Ovid: A Browser for Grids

Marios Dikaiakos and Artemakis Artemiou

Department of Computer Science, University of Cyprus, Nicosia, Cyprus
Email: `mdd@ucy.ac.cy, cs01aa2@ucy.ac.cy`

1 Introduction and Rationale

The formulation of parallel computing models has been a key topic of parallel computing research. Models abstract concepts such as the performance characteristics of hardware and system-software, the scheduling of tasks to processors, the size of input data and their partitioning. Typical abstractions include: graphs modeling the interconnection topology of multiprocessor systems and the structure of parallel programs, parameters like the number of processors in a multiprocessor system, the CPU processor speed, and the communication latency between two processors, etc. Parallel computing models can be used for the design of parallel algorithms and the performance analysis of parallel computations [4].

Developing models is a challenging task because the "right" models must remain simple and must have a proper level of abstraction, striking a balance between the incorporation of all significant capabilities and costs of parallel computation, and the omission of less important aspects and properties [4, 15]. Several approaches have been developed for modeling parallel computations theoretically and for proving theorems about parallel program complexity, such as the P-RAM [7], LogP [2], and the CTA [15]. Proposed theoretical models, however, have limited practical applicability: in order to provide mathematical abstractions amenable to analytical treatment and theorem proving, theoretical models often rely on unrealistic assumptions about parallel-machine properties or neglect important factors that affect application performance, such as the choice of parallel programming models and communication libraries [15]. Moreover, the theoretical treaty of realistic parallel applications that combine multiple algorithms and complex data structures is difficult and rarely provides analytical solutions. Consequently, application developers wishing to explore a wide space of system and program configurations, without having to incur the costs of implementation and testing, resort to modeling tools based on software-based abstractions of parallel architectures and programs [4, 3, 8, 9, 12].

With the emergence of the Grid [5] as a wide-scale, distributed computing infrastructure for resource sharing and coordinated problem solving in dynamic, multi-institutional Virtual Organisations (VOs), the problem of modeling resources and computations is placed in a totally new context. The dynamic nature, the heterogeneity, the complexity, the scale, and the open architecture of the Grid make the use of modeling abstractions of Grid infrastructures and computations an essential condition for harnessing the Grid. Modeling abstractions for the Grid must hide the numerous details involved in Grid computations, while maintaining and providing application developers with high-level "views" of important aspects of Grids. Even though efforts to formulate models for describing Grid infrastructures and computations have appeared in recent literature [11], we believe that the feasibility of effective theoretical models for the Grid is questionable; furthermore, that the scope of such models will be different from that of parallel computing models. Arguably, a Grid application developper will be interested in high-level (summary) representations of numerous aspects involved in a Grid computation, besides the complexity analysis of a given algorithm mapped upon a homogeneous abstract machine model. Issues of interest may include the capacity of resources available in sites participating to a Virtual Organization, the measured performance of Grid nodes and VOs, the average network distance and bandwidth between two nodes, the network topology linking the nodes of a VO, the type of job queues available at a given site, usage and charging policies followed by different sites and VOs, the availability of software libraries, histories of resource availability and usage, representations of workflow-like Grid computations, etc.

The need for effective Grid abstractions will result to a variety of modeling tools capturing essential aspects of Grid infrastructure and computation. In contrast to the implementation of traditional parallel computing modeling tools, which represent and store their modeling abstractions in internal data structures, we anticipate that Grid modeling tools will encode their abstractions in open, standardized, and extensible metadata formats. Another important difference between Grid and parallel computing modeling tools will be their typical scenarios of use. Parallel computing modeling tools are used in parametric studies that involve "model executions," i.e., analytic calculations, trace-driven or event-driven simulations, etc. Model executions provide estimates of performance metrics that describe the scalability of a computation with respect to input size and machine configuration, its computation-to-communication ratio, its bottlenecks, etc. In contrast, the emphasis of Grid modeling tools will be on the representation and integration of modeling abstractions for the different issues that pertain to the deployment of a computation on the Grid: infrastructure capacity and availability, measured performance of Grid resources, policies enforced by various Grid nodes, statistics of previous job runs, the availability of software libraries invoked by the computation, etc. The complexity and diversity of these issues make it very hard to derive "metrics" that can be used effectively by Grid application developers. Therefore, we anticipate a shift in the paradigm of modeling-tools functionality and use, from model execution to model navigation. We define as *model navigation* the process undertaken by an end-user who wishes to browse through the space of models and parameters, which represent abstractly the factors that affect the deployment and execution of a computation on the Grid. Such a navigation process must be supported by a simple graphical user interface representation and a core set of simple interaction primitives.

