Hierarchical resource management in grid computing

Korkhov, V.V.

Publication date
2009

Document Version
Final published version

Link to publication

Citation for published version (APA):
Korkhov, V. V. (2009). *Hierarchical resource management in grid computing*. [Thesis, fully internal, Universiteit van Amsterdam].

Chapter 1. Introduction

1.1 Multi-layered applications on the Grid

The rapid development of computing and networking technologies brings more power of computational resources to potential users every year. To be able to make a good use and get the most of the variety of accessible computing systems and networks a thoughtful approach is needed. On the other hand, the complexity of scientific applications, often with multi-component and hierarchical structure, grows as well. The consequence of both trends is the need for resource management strategies and methods that enable proper distribution and control of the applications on heterogeneous and dynamic resources.

This thesis addresses the issues of multi-layer resource management and workload balancing for parallel and distributed applications in heterogeneous distributed environment.

Lots of complex scientific applications operate in multi-scale, multi-time, multi-physics domains which affects the dynamics of the application requirements. The architecture of modern distributed systems features multiple resource layers that show their dynamics in performance and availability. Thus the core issue of executing scientific applications on contemporary distributed systems is in proper mapping of the applications to the environment. This thesis presents a view on the approaches to identify the layers of application's hierarchy and the ways of mapping each layer to the computing environment.

The main questions we pose and try to find answers to are:

- What are the technology invariant approaches to efficient resource management for complex parallel applications in a distributed heterogeneous environment?
- How to map different application layers to the computing infrastructure and manage these layers?
- When is a distribution of parallel applications across a set of Grid resources beneficial?

We start from exploring the basics: the case of a single parallel application running on a set of heterogeneous resources. Usually it is the lowest layer in the complex application hierarchy. A countless number of parallel applications have been developed for traditional (i.e. static homogeneous) parallel systems. The real problem in porting such applications to Grid environments is to keep up a high level of parallel efficiency. To assure efficient utilization of Grid resources, special methods for workload distribution control should be applied. Proper workload optimization methods should take into account two aspects: (1) the application characteristics (e.g. the amount of data transferred between the processes, amount of floating point operations and memory consumption) and (2) the resource characteristics (e.g. processors, network and memory capacities, as well as the level of heterogeneity of the dynamically assigned resources). The method should be computationally inexpensive not to impose too high overheads.

We propose and evaluate the method of adaptive workload balancing that depends on dynamic characteristics of both application and resources. Traditionally there are different approaches to the issue of workload balancing: a) carefully calculate the distribution of the workload taking into account all the properties of environment and application - the task that might be time and resource consuming by itself; b) distribute the workload in a straight forward way, considering only processing power of the worker nodes at most - the fast but not very efficient way in terms of resulting performance of workload distribution. We study an intermediate approach that combines the fast calculation of initial approximate workload distribution together with the precision of getting this distribution close to the optimal one during several refining iterations.

Moving to a larger scale, more questions appear. Shall we use several parallel computers together for a single parallel application? When and why will it speedup the parallel program? To check that we evaluate the theory of parallel applications speedup in a multi-cluster environment. Here a single parallel application spans over a set of computational clusters acquiring a number available nodes at each of them. Knowing the parameters of computational environment and parallel application model it is possible to predict if such distribution is efficient. This is the extension of a simple single parallel application, a horizontal layer of expansion in the application hierarchy.

Data processing is often organized not within a single parallel application but in a set of separate components that perform information exchange only at the start and finish of the execution. We call this type of software a multi-job application, and each job can be a separate parallel application in turn. This forms another layer in a vertical application hierarchy rising over a single parallel application. To examine the efficiency of multi-job applications processing divisible workload we bring the concepts developed for adaptive workload balancing of a single parallel application to the multi-job environment and compare it to standard self-scheduling mechanism. The multi-job application execution environment combined with user-level scheduling enables such features as transparent workload balancing, resource pooling and dynamic resource selection during application run-time.

While multi-job applications described above typically perform similar type of processing on a divisible workload, another type of applications in distributed environments is the Grid workflow: a scheduled sequence of functionally decomposed communicating processes coordinated by control or data flow. This feature of multi-functionality together with the variety of execution schemes (sequenced or pipelined, scattered in time and location) builds another layer of complex application hierarchy. Our research in this field is focused on data-driven workflows with direct data streaming between the workflow components. We study a model of a data-driven workflow, evaluate different methods of workflow components scheduling, describe and validate the environment developed for data-driven workflows enactment and resource management.

1.2 Hierarchical structure of large-scale distributed applications

Distributed e-Science applications can be composed of components that have different functionalities and employ different types of processing: high-performance or high-throughput computing, batch or interactive processing, computations or communications intensive tasks, distributed or local execution of application components etc. The combination of functionalities and technologies, within a single large application, forms a multi-layer structure. Typically the lowest layer represents simple computing jobs on distributed and parallel resources. The next layer is formed by a set of jobs often required to perform the exploration of large parameter space or divide working area between multitude of worker processes performing similar task. The highest layer combines all the diverse functionalities together forming a distributed workflow.

