Chapter 4

ATTRIBUTES FOR COMMUNICATION BETWEEN GRID SCHEDULING INSTANCES

Uwe Schwiegelshohn and Ramin Yahyapour
Computer Engineering Institute, University Dortmund

Abstract

Typically, Grid resources are subject to individual access and usage policies because they are provided by different owners. These policies are usually enforced by local management systems that maintain control of the resources. However, few Grid users are willing to deal with those management systems directly in order to coordinate the resource allocation for their jobs. This leads to a Grid scheduling architecture with several layers. In such an architecture, a higher-level Grid scheduling layer and the lower-level layer of local scheduling systems must efficiently cooperate in order to make the best use of Grid resources. In this chapter we describe attributes characterizing those features of local management systems that can be exploited by a Grid scheduler.

1. INTRODUCTION

Some computational Grids are based on compute and network resources of a single owner, for example in the case of Enterprise systems. But many computational Grids consist of resources with different owners in order to provide a high degree of flexibility and to allow efficient sharing of resources. However, in these cases, most owners are not willing to devote their resources exclusively for Grid use. For instance, a computer may temporarily be removed from a Grid to work solely on a local problem. In order to react immediately in such situations, owners typically insist on local control over their resources, which is achieved by implementing a local management system.

On the other hand, it can be a cumbersome and tedious for a potential Grid user to manually find, reserve and allocate all the resources needed to run an application. To automate this process, a specific Grid management system is needed. Ideally, such a Grid management system includes a separate scheduling layer that collects the Grid resources specified in a job request, checks the considerations of all requirements, and interacts with the local scheduling systems of the individual Grid resources. Hence, the scheduling paradigm of such a Grid management system will significantly deviate from that of local or centralized schedulers that typically have immediate access to all system information. Although it seems obvious that a Grid scheduler may consist of more than a single scheduling layer, many details of an appropriate scheduling architecture have not yet been established.

Nevertheless, it is clear that some layers are closer to the user (*higher-level scheduling instance*) while others are closer to the resource (*lower-level scheduling instance*). And of course those different layers must exchange information.

The information that passes between the scheduling layers of a Grid management system is subject of this chapter. Clearly, the flow of information between the layers is not symmetric: A query flows from a higher-level to a lower-level scheduling instance, while a confirmation takes the opposite direction. However, this observation does not necessarily indicate a fixed or static hierarchy of schedulers. Also the concept of scheduling layers is not restricted to two levels of scheduling instances in general. For instance, consider a large computational Grid consisting of several Virtual Organizations (VOs). Each of those VOs may have a separate Grid management system. A user request needs not to be executed in the VO in which the request was issued. Therefore, the Grid scheduler of one VO passes the request to the Grid scheduler of the second VO. Then this second Grid scheduler interacts with the local scheduling systems of the resources in the VO it works for. Clearly, three scheduling instances are involved in this situation. The Grid scheduler of the second VO is a lower-level scheduling instance with respect to the Grid scheduler of the first VO, while it is a higher-level scheduling instance in the communication process with the local scheduling systems of the second VO. Of course the scheduling instance in the lowest layer is always a local scheduling system.

Nowadays, a variety of different local scheduling systems, for example PBS [PBS], LoadLeveler [IBM01], LSF [Xu01], or Condor [LLM88], are installed on the individual computer systems. Some of these local scheduling systems are presented in this book. See Chapter 11 for information on Maui/Silver, Chapter 12 for information on LSF, and Chapter 13 for information on PBS.

A Grid scheduling layer must exploit the capabilities of these local scheduling systems to make efficient use of the corresponding Grid resources. However, those capabilities are not the same for all local scheduling systems due to system heterogeneity. Therefore, we introduce attributes to describe, in general, the features of a lower-level scheduling instance that can be used by a higher-level scheduling instance. The attributes are also the first step towards a classification of Grid-relevant properties of local schedulers. Further, they can be used to indicate what future enhancements of local scheduling systems may improve their Grid-readiness.

Note that attributes can be combined: if more than two levels of scheduling instances are used as described above, then a scheduling instance in the middle collects attributes from possibly several lower-level scheduling instances, combines them, and provides those combined attributes to a higher-level scheduling instance. As the combination of attributes is not subject of this chapter, we restrict ourselves to two scheduling instances in the following sections.

