u.de`

² TEMEP, School of Engineering, Seoul National University, South-Korea
`jorn.altmann@acm.org`

Abstract. In the business Grid environment, the business relationship between a customer and a service provider should be clearly defined. The responsibility of each partner can be stated in the so-called Service Level Agreement (SLA). In the context of SLA-based workflows, the business model is an important factor to determine its job-resource-mapping policy. However, this aspect has not been described fully in the literature. This paper presents the business model of a system handling SLA-based workflow within the business Grid computing environment. From this business model, the mapping policy of the broker is derived. The experiment results show the impact of business models on the efficiency of mapping policies.

1 Introduction

Service Level Agreements (SLAs) [1] are currently one of the major research topics in Grid Computing, as they serve as a foundation for a reliable and predictable job execution at remote Grid sites. The SLA is a business contract between a user and a service provider. Thus, a SLA context represents a business environment. In our system of SLA-based workflows [3–7], the user wants to run a SLA-based workflow, which can be represented in Directed Acyclic Graph (DAG) form. The user has responsibility to specify the estimated runtime of the sub-jobs correlating with the specific resource requirements. He also provides the number of data to be transferred among sub-jobs. In the case of communication workflows with light communication, the data that will be transferred between sub-jobs is very small. The main requirement of the user is finishing the whole workflow within a specific period of time. Figure 1 depicts a sample scenario of running a workflow on the Grid environment. To ensure the deadline constraints, sub-jobs of the workflow must be distributed over many high performance computing centers (HPCCs) in the Grid. The resources in each HPCC are locally managed by the software called Resource Management System (RMS). To ensure that a sub-job can be executed within a dedicated time period, the RMS must support advance resource reservations such as CCS [2]. As HPCCs

usually connect with the network through a broadband link, the data transfer of the workflow with light communication can easily be executed in one time slot without the necessity to reserve network capacity. In our system, the time slot that executes the data transfer is right after the slot within which the source sub-job finished.

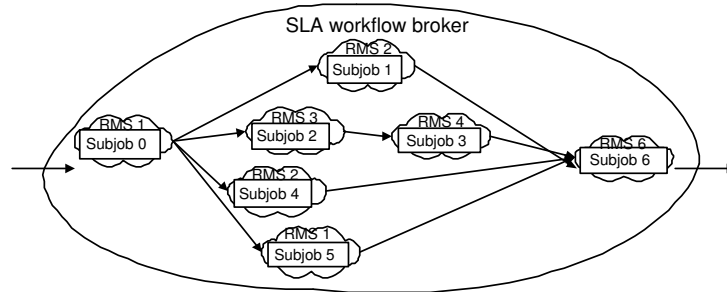


Fig. 1. Sample running workflow scenario

Assigning sub-jobs of the workflow to resources requires considering many constraints such as workflow integrity, on-time conditions, and optimality conditions. To free users from those tedious stuff, it is necessary to have a SLA workflow broker performing the co-operation task of many entities in the Grid. The business relationship of the SLA workflow broker with the users and the Grid service providers will determine the core working mechanism of the broker, the mapping policy. However, this issue has not been fully considered in most of the previous works about SLA-based workflows [9–15]. This paper, which is belongs to a series of efforts supporting SLA -based workflow [3–7], will analyze this problem to fill the gap. The paper is organized as follows. Section 2 describes the business model and SLOs. Section 3 presents the mapping policy and Section 4 describes the experiment. Related works are described in section 5. Section 6 concludes the paper with a short summary.

2 Business model and SLOs analysis

2.1 Business model

The business relationship between entities in the system running the SLA-based workflow is depicted in Figure 2. There are three main types of entities: end-user, SLA workflow broker and 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 how much he has to pay to each service provider. He only needs to know the total amount. This number depends on the urgency of the workflow and the financial ability

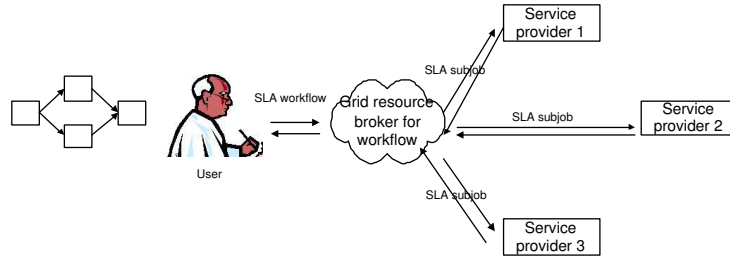


Fig. 2. Stakeholders and their business relationship

of the user. If there are problems with the workflow execution, for example the deadline is violated, the user will ask for compensation from the broker. This is clearly defined in the Service Level Objectives(SLOs) of the SLA.