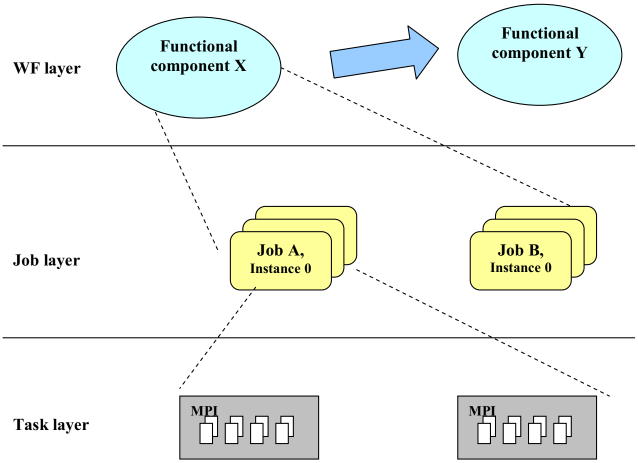

Figure 1.1: Hierarchical structure of multi-layered applications. The lowest layer represents simple computing jobs on distributed and parallel resources. The next layer is formed by a set of jobs to perform the exploration of large parameter space or divide workload between multitude of worker processes performing similar task. The highest layer combines all the diverse functionalities forming a distributed workflow.

We consider a three-layer structure of distribution and parallelism (Figure 1.1): task (e.g. single parallel application), job (e.g. parameter sweep and task farming) and workflow (functional decomposition of the application). Each layer of this structure has its own requirements to maximize the performance and enable good scalability. Thus appropriate resource management and workload balancing has to be addressed while building the distributed multi-layer and multi-component application as a whole:

- Task layer: adaptive workload balancing of parallel applications on dynamic heterogeneous resources
- Job layer: workload distribution management for job farming and parameter sweeps
- Workflow layer: execution of functionally decomposed communicating components coordinated by control or data flow

1.3 Grid architecture hierarchy

The hierarchical structure of complex applications is supplemented with the hierarchy of the Grid environment architecture. The classical architecture layering is introduced in [34]: fabric, connectivity, resource, collective, application layers. Components within each layer share common characteristics but can build on capabilities and behavior provided by any lower layer. In this thesis we refer to this classical definition of the Grid architecture but acknowledge the existence of other flavors oriented to Web services concepts and technologies.

Foster and Kesselman describe the following functionality of the classical Grid architecture layers [34]:

- Fabric layer: The Grid Fabric layer provides the resources to which shared access is mediated by Grid protocols: for example, computational resources, storage systems, catalogs, network resources, and sensors. A "resource" may be a logical entity, such as a distributed file system, computer cluster, or distributed computer pool; in such cases, a resource implementation may involve internal protocols (e.g., the NFS storage access protocol or a cluster resource management systems process management protocol), but these are not the concern of Grid architecture. Fabric components implement the local, resource-specific operations that occur on specific resources (whether physical or logical) as a result of sharing operations at higher levels.
- Connectivity: The Connectivity layer defines core communication and authentication protocols required for Grid-specific network transactions. Communication protocols enable the exchange of data between Fabric layer resources. Authentication protocols build on communication services to provide cryptographically secure mechanisms for verifying the identity of users and resources.
- Resource: The Resource layer builds on Connectivity layer communication and authentication protocols to define protocols for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operations on

individual resources. Resource layer implementations of these protocols call Fabric layer functions to access and control local resources. Resource layer protocols are concerned entirely with individual resources and hence ignore issues of global state and atomic actions across distributed collections; such issues are the concern of the Collective layer discussed next.

- Collective: While the Resource layer is focused on interactions with a single resource, the Collective layer contains protocols and services that are not associated with any specific resource but rather are global in nature and capture interactions across collections of resources. Because Collective components build on the narrow Resource and Connectivity layer "neck" in the protocol hourglass, they can implement a wide variety of sharing behaviors without placing new requirements on the resources being shared. For example, collective layer provides:
  - Directory services that allow Virtual Organization (VO) participants to discover the existence and/or properties of resources;
  - Co-allocation, scheduling, and brokering services that allow requesting the allocation of one or more resources for a specific purpose and the scheduling of tasks on the appropriate resources;
  - Monitoring and diagnostics services to support the monitoring of VO resources for failure, overload, etc.;
  - Data replication services to support the management of VO storage resources to maximize data access performance with respect to metrics such as response time, reliability, and cost.
  - Workload management systems and collaboration frameworks also known as problem solving environments ("PSEs") that provide for the description, use, and management of multi-step, asynchronous, multi-component workflows.
- Application: The final layer in our Grid architecture comprises the user applications that operate within a VO environment.

The key problem of complex Grid computing applications is to properly correlate all their components with the Grid architecture layers. Most of these aspects can be taken care of by middleware, libraries and integrated environments for composition and execution of complex distributed applications on the Grid. The latter are usually referred as Problem Solving Environments (PSE).

