Resource Selection Based on Application Features

Rodrigo Grumiche, Patrícia Vilain, M. A. Dantas
Department of Informatics and Statistics
Federal University of Santa Catarina
Florianópolis - Santa Catarina - Brazil
Email: {grumiche,vilain,mario}@inf.ufsc.br

***Abstract*—Grid middlewares and meta-schedulers provide job submission services for applications to be run on the computational resources of a grid. The resource selection mechanism on the meta-schedulers is usually based on syntactic or semantic descriptions from required hardware and software indicated by users. However this approach requires that users have a good knowledge about details of computational resources characteristics and how these affects the user application. Applications have some structural and behavioral features that restrict what resources are able to run it. And those features also may affect the performance of applications when running on resources, given their characteristics. In this paper is proposed an alternative heuristic approach for selecting resources in multi-cluster grid environments, where users describe the application features while the selection mechanism evaluates the levels of adequacy of available resources. The GRADI middleware is utilized as a testbed for evaluating improvements on the quality of experience of the user, on the execution time of the application and on the overall improvement on utilization of resources on the grid.**

I. Introduction

Computational grids are a complex infrastructure built on hardware and software that provides dependable, pervasive, and inexpensive access to high-end computational capabilities [1]. Grid middlewares are part of that infrastructure and they are responsible for hiding the complexity and heterogeneity of a grid environment from the user.

Grid middlewares [2] provide basic infrastructure services that allow users to manually find a subset of computational resources, select some of them based on their perception of what resources characteristics are better to run their application and finally submit a job to be run on those resources.

Meta-schedulers like VIOLA [3], CSF4 [4] and GridWay [5] were developed to improve user experience in the task of resource selection and job submission of applications to be run on grid infrastructure. First the user writes a syntactical description [6] of hardware and software requirements that one or more resources has to comply to be a candidate to run the user application. Then based on the user input, the meta-scheduler searches for resources that matches the specification, selects a subset of the matched resources based on some criteria and finally submits the application job to the selected resources. The issue with the syntactical description of resources is that a grid computing environment is composed of multiple Virtual Organizations, and each one may describe resource characteristics in their own particular form. The usage of ontologies for a semantic description of computational resources as proposed by [7] and the usage of semantic matching mechanism able to integrate different ontologies of resource description used by different Virtual Organizations to do resource selection [8] are solutions for this issue.

The usage of syntactic or semantic descriptions to select a resource for the job scheduling on a grid requires that users have a good knowledge about the details of computational resources characteristics and how these details affect ability to run applications. Further, an application has structural and behavioral features that affect the ability and performance of computation resource to run the application and that cannot be directly represented by such descriptions.

In this paper we propose an alternative approach to the problem of resource selection and job submission, based on heuristics about structural and behavioral features of an application and levels of adequacy to resource characteristics on a multi-cluster grid environment. We utilize ontologies to describe application features and fuzzy logic to define the levels of adequacy of a resource to run an application based on the application features and resource characteristics. So instead of describing technical requirements that need to be met by resources, the user describes application features. Then, the co-reservation and/or co-allocation of resources is done by applying the resource adequacy level heuristics to the given application features.

The implementation of this approach is been integrated with the GRADI middleware [9]. It is expected that the approach will simplify the task of job submission and improve application execution speed on a multi-cluster environment. Offering a better quality of experience [10] to the users of the grid.

II. Related Work

The Intelligent Scheduling Service (ISS) [11] research work is part of the Intelligent Application Oriented Scheduling for HPC Grids (IANOS) and it aims to detect what kind of resources are the best to run an application. The definition of the best resources is based on a cost functional model that takes as input different variables: historical application performance execution on resources normalized as the division of CPU time and communication times, performance requirements of the user, electricity costs, communication costs, software licenses and others [12]. However, it does not avoid the necessity of resource specification for a job submission, neither allows users to explicitly inform the application features. To be effective, it requires that a database of historical performance data of applications is built and maintained for each available resource on the grid. Instead of evaluating monetary costs of energy, software licenses and communication, and maintaining historical performance data of applications, our approach allows users to select resources by identifying the application features and then using heuristics to improve resource selection.

