

Mapping heavy communication Workflows onto Grid Resources within an SLA context

Dang Minh Quan

Paderborn Center of Parallel Computing, University of Paderborn, Germany

Abstract. Service Level Agreements (SLAs) are currently one of the major research topics in Grid Computing. Among many system components for supporting SLA-aware Grid jobs, the SLA mapping mechanism receives an important position. It is responsible for assigning sub-jobs of the workflow to Grid resources in a way that meets the user's deadline and as cheap as possible. With the distinguished workload and resource characteristics, mapping a heavy communication workflow within SLA context defines new problem and needs new method to be solved. This paper presents the mapping algorithm, which can cope with the problem. Performance measurements deliver evaluation results on the quality and efficiency of the method.

1 Introduction

Mapping and running jobs on suitable resources are the core tasks in Grid Computing. With the case of Grid-based workflows, where a single job is divided into several sub-jobs, the majority of efforts for this issue concentrate on finding a mapping solution in best effort manner [1–3]. In the SLA (Service Level Agreement) context, where resources are reserved to ensure the Quality of Service (QoS), mapping a workflow requires different mechanism. The literature recorded some proposed solutions for this problem in [4–6]. Most of the proposed mechanisms suppose a workflow including many sub-jobs, which are sequent programs, and a Grid service having ability to handle one sub-job at a time. This is not sufficient enough as sub-jobs in many existed workflows [7–9] are parallel programs, and many High Performance Computing Centers (HPCCs) provide computing service under single Grid service [10]. It is obvious that a HPCC can handle many sub-jobs, which can be either sequent programs or parallel programs, at a time. Moreover, all of them did not consider the case of having heavy communication among sub-jobs in the workflow. This paper, which is a continuous work in a series of efforts supporting SLA for the Grid-based workflow [12–14], will present a mechanism to handle all stated drawbacks.

1.1 Workflow model

Like many popular systems handling Grid-based workflow [1–3], we also suppose Directed Acyclic Graph (DAG) form of the workflow. User describes the

specification about the required resources to run sub-jobs, data transfer among sub-jobs, the estimated runtime of sub-jobs and impose the expected runtime of the whole workflow. User wants the system to finish running the whole workflow in time. In the scope of this paper, the time is computed in slot. Each slot equals with a specific period of real time. Figure 1 presents a concrete Grid workflow. Each sub-job of the workflow has different resource requirements as described in table 1.

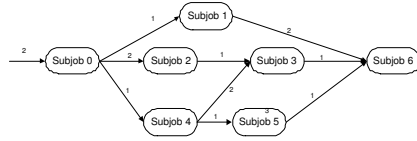


Fig. 1. A sample workflow

| Sj_ID | CPU | Storage | exp | runtime |
|-------|-----|---------|-----|---------|
| 0 | 18 | 59 | 1 | 6 |
| 1 | 16 | 130 | 3 | 8 |
| 2 | 20 | 142 | 4 | 5 |
| 3 | 22 | 113 | 4 | 8 |
| 4 | 18 | 174 | 2 | 9 |
| 5 | 20 | 97 | 3 | 14 |
| 6 | 16 | 118 | 1 | 4 |

Table 1. Resource requirements for sub-jobs

It is noted that a sub-job of the workflow can be either a single program or a parallel program and the data to be transferred among sub-jobs is very large, usually in GB scale. The case of light communication among sub-jobs of the workflow was handled in [14]

1.2 Grid service model

The computational Grid includes many High Performance Computing Centers (HPCCs). We believe that only HPCCs have enough conditions to support SLA for sub-jobs of the workflow. The resources in each HPCC are managed by software called local Resource Management System (RMS). In this paper, the acronym RMS is used to represent the HPCC as well as the Grid service of that HPCC. Each RMS has its own resource configuration and this configuration is usually different from other RMSs. Those differences include number of CPU, number of memory, storage capacity, software, expert, service price, etc. To ensure that the sub-job can be executed within a dedicated time period, the RMS must support advance resource reservation, for example CCS [10]. Figure 2 depicts a sample CPU reservation profile in such RMS. Queuing-based RMSs are not suitable for our requirement, as no information about the starting time is provided. In our system, we reserve three main types of resource: CPUs, storages and experts. An extension to other devices is straightforward.

