An ADL-based Support for CCA Components on the Grid

Maciej Malawski, Tomasz Bartyński, Eryk Ciepiela, Joanna Kocot, Przemysław Pelczar, Marian Bubak
Institute of Computer Science and ACC CYFRONET
AGH University of Science and Technology
Kraków, Poland

Abstract

Moccaccino project described in this paper provides a novel ADL specification and a manager tool that facilitates the deployment and management of component applications on the Grid. The concept of hierarchical component groups and connections indexed as lists or maps is introduced in the ADL. The designed and implemented Moccaccino Manager is suited for MOCCA, a CCA compliant framework build upon H2O, which is a secure and lightweight middleware platform providing a flexible component container. The interoperability with GCM and CoreGRID research group on component deployment is addressed as well.

1 Introduction

Component-based programming paradigm becomes increasingly popular model in scientific computing [6]. A component model supports software composition, deployment and reconfiguration, which are the essential requirements for building and running applications on such environment as Grid. Running component based applications on the Grid environment still remains a challenging problem involving resource discovery, component deployment and application management in response to changes of environment.

The main characteristics of the problem may be briefly summarized as follows:

- The application components and their connections may have various demands for computing power and communication cost;
- The application may include collections of components;
• The number of components in a collection may depend on the number of available computing nodes;

• The number of nodes may change in time.

The main assumption is that the component application should be possibly unaware of the complexity of the environment it is running on, so the responsibility for dealing with deployment and adaptation process should be thus delegated to the specialized manager tool.

Consequently, such a manager tool should be aware of both the environment and the application structure. Environment awareness can be achieved by using an appropriate discovery and monitoring techniques. The component application structure respectively can be expressed using an Application Description Language (ADL). In this paper, we describe our approach to development Moccacino: a manager for a component application running on the Grid and the ADL designed for this purpose. The background for our research is the CCA [2] model and its distributed framework MOCCA [9, 8].

The advantage of the CCA modes is that it was designed to satisfy the requirements of scientific applications. Moreover, the simple specification and support for reconfiguration of application at runtime make CCA useful basis for our research. MOCCA is a CCA compliant distributed framework, based on H2O platform which offers lightweight component containers called H2O kernels. Current version, called MOCCA_Light, is a pure-Java implementation and provides some extensions of CCA for multiple ports and connections.

2 Related work

A number of projects is being developed, that aim in supporting automated deployment of component applications in Grid environment. In order to fulfill this objective, they produce their own Architecture Description Languages.

The Fractal ADL [3] defines the component types, their connections and containment relationships between hierarchical components. Recent work on extending the Fractal towards the Grid Component Model also addresses the deployment problem and proposes a common solution for automating this process [4].

Grid Application Factory Service [5] for CCA addresses the problem of building reliable, scalable Grid applications. This solution is based on separating the process of deployment and hosting from application execution, and uses XML documents for describing these stages.

Corba Component Model (CCM) also has the concept of IDL3 language, which is responsible for application assembly and deployment. Within the Grid-

In addition to the examples presented above, other ADLs were inspected and taken into consideration. A good overview of ADLs can be found in [1].

Despite the number of ADLs, none of them fully satisfies our requirements. Problems such as describing components characteristics (computational and communicational needs) or building structured component-based applications are not solved. Furthermore, specifying number of components in a way that would enable adapting quantity of components to available resources is not supported.

3 Our approach - ADL concept

Considering the advantages, disadvantages and limitations of above-mentioned ADLs we identified the following goals to achieve when designing the ADL for Moccaccino (ADLM):

1. to make ADLM as concise as possible,
2. to facilitate modification of the quantitative properties of architecture,
3. not to lose expressiveness of the language, therefore to support a breadth of application architectures,
4. to link resources such as executables and documents using URLs,
5. to allow containing information related to other aspects of deployment such as scheduling or policies of adaptation.

To facilitate document processing and programming the tools the XML-based syntax was developed. The main elements of the description are described below. Component class is a basic element that defines type of a component. To facilitate the parametrization of application description the term of component instances group is introduced.