A relevant contribution for the proposed approach is the work of [13]. It deals with the issue of co-allocation of resources on a homogeneous multi-cluster environment, based on the saturation level of communication links that interconnect the clusters. The co-allocation process uses a Maximum Bandwidth Adjacent Strategy (MBAS) to select clusters with the least communication saturation level to run a job. The thresholds of the saturation levels are adjusted dynamically by the Adaptive Threshold Control System (ATCS) [14] based on the overall system state and the communication profile of the application to be run as a job in multiple clusters, using fuzzy logic. The ATCS evaluates two communication profiles which are described by users: master-slave and all-to-all. The goal here is to improve the co-allocation of resources instead of resource selection, because the environment is homogeneous. It is similar to our approach in taking a specific application feature as a parameter for the ATCS co-allocation strategy: the master-slave and all-to-all communication profile. What we propose here is to tackle the resource selection problem by making it based only on the application features. The selection mechanism discussed on this paper is based on the fuzzy control model proposed by [14].

Another important contribution is the work of [15] and [9] on resource selection, co-reservation, co-allocation and the development of the GRADI grid middleware. The GRADI platform does semantic resource selection using ontologies, co-reservation and co-allocation of computing resources on a grid environment. It integrates the semantic description and selection of resources using ontologies [7], integration of multiple ontologies with the goal of resource selection by semantic matching [16], co-allocation of resources applying the strategies proposed by [14] and implements the co-reservation approach proposed by [9]. Our selection mechanism is been developed on the GRADI platform and utilizes its ontological services. But its resource selection mechanism is based on ontological description of computational resources. Our approach extends the GRADI middleware by allowing the resource selection to be done based on application features.

III. Application Features

In [17] it is presented the concept of design patterns. The design patterns are classified based on their goal: creational, structural and behavioral. Creational patterns deal with object instantiation; structural patterns deal with the composition of the classes and objects; and the behavioral patterns characterize how the classes and objects communicate and distribute responsibility. In [18] the Unified Modeling Language(UML) diagrams are classified based on their features: structural and behavioral. Structural features represent static features, while behavioral features represent dynamics features.

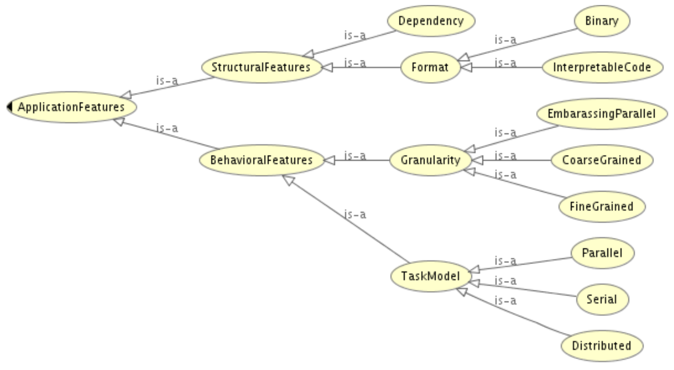

Figure 1. Part of the application features ontology

The figure 1 shows part of our proposed ontology that classifies application features in two groups, similar to the design pattern and UML diagram classification:

- Structural: features emerging from the body of the application.
- Behavioral: features emerging from the execution of the application.

The structural features commonly restrict which resources are able to run applications. For example, an application can exist on a binary form that resulted from the compilation of code written on a programming language to an Instruction Set Architecture (ISA) to be run on a family of processors, or as byte-code to be executed by some type of virtual machine. It could also be a sequence of instruction lines to be executed by an interpreter. It may depend on other applications or libraries, and may be built to be executed on a specific operating system.

The behavioral features are those visible when the application is executed, showing how they use the computational and communication resources to solve some specific problem. The parallelism and/or distributed capabilities of an application are classified as behavioral features. They may restrict too what resources are able to run the application. For example, parallel applications are unable to use distributed resources to solve a problem, but can use resources with multiple processors and with shared memory architecture.

However, behavioral features may also define how efficient a resource will be to execute the application. An example is the granularity of distributed applications. This feature is defined as the ratio between the amount of computational steps by the amount of communication steps during the execution time. Fined grained applications are those with higher number of communication steps while coarse grained are the applications with lesser communication steps. Distributed applications that do very little or none communication are defined as embarrassing parallel.

Clusters on a grid multi-cluster environment may use of-the-shelf interconnection technology as Gigabit Ethernet or use a high performance and low latency interconnection technology as Myrinet or Infiniband. Those cluster resource characteristics can affect the performance of parallel and distributed applications based on their granularity as shown by [19] and [20].

