Optimization of Application Execution in the ViroLab Virtual Laboratory

Maciej Malawski[1], Joanna Kocot[2], Eryk Ciepiela[2], Marian Bubak[1,2]

[1] Institute of Computer Science, AGH, al. Mickiewicza 30, 30-059, Kraków, Poland
[2] Academic Computer Center CYFRONET, ul. Nawojki 11, 30-950 Kraków, Poland

Abstract

The objective of the presented work is to describe an optimization engine for the ViroLab Virtual Laboratory runtime.
The Laboratory specific model – invocation of operations on special objects which reside on Grid resources – imposes a new approach to optimization of Grid application execution. Such an approach is presented in the shape of the GridSpace Application Optimizer (GrAppO) – a module responsible for optimization of exectution of the applications developed in the ViroLab Virtual Laboratory – supported by monitoring and provenance systems. We describe the architecture of the optimizer, and report on the results of tests which demonstrate the need for the monitoring system.
Keywords: virtual laboratory, ViroLab, Grid Computing, execution optimization

1 Introduction

An opitimizer designed to work in a Grid environment such as the ViroLab Virtual Laboratory [1] has to face the challenges imposed by Grid environment itself – e.g. dynamic nature, requiring distributed sources of information – as well as those specific only to ViroLab [2] and related to its specific programming model. The ViroLab model introduces several levels of abstraction represented by Grid Object [3] entities.

The core functionality of ViroLab runtime system (called GridSpace [4] Engine) offers a possibility of executing a ViroLab Experiment. However, the source code of the experiment provides only the information on Grid Object Classes [3] (the highest abstraction over a Grid service) whose instances have to be used to invoke certain operations on them. A non-trivial choice of instance, considering the structure of relations between the experiment's entities is left to the optimizer.

2 State-of-the-art

The problem of optimization of application execution and resource utilization is the subject of wide research on Grid scheduling and load-balancing [6]. In traditional Grid environments a dedicated component – scheduler of jobs or resource broker – is responsible for the choice of an appropriate resource for a given job. However, the functionality of such a component is much wider and usually includes many issues that were already assigned to other modules in ViroLab (e.g. submission and execution of jobs). At the same time, these components do not cover some of the problems that still need to be solved in the Virtual Laboratory (e.g. they do not have a notion of the structures introduced in the ViroLab experiment).

Despite these differences, the problem of optimization in the ViroLab Virtual Laboratory may be described by generic definition of optimization on the Grid formulated in [7]: *The process of discovering of the best combination of a job and resources in such a way that the user and application requirements are fulfilled, in terms of overall execution time (throughput) and cost of the resources utilized.* Therefore some algorithms that attempt to realize this combination could be applicable to the ViroLab environment. One of the branches of algorithms that can be used in the Virtual Laboratory is simple heuristics optimizing execution of independent jobs based on the job's Minimum Completion Time (MCT) [8]: *Min-min* (enables good load balancing), *Max-min* (minimizes the impact from tasks with longer execution times [9]), *Suffrage* [9], *XSuffrage* [10]. Algorithms for dependent jobs optimization require the precedence order of jobs in advance. Such algorithms are: *List heuristics* – e.g. *Heterogeneous Earliest Finish Time* [11] and *Dynamical Critical Path (DCP)* [12], *Clustering-based heuristics* [13, 9] – e.g. *Dominant Sequence Clustering* and *CASS-II* [9], *Duplication-based algorithms* – e.g. *Task Duplication-based Scheduling algorithm (TDS)* and *Task duplication-based scheduling Algorithm for Network of Heterogeneous systems (TANH))* [9].

3 Structure of the GridSpace Application Optimizer

According to the abstractions introduced in GridSpace [3] (see section 1), GrAppO is designed to select a Grid Object Instance optimal for invocation of an operation of a specified Grid Object Class. There may be many instances corresponding to one grid object class, with the same functionality, but of different performance. The decision produced by GrAppO is based on the information retrieved from the registry (Grid Resource Registry – GRR), monitoring and provenance (Provenance Tracking System – PROToS [5]) components of virtual laboratory and depends on implemented optimization policy. Optimization modes offered in the Gridspace Application Optimizer include so called short-, medium- and far-sighted optimization. The first one assumes an optimum solution is chosen only for one Grid Object Class at a time. In the second mode solutions are assigned for a group of Grid Object Classes without reordering tasks or using queues, still considering the previous choices. The far-sighted mode requires the whole application to be analyzed and Grid Object Classes to be reordered taking into account dependencies between them.

