Real-Time Performance Support for Complex Grid Applications

Marian Bubak[1,2], Włodzimierz Funika[1], Bartosz Baliś[1], Tomasz Szepieniec[2], Krzysztof Guzy[2], and Roland Wismüller[3]

[1] Institute of Computer Science AGH, al. Mickiewicza 30, 30-059 Krakow, Poland
[2] ACC CYFRONET AGH, Nawojki 11, 30-950 Krakow, Poland
[3] Universität Siegen, Hölderlinstr. 3, 57068 Siegen, Germany
{bubak,funika,balis}@uci.agh.edu.pl,
t.szepieniec@cyfronet.edu.pl, kjguzy@gmail.com,
roland.wismueller@uni-siegen.de

Abstract. One of the tasks in the EU IST int.eu.grid project is to create a pilot application which exploits Grid computational resources, based on the HEP application coming from the ATLAS project. When bringing the HEP application to the Grid, one of the problems is how to distribute the computations in an effective way to make the most of available Grid resources. Some support from monitoring is needed to assist in an optimal computation distribution. The HEP application is supported with monitoring information coming from two systems: JIMS and OCM-G. In this paper we present the issues of providing data for Real Time Dispatcher, based on the functionality of the two monitoring systems systems coupled into a single infrastructure. It is aimed to support fulfilling the real-time requirements posed by the HEP application.

Keywords: grid, monitoring, HEP, LHC, application optimization, MBeans.

1 Introduction

The objective of the EU IST Interactive European Grid (int.eu.grid) project [1] is to deploy an advanced Grid empowered infrastructure in the European Research Area (ERA). Within the project, research on its pilot application from the domain of High Energy Physics (HEP application) [4] is aimed at using the interactive grid environment to support processing of data coming from the biggest particle accelerator - the Large Hadron Collider (LHC). The current architecture of the High-Level Trigger (HLT), part of this system, requires building a massive computer farm of order of 1000s processors employed just for the selection of the most interesting collisions; it heavily involves the network infrastructure. So far in the local processing model exploited in the ATLAS TDAQ system, the data on the events were transferred for pre-processing to local computing farms. The idea behind the HEP application gridification is to delegate part of pre-processing tasks to computer resources available on the Grid.

E. César et al. (Eds.): Euro-Par 2008 Workshops, LNCS 5415, pp. 329–338, 2009.

The High Energy Physics application (HEP) poses hard challenges related to the Grid environment, specifically, in case of LHC experiments, their requirements go far beyond the ones typically involving storage and computational issues. The main task of the ATLAS TDAQ system is to select interesting events out of 1 Giga interactions per second generated in collisions at the LHC accelerator and record them on permanent storage with frequency of O(300 Hz). The system is based on two levels of online selection algorithms. The TDAQ system is logically divided into a fast First Level Trigger (L1), and High Level Trigger system (L2) which involves the next two selection stages. The First Level Trigger (L1) provides an initial reduction of the LHC's 40 MHz nominal bunch crossing rate to 75 kHz. Given a mean ATLAS event size of 1.6 Mbyte, this corresponds to a throughput of ca. 120 GB/s. The first stage of L2 reduces the event rate further to 3.5 kHz, which corresponds to a total throughput of about 6 GByte/s out of the event building system. The data fragments of events accepted by the first stage of L2 algorithms are collected by the event building nodes (SFIs) (see Fig. 1) from detector buffers. The resulting complete event fragments are then sent to the Event Filter Processors (EFPs) (see Fig. 1) for the last selection stage (i.e. the second stage of L2).

To enable the filtering of events during the second stage of L2, a need of 1600 EFP nodes (dual-CPU machines) is foreseen. To meet this challenge, it is proposed to extend the EFP system by the possibilities of a Grid infrastructure to allocate necessary processing power with fast access to get the data processed in realtime (O(1s)). A non-timely response to an event leads to the irreversible loss of the data coming from the real LHC experiment. Taking into account that submission of jobs to the Grid may take quite an amount of time, the use of the pilot jobs idea [2,3] is a proper solution. Just before the experiment starts, jobs are submitted to the Grid. These jobs allocate resources and establish a communication channel. They start waiting for tasks obtained from the experiment.