The SLA workflow broker takes on the task of a user as specified in a SLA with the user. It controls the workflow execution. The SLA workflow broker also has to perform 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 between boths. 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 fix the price for its resources at the time of the SLA negotiation. As the resources of one HPCC usually have the same configuration as well as quality, each service provider has a fixed policy for fining SLA violation, for example $n\%$ of the total cost for each late time slot.

Concluding, all activities between end-users, SLA workflow broker and service providers are specified in legally binding contracts (SLAs). We think that the consideration of business models is important in order to establish incentive structures for users to consume HPCC services at low risk, and for brokers to perform efficiently. That means:

- If the broker does not get any monetary compensation for managing the workflow, there is little incentive for the broker to find a high-quality mapping solution. On contrary, it will encourage the broker to find some unreliable solutions, increasing its income.
- With the proposed business model, the broker takes responsibility for the mapping solution. When a failure happens and the workflow does not finish within the desired period, the user can fine the broker and the broker would not get compensated. This method allows the user to get a guaranteed service.
- The described business model frees the user from the hard work of managing the workflow execution. He only signs an SLA with the SLA workflow broker and then waits for the result.

2.2 SLOs analysis

Based on the analysis of the kind of entities in our model as described in the previous section, we identified five main SLOs, which ensure a successful execution of the workflow:

SLO1:

Between broker - provider: *If the storage usage of the sub-job exceeds the pre-determination space, the sub-job will be canceled and the broker has to pay resource reservation cost.* The provider has to cancel the sub-job, since otherwise, other sub-jobs would be affected through less-available resources.

Between user - broker: *If the storage usage of a sub-job of the workflow exceeds the pre-determination space, the workflow will be canceled and the user has to pay for the cost of the executed sub-jobs.* As the wrong estimated sub-job is canceled, the whole workflow must also be canceled because of the workflow integrity characteristic.

SLO2:

Between broker - provider: *If the memory usage of the sub-job exceeds the pre-determined space, the sub-job will be canceled and the broker has to pay resource reservation cost.* If the memory usage of the sub-job exceeds the pre-determined space, the data has to be swapped in and out of the harddrive and, thus, slows down the processing speed and exceeds the specified runtime. Therefore, the provider has to cancel it.

Between user - broker: *If the memory usage of a sub-job of the workflow exceeds the pre-determined space, the workflow will be canceled and the user has to pay for the cost of the executed sub-jobs.*

SLO3:

Between broker - provider: *If the sub-job cannot be finished because of computing system failure, the sub-job will be canceled and the provider is fined.* This case happens when the whole system is down. For example, the electric power or the network link to the Internet is broken.

Between user - broker: *If the workflow cannot be finished because of computing system failure, the workflow will be canceled and the broker is fined.* This case happens when many RMSs are down and the remaining healthy RMSs cannot execute the sub-jobs of the workflow. In general, the probability of this kind of problem is very small.

SLO4:

Between broker - provider: *If the runtime of the sub-job exceeds the limitation because of a wrong estimation, the sub-job can run some more time*

slots if there are sufficient resource free. The broker has to pay the extra cost for computation. If there are insufficient resources available, the sub-job will be canceled and the broker has to pay the cost for the computation.

Between user - broker: *If the runtime of a sub-job of the workflow exceeds the limit because of customer's wrong estimation and the sub-job is cancelled, the workflow will be cancelled and the user has to pay the computation cost. If the sub-job is not cancelled, the broker will change the mapping solution to ensure the workflow integrity. The user has to pay the extra cost and accept the runtime delay of the entire workflow.*

To check the conditions for this SLO, we have to monitor the entire system. Therefore, if there is no failure in the computing system and the sub-job finished late nonetheless, it can be deduced that the user estimated the resources needed wrongly.

