ROMAI J., **2**, 1(2006),
1–7

MOTIVATION FOR MULTICRITERIA STRATEGIES IN DISTRIBUTED SYSTEM RESOURCE MANAGEMENT PROBLEMS

Costel Aldea

University "Transilvania" of Brasov

costel@info.unitbv.ro

Abstract In the resource management systems the following aspects must be considered: the virtual organization policies and the application requirements and, in some cases, the user preferences. According to these multicriteria strategies for distributed resource management should be formulated and considered. Some available multicriteriea optimization techniques and methods are discussed. The dynamic behaviour, uncertainty and incomplete information are important aspects of the distributed system nature that gives the core of the resource management process. Some aspects of the resource management process are: complete information about the accesible resource, applications requirements and user preferences, association of the tasks with the resource in the best possible ways, supporting the changes in the distributed system with respect to distributed application control.

1. INTRODUCTION

In order to describe the resource management process in a distributed system it is important to know the participants to this process. There are also many aspects in modelling their preferences in order to obtain a good association between the tasks and the resources. Nevertheless, one must observe other techniques for satisfying the requirements given by the dynamic behaviour of the tasks related to the resources as well as for the uncertainty and incomplete information.

Some basic definition and notions related to multicriteria approach are introduced. The aim is to motivate the importance of multicriteria strategies in distributed systems resource management.

The performance and the high-throughput of the resource management implies the following comparation between two main resource management strategies: application level scheduling and job scheduling. It is obvious that the combination between those two strategies can bring together all the benefits of the both strategies using multicriteria strategies approach and the suport of artificial inteligence techniques. An analysis of possible criteria that can be used in distributed system resource management process is presented

and how the preference models can be exploited in order to assign resources to the tasks in the most appropiate way is shown.

2. MULTICRITERIA APPROACH

A formulation for the multicriteria decision problem is made and some basic definition and various ways of modelling decisions maker's preferences are described.

Multicriteria decision problem. If there are several different ways to achieve a goal, a decision problem needs to be solved. More then that, to reach the goal, optimized choices of the best method must be used. These different ways are decision actions. They are also called the compromise solutions or schedules.

In general the main goal of the multicriteria decision making process is to build a global model of the decision makers' preferences and exploit them in order to find the solution that is accepted by a decision maker with respect to the values on all criteria. In order to accomplish this, preferential information concerning the importance of particular criteria must be obtained from a decisionmaker.

If these preferences are not available, then the only objective information is the dominance relation in the set of decision actions.

Basic definitions. For the choice of the compromise solution the concepts of non-dominated and Pareto optimal are used. There are two important spaces to consider: decision variable space and the criteria space. The former contains possible solutions along with values of their attributes. Let us denote by D the set of solutions. The set D can be easily mapped into its image in the criteria space, which creates the sets of points denoted by C(D). The functions $\phi(x)$ represent values of criteria and their number is denoted by k.

Definition 2.1 *(Pareto dominance). The point* $z \in C(D)$ *dominates* $z^{'} \in C(D)$, *denoted as* $z \succ z^{'}$, *if and only if* $(\forall)\, j \in \{1, \cdots, k\}, z_j \geq z_j^{'} \wedge (\exists)\, i \in \{1, \cdots, k\} : z_i > z_i^{'}$ *(*$z^{'}$ *is partially less than z). Thus, one point dominates another if it is not worse with respect to all criteria and it is better for at least one of them.*

Definition 2.2 *(Pareto optimality). The point* $z^{'} \in C(D)$ *is said to be non-dominated with respect to the set* $C(D)$ *if there is no* $z \in C(D)$ *that dominates* $z^{'}$. *A solution is x Pareto optimal (efficient) if its image in the criteria space is non-dominated.*

Definition 2.3 *(Pareto-optimal set). The set of* P^* *of all Pareto-optimal solutions is called the Pareto optimal set. Thus, it is defined as*
$P^* = \left\{ x \in D \,\middle|\, \neg\,(\exists)\, x^{'} \in D : z^{'} \succ z \right\}$, *where* $z^{'}$ *and* $z \in C(D)$

Definition 2.4 *(Pareto Front). For a given Pareto optimal set* P^* , *Pareto front* (PF^*) *is defined as* $PF^* = \{z = \{f_1 = z_1, \cdots, f_k(x) = z_k\} \,|x \in P^*\}$. *A*

Pareto front is also called a non-dominated set because it contains all non-dominated points.

