Advanced Grid Management Software for Seamless Services

Sahin Albayrak¹, Silvan Kaiser¹, Jan Stender¹

¹DAI-Labor, Technische Universität Berlin, Secretary GOR 1-1, Franklinstrasse 28/29, D-10587 Berlin, Germany
{sahin.albayrak, silvan.kaiser, jan.stender}@dai-labor.de
Fon: +4930 31421758, Fax: +4930 31421799

Abstract

Even though common Grid Management Software, like Condor or the SUN Grid Engine, features sophisticated functionality to execute and observe jobs on a computing Grid, it often lacks a comprehensive support for multi-modal access, security, accounting and advanced messaging. To tackle these shortcomings, the concept of an agent-based Advanced Grid Management Software is introduced. The approach envisions a multi-agent architecture forming an additional access and management layer which rests on top of conventional Grid Management Software implementations. Agents are supposed to remedy the aforementioned deficiencies by providing value-added services which extend the basic functionalities of the underlying Grid. An agent-based service platform for multi-modal access is used to feature an enhanced user interface to the functionality offered by the Advanced Grid Management Software.

Keywords: Grid Management Software, Agent, Multi-modal Access, Service
1 Introduction

As high performance computing is becoming more and more prominent in different sciences like astronomy, medical and biological analysis, etc., Grid technology becomes more important than ever as a way of providing massive computing power. Mainly because of smaller financial cost and better flexibility than provided by monolithic computing systems, Grids allow interactions like resource sharing on levels unknown so far or even realise virtual organisations as envisioned by Foster, Kesselman and Tuecke [18]. Based on the idea of Cluster Computing that relies on an integration of local homogeneous resources, Grid technology strives for merging distributed computing resources to a more powerful abstract machine that can handle extremely compute-intensive tasks. This is realised by a Grid Management Software (GMS), also known as a Distributed Resource Manager (DRM), a layer that offers a transparent access mechanism to distributed hardware resources.

In order to use this approach for novel solutions based on services, an additional software layer is introduced. This layer, in the following referred to as Advanced Grid Management Software (AGMS), realises several service-specific functionalities like flexible notification messaging, AAA and QoS. In this work the basic system architecture of the AGMS is introduced. It consists of a multi access service platform (MASP) [11] and a service delivery platform (SDP). The MASP provides multi modal access to the services offered by the SDP, so that these services can be used with different devices in order to enhance flexibility. The SDP hosts agents that provide a range of basic and value-added services of the Grid. While basic services are an abstraction of the normal Grid functionalities provided by the underlying GMS, value-added services offer improved functionalities like intelligent message notification. A deeper look at these services will be given later in the document.

In section 2, related technical problems are discussed. Section 3 gives an overview of state-of-the-art Grid and agent technologies which is followed by a detailed description of AGMS architecture in section 4. A final consideration of the ideas introduced in this paper follows in section 5.

2 Problem Description

Although Grids provide massive computing power, there is a range of problems associated with the use of Grid computing. Most programs run on Grids are custom applications which are exclusively designed for the underlying GMS. Such GMS generally lacks the functionality needed for a flexible service-based approach. Grid jobs are typically batch jobs that generally perform calculations on a given set of data with minimal or even no user interaction. In a service-driven environment the
focus changes to more interaction-oriented applications. Corresponding jobs usually have a more complex behaviour during their lifetimes, in comparison with common batch jobs. A Grid is a complex distributed system and therefore demands a close observation and administration in order to identify and handle errors as fast and efficiently as possible.

As Grid infrastructures are generally used at day and night time, powerful notification mechanisms are needed in order to keep administration personnel and users informed anytime and anywhere. However, GMS generally offers only e-mail as a means of notification. Here, more dynamic and flexible functionalities are needed in order to alert and inform faster if problems arise in the system. The delivery of job information to users that submit jobs to the Grid system can be an important issue, especially if execution problems occur in connection with costly jobs. Since notification mechanisms are mainly reduced to job logs and status e-mails, more sophisticated reporting mechanisms are necessary. SMS or voice notification can provide a faster flow of information.

Access to the Grids UI is generally provided by a native interface application. In order to provide more flexibility, web interfaces are offered by some systems. The Gridsphere project [22] even tries to provide a generic framework for web access to heterogeneous Grid installations. However, HTML based access is not enough. Here, solutions adopting to different devices and media are required, as ubiquitous computing environments turn out to become more and more common. Since submission and observation of Grid jobs does not make high demands on the devices that are used, ubiquitous access can be a major improvement. The opportunity to integrate mobile devices and the like should not be missed.