SLO5:

Between broker - provider: *If the runtime of the sub-job exceeds the limit because of computing system failure, the sub-job can continue running and the provider has to pay fining for late time slots.*

Between user - broker: *If the runtime of the workflow exceeds the limit because of computing system failure, the workflow can continue running and the broker has to pay fining k\$ for each additional time slot.*

To check the condition of this SLO, we also have to monitor the system status. If there is a failure of the computing system and the sub-job cannot finish on time, we can deduce that any delay detected is caused by the failure. Usually, the latency caused by this error is small. However, if the runtime of the sub-job exceeds a threshold (which includes the extra time for error recovery), we can say that the user had wrong estimation and the sub-job is cancelled.

Each SLO stated above can be implemented by using the structures described in [4].

2.3 The influence of SLOs on mapping policies

In most of the existing systems [9–15], the broker tries to find a solution which is as inexpensive as possible. As the number of data to be transferred among sub-jobs in the workflow is very small, we can omit the cost of data transfers. Thus, the cost of running a workflow is the sum of three factors: the cost of using: (1) the CPU, (2) the storage and (3) the expert knowledge. The runtime of the sub-job depends on the resource configuration. The runtime estimation mechanism can be based on the one described in [17].

However, within the SLA context, the formal policy is not always effective. First, the deadline of the workflow has different meanings for different people. The importance of a deadline depends on the urgency of the workflow. If the workflow is critical, the user is willing to pay more but also imposes stricter fining rates. Otherwise, he pays less and requires lower fining rates. Thus, having a fixed mapping mechanism is not enough. The broker must have suitable policies to

handle different user requirements. In this way, the fining rate, as defined in the SLO, can effect seriously the mapping policy of the broker. Assume that the broker choses a hardware resource with a higher processing power than specified by the user. Assume as well that the broker adjusts the expected runtimes of the sub-jobs according to the processing power. If a sub-job is finishing late (although no failure in any RMS has been detected), the broker cannot impose the responsibility to the user. It is not possible to determine whether the delay is caused by a poor initial estimation of the user or by a faulty adjustment of the broker. Therefore, the broker should pay in this case.

3 Mapping policies

In our system, we define three types of urgencies: high, medium, and low. It depends on the user to select the level of urgency for his workflow. Depending on the preference of the user, the broker will have different mapping policies. The goal of those mapping policies is securing the deadline of the workflow under different constraints.

The most common cause that affects the deadline of a workflow is the crash of some nodes within a RMS. According to Google system [18], there is a node down every hour. When a node is down, the sub-jobs cannot continue running. In this case, the RMS will restart the sub-job from its checkpoint, an image of the process states from some time in the past [16]. This event will prolong the previously estimated runtime of the sub-job. Consequently, the deadline of the workflow may be violated. A mapping policy could deal with this situation in different ways. For example, a mapping policy could be to introduce spare time slots between sub-jobs of high urgency workflows. In the following, we propose three mapping algorithms correlating to three urgency levels of workflows.

3.1 Mapping workflow of high urgency

For high urgency workflows, each sub-job of the workflow should have a spare time period. In case of light communication workflows, we can distinguish two types of spare time. The first type is for sub-jobs having flexible start time and end time. For example, sub-job 1 can start in a wide range of time slots without effecting the deadline of the workflow. Therefore, if this sub-job is scheduled earlier than the latest end time, we could introduce a spare period. The second type of spare time is for dependent, sequential sub-jobs (e.g. sub-jobs in the critical path 0, 2, 3, 6 of figure 1. The spare period can only be introduced if those sub-jobs are assigned to run on a more powerful RMS than initially specified. Thus, the actual runtime will be shorter than originally specified and, therefore, opening up the opportunity for spare time periods.

From the two above observations, we propose the algorithm called EL-Map (E presents Extreme, L presents Light), which is based on L-Tabu algorithm [3], to do the mapping task. At the step of determining RMS candidates for each sub-job, we refine further the solution space in order to satisfy the spare time period

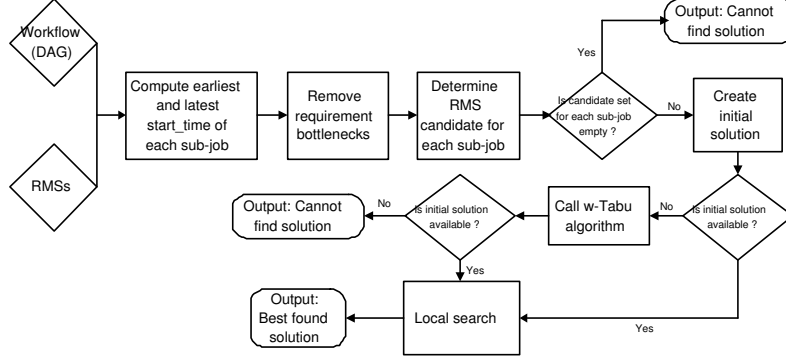


Fig. 3. EL-Map algorithm

requirements. For critical sub-jobs (i.e. a sub-job on which many subsequent sub-jobs depend), we choose only RMSs having more powerful configuration than required by the user.