Preference models. A preference model defines a preference structure in the set of decision actions. Preference models can aggregate evaluations criteria into:
-functions (e.g weighted sums);
-statements (e.g. binary relation: action a is at least as good as action b);
-logical statements (e.g. decision rules: if condition on criteria then decision, where if condition on criteria is the conditional part of the rule and then decision is the decision part of the rule).

3. MOTIVATION FOR MULTIPLE CRITERIA

Multicriteria approaches focus on a compromise solution (in this case compromise schedule). In this way one can increase the level of satisfaction of many stakeholders of the decision making process and try to combine various points of view, rather than provide solutions that are very good from only one specific perspective as is curently common in other distributed system resource management approaches (grids). Such specific perspective result in different, often contradictory, criteria (along with preferences) and make the process of mapping jobs to resources difficult or even impossible. Consequently all the different perspectives and preferences must be somehow aggregated to please to all participants of the distributed system resource management process.

Various stakeholders and their preferences. Distributed systems scheduling and resource management potentially involves the interaction of many human players (although possibly indirectly). These players can be devided into three classes:
- end users making use of distributed system applications and portals;
- resource administrators and owners;
- virtual organization (VO) administrators and VO policy makers.

One goal of the resource management process is to automate the scheduling and resource management process in order to maximize stakeholders participations in the entire process. This is why the final decision is often delegated to such systems. By decision maker notion one means a scheduler and all human players listed above are stakeholders of a decision making process. This model assumes a single artificial decision maker in a VO. However, in real distributed systems, multiple stakeholders (agents) may often act according to certain strategies, and the decision may be made by means of negotiations between these agents. Such an approach requires distributed decision making models with multiagent interactions [1], which will not be discussed here.

One refers to the assignment of resources to tasks as a *solution* or a *schedule*. Since one considers many contradictory criteria and objective functions,

finding the optimal solution is not possible. The solution that is satisfactory from all the stakeholders points of view and that takes into consideration all criteria is called the *compromise solution.*

The need for formulation of distributed systems resource management as a multicriteria problem follows from the characteristics of the distributed system environment itself. Such an environment has to meet requirements of different groups of stakeholders listed at the beginning of this section. Thus, various points of view and policies must be taken into consideration, and results need to address different criteria for the evaluation of schedules. Different stakeholders have different, often contradictory, preferences that must be somehow aggregated to please all stakeholders. For instance, administrators require robust and reliable work of the set resources controlled by them, while end users often want to run their applications on every available resource. Additionally, site administrators often want to maintain a stable load of their machines and thus load-balancing criteria become important. Resource owners want to achieve maximal throughput (to maximize work done in a certain amount of time), while end users expect good performance of their application or, sometimes, good throughput of their set of tasks. Finally, VO administrators and policy makers are interested in maximizing the whole performance of the VO in the way that satisfies both end users and administrators.

Different stakeholders are not the only reason for multiple criteria. Users may evaluate schedules simultaneously with respect to multiple criteria. For example, one set of end users may want their applications to complete as soon as possible, whereas another one try to pay the minimum cost for the resources to be used. Furthermore, the stakeholders may have different preferences even inside one group. For example, a user may consider time of execution more important than the cost (which does not mean that the cost is ignored).

Job Scheduling involves the use of one central scheduler that is responsible for assigning the distributed system resources to applications across multiple administrative domains. There are three main levels: a set of applications, a central scheduler, and the distributed system resources. In principle, this approach is in common use and can be found today in many commercial and public-domain job-scheduling systems. A detai1ed analysis and their comparative study can be found in [2].

Note that the dynamic behaviour seen in distributed systems environments is most often the result of competing jobs executed on the resources. By having a control over all the applications, as happens in the job scheduling approach, a central scheduler is able to focus on factors that cause variable, dynamic behavior, whereas application-level schedulers have to cope with this effects.

Application -Level Scheduling. With application-level scheduling, applications make scheduling decisions themselves, adapting to the availability of

resources by using additional mechanisms and optimizing their own performance. That is, applications try to control their behavior on the distributed system resources independently. Note that all resource requirements are integrated within applications.