Although we have stated above that the attributes can be used to evaluate Grid-readiness of a scheduling system, we want to strongly emphasize that the features described by the attributes are neither an obligation nor a limitation for the design of a lower-level scheduling system. For instance, some features are not relevant for certain resource types and need not to be considered in the corresponding local schedulers. The presence of an attribute simply indicates that the specified feature is provided by the lower-level scheduling instance and can be used by a higher-level scheduling instance. On the other hand, if such an attribute is not present for a lower-level scheduling instance then the higher-level scheduling instance cannot use the specified feature.

Also note that it is not the purpose of this chapter to address mechanisms for the communication between scheduling layers. Similarly, we do not define the structure and syntax for the description of resources. This kind of information can be accessed through other Grid information services which are currently subject to a standardization process in the Information Systems and Performance area [ISP] and the Architecture area [ARC] of the Global Grid Forum [GGF]. Consequently, there are no attributes in this description to determine, for instance, the set of resources for which a lower-level scheduler is responsible.

In the next section we present an example a Grid application. This example is used to illustrate the meaning of some of the attributes which are listed in the following sections. These are classified into 4 groups:

1 Attributes used when accessing the available scheduling information (Section 3),

2 Attributes used when requesting the resources (Section 4),

3 Attributes used when querying for allocation properties (Section 5), and

4 Attributes used when manipulating the allocation execution (Section 6).

The concepts described in this chapter are a result of the work done by the Scheduling Attributes Working Group of the Global Grid Forum, and they have been published as a Grid Forum Document [SY01]. The Working Group stated however, that this list of attributes while sufficient may not yet be complete.

2. TYPICAL USE OF A GRID

For a purpose of illustrating some of the attributes we consider a simple example where a job is executed on a computational Grid that includes the following resources with different owners:

- Several clusters of workstations,
- A visualization cave [LJD+99],
- A database, and
- A bandwidth broker for the network that connects other resources.

Our example job is a workflow job consisting of three stages:

1 Data transfer from the database to a workstation cluster,

2 Preprocessing of the basic data on the workstation cluster in batch mode and generation of some temporary data, and

3 Processing of the temporary data on the workstation cluster and, in parallel, online visualization of the results in the visualization cave.

A user generates a request and submits it to the Grid scheduler that tries to find a suitable allocation of resources. As not all of the requested resources are available at a single site, the job requires multi-site processing in Stage 3. To this end the Grid scheduler requires information about the properties of the involved local scheduling systems. Scheduling attributes described in this chapter provide this information.

Before addressing the task of the Grid scheduler in detail we define an *allocation* to be an assignment of resources to a request. An allocation is tentative until it is executed, that is, until resources are actually consumed. A *schedule* gives information about planned or guaranteed allocations. Such a guarantee of an allocation means that there is a guaranteed assignment of resources. This does not necessarily guarantee the completion of a job, as the actual job processing time may exceed the requested allocation time.

Note that there is a strict dependence between the stages in the example above. In Stage 1 the database, the workstation cluster and sufficient bandwidth in the network must be available at the same time. The length of this stage is mainly determined by the network bandwidth. However, this step is not necessary if the database is deployed on the same cluster that is used in Stages 2 and 3. The efficient execution of Stage 3 requires concurrent access to the workstation cluster, the visualization cave, and the network. For Stage 3, the visualization cave and the network require concurrent access to the workstation cluster. Moreover, the network must provide the bandwidth required by the job.

The Grid scheduler used in the example tries to obtain an allocation with a *guaranteed completion time* for Stage 2 in order to make an *advance reservation* for the visualization cave and the network in the next stage. In Stage 3, the Grid scheduler requests an *exclusive allocation* that will *run-to-completion* on the processing resources to prevent any negative influence of other independent jobs on the visualization. Finally, the Grid scheduler asks for a *tentative schedule* of the visualization cave and the network in order to find a sufficiently large time slot during which all the resources required for Stage 3 are available.

This example shows how a Grid scheduler can use certain features of local scheduling systems to generate an efficient schedule. The availability of those features is described by attributes. To this end, a Grid scheduler needs to know which features are available for a local scheduling system.

3. ACCESS TO AVAILABLE SCHEDULING INFORMATION

The first set of attributes addresses local schedule information that is made accessible to a higher-level scheduling instance.

3.1 Access to a Tentative Schedule

Some local management systems are able to return on request the complete schedule of current and future allocations. This information allows a Grid scheduler to efficiently determine suitable timeslots for co-allocation of resources in multi-site computing, as needed for Stage 3 of our example. Note that a local management system may not return the complete schedule due to architectural limitations or due to system policy restrictions.