Figure 1 shows the architecture of GridSpace Application Optimizer together with its connections to other components of the ViroLab Runtime and Middleware. During short- or medium-sighted optimization, *GrAppO Manger* receives calls from the Grid Operations Invoker [3] (that contain the name of Grid Object

Class found in the experiment script), collects the data concerning Grid Object Implementations of that class and their instances from the Grid Resource Registry. This data are then passed to the *Optimization Engine*, which executes dedicated optimization algorithms based on estimations of each solution's performance prepared by the *Performance Predictor*. The *Performance Predictor* itself makes predictions on the basis of data obtained from PROToS (previous performance of the GOb Class's instances) – through *Historical Data Analyzer*, and from the monitoring system (the condition of the available Grid resources) – through *Resource Condition Data Analyzer*. *Application Structure Analyzer* is used only during far-sighted optimization for analysis and reorganization of application structure.

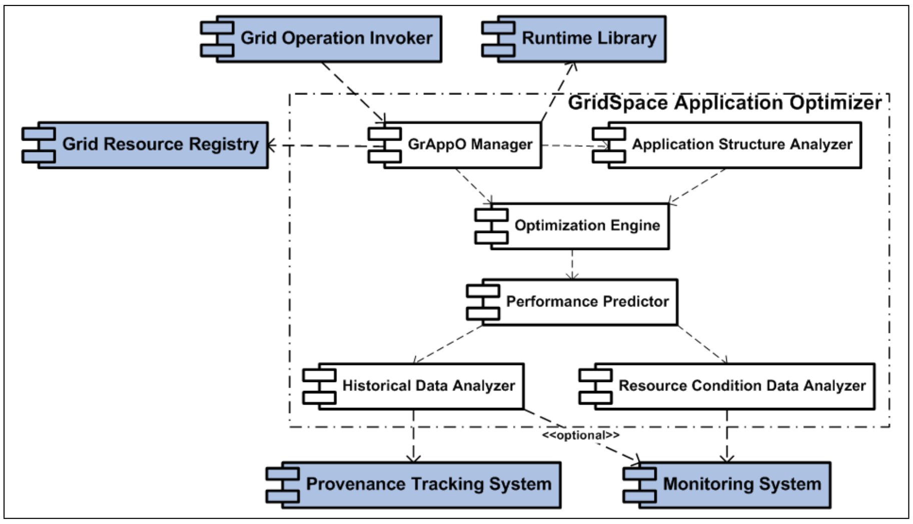

Fig. 1: Architecture of GridSpace Application Optimizer.

4 Monitoring System

As the experiments with GrAppO demonstrated (see also Section 5), the important source of information is the monitoring system. It is intended to provide up-to-date information on the resources for the GrAppO optimizer as well as to monitor the state of the application being executed. Therefore, the monitoring system should be interoperable with a variety of grid object technologies supported, such as Web Services, LCG jobs, MOCCA components as well WSRF and AHE in the near future. Such a system has to preserve high level of abstraction of grid object notion, at the same time providing low-level and technology-specific information about the monitored entities when needed.

In addition to GrAppO, the monitoring information is used by the Provenance Tracking System – PROToS (also used by GrAppO) to collect data concerning the course of applications. Another important element of functionality of the monitoring system is to provide application users with tools enabling multilevel insight into the application execution.

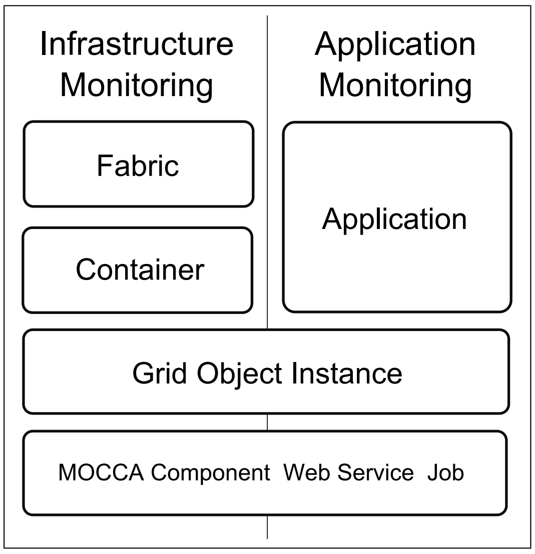

Fig. 2: Architecture of leMonAdE.

In order to achieve these goals, the proper means of instrumentation, monitoring data exposure, and monitoring data accessing are provided by the dedicated agiLE MONitoring ADherence Environment (leMonAdE), which applies the Aspect-Oriented Programming (AOP) approach. The leMonAdE architecture (presented on 2) can be divided into two main parts, according to their functionality: *Infrastructure monitoring* – instrumentation of containers of multiple technologies involved within the ViroLab Virtual Laboratory and *Application monitoring* – Aspect-Oriented Programming techniques applied by leMonAdE monitoring system.