3.2 Mapping workflow of medium urgency

For medium-urgent workflows, each sub-job of the workflow does not need to have a spare time period. The broker will reserve the resources according to the user's estimated time period. The mapping algorithm for this case is L-Tabu algorithm [3].

3.3 Mapping workflow of low urgency

For workflow of low urgency, we suggest to use existing algorithms, which reduce the cost by considering the runtime on sub-job-suitable hardware resources. The algorithm that we use is called L-Map and is presented in Figure 4 [5].

Step 0: With each sub-job of the workflow, we sort the RMSs in the candidate set according to the cost of running.

Step 1: We form the first configuration by assigning each sub-job to the RMS that has the lowest cost in the candidate list.

Step 2: We compute the earliest start time and the latest stop time of each sub-job using the conventional graph algorithm.

Step 4: If the reservation profile has conflict periods, we have to move sub-jobs by adjusting the earliest start time slot or the latest end time slot of the sub-jobs.

Step 5: We have to adjust the earliest start time and latest stop time of each sub-jobs that is affected by moving of sub-jobs in step 4 and then repeat step 3 and 4 until we cannot adjust the sub-jobs any more.

Step 6: If after adjusting phase, there are still some conflict periods, we have to move some sub-jobs contributing to the conflict to other RMSs. If a sub-job

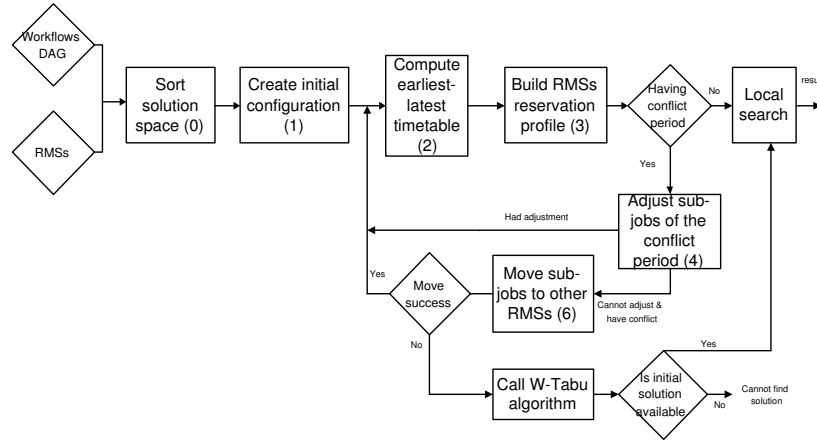


Fig. 4. L-Map algorithm architecture

cannot be moved to other RMS, we can deduce that the Grid resource is busy and the w-Tabu algorithm is invoked. If w-Tabu cannot find an initial solution, the algorithm will stop.

Step 7: The process from step 3 to step 6 is repeated until there is no conflict period or w-Tabu algorithm is invoked. After this phase, we have a feasible candidate solution.

Step 8: A local search procedure is used to improve the quality of the solution as far as possible.

4 Experimental results

The goal of the experiment is to measure the difference in cost and the influence on business decisions when applying different mapping policies. The experiments are executed with 18 workflows. Each workflow differs in its topology, the number of sub-jobs, the sub-job specifications, and the amount of data to be transferred. These workflows are then mapped to 20 RMSs with different resource configurations and initial resource reservations by 3 algorithms: EL-Map, L-Tabu and L-Map. In the experiment, 30% of all RMSs have a CPU performance equal to the requirement, 60% of RMSs have a CPU performance is twice as powerful than required, 10% of RMSs have a CPU performance that is 200% more powerful than required. Along with the increase in performance, we assume that the price for each CPU class increases linearly, however, slower than $f(x)=x$. The information about RMS and workflow is available online at http://it.i-u.de/schools/altmann/DangMinh/desc_expe1.txt.