In this paper, we present initial efforts to design and develop *Ovid*, a tool for modeling, visualizing, and browsing properties of Grid infrastructures. At

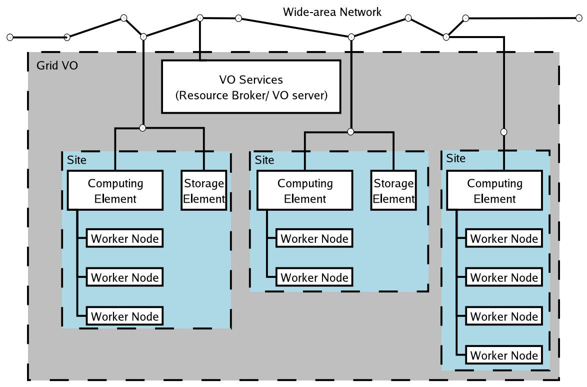

Fig. 1. A model of a Grid infrastructure.

the heart of Ovid lies an abstraction of the Grid infrastructure, modelled as a constellation of clusters that belong to different Virtual Organizations (see Figure 1). This model reflects the architecture of the European DataGrid and CrossGrid testbeds [6]. Inside Ovid, this model is represented as an ontology [13] described in RDF Schema [1, 10]. The ontology is instantiated with the help of a plug-in, which invokes various Grid Information Services [14] and retrieves values of model parameters. In addition to the main ontology model of the Grid infrastructure, Ovid supports other models of relevance to the Grid, which are also represented in RDF/S and instantiated by special plug-in modules. In particular, the first prototype implementation of Ovid integrates abstractions that represent the membership of cluster-sites to Virtual Organizations and the performance properties of Grid nodes, described in terms of metrics derived by experiments conducted with the GridBench suite of Grid benchmarks [16]. Both the VO and the GridBench models are represented in RDF/S. The former is instantiated with queries to the Grid Information Services of available VOs. The later is instantiated with queries to a service that provides access to an XML database with GridBench specifications and results from GridBench experiments.

Ovid comprises a user interface module, which undertakes the graphical representation of its supported models. The graphical representation of the Grid-infrastructure model is used as an "index" to the properties of other abstractions supported by Ovid and as a basis for their graphical representation. Starting from the Grid-infrastructure model's graphical representation and using a pointer, the end-user is able to navigate through the properties of other Grid-related models. The graphical representations of these models and of their interrelationships are stacked upon the Grid-infrastructure graphical model, according to visualization guidelines associated with each model. The GUI of Ovid implements a number of primitive operations that enable end-users to "zoom" in or out of a model, selecting the proper focus and level of abstraction.

In this paper, we present a brief overview of early implementation effort for Ovid. We describe the functionality provided by Ovid, its software design, the representation of the core Grid infrastructure model, and current implementation status.

2 Towards a First Prototype

We have started implementing a prototype of Ovid, based on the ideas presented above. The functionality of this prototype is described below.

When running Ovid, the end-user is presented with a main control panel, which can be used for selecting and running a plug-in that will instantiate the Grid infrastructure ontology. Currently there are three available plug-ins (see Figure 2): a) One consists of a MDS Query module that performs direct queries to MDS servers of various VOs. b) A Direct file retrieval module for retrieving information captured by previous MDS queries and stored on disk. c) A Random data module, which allows the experimenting with arbitrary Grid configurations.

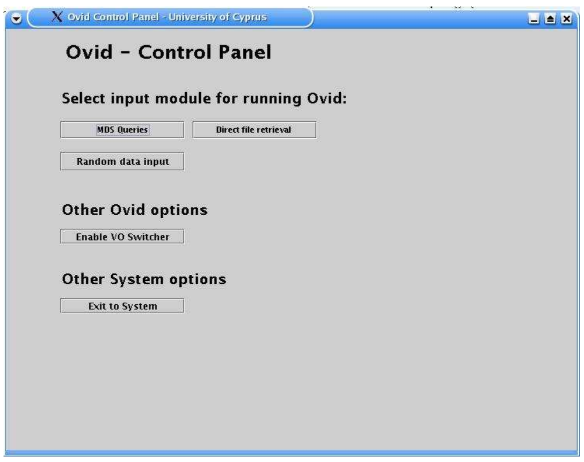

Fig. 2. Ovid: Control Panel.

