Mapping Workflows onto Grid Resources Within an SLA Context

Dang Minh Quan and Odej Kao

Department of Computer Science, University of Paderborn, Germany

Abstract. The mapping of workflows on Grid resources differs from mapping a single job, as dependencies between the subjobs has to be considered and resolved. This is a major task, if underlying Service Level Agreements (SLAs) have to be satisfied, which define the deadline for the entire workflow but also allow flexibility while assigning the subjobs to resources. Therefore, new requirements regarding selection of optimal resource destination and satisfying multiple time constraints are important issues, which are not completely addressed by existing methods.

This paper presents a method which performs an efficient and precise assignment of workflow subjobs to Grid resources with respect to SLAs and subjob dependencies. The quality of the created results is evaluated with a number of experiments and compared to the results of existing planning tools.

1 Introduction

The mapping of jobs to suitable resources is one of the core tasks in Grid Computing. A substantial research in this area led to a number of methods, which allow the specification of job requirements, the description of available resources and the matching in order to find an appropriate execution platform [1, 2, 3]. However, the majority of the research is related to the mapping of singular jobs, which do not exhibit dependencies to other jobs regarding input/output data. The mapping of workflows, where a single job is divided into several subjobs, is the next research step. Demands for Grid-based workflows result from many scientific and business application, where data or suitable resources are distributed over multiple administrative domains. Thus, supporting workflows in Grid environments increases the applicability of Grid Computing for a large number of processes.

An intuitive approach of treating each subjob as an independent job and its mapping with the traditional methods reaches the limits in case of SLA-based processing. An SLA defines the start and end time of the workflow execution, which has to be considered within the execution chain. Thus, the subjobs has to be mapped in a way, that the final workflow deadline will be reached, regardless of the actual execution resource for the subjobs and the time needed for the transmission of output/input data between the subjobs. Thereby, two main problems arise. Firstly, for each subjob involved in the workflow the start and

P.M.A. Sloot et al. (Eds.): EGC 2005, LNCS 3470, pp. 1107–1116, 2005.

end time has to be computed and verified with respect to the global deadline. According to results the corresponding time slots on the selected resources have to be reserved. For this purpose we assume that the user provides the maximal runtime of each subjob and that the underlying resource management system (RMS) supports advance reservations such as CCS [5]. Queuing-based RMS are not suitable, as no information about the starting time is provided. Secondly, the selection of the Grid resources should be based on a minimal cost for the resource usage. For this purpose well-known models such as the flexible job shop scheduling [4, 7] are applied, which minimize the runtime of the workflow. In this paper, time is computed in slots, where each slot corresponds to a-priori defined time period. Reserved resources include number of CPUs, required memory size, availability of experts etc. An extension to other devices, software licenses, and other related resources is straightforward.

This paper is organized as follows. Section 2 presents a formal problem statement. Section 3 and 4 describe the related work and the proposed algorithm respectively. Empirical measurements and comparisons with existing planning methods are subject of Section 5.

2 Formal Problem Statement

The formal specification of the described problem includes following elements:

- Let K 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 subjobs in a given workflow including all subjobs with the current resource and deadline requirements.
- Let E be the set of connections (edges) in the workflow, which express the dependency between the subjobs and the necessity for data transfers between the jobs.
- Let T_i be the set of time slots for the subjob $S_i, S_i \in S$.
- Let K_i be the set of resource candidates of subjob S_i . This set includes all resources which can run subjob S_i, $K_i \in K$.