Component instances group is established by zero or more component instances. These instances are physically separate and independent, but their connections follow the same pattern. For example in case of master-worker architecture, every single worker is connected with master, therefore despite that workers are individual they are connected according to the same pattern. If so, we may logically treat them as a component instances group and represent it as an ensemble in the qualitative component diagram.

When a component is connected to a collection, there is a need for an addressing scheme for indexing the multiple connections. Connection qualifier defines
the addressing scheme in pluggable mechanism, and such basic qualifiers as list or map are provided. The connections are parametrized by qualifier-specific properties such as list length or map key set.

The important rule in our system is that the connection multiplicity implies how many of component instances of a group are to be created. For example, if a master component is connected to the workers collection using a list qualifier, then the number of workers will be equal to the list length.

Obviously, since the qualitative component diagram may not be a simple tree, the connection cycles may exist. It may result in the unambiguity of the number of component instances. This issue amongst the others is covered by Component Graph Builder (CGB) which validates the diagram.

Moccaccino can support as well ordinary quantitative component diagrams by describing all connections as individual (with multiplicity equal to one) and then treating component instances group as a single component instance.

It is also worth noting that in ADLM number of Moccaccino-specific decorations are present. These include attributes associated with component instances group and connections which provide information about computational power needs and connection load which are used for deployment optimization.

4 Architecture of Moccaccino

In order to validate the proposed ADL approach, the Moccaccino Manager which was designed and implemented. Figure 1 presents the Moccaccino components’ dependencies along with results of their activities.

The Moccaccino Manager is responsible for deployment, execution and management of the application. Its functional components are: ADLM Unmarshaller, Component Graph Builder (CGB), Kernel Information Provider (KIP), Deployment Planner (DP) and Application Deployer (AD).

The ADLM Unmarshaller component is responsible for parsing the file that contains the application description ADLM file (XML-based). While parsing, the application model is built using Component Graph Builder to create components graph. The model that represents application is called Application Object Model (AOM). It is a data structure that can be exchanged between functional components to construct and deploy the application.

The deployment process itself is handled by the Application Deployer component. The result of his action is a deployed application and a handle to it, which Manager can further use. Application Deployer uses a Deployment Plan, which assigns H2O kernel to each application’s MOCCA component. This plan is built by the Deployment Planner, that can implement different policies to make the plan possibly optimal. The H2O kernel information needed for the construction for
Deployment Plan are obtained from the Kernel Information Provider component which may be configured to run in static mode - by providing a list of available kernels or dynamic mode in which case it uses the HDNS registry.

Moccaccino provides also a tool for visualizing the application model, expressed either in AOM or ADLM form.

5 Integration within the STE Institute

The approach described here fits well in the context of the Systems, Tools and Environments Institute of CoreGRID mainly for the reason that it follows the generic component paradigm. There are also two more specific integration activities, which strengthen the relevance of the presented work within the Institute.

The first is the response to the newly drafted GCM (Grid Component Model) specification. Our recent work shows that the CCA and GCM components can be integrated into one component system, and moreover two frameworks: one CCA-based as MOCCA and another Fractal-based as ProActive can interoperate at runtime and build one large distributed component applications.

The second one is the activity of the Application Deployment Component research group. The Application Manager tool and ADL-based description within Moccaccino project as an alternative solution bring a new experience to this re-

Figure 1: Moccaccino components
search group and helps better understand and analyze the process of component deployment on Grid.

6 Summary and future work

In this paper we presented Moccaccino: an application manager tool for supporting CCA components on the Grid. The tool required developing a new Application Description Language, which supports hierarchical parametrized component groups and connections as well as performance hints for optimizing the deployment. The current and future work includes development of deployment and adaptation algorithms, introducing on-line monitoring support, tests on real applications and further integration with deployment solutions within GCM.

Acknowledgments: This work was supported by EU IST CoreGRID project and Polish grant SPUB-M. Maciej Malawski acknowledges the support from the Foundation for Polish Science.

References