A second panel, provides a selection of visualization modules, which determine the presentation of values on top of the Grid-infrastructure graphical representation. Examples of possible modules are: a graph displaying benchmark results, text describing Grid resources, etc. If no module is selected, Ovid uses a default representation with the name of the selected component and the site in which it belongs to.

Ovid's main screen displays all the nodes of Virtual Organizations in a faded out form. This mode is called "Compact Mode". The end-user can use a "VO Switcher" panel for selecting one or more Virtual Organizations. Upon selection of a VO, the nodes of this VO are activated on screen and allow further

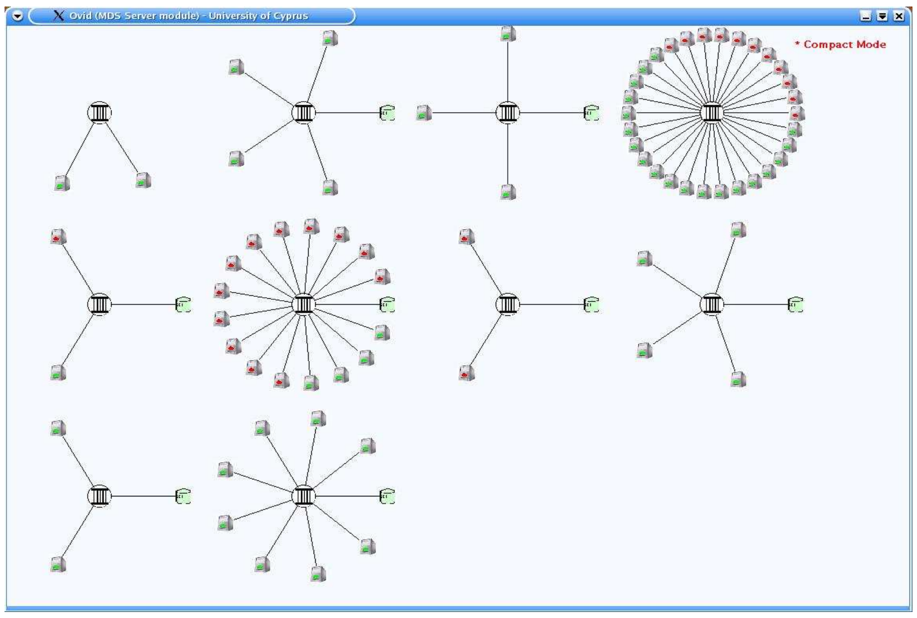

Fig. 3. Main Screen (Compact Mode).

interaction via the GUI (see Figure 3). For instance, when the user clicks at the center of a node of a selected VO, Ovid switches into an "Expanded Mode," in which the selected node is displayed isolated, providing more details about its characteristics and status (Figure 4. The user can select the "Back" button to return back to Compact Mode.

Ovid implements a "mouse-over" functionality: when the mouse is over a component, a tool tip appears providing extra information about the underlying abstraction. For example when the mouse is over a worker node, it displays whether the worker node is "Free," "Busy" or "Down," using a color encoding. Other information can be easily displayed as well.

We are currently working in finalizing the software design, the functionality of Ovid, the graphical interface, and the interaction primitives provided by Ovid. In particular, we plan to implement: an input box that will enable the selection of some view of the models supported by Ovid using textual or URI descriptions, a "History" function for viewing the most recent navigation sessions, and a "Favourites" function for storing parts of models of interest of the user.

Acknowledgement: This work was supported in part by the European Union through contract number IST-2001-32243, under the CrossGrid project.