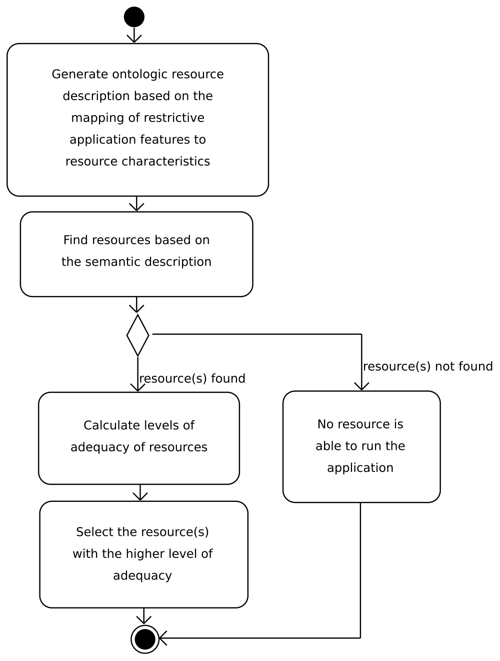
Generate ontologic resource description based on the mapping of restrictive application features to resource characteristics
Find resources based on the semantic description
resource(s) found
resource(s) not found
Calculate levels of adequacy of resources
No resource is able to run the application
Select the resource(s) with the higher level of adequacy

Figure 2. Selection mechanism activity diagram

Finally, some behavioral features may have a structural form. One example is the communication pattern of parallel applications: Cartesian, 2D, 3D, Fat-Tree, hypercube and others. However, we keep them classified as behavioral features.

IV. Selection Mechanism

An ontology was build to describe application structural and behavioral features. Users describe application features by identifying what concepts of the ontology match which application features. This ontology extends the reference resource description ontology proposed by [8] in order to express more information about the computational resources.

For those features that do not have levels of adequacy to computational resources, but restrict what resources are able to run the application, auxiliary ontologies are utilized to establish a direct map between the concept and attributes representing an application feature and the concept and attributes representing a resource characteristic. One example of such directly mapped features is the operating system the application was build to run upon and the actual operating system of the resource.

For the features that have levels of adequacy to computational resource characteristics, a fuzzy model based on [13] is utilized to compute the level of adequacy of a resource to run the application.

The proposed resource selection mechanism based on application features is described by the activity diagram shown in Figure 2. According to this diagram the selection is carried out in four steps. In the first step, the mechanism reads the application features provided by the user and utilizes the mapping table to build an ontological resource specification.

In the second step, the ontological resource specification that was generated in the first step is utilized to find a subset of resources by means of the semantic resource selection proposed by [8].

In the third step, a fuzzy model is applied to obtain the level of adequacy of each one of the selected resources on the second step. This model is based on the proposal of [14] and works as follows: first, it takes as input the application features and the resource characteristics that are related by some level of adequacy. This input is translated into linguistic variables with linguistic values by means of a fuzzyfication mechanism. Next, an inference process occurs using a fuzzy rule base to obtain the level of adequacy of the resource. A sample of the rules is shown in listing 1. Currently all the rules have the same weighting factor, i.e., they have the same level of relevance. After, the result of the inference process is defuzzified to an adequacy number in the interval of 0 to 100. Where 0 represents the lowest level of adequacy and the 100 represents the higher level of adequacy. This defuzzification process takes in account the result of each rule and applies the Center of Gravity (GOC) strategy for translation. The fuzzy model is implemented using Fuzzy Control Language (FCL).

```
RULE 1: IF applicationGranularity IS
    embarassingParallel AND
resourceInterconnectLatency IS small THEN
    adequacy IS low;
RULE 2: IF applicationGranularity IS
    embarassingParallel AND
resourceInterconnectLatency IS medium THEN
    adequacy IS medium;
RULE 3: IF applicationGranularity IS
    embarassingParallel AND
resourceInterconnectLatency IS high THEN
    adequacy IS high;
```

Listing 1. Sample of the fuzzy rule base written in FCL

Finally, in the fourth step, the resource with a higher value of adequacy level is selected as the best candidate to run the application.

V. Conclusion and Future Work

In this paper we presented a proposal to enhance the resource selection mechanism in a multi-cluster grid environment based on application features. The goal is to make the meta-scheduler aware of application features. According to our proposal, users are not required to specify resources characteristics to submit applications to run on a grid. Instead, application features are specified so that the meta-scheduler is able to find and select the resources with better levels of adequacy between their characteristics and the application features. In other words, it tries to select the best suited resource for the application.

With applications running on suited resources, it is expected a reduction on their execution time. Consequently, there would be an improvement on job execution throughput of resources, causing an overall optimization on resource usage and job execution throughput of the grid. The evaluation of those improvements by the proposed selection mechanism utilizes the GRADI Middleware and the GridSim as a testbed.

In our future research we plan to evaluate the usage of Semantic Web Rule Language(SWRL) [21] as a replacement to the mapping between application features and resource characteristics. The rules would be written to map classes and properties from application feature ontologies to classes and properties of the resource description ontologies. The Pellet [22] reasoner would be utilized to perform resource description queries based on application features, replacing the first step of the proposed resource selection mechanism.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *JHPC*, vol. 15, no. 3, p. 200, 2001.