Even if the complete schedule of a local resource cannot be made available, the local management system may return some information that can be used by a higher-level scheduler. The type and the amount of this information can be specified with additional options of this attribute. As an example, a local management system may provide an authorized higher-level scheduling system with the projected start time of a specified allocation.

In other systems, the returned schedule information is subject to change as there may be a different access to those local resources that is not controlled by the lower-level scheduling instance.Then this information has a limited reliability, which can also be described by an option of this attribute.

3.2 Exclusive Control

The local scheduler has exclusive control over its resources, that is, no allocations can be scheduled on those resources without using this scheduler instance. In this case, the reliability of schedule information from the local scheduler is very high as there can be little outside interference. Then, a Grid scheduler can better exploit the knowledge about a local schedule to generate efficient allocations for complex job requests.

3.3 Event Notification

Some unforeseeable events may influence a local schedule, for example a sudden failure of a resource or the early termination of a job. A local scheduler may pass information on to a higher-level scheduling instance if this scheduling instance has subscribed to the appropriate event notification. If so, the higher-level scheduling instance can quickly react and reschedule some allocations. Those events may also include notification of an allocation change, for example cancellation or changed execution time, as well as information about modified resource conditions. This attribute states that the local scheduling system supports event notification. However, note that it does not specify any event types nor define interfaces to query supported event types.

4. REQUESTING RESOURCES

Local scheduling systems do not only differ in the type of functionality they provide but also in the type of information they need when resources are requested. This information is characterized by the second set of attributes.

4.1 Allocation Offers

Local management systems may support the generation of potential resource allocations on request. In our example, several workstation clusters with different characteristics may be available for Stage 1. A Grid scheduler can determine the best suited cluster to use by considering different offers returned from the corresponding local schedulers, especially with regards to the location of the database and the available bandwidth on the network. To this end, the local scheduling instance may offer different resource allocations to the Grid scheduler.

Local management systems can further be classified according to the number of offers they generate for a request. For instance, local management systems may provide several offers for a request with possibly overlapping allocations. It is then up to the higher-level scheduler to select a suitable offer from them. In our example, the local scheduling system may generate different offers that differ in price or execution time. The Grid scheduler may use this feature for the multi-site allocation in our example in Stage 3 where corresponding allocations must be found for different resources. If several allocation offers are provided, the Grid scheduler has more flexibility to select a suitable offer to allow for an efficient and concurrent execution of a job.

4.2 Allocation Cost or Objective Information

The local management system returns cost or objective information for an allocation. In case of several allocation offers, a Grid scheduler can, for instance, use this information for the evaluation of different offers. The cost for a specified allocation usually relates to the policy that is applied by the lower-level scheduling instance. This represents the scheduling objective of the owner of the resource. It is obvious that costs for an allocation will be an important criterion for a Grid scheduler in selecting a suitable allocation for a job.

4.3 Advance Reservation

This feature is of great help for any Grid scheduler that must schedule multi-stage or multi-site jobs. The Grid scheduler in the example can ensure that the network connection between visualization cave and workstation cluster is sufficient in Stage 3 of our example by obtaining a reservation for appropriate bandwidth.

An Advance Reservation API is already subject of a document from the Global Grid Forum [RS02]. With such an interface a higher-level scheduling instance is able to access the Quality of Service features of Grid resources.

4.4 Requirement for Providing Maximum Allocation Length in Advance

As already mentioned, some local management systems require that a maximum allocation length is provided together with the resource request. In our example, the application in Stage 2 does not require any user interaction and is therefore a batch job. For this type of job, an efficient scheduler needs information about the maximum execution length. Historically, resource requests have often been submitted without additional information about the amount of time that the resources would be used. These jobs are started and run until completion. However, current scheduling algorithms, for example backfilling [Lif96], require additional information on the maximum allocation length in order to generate an efficient schedule.

4.5 Deallocation Policy

A deallocation policy for pending allocations applies to some local management systems. Such systems establish requirements that must be met by the user or the higher-level scheduling instance to keep the allocation valid. The requirement that an allocation must be repeatedly confirmed until the start of its execution is an example of such a policy. For instance, in our example the visualization cave might require complying to such a policy to ensure a high degree of utilization of the cave itself. Of course, these policies must be further specified to allow the higher-level scheduler to keep its allocations. Attribute 5.1: Revocation of an Allocation, in comparison, describes the reliability of an allocation as given by the local management system.