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

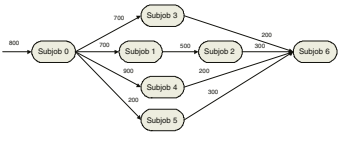


Fig. 1. A sample workflow

Table 1. RMSs resource reservation

ID	ID_hpc	CPUs	mem	exp	start	end
31	2	128	256000	8	0	1000000
23	0	128	256000	9	0	1000000
30	1	128	256000	6	0	1000000

Table 2. Resource requirements for subjobs

Sj_ID	CPU	mem	exp	runtime	earliest	latest
0	51	59700	1	21	5	35
1	62	130030	3	45	27	57
2	78	142887	4	13	57	87
3	128	112933	4	34	28	66
4	125	171354	2	21	28	65
5	104	97560	3	42	27	62
6	45	117883	1	55	71	101

Table 3. Connections

From	To	Transfer_time	Data
0	1	1	636
0	3	2	638
0	4	2	892
0	5	1	192
1	2	2	401
2	6	1	300
3	6	1	200
4	6	2	200
5	6	1	271

$$R = \{(S_i, k_{ij}, t_{il}) | S_i \in S, k_{ij} \in K_i, t_{il} \in T_i\} \tag{1}$$

A feasible solution must satisfy following conditions:

- For all $K_i \neq 0$ at least one RMS exists which can satisfy all resource requirements for each subjob.
- The earliest start time slot of the subjob $S_i \leq t_{il} \leq$ the latest start time slot of S_i. Each subjob must have its start time slot in the valid period.
- The dependencies of the subjobs are resolved and the execution order remains unchanged.
- Each RMS provides a profile of currently available resources and can run many subjobs of a single flow both sequentially and parallel. Those subjobs which run on the same RMS form a profile of resource requirement. With each RMS k_{ij} running subjobs 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 in this specific example is defined as a sum of four factors: compute time, memory usage, cost of using experts knowledge and finally time required 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 neglected.

An example of a specific instance of the described problem is presented in Figure 1 and Tables 1-3. Figure 1 visualizes the sequence of subjobs, the corresponding parameter are summarized in Table 3, where the data to be transferred between subjobs as well as the estimated duration of each task is presented. Table 2 describes the resource requirements and the range of valid start time slots of each subjob. Information about the available resources of the three involved RMS, which will execute the flow, is presented in Table 1.

Considering the RMS's ability to run several subjobs in parallel and the evaluation of resource profiles increases the complexity of the flexible job shop

scheduling problem. It can be shown easily that the optimal mapping of Grid-based workflows as described above is a NP hard problem.

3 Related Work

Cao et al. presented an algorithm that maps each subjob separately on individual Grid resources [2]. The algorithm processes one subjob at a time, schedules it to a suitable RMS with start time slot not conflicting the dependency of the flow. The selection of the destination resources is optimised with respect to a minimal completion time. When applying this strategy to the specified problem, each subjob will be assigned separately to the cheapest feasible RMS. This strategy allows fast computation of a feasible schedule, but it does not consider the entire workflow and the dependencies between the subjobs.

The mapping of Grid workflows onto Grid resources based on existing planning technology is presented in [1]. This work focuses on coding the problem to be compatible with the input format of specific planning systems and thus transferring the mapping problem to a planning problem. Although this is a flexible way to gain different destinations, significant disadvantages regarding the time-intensive computation, long response times and the missing consideration of Grid-specific constraints appeared. The latter is the main cause that the suggested solutions often do not express the expected quality.

The described problem can also be seen as a special case of the well known job shop scheduling problem [4, 7]. For solving this problem, two main methods – complete and incomplete method – exist. A complete method explores systematically the entire search space, while the incomplete (non-exact) method examines as rapidly as possible a large number of points according to selective or random strategy. Local search is one of the most prominent examples for this approach, which is realized by a number of methods such as Tabu Search [11, 12], Simulated Annealing [8], GA [6] etc. However, with the appearance of resource profiles, the evaluation at each step in local search is very hard. If the problem is big with highly flexible variable, the classical searching algorithm need very long time to find a good solution.

4 Planning Algorithm for Workflows

The proposed algorithm for mapping workflows on Grid resources uses Tabu search to find the best possible assignment of subjobs to resources. In order to shorten the computation time caused by the high number of resource profiles to be analysed and by the flexibility while determining start and end times for the subjobs, several techniques for reducing the search space are introduced.