To use this idea, some middleware support for the HEP application is needed. The Real-Time Dispatcher (RTD) is a software solution dedicated to facilitate the delegation of computations to the Grid infrastructure. It coordinates the distribution of data obtained from the ATLAS TDAQ system to remote processing tasks (PT). A remote processing task is a grid job which is submitted in advance to wait for data to work on. The data is sent by a *proxy processing task* (proxy PT) which is running on local resources in CERN. The proxy PT is behaving in the same manner like an ordinary local processing task with the exception of delegating an actual computation to a remote PT. When remote PTs obtain the data from the proxy they perform the computations and return the result which is further used by the ATLAS TDAQ system to judge whether this data could describe some interesting collisions from the scientific viewpoint. On having returned a result the remote PT is ready for getting the next data to process. So the main RTD's purpose is to bind the proxy and a remote PT. When the proxy PT starts, it requests a remote PT to be assigned. As RTD has a pool of remote PTs, it must decide which one will be connected with the requesting proxy PT. The selected remote PT should process the coming events' data at

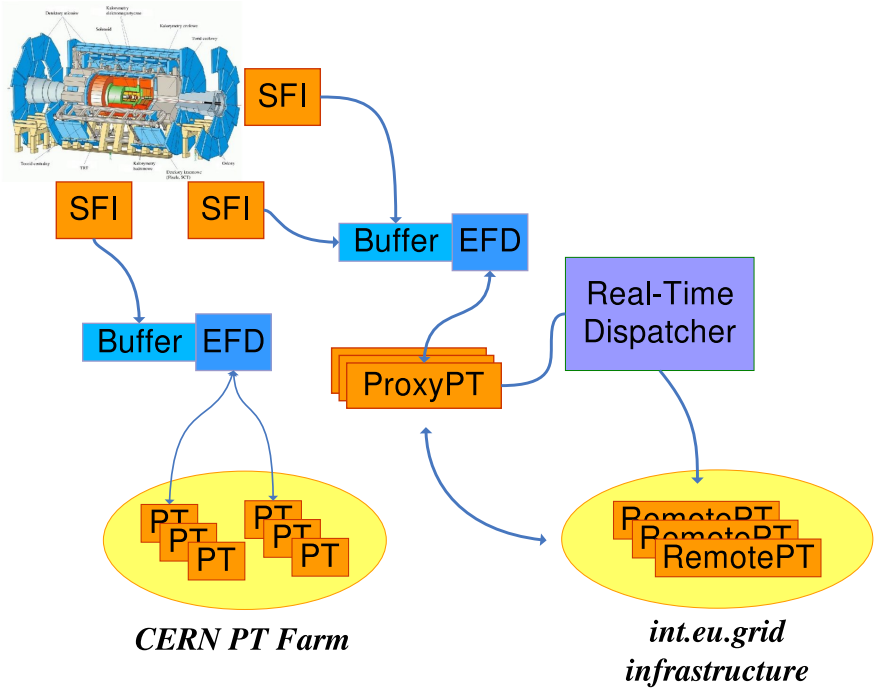

Fig. 1. The idea of HEP application

the fastest rate. Thus to make proper decisions, the RTD needs to be provided with needed information by a monitoring facility. For this goal we exploited two monitoring facilities, one for obtaining data on network load, while for better performance and less resource usage (mainly, main memory) we decided to use another lightweight monitoring facility which provides fast access to the data regarding the processing performance of the nodes where remote PTs run.

Fig. 1 presents a general concept of the HEP application. One can see how the local processing (on CERN PT Farm) is augmented by the use of Grid resources (based on the int.eu.grid's infrastructure) with RTD, proxy, and remote PTs.