5 Results of tests

The usability of the presented solution was inspected by various quality and integration tests. The quality tests performed on GrAppO were based on measuring makespan defined as: $\max_{j \epsilon J}(CT_{jr})$, where CT_{jr} is the expected completion time of the job j on the resource r – the moment of time when the resource r completes the job j [14]. The Max-min heuristics (see section 2) was used as a GrAppO optimization algorithm. It proved to give similar results to the

Min-min heuristics (see section 2), still the first gives (approximately 5.6%) better solutions when, among the Grid Object Classes to optimize appears one or relatively small group with significantly longer execution time than the others.

The tests showed a vulnerability of GrAppO's optimization results to the lack of data that should be provided by monitoring system (see table 1). That is understandable, as when a significant amount of information is missing, an accurate prediction of performance becomes impossible.

Percentage of removed data	Deterioration of makespan
10%	9.81%
20%	20.90%
30%	42.93%
40%	88.12%
50%	124.26%
...	...
100%	over 200%!

Tab. 1: Influence of the availability of information on the makespan.

The prototype of GrAppO was implemented and integrated with the ViroLab Virtual Laboratory. In this way, the ViroLab experiments appeared to be good integration tests.

6 Conclusions and future work

The GridSpace Application Optimizer, presented in this paper is restricted by multiple limitations imposed by the Grid environment characteristics and by the structure and attributes of the ViroLab Virtual Laboratory (see section 1. Nevertheless, as the test showed (section 5), in cooperation with the monitoring system it should be able to significantly decrease the time needed for executing an application in the GridSpace Engine (the ViroLab runtime system) and increase its quality. To fully take advantages of this optimization mechanism, the following actions are going to be taken in the nearest future:

- Realization of connections to the monitoring system and other ViroLab components (most of them are now unavailable) – a critical issue, as data from the external components significantly influence the optimizer's performance (compare section 1);
- Reaserch on far-sighted mode of optimization – the most interesting issue in the GrAppO optimization idea;
- Extending the monitoring system to support a broader range of technologies available in ViroLab;
- Creating a graphical interface for easier configuration, which can also be used to control the results of GrAppO operations.

Acknowledgements. This work was supported by the EU Virolab project IST-027446 with related Polish grant SPUB-M and by the Foundation for Polish Science.

References

1. ViroLab Virtual Laboratory, http://virolab.cyfronet.pl
2. ViroLab - EU IST STREP Project 027446, http://www.virolab.org
3. Maciej Malawski, Tomasz Bartyński, Marian Bubak: Invocation of Grid Operations in the ViroLab Virtual Laboratory. In *Proceedings of Cracow Grid Workshop 2007*, This volume.
4. Tomasz Gubala and Marian Bubak: GridSpace - Semantic Programming Environment for the Grid; Proceedings International Conference of Parallel Processing and Applied Mathematics (PPAM'05), Poznan, Poland, Springer, Sept., 2005
5. Bartosz Baliś, Marian Bubak, Michał Pelczar, Jakub Wach: Provenance Tracking and Querying in ViroLab. In *Proceedings of Cracow Grid Workshop 2007*, This volume.
6. Jarek Nabrzyski, Jenniffer Schopf, Jan Weglarz. (eds): Grid Resource Management. State of the Art and Future Trends, Kluwer Academic Publishers, 2003
7. M. Li and M. Baker: The Grid: Core Technologies; John Wiley & Sons, 2005
8. M. Maheswaran, S. Ali, H.J. Siegel, D. Hensgen, and R. Freund: Dynamic mapping of a class of independent tasks onto heterogeneous computing systems; IEEE Heterogeneous Computing Workshop 1999, 30-44
9. D. Fangpeng, G. Selim: Scheduling Algorithms for Grid Computing: State of the Art and Open Problems; School of Computing, Queen's University, Kingston, Ontario; January 2006
10. Henri Casanova, A. Legrand, D. Zagorodnov, and F. Berman: Heuristics for Scheduling Parameter Sweep Applications in Grid Environments; Proc. of the 9th Heterogeneous Computing Workshop 2000, 349-363
11. J. Kim, J. Rho, J. Lee, M. Ko: Effective Static Task Scheduling for Realistic Heterogeneous Environment; 7th International Workshop on Distributed Computing, IWDC 2004, Khargpur, India, December 2005
12. Y.-K. Kwok, I. Ahmad: Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors; IEEE Trans. Parallel Distrib. Syst., 7(5):506-521, 1996
13. Z. Shi: Scheduling Tasks with Precedence Constraints on Heterogeneous Distributed Computing Systems; The University of Tennesse, Knoxville, December 2006
14. X. He, X. Sun, and G. Laszewski: A QoS guided scheduling algorithm for grid computing. Workshop on Grid and Cooperative Computing 2002