Such an assumption causes many ambiguities in terms of the classic scheduling definitions. When one considers classic scheduling problems, the main goal is to effectively assign all the available task requirements to available resources so that they are satisfied in terms of particular objectives. In general, this definition fits more the job scheduling approach, where job scheduling of systems take care of all the available applications and map them to resources in a suitable way. In both cases application-level scheduling differs from the job-scheduling approach in the way applications compete for the resources. In application-level scheduling, an application schedule have no knowledge of the other applications, and in particular of any scheduling decisions made by another application. Hence, if the number of applications increases (which is a common situation in large and widely distributed systems like a Grid), one observe worse scheduling decisions from the system perspective, especially in terms of its overall throughput. Moreover, since each application must have its own scheduling mechanisms, information processes may be unnecessarily duplicated (because of the lack of knowledge about other applications' scheduling mechanisms).

Hence, a natural question arises: why does one need the application level scheduling approach? In practice, job scheduling systems do not know about many specific internal application mechanisms, properties, and behaviors. Obviously, many of them can be represented in the form of before mentioned hard constraints, as applications requirements, but still it is often impossible to specify all of them in advance. More and more, end users are interested in new Grid-aware application scenarios. New scenarios assume that dynamic changes in application behavior can happen during both launch-time and run-time. Consequently, Grid-aware applications adapt on-the-fly to the dynamically changing Grid environment if a pure job scheduling approach is used. The adaptation techniques, which in fact vary with classes of applications, include different application performance models, and the ability to checkpoint and migrate.

Consequently, job scheduling-systems are efficient from high-throughput perspective *(throughput based criteria)* where as application-level schedulers miss this important criterion, even though internal scheduling mechanisms can significantly improve particular application performance *(application performance based criteria)*.

Hard and soft constraints. By extending present job-scheduling strategies (especially to the war application requirements are expressed), one is able to deal with both throughput and application-centric needs. As noted in the

previous section, a central scheduler receives many requests from different end users to execute and control their applications. Requests are represented in the form of *hard constraints*, a set of applications and resources requirements that must be fulfilled in order to run an application or begin any other action on particular resources. These constraints are usually specified by a *resource description language* such as RSL in the Globus Toolkit, ClassAds in Condor, or JDL in DataGrid. Simply stated, a resource description language allows the definition of various bald constraints, such as the operating system type, environment variables, memory and disk size, and number and performance of processors, and generally varies with classes of applications on the distributed system. Many extensions to the resource description languages are needed to represent *soft constraints*, the criteria for resource utilization, deadlines, response time, and so forth needed when many stakeholders are involved. Unfortunately, soft constraints are rarely expressed in the resource description language semantics, and consequently they are omitted in many resource management strategies.

The main difference between hard and soft constraints is that hard constraints must be obeyed, where as, soft constraints should be taken into consideration in order to *satisfy* all the stakeholders as far as possible. In other words, *one is able to consider soft constraints if and only if all hard constraints are met.* Yet, soft constraints are actually criteria in the light of decision making process, and they are tightly related to the preferences that the stakeholders of the process want to consider.
We have indicated that resource management in a distributed environment requires a multicriteria approach. However, proposition of a method copying with multiple criteria during the selection of the compromise schedule does not solve the resource management problem entirely.

References

[1] Th. W. Sandholm, *Distributed rational decision making*, Multiagent Systems: a modern Approach to Distributed Artificial Intelligence, MIT Press, 1999, 201-258.

[2] G. Kris, El-Ghazawi Tarek, N. Alexandridis, F. Vroman, J. R. Radzikowski, Preeyapong Samipagdi, S. A. Suboh, *An empirical comparatie study of job management systems. Concurrency: Practice and Experience*, to appear, 2003.

[3] G. Allen, D. Angulo, T. Goodale, T. Kielmann, A. Merzky, J. Nabrzyski, J. Pukacki, M. Russel, T. Radke, E. Seidel, J. Schaf, I. Taylor, *GridLab:Enabling application on the grid,* in Proceedings of the Third International Workshop on Grid Computing (Grid2002), November 2002.

[4] I. Foster, C. Kesselman, *The Globus Project: a status report*, Proceedings of the Seventh Heterogenous Computing Workshop, 1998.

[5] C. Aldea, *Measuring the performance for parallel matrix multiplication algorithm*, Proceedings of the International Conference on Theory and Applications in Mathematics and Informatics, Alba-Iulia, Romania, 2005.

[6] *European DataGrid Project*, http://www.eu-datagrid.org.