In the following sections, we focus on the requirements for the monitoring support from the HEP application and make a brief overview of the monitoring systems we used to fulfill these requirements. Then the issues of coupling these systems are discussed. We will give details of co-operation of these facilities over the application life-time to match the real-time application requirements. The last section includes conclusions and future work.

2 Monitoring Requirements for the HEP Application

As mentioned above, RTD needs some monitoring information to choose a remote PT which should process the events' data at the fastest rate. To make this

choice, the RTD is to be provided with the information on the network load and resources usage of the worker nodes where remote PTs are running. Information on the load of a network link is important to determine whether a given grid site is able to receive the data fast enough. On the other side, knowing the resource usage of worker nodes (CPU load and the amount of available memory), we can distribute computations to less loaded nodes. Making use of this information, RTD can choose the most appropriate remote PT at the moment. To fulfill the monitoring requirements from the HEP application we decided to use two monitoring systems: JIMS and OCM-G. These monitoring systems will be described concisely in the following subsections before we proceed to the discussion on an integrated monitoring infrastructure for the HEP application.

2.1 JIMS Monitoring Infrastructure

JIMS – the JMX-based Infrastructure Monitoring System [5] is based on the Java Management Extensions platform (JMX) where monitored resources are represented as MBeans (Managed Beans), simple Java objects, installed on MBean Servers. JIMS can automatically adapt itself to operating systems, kernels, and IP protocols. It provides an auto-configuration facility (cluster and Grid level auto-configuration) and dynamic deployment of proper monitoring sensors. All these features make it well suited for Grid environments.

Figure 2 presents the JIMS architecture. It consists of three types of agents. The role an agent performs depends on the modules where the agent is deployed. One agent provides the global discovery service which delivers an integration layer, it finds all running SOAP Gateway agents, thus provides an overall view of the Grid. The proxy agent, in turn, finds and provides access to all currently running monitoring agents on a cluster. An ordinary monitoring agent, which is running on a worker node, provides monitoring information collected by its

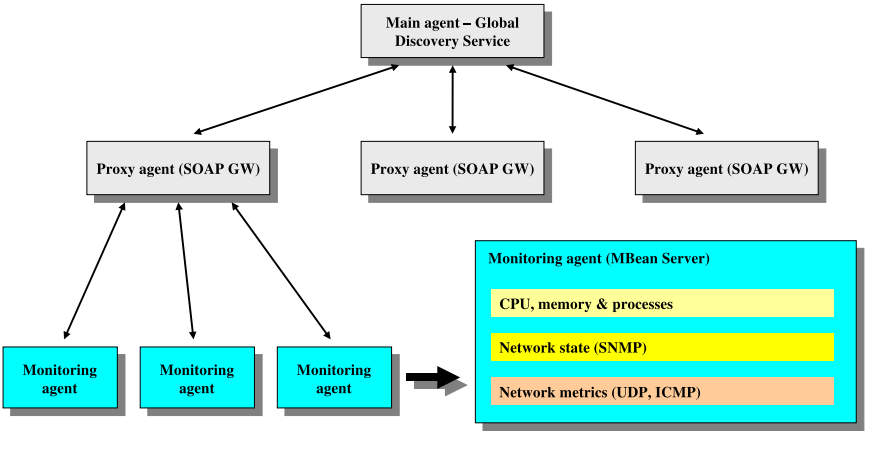

Fig. 2. JIMS components

sensor modules, i.e. MBeans. The figure gives a few examples of the information the sensors can provide.

2.2 OCM-G Monitoring System

OCM-G – OMIS-Compliant Monitoring system for the Grid [6,7] is a system for on-line monitoring of interactive Grid applications. It supports parallel/ distributed applications running across multiple sites. Due to the efficient request-reply mode of operation, OCM-G can underly development-support tools, specifically, performance analysis tools, like the GPM [8] performance measurement tool. It provides special support for monitoring applications exploiting message-passing and other programming paradigms. OCM-G can also provide information on the nodes where the application processes are running.