[2] R. G. Silva *et al.*, "Grid computing middleware survey," *Proceedings of the 7th International Information and Telecommunication Technologies Symposium*, 2008.

[3] O. Wäldrich, P. Wieder, and W. Ziegler, "A meta-scheduling service for co-allocating arbitrary types of resources," in *Parallel Processing and Applied Mathematics*, ser. Lecture Notes in Computer Science, R. Wyrzykowski *et al.*, Eds. Springer Berlin / Heidelberg, 2006, vol. 3911, pp. 782–791, 10.1007/11752578_94. [Online]. Available: http://dx.doi.org/10.1007/11752578_94

[4] W. Xiaohui *et al.*, "Csf4: A wsrf compliant meta-scheduler," in *In Proc. of World Congress in Computer Science Computer Engineering, and Applied Computing*, 2006, pp. 61–67.

[5] E. Huedo, R. S. Montero, and I. M. Llorente, "A framework for adaptive execution in grids," *Softw. Pract. Exper.*, vol. 34, no. 7, pp. 631–651, 2004.

[6] A. Anjomshoaa *et al.*, *Job Submission Description Language (JSDL) Specification, Version 1.0*, Global Grid Form, november 2005.

[7] A. Pernas and M. Dantas, "Using ontology for description of grid resources," may. 2005, pp. 223 – 229.

[8] A. Silva and M. Dantas, "A selector of grid resources based on the semantic integration of multiple ontologies," oct. 2007, pp. 143 –150.

[9] D. Janson *et al.*, "Dynamic resource matching for multi-clusters based on an ontology-fuzzy approach," in *High Performance Computing Systems and Applications*, ser. Lecture Notes in Computer Science, D. Mewhort *et al.*, Eds. Springer Berlin / Heidelberg, 2010, vol. 5976, pp. 241–250, 10.1007/978-3-642-12659-8_18. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-12659-8_18

[10] L. Alben, "Quality of experience: defining the criteria for effective interaction design," *interactions*, vol. 3, no. 3, pp. 11–15, 1996.

[11] R. Gruber *et al.*, "Towards an Intelligent Grid Scheduling System," in *GRMW?2005*, vol. 3911/2006. Berlin: Springer Berlin / Heidelberg, 2006, pp. 751–757. [Online]. Available: http://ppam.pcz.pl/ppam2005/workshops.htm

[12] ——, "Integration of grid cost model into iss/viola meta-scheduler environment," in *Euro-Par'06: Proceedings of the CoreGRID 2006, UNICORE Summit 2006, Petascale Computational Biology and Bioinformatics conference on Parallel processing*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 215–224.

[13] J. Qin and M. A. Bauer, "Job co-allocation strategies for multiple high performance computing clusters," *Cluster Computing*, vol. 12, no. 3, pp. 323–340, 2009.

[14] ——, "An evaluation of communication factors on an adaptive control strategy for job co-allocation in multiple hpc clusters," in *ICPADS '09: Proceedings of the 2009 15th International Conference on Parallel and Distributed Systems*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 391–398.

[15] D. Janson *et al.*, "Toward resource management in multi-cluster grid configurations through an ontology-fuzzy approach," in *GCA*, H. R. Arabnia and G. A. Gravvanis, Eds. CSREA Press, 2009, pp. 10–16.

[16] A. Silva and M. Dantas, "An efficient approach for resource set-matching in grid computing configurations," may. 2006, pp. 5 – 5.

[17] E. Gamma *et al.*, *Design patterns: elements of reusable object-oriented software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[18] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language user guide*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1999.

[19] L. Pinto, R. Mendonca, and M. Dantas, "Impact of interconnection networks and application granularity to compound cluster environments," jul. 2008, pp. 468 –473.

[20] J. Ploski *et al.*, "Grid-based deployment and performance measurement of the weather research & forecasting model," *Future Generation Computer Systems*, vol. 25, no. 3, pp. 346 – 350, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6V06-4SJ2WPN-3/2/a08248530ec9ecc99547247b9ae3b94f

[21] I. Horrocks *et al.*, "Swrl: A semantic web rule language combining owl and ruleml," W3C Member Submission, World Wide Web Consortium, Tech. Rep., May 2004.

[22] E. Sirin *et al.*, "Pellet: A practical owl-dl reasoner," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 2, pp. 51 – 53, 2007, software Engineering and the Semantic Web. [Online]. Available: http://www.sciencedirect.com/science/article/B758F-4NF7Y7R-2/2/684715cf5485269b827d65332803523c