4.6 Remote Co-Scheduling

A local management system may allow for co-scheduling where the actual resource allocation and the schedule are generated by a higher-level scheduling instance. In this scenario, the local management system provides interfaces to the higher-level scheduling instance to delegate certain modifications to the local schedule. This includes the generation and cancellation of allocations on the local schedule by the higher-level scheduler. In our example, the Grid scheduler can use this property to quickly co-allocate resources for the multi-site allocation in Stage 3. In this case, some part of the authority of a lower-level scheduling instance is delegated to the higher-level scheduling instance.

4.7 Consideration of Job Dependencies

A local scheduler may take dependencies between allocations into account if they are provided by the higher-level scheduling instance. For instance, in case of a complex job request, the lower-level scheduler will not start an allocation if the completion of another allocation is required and still pending. These dependencies are sometimes also referred to as the workflow of a job. For Grid jobs, different steps often depend on each other. In our example, the dependencies of Stage 1 and 2 may be considered by a local management system as precedence relations.

Note that the complete workflow graph of the job may not be initially available to the Grid scheduler since the results of some job steps may influence the following steps. In this case, some of the tasks of a higher-level scheduling instance are delegated and managed by local management systems.

5. ALLOCATION PROPERTIES

Allocations of resources may differ with respect to timing and reliability. For instance, a reliable allocation is important for multi-stage and multi-site jobs. Therefore, we use a third set of attributes to describe allocation properties.

5.1 Revocation of an Allocation

Some resource owners may assign a low priority for Grid use. In such a case local management systems may reserve the right to revoke an existing allocation until the resources are actually being used. Here we address only a revocation of an allocation that cannot be prevented by actions of the higher-level scheduling instance. Therefore, the revocation is independent of any process that must be executed by the user system or a higher-level scheduling instance to prevent deallocation according to the deallocation policy of a local scheduling system, see Attribute 4.5: Deallocation Policy. For our example, the cluster in Stage 3 may be withdrawn from Grid use or may be used to execute another job with a higher priority. In this case, the allocation was not guaranteed. Note that a local management system may support both revocable and irrevocable allocations.

5.2 Guaranteed Completion Time of Allocations

Some local management systems guarantee the completion time of an allocation. While the system reserves the right to individually determine the actual start and end time of an allocation within a given time frame, it guarantees that the requested resource allocation will be executed before a given deadline. For instance, scheduling systems that are based on the backfilling strategy [Lif96, FW98] can provide information on the maximum completion time of a newly submitted resource request while not being able to predict the actual starting time. In those cases it is necessary for the user to provide information in advance on the maximum job run-time, see also Attribute 4.4: Maximum Allocation Length. With this information, an upper bound for the completion time of the jobs can be calculated by the local scheduling system. In our example, the completion time of Stage 2 is available to the Grid scheduler and used to coordinate the start of Stage 3.

5.3 Guaranteed Number of Attempts to Complete a Job

Some tasks, for instance, the transfer of data over a network link, cannot always be completed on the first try. In those situations it is useful if the local management system guarantees a minimum number of attempts before informing the user of the failure or requiring a re-scheduling from the higher-level scheduling system. This reduces the work of the higher-level scheduling instance. In our example, this feature can be used in Stage 1 to ensure that data has been successfully transferred to the workstations cluster.

5.4 Allocations Run-to-Completion

The local management system will not preempt, stop, or halt a job after it has been started. In particular, once started, the allocation will stay active on the given resources until the end of the requested allocation time frame or the completion of the job. This information about the way a job is executed helps a higher-level scheduling instance to more reliably predict the completion time of a running job, and is therefore useful for the coordination of concurrent allocations on different resources. In our example, the Grid scheduler selects a workstation cluster with this property for Stage 3 to ensure the uninterrupted execution of the job concurrently with the allocation of the visualization cave.

5.5 Exclusive Allocations

The allocation runs exclusively on the provided set of resources, that is, the resources are not time-shared, and this allocation is not affected by the execution and resource consumption of another allocation running concurrently. Similar to the previous attributes, this information may be helpful to estimate the required run-time of a job on a specific resource. In a time-shared scenario, the run-time of a job may significantly deviate from an estimate due to the interference of other jobs on the same resource. In our example, the efficient use of the visualization device depends on the reliable performance of the workstation cluster in Stage 3. Therefore, the Grid scheduler can require that the workstation cluster is exclusively available during the allocation in this stage.