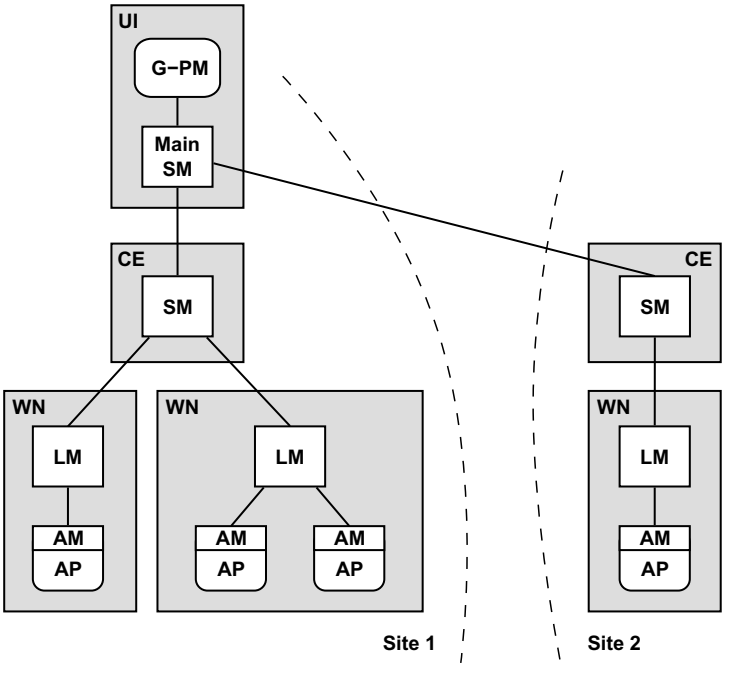

Fig. 3. OCM-G components distributed on the Grid

As shown in Figure 3, the OCM-G system comprises three types of monitor modules. *Main Service Manager* (MainSM) is a central component, one per each user. It is an access point to the system for a user. The location of MainSM is configurable. *Service Managers* (SM) are located in each site of the Grid, typically, on Computing Element machines (CE). *Local Monitors* (LM) are located on each Worker Node (WN) where the application processes (AP) under monitoring run and they actually perform the monitoring tasks. *Application Modules* (AM) are parts of OCM-G that are linked to the application processors.

2.3 Monitoring Strategy for the HEP Application

As the main monitoring data provider for RTD, the JIMS system has been chosen. JIMS is meant to be deployed in each site of the Virtual Organization, intended to support the HEP application (HEP VO). HEP VO is responsible for providing a necessary environment to run the HEP application on the Grid. Available grid resources are carefully checked during a certification procedure. Next, SLA [10] is signed and resources are included into a HEP VO. The HEP VO is endowed with a portal which simplifies the usage and management of VO.

JIMS will measure the available bandwidth to sites where events' data are processed. For this purpose, only one JIMS agent is needed, probably, on a computing element. In addition to the network load, RTD needs to obtain some data about worker nodes status: CPU load and memory usage, very frequently and rapidly. JIMS could provide this information but to do this it should be installed also on every worker node of HEP VO. To save resources of worker nodes (mainly, main memory) we decided to take the advantages of the OCM-G monitoring system, instead. For collecting monitoring information, OCM-G uses a monitor which is much more lightweight than a JIMS agent as to the memory required for its operation. Moreover, on a worker node the OCM-G monitor is started together with an application process, thus, the monitoring system don't exploit resources of the worker nodes where the application in not running at the moment. For this reasons we decided to monitor WNs using OCM-G which is intended to provide monitoring information to a JIMS agent. Next, RTD will obtain this information from JIMS. In the next section, we will give some details of the integration of OCM-G into JIMS .

3 Integration Mechanism

To integrate OCM-G with JIMS we needed to implement a new sensor module for JIMS. As sensor modules JIMS exploits MBeans. An MBean runs in a JIMS monitoring agent (MBean server) and provides some monitoring functionality. Thus, we created an MBean which hides the OCM-G system and provides the data about nodes' CPU load and memory usage through its exposed interface.