If two sequent sub-jobs are executed in the same RMS, it is not necessary to do data transfer task and the time used for this task equal to 0. Otherwise, the data transfer task must be performed. To make sure that a specific amount of data will be transferred within a specific period of time, the bandwidth must also

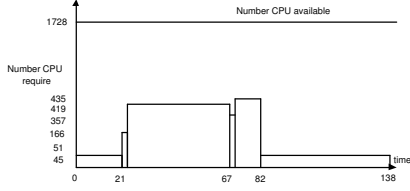


Fig. 2. A sample CPU reservation profile of a local RMS

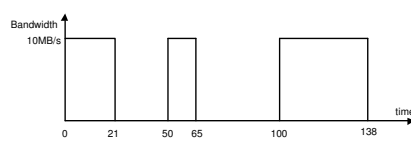


Fig. 3. A sample bandwidth reservation profile of a link between two local RMSs

be reserved. Unfortunately, up to now, there is no mechanism responsible for that task in the worldwide network. Here, to overcome that elimination, we use central broker mechanism. The link bandwidth between two local RMSs is determined as the average bandwidth between two sites in the network. Whenever having a data transfer task on a link, the SLA broker will determine which time slot is available for that task. During that specified period, the task can use the whole bandwidth and other tasks must wait. Using this principal, the bandwidth reservation profile of a link will look similar to the one as depicted in Figure 3. A more correctly model with bandwidth estimation [11] can be used to determine the bandwidth within a specific time period instead of the average value. In both cases, the main mechanism is unchanged.

1.3 Mapping mechanism requirement

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

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

A feasible solution must satisfy following conditions:

- The total runtime period of the workflow must be within the expected period given by user.

- All $K_i \neq \emptyset$. There is at least one RMS in the candidate set of each sub-job.
- The dependencies of the sub-jobs are resolved and the execution order remains unchanged.
- 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.

In the next phase the feasible solution with the lowest cost is sought. The cost of a Grid workflow is defined as a sum of four factors: money for using CPU, money for using storage, cost of using experts knowledge and finally money for transferring data between the involved resources. If two sequent subjobs run on the same RMS, the cost of transferring data from the previous subjob to the later subjob is neglected. It can be shown easily that the optimal mapping of the workflow to Grid RMS with cost optimizing is a NP hard problem.

2 Related work

In two separated works [5, 6], 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 an algorithm which mapping a light communication workflow to Grid resources in [14]. 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 in this paper because of different workload context.

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 [4]. 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. In the scope of this paper, we apply several standard Metaheuristics to our problem as means of comparing.

3 Planning algorithm for heavy communication workflows

The input of the mapping procedure includes information about workflow and information about RMSs. Information about workflow is provided in a file describing sub-jobs and a file describing the dependence. Information about RMSs is stored in a relational database. They include the description of the resource configuration in each RMS, the resource reservation profile of each RMS and the bandwidth reservation profile of each link. The information is collected from RMSs by the monitoring module. Based on this information, the system will do mapping. The overall mapping mechanism, which is called H-Map, is presented in Figure 4.