Another topic of interest are AAA aspects. User authentication and authorisation systems need to be bound to sophisticated accounting mechanisms and have to provide flexible and secure user management functionalities. Basic user management relying on the operating system is insufficient when thinking about business cases which offer Grid resources to external customers. Here a separate user space is needed, offering several levels (company, department, working group, etc.) for grouping users that don’t get direct system access. For the same reasons, accounting is needed which is not or only partially supported by most GMS. Without an comprehensive accounting framework sophisticated accounting tariffs cannot be employed which hampers business opportunities.
3 State of the Art

This section describes the technologies used in the AGMS approach: Grid, agents and multi-modal access technologies.

3.1 Grid Technology

Grid Technology currently focuses on two basic concepts. On the one hand, low level Grids provide computing power on homogeneous systems which are usually connected in a local area network. Examples include the Sun Grid Engine [4] and Condor [27]. On the other hand, there is the high level approach of the Global Grid as promoted by the Globus Alliance [5] which focuses on a Web Service-based network of distributed services. With regard to this work, the examination of the low level Grids is more important, so a closer look is taken at these concepts.

The Open Source Sun Grid Engine (SGE) is among the most widespread Grid management systems in use today. While the system itself is developed by open source methods supported by Sun’s staff, Sun offers commercial versions with additional tools. The Sun Grid Engine is available for several platforms. Currently supported are Solaris, Linux, Mac OS/X and various other Unix derivatives. Jobs are scripted and then scheduled through command line or GUI tools and sent to the executing nodes in the Grid. On the executing node the script is started and the job executed. The management software allows to submit not only single jobs but also queues of jobs. Jobs are automatically spread throughout the Grid based on its load balancing mechanisms. Exit status and output of the jobs are returned after the scheduled scripts have finished. This approach allows the distribution of arbitrary jobs. Basically there is no need for a Grid-specific implementation, as this concept distributes the call of the scheduled program, not the program itself. Thus any program installed can be started. However, all nodes in the Grid have to be able to execute the scheduled call, so generally a common file mount is used as the program and data storage space. The SGE features a sophisticated user management with several organisation levels and resource handling mechanisms. Checkpointing is supported in a basic manner, i.e. the application has to handle checkpointing operations like saving and loading status information by itself. Unlike other GMS, the SGE supports X Window System display forwarding [7] for jobs with GUIs.

Condor [27] basically uses similar techniques and relies on the same concept for realising a Grid. It supports a wide range of Unix and Windows systems. However, Condor differs from the SGE in several points. For example, checkpointing support does not require source changes to an application. Using the Condor checkpointing libraries, applications can be recompiled to allow saving and relocating a running job to a different node. This feature, however, requires absolutely
homogeneous nodes as saved memory pages, etc. have to be executable on the new node, too, which largely reduces its effectiveness in heterogeneous Grids. Condor does not transport execution scripts but application binaries to the execution nodes. This allows more a flexible usage of the Grid since the jobs can be executed at nodes at which the application has not been previously installed. In order to ensure that only nodes with environments that match the application requirements are chosen for execution, Condor uses a configuration mechanism. Nodes are defined as members of so-called universes. By specifying a defined universe in the configuration of a given job, Condor delegates the job only to nodes with a matching universe.

An example of a commercial Grid computing framework is the Load Sharing Facility (LSF [29]) developed by Platform Computing Inc. LSF belongs to a family of products constituting a complex workload management framework supporting submission, scheduling and monitoring of jobs in communities of heterogeneous distributed computing resources. Particular importance is attached to the interconnection and interoperability with different operating systems and Grid middleware infrastructures. Like Condor, LSF features support for Unix as well as Windows systems. OGSI-compliant resource managers facilitate compatibility of the LSF framework with existing meta-scheduling architectures, such as GRAM [2]. This offers a means for job handling in wide area networks. Support of advanced resource reservation and allocation based on service level agreements exists, just as security and accounting functionality. Self-management, including self-healing and fail-over mechanisms as well as automatic adaptation to changes in the infrastructure are another property of the LSF. Job state notification, however, is reduced to command line outputs. In addition to the possibility of a command line job interaction, a web-based user interface facilitates access to the infrastructure.