When loaded and started by a JIMS agent, the MBean spawns OCM-G's Main Service Manager process and connects to it. Then the Main Service Manager is waiting for Local Monitors to connect to (a Local Monitor is actually connected to Main Service Manager using a Site Service Manager). A Local Monitor is started whenever a remote PT is launched on a worker node. This is done by a script which also stops the Local Monitor when the remote PT exits. Local Monitors find the Main Service Manager's contact by reading a configuration file with host and port information. The list of currently monitored worker nodes can be obtained by reading the `NodeList` MBean's attribute. The value of this attribute is a comma separated list of hosts' names.

The MBean provides the user with the operations to start (`start`) and stop (`stop`) the underlying OCM-G system (Main Service Manager process). But these operation are rather not used directly as they are invoked by the MBean

when JIMS agent starts and stops. For the client, the interesting operations are those which provide needed monitoring information. At the moment, there are six operations provided by the MBean. All these operations take a host name as an argument. The first three ones: `getNodeLoadAvg1`, `getNodeLoadAvg5`, `getNodeLoadAvg15` return the load average values for a given worker node (i.e. the number of jobs in the run queue or waiting for disk I/O averaged, respectively, over 1, 5, and 15 minutes). They are the same as the load average values returned by *uptime* and other programs. The next operation, `getNodeMemFree`, returns the amount of available host's main memory. Invoking the last operation, `getNodeSwapFree`, we can get the amount of remaining swap space. When one of the MBean's operations is called, the MBean constructs a proper request and submits it to the OCM-G system. When an OCM-G's reply arrives, it is processed and returned within an MBean's operation result.

OCM-G is installed on HEP VO sites as an experiment software [9]. Therefore, the VO software manager is responsible for installation, configuration, and