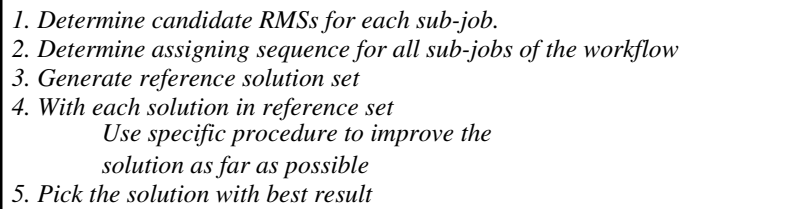
- 
1. *Determine candidate RMSs for each sub-job.*
 2. *Determine assigning sequence for all sub-jobs of the workflow*
 3. *Generate reference solution set*
 4. *With each solution in reference set*
Use specific procedure to improve the
solution as far as possible
 5. *Pick the solution with best result*

Fig. 4. Mapping mechanism overview

3.1 Determining candidate RMSs for each sub-job

Each sub-job has different resource requirement about type of RMS, type of CPU, etc. There are a lot of RMSs with different resource configuration. This phase finds among those heterogeneous RMSs the suitable RMSs, which can meet the requirement of each sub-job. Each resource parameter of an RMS is represented by number value and is stored in a separate column in the database table. For example, with the parameter Operating System, Linux, Sun, Window, Unix are represented by value number 1, 2, 3, 4 respectively. The co-relative resource requirement parameter of a sub-job is also represented by number value in the same manner. Thus, the matching between sub-job's resource requirement and RMS's resource configuration is done by several logic checking conditions in the WHERE clause of the SQL SELECT command.

3.2 Determining the assigning sequence of the workflow

When the RMS to execute each sub-job, the bandwidth among sub-jobs was determined, the next task is determining time slot to run sub-job in the specified

RMS. At this point, the assigning sequence of the workflow becomes important. The sequence of determining runtime for sub-jobs of the workflow in RMS can also affect the total runtime especially in the case of having many sub-jobs in the same RMS.

In general, to ensure the integrity of the workflow, sub-jobs in the workflow must be assigned basing on the sequence of the data processing. However, that principle does not cover the case of a set of sub-jobs, which have the same priority in data sequence and do not depend on each other. To examine the problem, we determine the earliest and the latest start time of each sub-jobs of the workflow in ideal condition. The time period to do data transfer among sub-jobs is computed by dividing the amount of data to a fix bandwidth. The earliest and latest start, stop time for each sub-job and data transfer depends only to the workflow topology and the runtime of sub-jobs but not the resources context. Those parameters can be determined by using conventional graph algorithms. A sample of those data for the workflow in Figure 1, in which the number above each link represents number of time slots to do data transfer, is presented in Table 2.

```

assign_number of each candidate RMS =0
While m_size < max_size {
  clear similar set
  foreach sub-job in the workflow {
    foreach RMS in the candidate list {
      foreach solution in similar set {
        if solution contains sub-job:RMS
          num_sim++
          store tuple (sub-job, RMS, num_sim) in
            a list }}
    sort the list
    pick the best result
    assign_number++
    If assign_number > 1
      find defined solution having the same
      sub-job:RMS and put to similar set
  }}

```

| Sub-job | Earliest start | Latest start |
|---------|----------------|--------------|
| 0 | 0 | 0 |
| 1 | 7 | 22 |
| 2 | 8 | 17 |
| 3 | 18 | 23 |
| 4 | 7 | 7 |
| 5 | 17 | 17 |
| 6 | 32 | 32 |

Table 2. Valid start time for sub-jobs of workflow in Figure 1

Fig. 5. The algorithm generate reference set

The ability of finding a suitable resource slot to run a sub-job depends on number of resource free during the valid running period. From the graph, we can see sub-job 1 and sub-job 4 having the same priority in data sequence. However, from the data in table 2, sub-job 1 can start at max time slot 22 while sub-job 4 can start at max time slot 7 without affecting the finished time of workflow. Suppose that two sub-jobs are mapped to run in the same RMS and the RMS can run one sub-job at a time. If sub-job 4 is assigned first at time slot 7, sub-job 1 will be run from time slot 16 thus the workflow will not be late. If

sub-job 1 is assigned first, in the worse case at time slot 22, sub-job 4 can be run at time slot 30 and the workflow will late 23 time slots. Here we can see, the latest time factor is the main parameter to evaluate the full affection of the sequence assignment decision. It can be seen through the affection, mapping the sub-job having smaller latest start time first will make the latency smaller. Thus, the latest start time value determined as above can be used to determine the assigning sequence. The sub-job having smaller latest start time will be assigned earlier.