5.6 Malleable Allocations

Some local management systems support the addition or removal of resources to/from applications during run time [FRS+97]. In this case, the size of an allocation can change during the execution of a job. If such a modification of the allocation is not controlled by the higher-level scheduling instance it has a similar impact on the reliability of the estimated run-time of a job as time-sharing of resources. In addition not all applications may be able to handle a reduced allocation size. Therefore, those resources are not suitable for all job requests.

This problem is less severe if the local management can only increase the resource set of an allocation during run time, that is, if resources are not taken from a job. Feitelson et al. [FRS+97] describe such an allocation by the term *moldable*.

6. MANIPULATING THE ALLOCATION EXECUTION

For multi-stage or multi-site Grid jobs, unexpected changes of an allocation may influence other allocations and require quick re-scheduling. In this case it may be beneficial if the higher-level scheduling instance can directly modify other running allocations in order to better coordinate the execution of the jobs. The fourth set of attributes describes actions that a higher-level scheduling instance may directly initiate on resources without involving the local management system.

6.1 Preemption

Some local management systems allow for temporary preemption of an allocation by a higher-level scheduling instance. In this case, the corresponding application is stopped but remains resident on the allocated resources and can be resumed at a later time [WFP+96]. Such preemption is not synonymous with the preemption in a multitasking system that typically happens in the time range of milliseconds. It indicates only that the local management system offers the ability to remotely initiate a preemption of an allocation, for example to temporarily free resources for other use or to synchronize two allocations on different resources. Moreover, this kind of preemption does not necessarily require checkpointing. For instance, if machines of the allocated resource set go down, it is not guaranteed that the preempted allocation can be resumed on other resources.

6.2 Checkpointing

Other local management systems support the checkpointing of a job. In this case, a checkpoint file of the job is generated to allow for a later continuation of the job from the checkpoint. The generation of the checkpoint file can be independent of any actual preemption of the job. The checkpoint file may also be migratable to other resources, but this feature is not mandatory. Note that different types of checkpointing paradigms exist, and on some systems not all kinds of jobs can be checkpointed. Here, the application must cooperate and support the checkpointing feature.

6.3 Migration

Some local management systems support the migration of an application or part of an application from one resource set to another set. Hence, an application can be stopped at one location and the corresponding data is packed such that the application can be moved to another location and be restarted there. This migration process is only of relevance to a higher-level scheduling instance if it can initialize and control this process. Therefore, the attribute does not include any migration of allocations within the domain of the lower-level scheduling instances that is not influenced by the higher-level scheduling instance, see Attribute 5.6: Malleable Allocations. Also, the migration attribute does not necessarily require the presence of the Attribute 6.2: Checkpointing. However, similar to the remark on checkpointing, explicit support for migration by the application may be required to use this feature.

6.4 Restart

Here, the local management system supports the receiving and the restart of a stopped and packaged application from another resource set.

On some systems, the continuation of a job may be supported as so called Checkpoint Restart. Then, a restart is only possible from a checkpoint file, see Attribute 6.2: Checkpointing. That is, the system does not support migration on the fly.

7. CONCLUSION

Ideally, computational Grids consist of many different resources but look like a single system from a user's point of view. A similar structure can also be found in a Grid scheduler, an important component of any computational Grid. This has led to the concept of a layered scheduler consisting of higher-level scheduling instances that are close to the user and lower-level scheduling instances that are close to the resources. Due to the heterogeneity of available local scheduling systems that form the lowest layer of a Grid scheduler, information about the properties of those systems must be provided to the higher-level scheduling instances. This is done with the help of attributes which are presented in this chapter. Those attributes may also be helpful for the future development of local scheduling systems for Grid resources. Ideally, the existence of an attribute should not be determined by the features of a local scheduling system but by properties of the resource or by system policies.

Some chapters in the book address actual implementations of resource management systems with Grid support. More specifically, the MAUI/Silver scheduler is described in Chapter 11, Platform's LSF is presented in Chapter 12, and the Portable Batch System (PBS) can be found in Chapter 13. These systems represent the current state of the art for resource management systems that can be found in today's computational Grids.

Acknowledgments

To a large degree this chapter describes the results of discussions in the Scheduling Attributes Working Group of the Global Grid Forum, see also [SY01]. The authors would like to thank all contributors to this effort.