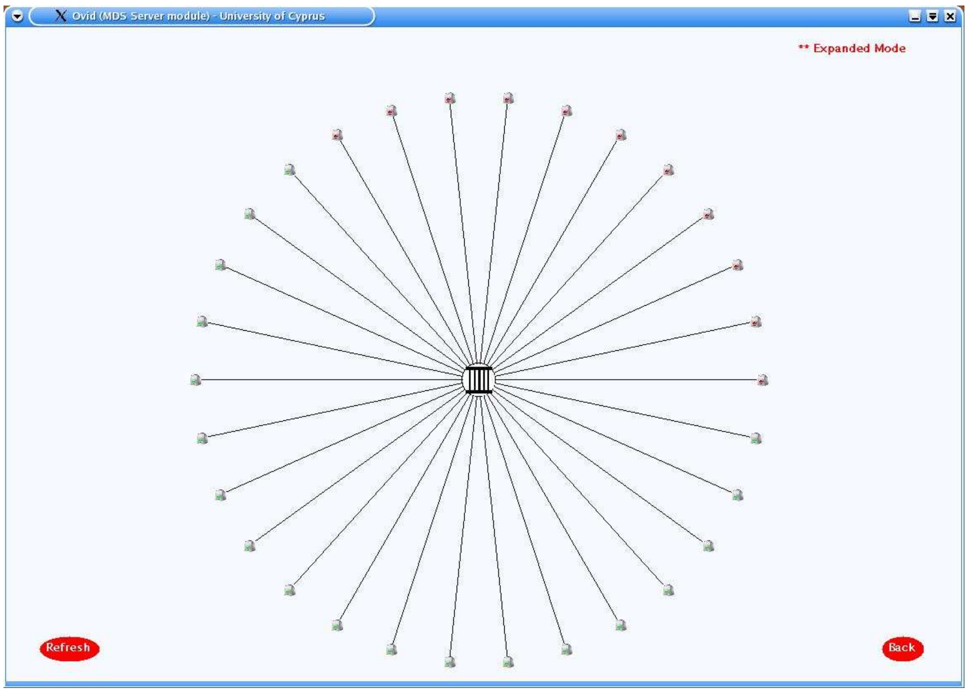

Fig. 4. Ovid: Expanded Mode.

References

1. D. Brickley and R.V. Guha (editors). RDF Vocabulary Description Language 1.0: RDF Schema. W3C Working Draft, October 2003. http://www.w3.org/TR/rdf-schema/.
2. D. Culler, R. Karp, D. Patterson, et al. LogP: Towards a Realistic Model of Parallel Computation. In *Proceedings of the Fourth ACM Sigplan Simposium on Principles and Practice of Parallel Programming*, May 1993.
3. M. Dikaiakos, A. Rogers, and K. Steiglitz. Functional Algorithm Simulation of the Fast Multipole Method: Architectural Implications. *Parallel Processing Letters*, 6(1):55–66, 1994.
4. Marios Dikaiakos, Anne Rogers, and Kenneth Steiglitz. Performance Modeling through Functional Algorithm Simulation. In G. Zobrist, K. Bagchi, and K. Trivedi, editors, *Advanced Computer System Design*, chapter 3. Gordon and Breach, 1998.
5. I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications*, 15(3), 2001.
6. Fabrizio Gagliardi, Bob Jones, Mario Reale, and Stephen Burke. European DataGrid project: Experiences of deploying a large scale testbed for E-science applications. *Lecture Notes in Computer Science*, 2459:480–499, 2002.
7. A. Gibbons and W. Rytter. *Efficient Parallel Algorithms.* Cambridge University Press, 1988.

8. B. R. Helm, A. D. Malony, and S. Fickas. Capturing and Atomating Performance Diagnosis: The Poirot Approach. Technical Report CIS-TR-93-25, Department of Computer and Information Science, University of Oregon, November 1993.
9. R. B. Irvin and B. P. Miller. A Performance Tool for High-Level Parallel Programming Languages. In *Proceedings of the IFIP WG 10.3 Working Conference on Programming Environments for Massively Parallel Distributed Systems*, pages 31.1–31.15, April 1994.
10. F. Manola and E. Miller (editors). RDF Primer. W3C Working Draft, October 2003. http://www.w3.org/TR/rdf-primer/.
11. Z. Nemeth and V. Sunderam. Characterizing Grids: Attributes, Definitions, and Formalisms. *Journal of Grid Computing*, 1:9–23, 2003.
12. D.M. Nicol and J.C. Townsend. Accurate Modeling of Parallel Scientific Computations. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems*, pages 165–170, May 1989.
13. N. Fridman Noy and D. McGuinness. Ontology Development 101: A Guide to Creating your First Ontology. Technical Report KSL-01-05, Stanford Knowledge Systems Laboratory, October 2001.
14. B. Plale, P. Dinda, and G. von Laszewski. Key Concepts and Services of a Grid Information Service. In *Proceedings of the 15th International Conference on Parallel and Distributed Computing Systems (PDCS 2002)*, 2002.
15. Lawrence Snyder. Experimental validation of models of parallel computation. In A. Hofmann and J. van Leeuwen, editors, *Special Volume 1000*, Lecture Notes in Computer Science, pages 78–100. Springer Verlag, 1995.
16. G. Tsouloupas and M. Dikaiakos. GridBench: A Tool for Benchmarking Grids. In *Proceedings of the 4th International Workshop on Grid Computing (Grid2003)*. IEEE Computer Society, November 2003. To appear.