The core algorithm requires a specification of the workflow and subjob requirements as well as the description of the available resources as input. This information is necessary in order to resolve the dependencies in the workflow and is stored in a file. The properties of the involved RMS are contained in a database.

This input information is processed according to the following algorithm steps:

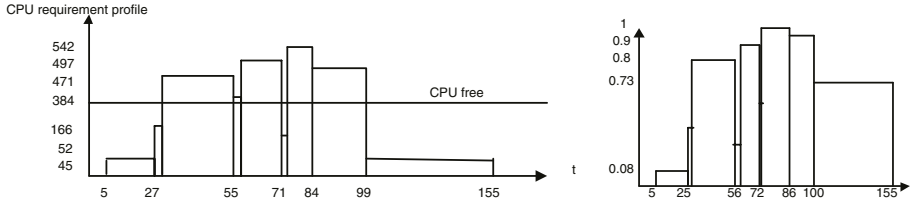
1. Based on the description in the database, all RMS are selected, which fulfil the requirements of at least one subjob.
2. Computation of the earliest start time and latest start time for each subjob by analysing the input workflow with traditional graph techniques.
3. Removing requirement bottlenecks. This task aims to detect bottlenecks with a large number of resource requirements, which can reduce the possible start/end times of a subjob. Based on this information subjobs are moved to other sites with more available resources in order to gain a longer time span for the positioning of the subjobs in the workflow.
4. Definition of the solution space by grouping all RMS candidates, which have enough available resources to start the subjobs within the determined slack time and to run the subjob until it is completed. This task is performed by establishing a connection with the site and retrieving the current and planned schedule.
5. The gained search space is evaluated with respect of the contained number of feasible solutions (large or small number of possible solutions). Subsequently, an initial solution is created.
6. Starting with the initial solution, a Tabu search is applied in order to increase the quality of the solution and to find the best possible and feasible assignment.

In following the individual algorithm steps are described in detail.

4.1 Resolving Requirement Bottlenecks

The generated requirement and resource profiles show those time slots, where a large number of resources is required, but not available. A sample of such situation is shown in Figure 2(a) on an example of required and available CPU instances. At each possible time slot, the total number of available CPUs over all involved RMS and the total number of required CPU as sum of the requirements of all subjobs possibly running in this time slot are computed. The contribution of each subjob in the profile is computed from the earliest start time to latest possible deadline. Figure 2(a) shows that at period 28-55 or 78-84 the number of required CPUs is larger than in other profile periods. This leads to a significantly reduced number of feasible start times and thus reduces the probability for finding an optimal solution or even a good solution. Therefore, the peak requirements have to be reduced by moving selected jobs to other time slots. Furthermore, by reducing the number of parallel running jobs on the same site, the probability for cost-effective execution of two subsequent jobs on the same site with a low communication effort increases.

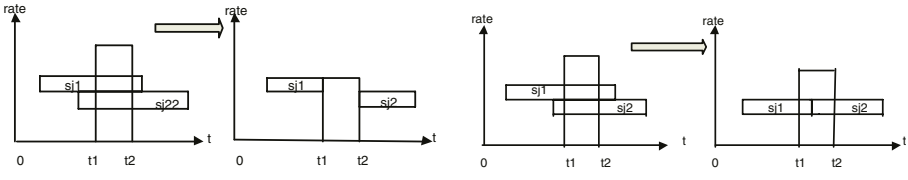
For resolving the requirement bottleneck the profiles of the required resources and the available resources are compared as shown in Figure 2(b). At each time slot, we define J as the set of m subjobs running at that time slot and R as the



(a) Relation between available and required CPUs

(b) Rate profile of the sample

Fig. 2. Profiles of the sample



(a) Moving subjobs

(b) Adjusting subjobs

Fig. 3. Removing possible requirement bottlenecks

set of n possible resource candidates for J. Subsequently, following measures can be computed.