In the experiment, the runtime of each sub-job in each type of RMS, which is used in L-Map algorithm, is assigned by using following formula:

Sjs	EL-Map	L-Tabu	L-Map		
			30-10-60	60-10-30	90-10-0
Simple level experiment					
7	660.18	629.28	624.97	616.80	612.47
8	830.01	753.90	749.29	741.12	733.42
9	828.47	776.87	772.73	763.25	755.55
10	938.15	831.62	829.36	822.30	814.60
11	972.64	886.94	884.14	875.34	870.44
12	1061.01	942.71	940.04	931.23	920.11
13	1136.78	996.76	993.82	987.07	972.58
Intermediate level experiment					
14	1250.05	1052.43	1050.61	1033.62	1025.05
15	1367.41	1176.33	1174.93	1159.48	1146.00
16	1431.28	1301.06	1299.31	1283.80	1270.36
17	1586.94	1355.84	1345.80	1336.27	1322.15
18	1707.27	1480.52	1470.12	1460.59	1439.00
19	1823.63	1503.77	1493.56	1480.60	1462.43
20	1762.85	1558.92	1557.85	1536.56	1515.27
21	1966.77	1582.08	1581.23	1559.94	1535.85
Advance level experiment					
25	2093.66	1775.10	1770.44	1747.02	1729.51
28	2257.40	1875.54	1865.57	1845.57	1824.08
32	2570.20	2202.74	2198.52	2167.76	2160.52

Table 1. Performance experiment result

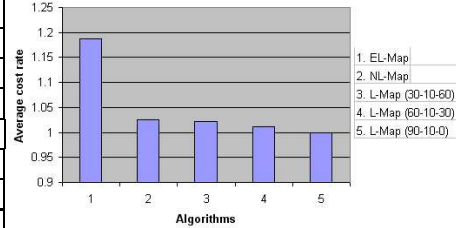


Fig. 5. Cost in average of each algorithm

$$rt_j = \frac{rt_i}{\frac{pk_i + (pk_j - pk_i) * k}{pk_i}} \quad (1)$$

pk_i, pk_j is the performance of a CPU in RMS r_i, r_j respectively. rt_i is the estimated runtime of the sub-job with the resource configuration of RMS r_i . k is a speed-up control factor, which depends on the structure of the application. In the experiment, k had three values: 0.5, 0.25, 0.1. We specified the workload configuration as a vector of numbers of sub-jobs in the workflow with different k . For example, the workload configuration 90-10-0 defines that 90% of sub-jobs have a $k = 0.5$, 10% of sub-jobs have $k = 0.25$, 0% of sub-jobs have $k = 0.1$. In this experiment, we use three workload configurations: (1) 30-10-60, (2) 60-10-30, (3) 90-10-0.

The result of the experiment is presented in Table 1. As the runtime of each algorithm is very small in few seconds, we only present the cost of the mapping solution with each algorithm. Figure 5 presents the average cost in relative value of each algorithm. From the experiment results in Table 1 and Figure 5, we can see that the cost of executing a workflow is lower for workflows with a lower level of urgency. This is caused by the fact that the time reserved for each sub-job is lower for workflows with lower urgency.

We can also see that in the case of the L-Map algorithm, the configuration of the workflow also effects the cost of the solution. If the workflow has many sub-jobs with large k (i.e. higher performance), the cost will be lower. The reason is that a sub-job with a large k needs less time periods to finish. In 5, we can see that the cost of using less-powerful resource is higher than the price of more powerful resource multiplied by their runtime.

In 5, we can also notice that the difference in cost between medium-urgent and low-urgent workflows varies depending on the configuration of the workflow. If the workflow has a small number of sub-jobs with big k , the difference in cost is not so much. This fact suggest that NL-Map algorithm can be applied even to low urgency workflows with low number of large k .

The price to pay for preventive actions against failures of nodes within in workflows of high urgency is high. As can be seen in 5, the cost for running the EL-Map algorithm is about 15% higher than the cost of the L-Map algorithm. However, it can be assumed that a user, who has to finish his workflow on time, is willing to pay this extra cost to him. Therefore, this extra cost is acceptable.