3.2 Agent Technology

Similar to Grid technology, the domain of software agents strongly relies on middleware concepts as well. The different agent technologies currently in use are much more diversified than the Grid concepts. Herein we focus on in Cougaar [25], as well as systems following the FIPA architecture [1] as implemented in JADE [10] or the JIAC [8] Serviceware Framework. These software agents provide development frameworks as well as scalable and flexible runtime environments for the developer. Based on the Java [6] platform, these implementations are OS independent and realise a powerful middleware concept.

In [24] JIAC is described as “...a generic agent toolkit which is intended as a service-ware framework for the realisation of applications in the areas of telecommunications, mobile services and electronic commerce.”. This toolkit includes a scalable component architecture, AAA services, a
security infrastructure and a wide range of development tools. Led by an Agent-Oriented Software Engineering (AOSE) methodology, JIAC applications are composed of services provided by agents. Such agents are located on agent platforms which may be distributed on different computers and constitute the runtime environment. Communication between agents can take place either on the local agent platform or over the network using different protocols like TCP, SSL or HTTP. Services are implemented in JADL (JIAC Agent Description Language), a language based on declarative concepts. For handling peripheral access, Java components can be added to JIAC agents, e.g. for accessing standard APIs like Java Mail. The toolkit contains a wide range of security functionalities in order to safeguard communication and apply authentication and authorisation services. The generic accounting framework included in JIAC IV relies on these implementations, so as to realise a secure and flexible accounting infrastructure.

Cougaar [25], funded by the DARPA, is a wide-spread, Java-based open source framework for the development of large-scale distributed agent-based applications. Although Cougaar is not completely FIPA-compliant, the basic architectural properties resemble to those of JIAC, as Cougaar also relies on a component model. However, it is reduced to a minimum complexity in terms of standard core functionality, which entails less restrictions on the way a Cougaar agent must be implemented. Special languages for service development, such as JADL in case of JIAC, are not explicitly encouraged. The main focus of Cougaar is set on aspects like security, stability and scalability, as Cougaar was designed with regard to military usage.

Data between Cougaar agents and Cougaar agent components respectively, is shared by means of a blackboard where objects representing tasks as well as resources can be stored. This may be done either by the agent itself or by a remote agent. Moreover, the blackboard supports a mechanism for event-based notification in case of a change in the blackboard content. As no particular emphasis is placed on service provision, however, the communication model significantly differs from the one which JIAC propagates.

### 3.3 Agents & Grid Technology

Similarities between Grid and Agent Technologies offer much room for combining the two domains. As pointed out in [16] both concepts have a range of analogies. For instance, both are inherently distributed and both focus on creating communities which are called Virtual Organisations in connection with the Grid and Multi Agent Systems in connection with the Agent domain. While the Grid domain focuses on basic technologies for interoperability, scalability, robustness and security, the Agent domain deals with coordination, interaction and dynamic organisation. The most suitable approach for combining both is seen in applying agents to the management tasks
in a Grid environment, using their strengths for controlling the complex and distributed structure while using the Grid infrastructure as a stable and reliable job execution environment. Examples for such work can be found in [14], [13], [12] where agents are used to manage Grid resources. Those agents negotiate with each other in order to find the most suitable resource for executing a given job. This is done by processing current system load data as well as information obtained from load prediction techniques. As shown in [12], such approaches can improve load balancing in large Grid structures. Further related work can be found in [19], [26], [20] and [21].

3.4 Multi Modal Access Technologies

An agent-based solution to multi-modal service usage is provided by the Multi Access Service Platform (MASP) [11]. Built on top of the JIAC agent framework, the MASP provides support for a wide range of different user interfaces, as e.g. HTML forms, voice and devices, such as PCs, PDAs and cell phones. XML-based interface and device description languages feature a generic approach to the device-independent development of multi-modally accessible JIAC services. In response to a user interaction, the MASP selects and invokes the adequate MASP-enabled agent service. Service parameters are generated from the user input, and results are rendered as defined in the user interface description, with regard to the user’s current access device. This approach affords the possibility of a high level device-independent access to agent services.