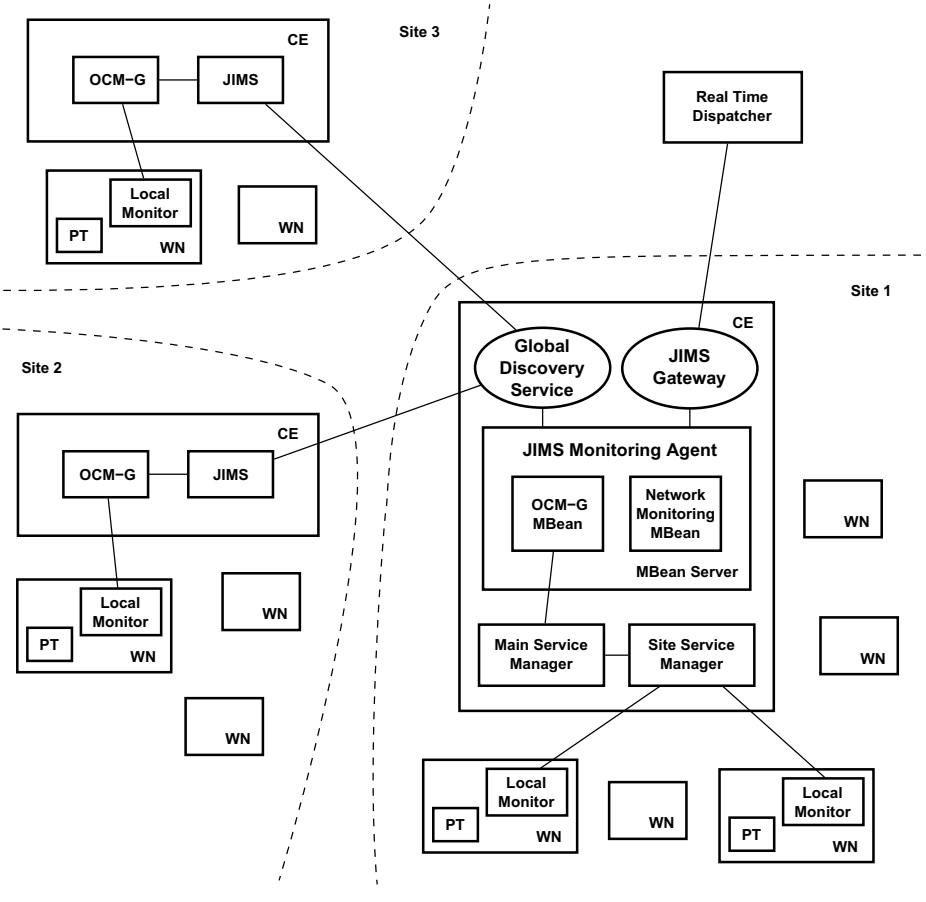

Fig. 4. Overview of JIMS and OCM-G integration

updating the software. The advantage is that a contact with a site administrator for these activities is not needed. JIMS agent is installed on a single host in the site (possibly, CE) and runs as a service. This agent has our MBean deployed and is an access point to the monitoring information collected by OCM-G's Local Monitors running on site's WNs. Using the JIMS infrastructure the data from all sites can be supplied to RTD. In order to facilitate a client's interaction with JIMS, there has been added a possibility to access all discovered agents using a single connection. Earlier, to find the address of an agent, a client had to connect to the Discovery Service first. Then, when the client obtained the address, it could connect to an agent directly. Now, all agents provide the Discovery Service and have an ability to forward a client request to other discovered agents. Thus, using a single connection with a chosen agent, a client can also interact with other agents. This makes easier gathering monitoring information from all the sites involved in the HEP application.

Figure 4 presents an overview of OCM-G and JIMS components and relationships between them. In one site its components are shown in details while in the two remaining ones the details are hidden. The RTD is connected to a JIMS agent. This agent serves as a JIMS Gateway for the site so it is accessible from outside. The agent also provides Discovery Service and owing to this the RTD can access other JIMS agents in the other sites. Within the depicted JIMS agent, there can be seen two MBeans. Our MBean is connected with the OCM-G system and there are OCM-G's monitors on the worker nodes where PTs are running.

The information provided by the OCM-G MBean can be accessed in different ways, e.g. through e.g. JIMS GUI. In Figure 5 a GUI, called JIMS Manager is

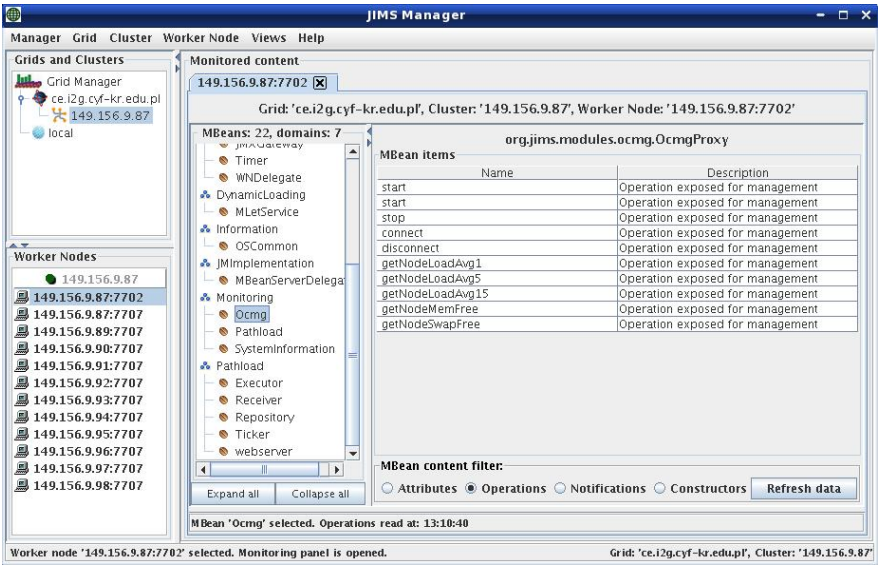

Fig. 5. OCM-G MBean in the JIMS Manager window

shown. On the left top panel (*Grids and Clusters*) there is a list of known JIMS agents which were specified in a configuration file. With selecting one of these agents we can start a process of finding other ones that will be shown on the left bottom panel (*Worker Nodes*). In the figure, on the left, there is an open tab for a one of the discovered agents. A list of MBeans the agent runs is presented. Once the OCM-G's MBean is selected, on the right there are shown our MBean's operations.