3.3 Generating reference solution set

A solution is found by determining each sub-job of the workflow run by which RMS. We do not consider time factor in this phase so a reference solution is defined as a set of map sub-job:RMS with all sub-jobs in the workflow. Each solution in the reference solutions set can be thought as the starting point for local search so it should be spread as wide as possible in the searching space. To satisfy the space spreading requirement, number of the same map sub-job:RMS between two solutions must be as small as possible. The number of member in the reference set depends on the number of available RMSs and number of sub-jobs. During the process of generating reference solution set, each candidate RMS of a sub-job has a co-relative *assign_number* to count the times that RMS is assigned to the sub-job. During the process of building a reference solution, we use a similar set to store all defined solution having at least a map sub-job:RMS similar to one in the creating solution. The algorithm is defined in Figure 5.

While building a solution, with each sub-job in the workflow, we select the RMS in the set of candidate RMSs, which creates minimal number of similar sub-job:RMS with other solutions in the similar set. After that, we increase the *assign_number* of the selected RMS. If this value larger than 1, which means that the RMS were assigned to the sub-job more than one time, there must exist solutions that contains the same sub-job:RMS and thus satisfying the similar condition. We search those solutions in the reference set, which have not been in the similar set, and then add them to similar set. When finished, the solution is put to the reference set. After all reference solutions are defined, we use a specific procedure to refine each of the solution as far as possible.

3.4 Improving solution quality algorithm

Before improving the quality of the solution, we have to determine specific run-time period for each sub-job and each data transfer task as well as the makespan of the present solution. The start time of a data transfer task depends on the finish time of the source sub-job and the state of the link's reservation profile. We use *min_st_tran* variable to present the dependence on the finish time of the source sub-job. The start time of a sub-job depends on the latest finish time of the related data transfer tasks and the state of the RMS's reservation profile. We use *min_sj_tran* variable to present the dependence on the latest finish time of

the related data transfer tasks. The task to determine timetable for the workflow is done with the procedure in Figure 6.

```

foreach sub-job k following the assign sequence {
  foreach link from determined sub-jobs to k{
    min_st_tran=end_time of source sub-job
    search reservation profile of link the
    start_tran > min_st_tran
    end_tran = start_tran+num_data/bandwidth
    store end_tran in a list
  }
  min_st_sj=max (end_tran)
  search in reservation profile of RMS running
  k the start_job > min_st_sj
  end_job= start_job + runtime
}

```

```

while (num_loop < max_loop) {
  foreach subjob in the workflow {
    foreach RMS in the candidate list {
      if cheaper then put (sjid, RMS id, improve_value)
      to a list }}
  sort the list according to improve_value
  from the begin of the list{
    Compute time table to get the finished time
    If finished time < limit
      break
  }
  Store the result
  num_loop ++;
}

```

Fig. 6. Algorithm determine timetable for workflow **Fig. 7.** Procedure to improve the solution quality

For each sub-job of the workflow in the assigning sequence, firstly, we find all the runtime period of data transfer task from previous sub-jobs to current sub-job. This period must be later than the finish time of the source sub-job. Note that with each different link the transfer time is different because of different bandwidth. Then, we determine the runtime period of the sub-job itself. This period must be later than the latest finish time of previous related data transfer task. The whole procedure is not so complicate but time consuming. The time consuming steps are the searching reservation profiles, and they make the whole procedure long time consuming.

The overall of solution quality improvement procedure is described in Figure 7. In one iteration, we can move only one sub-job to one RMS with the hope to decrease the cost. So we only consider the move, which can decrease the cost. With each solution we compute the time table, if it satisfies the deadline then update the result.