In addition to service access, the MASP offers sophisticated mechanisms to facilitate complex user interaction, supporting interruption and continuation of a user session on different devices through the Session Manager. Furthermore, it fulfills high requirements in terms of scalability and modularity, as the service presentation remains independent of its provision. This enables adding, changing and removing MASP-enabled JIAC services at runtime. Enhanced user interfaces based on the MASP exist within the scope of the projects BerlinTainment [28] and PIA [9].

A similar technique driven by the objective of device independence is UIML [23]. It proposes an XML language for device independent user interface design. UIML is not meant for on-the-fly generation of user interfaces but for a consistent user interface development across different platforms. The user interface description can be converted into languages like Java or HTML. Recent projects focus on the generation of user interfaces for different devices. Due to its static nature, this approach is unsuitable for our purpose, though.
4 Approach: Advanced Grid Management Software

As described in [16], there is a wide range of potential synergies between Grid and agent technologies. As described before, a promising approach to make use of these synergies is to build an agent-based layer on top of the Grid layer. Adding a layer that wraps basic services and offers agent-based advanced services yields benefits in terms of access to Grid functionality as well as observing and controlling Grid behaviour. In the following an Advanced Grid Management Software (AGMS) is introduced which forms a JIAC-based management layer which is designed to enhance existing GMS implementations.

4.1 Basic Concepts of the AGMS

To achieve the goal of a service-oriented job handling, an agent-based approach is pursued which relies a JIAC multi-agent architecture. The basic idea is to offer agent services which allow a seamless execution, monitoring and manipulation of jobs controlled by common GMS in a more device-independent and flexible fashion. Support of different GMS implementations is planned, in order to provide a uniform job submission and control mechanism. Compatibility with existing OGSA [17] implementations is envisioned by offering WSRF compliant Web Services.

As illustrated in figure 1, the concept involves two agent platforms constituting the AGMS: The Service Delivery Platform (SDP) and the Multi Access Service Platform (MASP). While the MASP can be run on an arbitrary node in the network, the SDP is run on the submission host. It comprises a set of agents providing services of two kinds, namely basic and value-added services. As a basic service, we consider an agent service providing an interface to a function which the GMS explicitly offers, e.g. job submission, job state querying or functionality relating to the administration of the Grid. Access is realised through Java APIs or by using the command line applications if no API can be used. The basic services are wrapped by the AGMS for transparency reasons, so that these services can be offered on AGMS level in an unchanged manner.

4.2 Value-added Services

As described in section 2 and shown in table 1, there are several problem areas which are not, superficially or only partially addressed by existing GMS. In order to deal with this issue, a set of special value-added services is introduced, extending the underlying Grid functionality.

One problem area covered by such a value-added service is user management and security, as like Condor, some low level Grid architectures barely address these aspects. An important task of the SDP is hence to provide unified authentication and authorisation methods allowing for
controlled access to the Grid functionality. Our plan is to make use of JIAC user management and security components. The latter include an implementation utilising the JAAS [3] framework which provides sophisticated mechanisms to handle security aspects within complex user interactions in a standardised, platform independent and flexible way. Thus, access and usage of an underlying GMS can be controlled and managed even if the GMS itself does not support comprehensive functionalities in this area.

Applying the accounting framework to basic services takes care of providing business data for billing systems, etc. By wrapping the basic services of the underlying GMS, the AGMS can collect accounting data for resources used, taking into account service dependent tariffs for flexible resource billing. Tariffs can rely on accounting operations in the AGMS or functions of the underlying GMS itself. If the GMS does not provide sufficiently detailed information about Grid nodes concerned with a certain job, agents are used to retrieve such information by observing such Grid nodes. The JIAC accounting framework is integrated in the aforementioned user management and security services.

Value added services for advanced notification include a list of functionalities. Upon receiving a message for an administrator or user, agents running on the AGMS try to inform the corresponding person or group. Different parameters matter with respect to this, such as the priority of the information and the personal preferences of the corresponding receivers. Agents attempt to change the media in case the chosen message transfer mode was unsuccessful. Urgent messages are sent via different modes by a cascading method in order to ensure that emergency information reaches the intended recipient quickly. If possible, this is done by monitoring the reception of the message and the reaction time of the corresponding person. If there is no reaction, other media are tried. This process cascades through the different media until the system has tried all available techniques.