$$TotalCPU_{require} := \sum_{i=1,m} J_i.CPU_{req} \quad \text{with} \quad J_i \in J \tag{2}$$

$$TotalMEM_{require} := \sum_{i=1,m} J_i.mem_{req} \quad \text{with} \quad J_i \in J \tag{3}$$

$$TotalEXP_{require} := \sum_{i=1,m} J_i.EXP_{req} \quad \text{with} \quad J_i \in J \tag{4}$$

$$TotalCPU_{avail} := \sum_{j=1,n} R_j.CPU_{avail}\frac{m_j}{m} \quad \text{with} \quad J_i \in J,\ m_j \leq m \tag{5}$$

$$Totalmem_{avail} := \sum_{j=1,n} R_j.mem_{avail}\frac{m_j}{m} \quad \text{with} \quad J_i \in J,\ m_j \leq m \tag{6}$$

$$TotalEXP_{avail} := \sum_{j=1,n} R_j.exp_{avail}\frac{m_j}{m} \quad \text{with} \quad J_i \in J,\ m_j \leq m \tag{7}$$

The parameter $J_i.CPU_{req}$, $J_i.mem_{req}$, and $J_i.EXP_{req}$ represent the number of required CPUs, the size of needed memory (in MB), the required experts for supervision of the of subjob J_i respectively. Finally, m_j is the number of subjobs which R_j can run simultaneously.

$$rate := \frac{\frac{TotalCPU_{require}}{TotalCPU_{avail}} + \frac{Totalmem_{require}}{Totalmem_{avail}} + \frac{TotalEXP_{require}}{TotalEXP_{avail}}}{3} \tag{8}$$

The removal of the requirement peak is performed by adjusting the start time slot or the end time slot of the subjobs and thus by moving out of the

bottleneck area. One possible solution is shown in Figure 3(a), where either the latest finished time of subjob1 is set to t_1 or the earliest start time of subjob2 is set to t_2. The second way is to adjust both subjobs simultaneously as depicted in Figure 3(b). A necessary prerequisite here is that after adjusting, the latest completion time - earliest start time is larger than the total runtime.

4.2 Initial Solution

The algorithm for determining the initial solution is based on a fail-first heuristic and the forward checking algorithm [10]. According to the Greedy strategy for each subjob under investigation, an RMS with the minimal cost is selected and assigned as described in the following four steps:

1. Determine the subjob in the workflow with the smallest number of RMS candidates, which can execute the subjob according to the provided specification.
2. If the set with RMS candidates for this job is empty, than assign one randomly selected RMS and mark the assignment as conflict. Otherwise, assign the RMS with the minimal cost to that subjob.
3. Repeat the process with the next subjob, until all subjobs are assigned.
4. Resolve the marked conflicts.

In case of conflicts, the Tabu search is modified in order to find a conflict-free and thus a feasible solution. The application of the Tabu search for finding at least one feasible solution is performed analogously with the one described in Section 4.3. Solely the cost function has to be replaced by function based on the number of remaining subjob conflicts in the workflow. If this is possible, the first found feasible solution is declared as initial solution and the mapping process proceeds with the Tabu search for the best solution. Otherwise the owner of the workflow is notified, that the workflow can not be executed according to the given conditions and rejected until new specification is submitted.

4.3 Tabu Search

The Tabu search is well-known method for finding feasible and best possible solutions for NP hard problems. In order to adapt Tabu search to the problem of workflow mapping, following parameters are set. The configuration $R_t \in R$ is any feasible assignment of the set $R = \{(S_i, k_{ij}, t_{il}) | S_i \in S, k_{ij} \in K_i, t_{il} \in T_i\}$. However, at this step for each S_i only one t_{il} is available leading to k_{ij} as a single variable factor. In the next step the neighbourhood to be evaluated is determined. Let R_t be a feasible configuration in R, then the neighbourhood $N(R_t)$ consists of all R'_t, where R'_t is a feasible configuration and R_t differs from R'_t at a single variable. The considered cost function is computed as stated in Section 2.