4 Performance evaluation

Performance experiment is done with simulation to check for the quality of the mapping algorithms. The hardware and software used in the experiments is rather standard and simple (Pentium 4 2,8Ghz, 2GB RAM, Linux Redhat 9.0, MySQL). The whole simulation program is implemented in C/C++. The goal of the experiment is to measure the feasibility, the quality of the solution and the time needed for the computation. To do the experiment, 18 workflows with different topologies, number of sub-jobs, sub-job specifications, amount of data transferring were generated and mapped to 20 RMSs with different resource configuration and different resource reservation context by 6 algorithms H-Map, Simulated Annealing (SA), Guided Local Search (GLS), Iterated Local Search (ILS), Genetic Algorithm (GA), Estimation of Distribution Algorithm (EDA)

[15]. The implementation of those algorithms is described in [16]. The final result of the experiment is presented in table 3 with column Wf (Workflow) presents the id of workflows, column Rt (Runtime) and Cost record the cost and runtime of solutions generated by each algorithm correlative with each workflow respectively.

| | H-Map | | SA | | ILS | | GLS | | GA | | EDA | |
|----|-------|---------|-----|---------|------|---------|------|---------|------|---------|-------|---------|
| Wf | Rt | Cost | Rt | Cost | Rt | Cost | Rt | Cost | Rt | Cost | Rt | Cost |
| 1 | 1 | 632.58 | 105 | 632.58 | 21 | 632.58 | 14 | 632.58 | 25 | 632.58 | 62 | 632.58 |
| 2 | 0.5 | 756.96 | 84 | 756.90 | 37 | 756.96 | 19 | 756.90 | 22 | 756.96 | 88 | 756.90 |
| 3 | 1 | 780.42 | 114 | 780.42 | 54 | 780.42 | 29 | 780.34 | 27 | 780.34 | 126 | 780.34 |
| 4 | 1 | 836.18 | 78 | 836.18 | 81 | 836.18 | 46 | 836.18 | 28 | 836.18 | 178 | 836.18 |
| 5 | 1 | 892.02 | 105 | 892.02 | 114 | 892.02 | 59 | 892.02 | 29 | 892.21 | 241 | 892.02 |
| 6 | 2 | 948.10 | 90 | 948.10 | 147 | 948.10 | 80 | 948.10 | 36 | 1005.27 | 390 | 947.86 |
| 7 | 1 | 1003.7 | 78 | 1003.99 | 201 | 1003.7 | 98 | 1003.99 | 36 | 1075.19 | 462 | 1003.7 |
| 8 | 2 | 1059.89 | 121 | 1059.89 | 250 | 1059.89 | 127 | 1059.89 | 32 | 1059.89 | 558 | 1059.89 |
| 9 | 2 | 1184.21 | 130 | 1184.21 | 307 | 1184.21 | 167 | 1184.21 | 44 | 1248.86 | 659 | 1183.92 |
| 10 | 2 | 1308.53 | 146 | 1332.53 | 398 | 1308.53 | 187 | 1308.53 | 47 | 1383.53 | 680 | 1308.53 |
| 11 | 3 | 1364.14 | 124 | 1376.42 | 502 | 1364.14 | 222 | 1377.63 | 52 | 1440.81 | 956 | 1364.42 |
| 12 | 2 | 1488.12 | 184 | 1551.74 | 462 | 1521.74 | 303 | 1501.95 | 51 | 1569.39 | 854 | 1536.74 |
| 13 | 7 | 1512.26 | 174 | 1512.26 | 620 | 1512.26 | 354 | 1566.09 | 56 | 1620.17 | 1136 | 1512.26 |
| 14 | 3 | 1567.74 | 162 | 1631.15 | 815 | 1567.74 | 392 | 1568.15 | 56 | 1663.81 | 1255 | 1601.15 |
| 15 | 6 | 1591.12 | 161 | 1675.67 | 876 | 1591.12 | 524 | 1621.67 | 70 | 1764.18 | 1663 | 1621.67 |
| 16 | 5 | 1786.56 | 180 | 1871.81 | 1394 | 1840.31 | 763 | 1843.55 | 85 | 1914.91 | 2845 | 1830.87 |
| 17 | 7 | 1889.78 | 197 | 1960.87 | 1695 | 1892.27 | 1258 | 1936.83 | 93 | 2028.06 | 4170 | 1961.30 |
| 18 | 10 | 2217.34 | 272 | 2276.33 | 2046 | 2283.67 | 1623 | 2256.53 | 1953 | 2406.97 | 10976 | 2276.33 |