1.4 Problem solving environments

A majority of modern computational engineering applications are in essence multidisciplinary problems challenging researchers from diverse knowledge domains and institutions to share their experience, methodologies, software, data, computational and instrumental resources. One of the most prominent features of such projects is that many components essential for solving one problem are distributed over multiple organizations. Gathering all the required pieces of a problem solving puzzle in one site is often not possible, nor is it desirable.

To solve the interdisciplinary problems, several aspects shall be addressed: First, organizing a collaborative environment that would facilitate the exchange of ideas, models and codes. Second, proving the participating groups with a unified transparent and secure way to share and use computational power and other resources of the consortium. Third, developing a user-friendly environment that would guide end-users through the multiple stages of solving the problem under investigation. These stages include: properly defining a physical engineering problem, finding appropriate simulation components, choosing optimal numerical methods, accessing data bases, initiating simulations, discovering computational resources, submitting and monitoring the jobs, analyzing and archiving the results. All these tasks can be semi-automated within generic problem solving environments or virtual laboratories for e-Science [72].

A problem solving environment (PSE) provides a complete integrated computing environment for composing, compiling, running, controlling and visualizing applications. It incorporates many features of an expert system and provides extensive assistance to users in formulating problems and integrating program codes, processes, data, and systems in distributed computer environments [41]. Information sources and codes may be widely distributed, and the data processing requirements can be highly variable, both in the type of resources required and the processing demands put upon these systems. Grid technology as integrating middleware is a major cornerstone of today's PSEs [35, 50]. By offering a unified means of access to different and distant computational and instrumental resources, unprecedented possibilities and benefits can be expected. Connectivity between distant locations, inter-operability between different kinds of systems and resources, and high levels of computational performance are some of the most promising characteristics of the Grid. Grid technology provides dedicated support such as strong security, distributed storage capacity, and high throughput over long distance networks. Besides these immediate benefits, the computational resources of the Grid provide the required performance for large-scale simulations and complex visualization in collaborative environments, which are expected to become of major importance to the modern hi-tech society. Scientific communities have been investing great efforts into development of Grid-aware problem solving environments for complex applications [75, 112, 122–124, 127, 132, 134, 139].

An important flavor of PSE is a virtual laboratory which is defined as a heterogeneous distributed problem solving environment that enables a group of researchers located around the world to work together on a common set of projects [53]. A virtual laboratory allows scientists in a number of different physical locations, each with unique expertise, computing resources, and/or data to collaborate efficiently not simply at a meeting but in an ongoing way. Such environment extends and pools resources while engendering orderly communication and progress toward shared goals.

This thesis begins from outlining the challenges posed to a complex application within a PSE deployed in the Grid environment. The original PSE we use to make a start from was initially developed for traditional parallel architectures, and we immediately dive into the problems of executing parallel applications in heterogeneous distributed environment. Step by step we examine the methods and approaches that can be used by a Grid-based PSE for execution of parallel and distributed applications. The growth of distributed application complexity leads to the formation of the concept of a Grid workflow within a Grid-enabled Virtual Laboratory [140]: a scheduled sequence of functionally decomposed distributed processes coordinated by control or data flow. Grid workflows and workflow management systems make up the most convenient, advanced and generic form of PSE on the Grid. Thus we close the cycle which started from the PSE created for traditional architectures, followed the layers of parallel processing in distributed heterogeneous environment and came to the Virtual Laboratory on the Grid.

1.5 Thesis outline

The outline of the thesis is as follows:

Chapter 2 gives the background of our research, presents the related work on resource management, scheduling and workload balancing.

Chapter 3 investigates the problem of running a dynamic parallel application on a set of dynamic heterogeneous resources. The dynamics means that the requirements of the application can vary during the runtime, and the resources can vary the performance. We propose the adaptive workload balancing algorithm (AWLB) for parallel applications with divisible workload and evaluate it on a case study of a parallel solver from the Virtual Reactor (VR) for Plasma Enhanced Chemical Vapour Deposition (PECVD).

Chapter 4 takes further steps in evaluating the possibilities of running parallel applications in distributed environment. Here a multi-cluster system is used as a resource, and a single parallel application spans over several sub-clusters. We outline the concept of Grid speedup and the theoretical approach for its evaluation introduced in [49] and perform its experimental validation. The Lattice Boltzmann Method (LBM) solver is used as a case study application; a simple model of this application is used for prediction of possible performance gain from multi-cluster distribution.

Chapter 5 applies the Adaptive WorkLoad Balancing (AWLB) method developed in Chapter 3 to multi-job applications with divisible workload. We present a hybrid resource management environment, operating on both application and system levels, developed for minimizing the execution time of parallel applications with divisible workload on heterogeneous Grid resources. The system is based on the AWLB algorithm incorporated into the DIANE User-Level Scheduling (ULS) environment.

Chapter 6 presents the VLAM-G workflow management system and its core component: the Run-Time System (RTS). The RTS is a dataflow driven workflow engine which utilizes Grid resources, hiding the complexity of the Grid from end users. Special attention is paid to the concept of dataflow and direct data streaming between distributed workflow components. We present and evaluate the resource management algorithms and solutions for data-driven workflows used in VLAM-G.

Chapter 7 brings a summary and conclusions of the research.