Subsequently, the created configuration is evaluated. In every iteration the most suitable candidate in the neighbourhood $N(R_t)$ is selected and the feasibility of the new configuration is checked. This is a compute-intensive step, as the

resource and job profiles (see Figure 2) have to be compared before determining the total processing cost. The Tabu tenure goes with each instance of resource in the solution space, thus after each move, the solution space is scanned and the Tabu list is updated. In order to consider very promising configurations, a simple aspiration criterion is applied. The Tabu status of a move is removed if the move leads to the solution with the lower cost than the best configuration found so far. The pseudo-code of the Tabu search algorithm is found in following:

```
begin {
  Get initial solution from previous step
  while(num_move < max){
     Select the best configuration from the current neighbourhood
     Update the Tabu list
     if (cost( ) > cost( ) )
          <---
     num_move+=1    }}
```

5 Performance Evaluation

Planning systems emerged as an effective and power tool for mapping Grid workflows to Grid resources[1]. Therefore, a sample planning-based method will be used as a basis for the comparison of the solution quality. A suitable data models were produced and given as input to a planning system and to the proposed algorithm. The planning method was selected from the list of systems, which participated in the international AI planning contests. Preliminary evaluations showed that solely Metric-FF [9] – well-known planner with very high performance – can handle fully the required Planning Data Description Language (PDDL 2.1) [13] and can solve the workflow mapping problem.

The hardware used for the experiments is rather standard and simple (Pentium 4 2,8Ghz, 1GB RAM, Linux Redhat 9.0). All necessary information about the resource requirements and resource specifications/reservations used in the experiments are on the web site wwwcs.upb.de/cs/ag-kao/en/persons/dang_minh/experiment1.html.

The goal of the experiment is to measure the feasibility of the solution, its cost and the time needed for the computation. For this purpose three different scenarios with increasing complexity level were analysed. The performance measurements started with a low-level experiment and a workflow with 7 subjobs and 3 RMSs as shown in Section 2. The total number of time slots is 138. The problem is coded in PDDL 2.1 and run using a simplified algorithm, where the feasibility check of resources is performed only at the start slot and the end slot of each subjob. The result of this experiment is presented in Table 4.

A second experiment is based on a Grid workflow presented in Figure 4, which includes 10 subjobs, 12 RMSs and 40 time slots. The results are presented in Table 5, where faster computation and slightly lower cost of the solution by the proposed algorithm were observed. Moreover, a significant difference in the

Table 4. Results of the simple level experiment

	Metric-FF		New method	
Subjob	RMS	TS start	RMS	TS start
0	RMS0	35	RMS2	5
1	RMS1	57	RMS2	27
2	RMS1	87	RMS2	57
3	RMS0	66	RMS1	28
4	RMS2	65	RMS1	65
5	RMS0	28	RMS0	62
6	RMS0	101	RMS2	101
Runtime	6.33 sec		< 1 sec	
Cost	171343.52		152870.61	

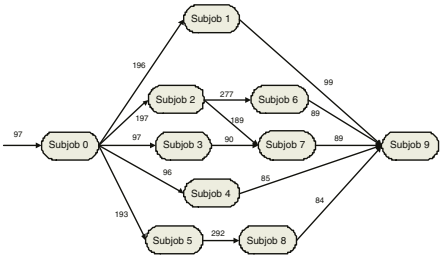


Fig. 4. Intermediate level flow

Table 5. Intermediate level flow

	Metric-FF		New Method	
Subjob	RMS	Start	RMS	Start
0	RMS5	14	RMS1	14
1	RMS0	36	RMS1	34
2	RMS0	33	RMS1	29
3	RMS5	34	RMS1	32
4	RMS5	21	RMS1	20
5	RMS0	21	RMS1	21
6	RMS3	40	RMS2	35
7	RMS0	37	RMS1	35
8	RMS0	32	RMS1	32
9	RMS0	43	RMS1	43
Runtime	67.13 sec		< 1 sec	
Cost	38802.96		38615	

required memory space was noticed, as Metric FF used about 500MB for the computation.