Table 3. Experiment results of the H-Map algorithm

The experiment results show that H-Map algorithm finds out higher quality solution with much shorter runtime than other algorithms in most cases. Some of the metaheuristics such as ILS, GLS, EDA find out equal results with small problems. But with big problem, they have exponent runtime and find out unsatisfied results.

5 Conclusion

This paper has presented a method, which performs an efficient and precise assignment of heavy communication workflow to Grid resources with respect to SLAs defined deadlines and cost optimization. In our work, the distinguished character is that a sub-job of the workflow can be a sequent or parallel program and a Grid service can handle many sub-jobs at a time. The performance evaluation showed that the proposed algorithm creates solution of equal or better quality than most standard metaheuristics and needs significantly shorter computation time. The latter is a decisive factor for the applicability of the method

in real environments, because large-scale workflows can be planned and assigned efficiently.

References

1. 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, Nicosia, Cyprus, January 28-30, 2004.
2. 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, 150(2), pp. 87–96, 2003.
3. 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, Nicosia, Cyprus, 2004.
4. 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, pp. 358–368, 2005
5. 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, pp. 311–327, may 2004
6. I. Brandic and S. Benkner and G. Engelbrecht and R. Schmidt, QoS Support for Time-Critical Grid Workflow Applications, Proceedings of e-Science 2005
7. S. Ludtke, P. Baldwin, and W. Chiu, "EMAN: Semiautomated Software for High-Resolution Single-Particle Reconstruction" , Journal of Structure Biology, v. 128, 1999.
8. G. B. Berriman, J. C. Good, A. C. Laity, "Montage: a Grid Enabled Image Mosaic Service for the National Virtual Observatory" , ADASS, v. 13, 2003.
9. L. Richter, "Workflow Support for Complex Grid Applications: Integrated and Portal Solutions", Proceedings of the 2nd European Across Grids Conference, 2004.
10. L. Burchard, M. Hovestadt, O. Kao, A. Keller, and B. Linnert, "The Virtual Resource Manager: An Architecture for SLA-aware Resource Management", Proceedings of the IEEE CCGrid 2004, IEEE Press, pp. 126–133, 2004.
11. R. Wolski, "Experiences with Predicting Resource Performance On-line in Computational Grid Settings", ACM SIGMETRICS Performance Evaluation Review, v. 30 n. 4, pp. 41–49, 2003.
12. D.M. Quan, O. Kao, "SLA negotiation protocol for Grid-based workflows", Proceedings of the International Conference on High Performance Computing and Communications (HPPC-05), LNCS 3726, pp. 505–510, 2005.
13. D.M. Quan, O. Kao, "On Architecture for an SLA-aware Job Flows in Grid Environments", Proceedings of the 19th IEEE International Conference on Advanced Information Networking and Applications (AINA 2005) , IEEE Press , pp. 287–292, 2005.
14. D.M. Quan, O. Kao, "Mapping Grid job flows to Grid resources within SLA context", Proceedings of the European Grid Conference,(EGC 2005), LNCS 3470, pp. 1107–1116, 2005.
15. C. Blum, A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison", ACM Computing Surveys, v. 35 n.3, pp. 268–308, 2003
16. D.M. Quan, "A Framework for SLA-aware execution of Grid-based workflows", PhD thesis, University of Paderborn - Germany, 2006.