4 Conclusion and Future Work

The solution for middleware support, in particular, this related to monitoring functionality, presented in this paper was designed for the HEP event processing problem, which poses some rigorous time requirements for data processing. Usually, large delays in the event processing end up in the overloading of the queues holding the data coming from the trigger.

In this paper we focused on the requirements for monitoring support for Real Time Dispatcher in choosing proper remote Processing Tasks for performing computations within the High Energy Physics application explored as a pilot application in the `int.eu.grid` project. The designed monitoring strategy and the idea of how we had integrated OCM-G monitoring system into the JIMS infrastructure for the monitoring of the HEP application were shown.

At the moment, a working version of our solution is tested with RTD and remote PTs. In the future we intend to improve the reliability of OCM-G by an elimination of Main Service Manager component. This component is important when OCM-G is used independently on the Grid. It gathers monitoring information coming form all Site Service Managers (one per site). In our scenario, we've got one OCM-G system per site so the Main Service Manager is not needed and the Site Service Manager should be accessed by a client (OCM-G MBean) directly. Our further focus will be on the comparative study of the functionality of the monitoring infrastructure under discussion and similar systems, which will involve experiments with MonALISA [11], a distributed, auto-configuring and auto-deploying monitoring system that can provide quasi-real-time information about huge number of nodes. This will also enable us to study the scalability of of our monitoring system.

Acknowledgements. We are very grateful to dr. Renata Slota for valuable discussions. This research is partly supported by the EU IST `int.eu.grid` project IST-031857, the corresponding SPUB-M and the EU IST CoreGRID project.

References

1. EU IST int.eu.grid project's web page, `http://www.interactive-grid.eu/`
2. Thain, D., Livny, M.: Building Reliable Clients and Services. In: Foster, I., Kesselman, C. (eds.) The Grid: Blueprint for a New Computing Infrastructure, pp. 297–299. Morgan Kaufmann, San Francisco (2004)

3. Globus Falkon framework page, http://dev.globus.org/wiki/Incubator/Falkon
4. Dutka, L., Korcyl, K., Zielinski, K., Kitowski, J., Slota, R., Funika, W., Balos, K., Skital, L., Kryza, B.: Interactive European Grid Environment for HEP Application with Real Time Requirements. In: Bubak, M., Turala, M., Wiatr, K. (eds.) Proceedings of Cracow 2006 Grid Workshop, Cracow, Poland, October 15-18, 2006, pp. 11–20. ACC Cyfronet AGH (2007)
5. Zielinski, K., Jarzab, M., Wieczorek, D., Balos, K.: JIMS Extensions for Resource Monitoring and Management of Solaris 10. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2006. LNCS, vol. 3994, pp. 1039–1046. Springer, Heidelberg (2006)
6. Balis, B., Bubak, M., Funika, W., Wismueller, R., Radecki, M., Szepieniec, T., Arodz, T., Kurdziel, M.: Grid Environment for On-line Application Monitoring and Performance Analysis. Scientific Programming 12(4), 239–251 (2004)
7. OCM-G project home page, http://grid.cyfronet.pl/ocmg/
8. Wismüller, R., Bubak, M., Funika, W.: High-level application specific performance analysis using the G-PM tool. In: Di Martino, B., Kranzlmüller, D., Dongarra, J. (eds.) EuroPVM/MPI 2005. LNCS, vol. 3666, pp. 317–324. Springer, Heidelberg (2005)
9. Experiment Software Installation, https://edms.cern.ch/file/498080/1.0/SoftwareInstallation.pdf
10. Skital, L., Janusz, M., Slota, R., Kitowski, J.: Service Level Agreement metrics for real-time applications on the Grid. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 798–806. Springer, Heidelberg (2008)
11. The MonALISA web site, http://monalisa.cern.ch/monalisa.html