5 Related works

The literature records many efforts supporting QoS for workflow. Imperial College e-Science Network Infrastructure (ICENI) is an end-to-end Grid middleware system developed at the London e-Science Centre [9]. AgFlow is a middleware platform that enables the quality-driven composition of Web services [10]. QoS-aware Grid Workflow is a project, which aims at extending the basic QoS support to Grid workflow applications [11]. The work in [12] focuses on mapping the sweep task workflow to Grid resource with deadline and budget constraints. However, all of them do not define business model for the system. Therefore, they just simply find a solution satisfying the deadline and optimizing the cost. Recently, there are many Grid projects working on SLA issue [13–15]. Most of them focus on single job and thus, only the direct relation between user and service provider is considered. The business role of the broker in such systems is not fully evaluated. More over, supporting SLAs for Grid-based workflows is just at the initial phase and is not fully exploited.

6 Conclusion

This paper presents a business model to execute SLA-aware workflows in Grid environments. In particular, we focused on the aspects related to Service Level Objectives. Within the SLA context, end-user can negotiate different fining rates with the service provider. Scoping with different levels of urgency, brokers have to use different mapping algorithm to find solutions with different risk levels. Our measurements have shown that using runtime adaptation gains only benefit in some special cases, namely where workflows have many sub-jobs with a large speed-up factor. For workflows of high urgency, the extra cost of 15% is still

acceptable, if we consider that user with high urgency jobs are willing to pay more for getting a higher quality.

References

1. A. Sahai, V. Machiraju, M. Sayal, L. J. Jin, F. Casati: Automated sla monitoring for web services. DSOM 2002, LNCS 2506, (2002) 28–41.
2. M. Hovestadt: Scheduling in hpc resource management systems:queuing vs. planning. Proc. 9th Workshop on JSSPP at GGF8, LNCS, (2003) 1–20.
3. D. Quan, O. Kao: Mapping grid job flows to grid resources within sla context. Proc. the European Grid Conference,(EGC 2005), LNCS 3470, (2005) 1107–1116.
4. D. Quan, O. Kao: On architecture for an sla-aware job flows in grid environments. Journal of Interconnection Networks, World scientific computing, (2005) 245–264.
5. D. Quan, J. Altmann: Mapping Grid-based workflows with light communication, onto Grid resources within an SLA context. Submitted to GSEM 2007 conference, (2007).
6. D. Quan: Error recovery mechanism for grid-based workflow within sla context. Accepted by International Journal of High Performance Computing and Networking (IJHPCN), (2006).
7. D.Quan: Mapping heavy communication workflows onto grid resources within sla context. Proc. International Conference of High Performance Computing and Communication (HPCC06), (2006) 727-736.
8. http://en.wikipedia.org/wiki/Service_level_objectives
9. 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, v.48 n.3, (2005) 358–368.
10. L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, H. Chang: QoS-Aware Middleware for Web Services Composition, IEEE Transactions on Software Engineering, v.30 n.5, (2004) 311–327.
11. I. Brandic and S. Benkner and G. Engelbrecht and R. Schmidt: QoS Support for Time-Critical Grid Workow Applications, Proceedings of e-Science 2005, (2005).
12. J. Yu and R. Buyya: Scheduling Scientific Workflow Applications with Deadline and Budget Constraints using Genetic Algorithms, Scientific Programming Journal, ISSN: 1058-9244, IOS Press, Amsterdam, The Netherlands (2006)
13. M. Surridge, S. Taylor, D. De Roure, and E. Zaluska: Experiences with GRIA, Proc. 1st IEEE International Conference on e-Science and Grid Computing, (2005).
14. <http://www.eu-egee.org/>
15. M. Hovestadt, O. Kao, and K. Vo. The First Step of Introducing Risk Management for Preprocessing SLAs, IEEE International Conference on Services and Computing 2006 (SCC'06), (2006) 36-43.
16. P. Garbacki, B. Biskupski and H. Bal: Transparent fault tolerance for grid application, Proc. the European Grid Conference (EGC 2005), LNCS 3470, (2005) 671-680.
17. D.P. Spooner, S.A. Jarvis, J. Cao, S. Saini, G.R. Nudd, Local grid scheduling techniques using performance prediction. IEEE Proc. Computers and Digital Techniques, (2003) 87–96.
18. <http://www.renci.org/publications/presentations/LACSI/prelimSlides.pdf>