Finally, the last model contains 20 randomly selected jobs together with randomly selected requirements for CPU, memory, etc. Unfortunately, it was not possible to find a feasible solution with Metric-FF, as the existing memory was not sufficient while the proposed algorithm created a solution in a less than a second.

6 Conclusion

This paper presented a method which performs an efficient and precise assignment of workflow subjobs to Grid resources with respect to SLAs defined dead-

lines and subjob dependencies. The performance evaluation showed that the proposed algorithm creates solution of equal or better quality than well-known planning system Metric-FF and needs significantly shorter computation time and less main memory. 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.

Future work sets a strong focus on the network transfer rates, as the transfer time has a major impact on the possible starting and ending time slots for every subjob. If it is possible to predict the network performance, the planning process and the required reservations can be executed more precisely.

References

1. E. Deelman, J. Blythe, Y.Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, S. Koranda: Mapping Abstract Complex Workflows onto Grid Environments, Journal of Grid Computing, Vol 1, no. 1, pp. 25-39, 2003.
2. J. Cao, S. A. Jarvis, S. Saini, G. R. Nudd: GridFlow: Workflow Management for Grid Computing, Proc. 3rd IEEE/ACM Int. Symp. on Cluster Computing and the Grid, Tokyo, Japan, pp. 198-205, 2003.
3. R. Lovas, G. Dzsa, P. Kacsuk, N. Podhorszki, D. Drtos: Workflow Support for Complex Grid Applications: Integrated and Portal Solutions, Proc. 2nd European Across Grids Conference, Nicosia, Cyprus, 2004.
4. J. W. Barnes, J. B. Chambers : Flexible Job Shop Scheduling by Tabu Search,Technical Report Series,Graduate Program in Operations Research and Industrial Engineering. The University of Texas at Austin, ORP96-09, 1996.
5. M. Hovestadt, O. Kao, A. Keller, A. Streit: Scheduling in HPC Resource Management Systems: Queuing vs. Planning, Proc. 9th Workshop on JSSPP at GGF8, LNCS 2862, pp. 1-20, 2003.
6. I. Kacem, S. Hammadi, P. Borne: Approach by Localization and Multi-objective Evolutionary Optimization for Flexible Job-Shop Scheduling Problems, IEEE Transactions on Systems, Man, and Cybernetics. Part C, Vol 32. N1, pp. 1-13, 2002.
7. S. Dauzere-Peres, J. Roux, J.B. Lasserre: Multi-resource shop scheduling with resource flexibility, European Journal of Operational Research, Vol 107, pp. 289-305, 1998.
8. N. Sadeh, Y. Nakakuki: Focused Simulated Annealing Search: An Application to Job Shop Scheduling, Annals of Operations Research, Vol 60, pp. 77-103, 1996.
9. J. Hoffmann: Extending FF to Numerical State Variables, Proc. 15th European Conference on Artificial Intelligence, Lyon, France, 2002.
10. V. Kumar: Algorithms for constraint-satisfaction problems: a survey, AI Magazine, Vol 13 n.1, pp.32-44, 1992.
11. F. Glover: Tabu search Part I, ORSA Journal on Computing, pp. 190-206, 1989.
12. F. Glover: Tabu search Part II, ORSAJournal on Computing, pp. 4-32, 1990.
13. M. Fox, D. Long: PDDL2.1: An extension of PDDL for expressing temporal planning domains, Journal of AI Research, Vol 20, pp. 61-124, 2003.