Besides controlling message delivery the agents check the number and frequency of messages for the individual users. Traffic is reduced when a large system failure threatens to result in massive spamming of a recipient. This, however, has to take into account the different message priorities, and meta reports need to be generated to keep the administrators informed without flooding them. Beside this messaging services the MASP provides the main user interface for the Grid.

4.3 MASP

The MASP is meant to form the second part of the AGMS. The JIAC Serviceware Framework already brings along the functionality required for a seamless multi-modal service access. This will facilitate the development of device-independent user interfaces based on HTML, WML or VoiceXML for advanced Grid level access, which allows ubiquitous interaction with the SDP. Basic
services as well as value-added services can thus be made available to the user.

The MASP implements a layered architecture. JIAC agent services which constitute the bottom layer are exposed to the user by means of so-called Human Services. A Human Service may comprise several JIAC services, supporting complex user interaction based on simple JIAC service usages. Semantic and structural information which is required to present service output to the user and user input to the service in a useful and perspicuous way, is added to the human service at the Generic Presentation Layer. The transformation layer takes care of the interface generation. For each human service, an XML-based user interface description for a device family defines how service input and output parameters are rendered. Albeit different modes and devices for user interaction may be based on the same rendering mechanisms, such as in case of HTML for both PDAs and PCs, the MASP takes care of selecting the proper interface description, depending on the user’s current access device constraints. For instance media providers supply scaled versions of graphical elements in a user interface, in order to adopt to a small display. A Central Access Portal constituting the top layer offers an interface to a directory of all MASP-accessible services which also provides support for service lookup. A more detailed overview of the MASP architecture is given in [15].

As the MASP is designed to hold different device profiles, user interfaces can easily be accessed by mobile devices. This offers opportunities like Grid-based image analysis of pictures taken by a mobile device utilising the resource of the Grid from a remote site. As stated in section 3.4, the session management of the MASP allows seamless interruption and continuation from different devices which gives the user extended flexibility when using AGMS services.

5 Roundup and Outlook

In this paper, an Advanced Grid Management Software has been introduced that acts as an access and management layer on state-of-the-art GMS implementations. After having described several shortcomings with existing GMS implementations, an overview of common Grid and agent technology has been given. A concept has been described which points out how agent technologies can be used to form an AGMS that deals with these shortcomings. A collection of value-added services provided by the AGMS aim for improving quality, usability and flexibility of the Grid.

Future research will be done on aspects associated with different kinds of value-added services as well as Meta Grid composition based on the AGMS layer. Each of these fields offers a wide range of working opportunities.
References


[22] Novotny, Jason; Russell, Michael; Wehrens, Oliver: *GridSphere: A Portal Framework for Building Collaborations*; In: j-CCPE; 16:5 (April 2004), pp. 503-513


[27] Tannenbaum, Todd; Wright, Derek; Miller, Karen; Livny, Miron: *Condor – A Distributed Job Scheduler*; In: Beowulf Cluster Computing with Linux, MIT Press, October 2001


Prof. Dr. Sahin Albayrak is founder and scientific director of the Distributed Artificial Intelligence (DAI)-Labor. Since July 2003 Prof. Albayrak owns the chair of Agent Technologies in Business Applications and Telecommunication at Technische Universität Berlin. In 2004 he established the Center of Excellence for Seamless and Intelligent Grid Services for e-Science at DAI Labor.

Silvan Kaiser is scientific assistant at the DAI Labor of the Technische Universität Berlin. In his past he worked on agent based distributed applications and messaging components, afterwards heading the development of the Java-based Intelligent Agent Componentware (JIAC) Project. His late work focuses on the combination of Grid and agent technologies.

Jan Stender is a student co-worker at DAI-Labor. He currently works in the JIAC Project, with his main research interests being distributed computing and Grid technology.
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Sun Grid Engine</th>
<th>Condor</th>
<th>LSF</th>
<th>AGMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>sophisticated user</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>management</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accounting support</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>advanced messaging</td>
<td>e-mail</td>
<td>e-mail</td>
<td>e-mail</td>
<td>e-mail, SMS, MMS, ...</td>
</tr>
<tr>
<td>multi-modal access</td>
<td>HTML interface</td>
<td>HTML</td>
<td>HTML</td>
<td>HTML, WML, VoiceXML</td>
</tr>
</tbody>
</table>

Table 1: Comparison of GMS concerning service-centric functionality
Figure 1: AGMS architecture
Figure 1: AGMS architecture