Computing and Informatics, Vol. 28, 2009, 353–367

ADAPTING A HEP APPLICATION FOR RUNNING ON THE GRID

Włodzimierz Funika[1], Krzysztof Korcyl[2], Jan Pieczykolan[3]
Łukasz Skitał[3], Kazimierz Bałos[3], Renata Słota[1]
Krzysztof Guzy[3], Łukasz Dutka[3], Jacek Kitowski[1,3]
Krzysztof Zieliński[1,3]

Institute of Computer Science AGH [1]
al. Mickiewicza 30, 30-059 Kraków, Poland

Institute of Nuclear Physics PAN [2]
ul. Radzikowskiego 152, 31-342 Kraków, Poland

ACC Cyfronet AGH [3]
ul. Nawojki 11, 30-950 Kraków, Poland
e-mail: funika@agh.edu.pl, Krzysztof.Korcyl@ifj.edu.pl

Revised manuscript received 16 May 2008

Abstract. The goal of the EU IST int.eu.grid project is to build middleware facilities which enable the execution of real-time and interactive applications on the Grid. Within this research, relevant support for the HEP application is provided by Virtual Organization, monitoring system, and real-time dispatcher (RTD). These facilities realize the pilot jobs idea that allows to allocate grid resources in advance and to analyze events in real time. In the paper we present HEP Virtual Organization, the details of monitoring, and RTD. We present the way of running the HEP application using the above facilities to fit into the real-time application requirements.

Keywords: Interactive application, real-time application, pilot job, virtual organization, SLA, infrastructure monitoring, application monitoring, grid middleware

1 INTRODUCTION

A High Energy Physics application (HEP) poses challenges related to the Grid environment, specifically, in case of LHC experiments, their requirements go far beyond those typically involving storage and computational issues. These requirements are also much more difficult to meet even than those coming from interactive applications. This is due to the fact that they become the requirements of real-time-like applications. The application which handles the LHC experiment has to process ca. 3 500 events per second while each event should be studied within less than 5 seconds. A non-timely response to an event leads to 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 is a proper solution. Just before the experiment starts, jobs are submitted to the Grid. These jobs allocate resources and establish communication channel. They start waiting for event data from the experiment. To use this idea, support for the HEP application is needed.

The goal of the EU IST int.eu.grid project [1] is to build middleware facilities which enable the execution of real time and interactive applications on the Grid. Within the research under discussion, to provide relevant support for the HEP application, Virtual Organization for HEP (HEP VO), monitoring system (MS), and real-time dispatcher (RTD) were developed. These facilities realize the pilot jobs idea that allows to allocate grid resources in advance and to analyze events in real time, concurrently with on-line communication with an experiment.

In the paper we will present HEP application, HEP VO with its certification procedure, SLA metrics, and VO management tools, the details of monitoring with JIMS and OCM-G, as well as the design and implementation of RTD. We will show the procedure of starting of HEP application with support of the above-mentioned facilities. We will give the details of co-operation of these facilities within the application lifetime to match the real time application requirements.

2 STATE-OF-THE-ART

The issues related to supporting the applications with interactivity-like and real-time requirements and exploiting grid environments are addressed nowadays by numerous researchers. Most of them refer to building grid-oriented middleware and applications supporting human users in interacting with a running experiment or simulation application. For example, the ViroLab project's [3] Virtual Laboratory aims to deliver a collaborative platform for e-Science, which is a set of dedicated tools and servers that form a common space of planning, building, improving and using in-silico experiments in the virology domain. Other environments refer to the research on enabling conducting long-running simulations on the grid, which need a high responsiveness rate and the ability of interactive steering. Such research was addressed by the CrossGrid project [2]. It was explicitly targeted towards interactive grid applications by providing tool and service support for layered distributed

compute- and data-intensive applications which need nearly real-time response. The software support involved tools for performance measurements and predictions, code verification, application specific and grid services: roaming access, scheduling agents, application and grid monitoring, data access optimization.

The existing environments do not meet fully the requirements stemming from the HEP application, since it features two special kinds of requirements. One of them is related to interactivity which is needed for the human operator to start/stop event analysis of the real experiment and to decide where the events will be sent to. On the other side the HEP application should meet the requirements for nearly real-time responsiveness, which are very important. This is because copies of events delegated to remote sites are buffered in local storage (just in case the forwarded data will be lost), and failures to return filtering decisions in time may result in introducing dead time in the system due to a lack of space for buffering. As the HEP application produces huge amounts of data, the obligatory requirement to meet the real-time demand is among others the quality of service provided by the network and communication protocols. Series of tests were conducted to check various types of networks and protocols: dedicated lightpath, VPN and WAN switched with various flavours of TCP/IP [4] and indicated that there were no basic limitations from the basic infrastructure. Another challenging issue for the HEP application is the size of the system reaching few thousands of nodes. To assess the scalability of the system, the HEP application was mapped onto a single grid farm [5]. There, the idea of pilot job (see later in the text) was introduced and used to assign roles of various components of the system to machines from the farm. That work demonstrated the ability of the system to run on a large scale with statically assigned machines and all components sharing a common file system. In our approach we aim at running the system on dynamically allocated machines scattered in different remote farms.

3 HEP APPLICATION IN THE GRID ENVIRONMENT

3.1 HEP Application

To support the processing of data coming from the largest particle accelerator – the Large Hadron Collider (LHC), the ATLAS Trigger and Data Acquisition System (ATLAS TDAQ) is created. 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 $O(300\,\mathrm{Hz})$.

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 (HLT) 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 MB, this corresponds to a throughput of ca. 120 GB/s.

The first stage of the second level reduces the event rate further to 3.5 kHz, which corresponds to a total throughput of about 6 GB/s out of the event building system. The data fragments of events accepted by the first stage of HLT algorithms are collected by the event building nodes (SFIs) from detector buffers. The resulting complete event fragments are then sent to the Event Filter Processors (EFPs) for the last selection stage (i.e. the second stage of HLT). The accepted events (ca. 0.2 KHz, 300 MB/s) are finally sent to the output nodes (SFOs) to be permanently saved on mass storage.

To enable the filtering of events during the second stage of L2, a need of 1 600 EFP nodes (dual-CPU machines) is foreseen. To meet this challenge, we propose to extend the EFP system by the possibilities of using a Grid infrastructure to allocate necessary processing power with fast access to get the data processed in real time ($O(1\,\mathrm{s})$).

Each EFP hosts an Event Filter Demon (EFD) instance, which provides the data flow functionalities, and several Processing Tasks (PTs) in charge of the actual data processing and event selection. The EFD manages the communication with the SFI and SFO elements and makes the events available to the PT processes via PT I/O. PT runs the EF algorithm in the standard ATLAS off-line framework (Athena).

3.2 Interactive Grid as a Solution

To use a grid environment for filtering events (second stage of L2), we need to address two main problems. First, how to delegate computations to the Grid, while avoiding drastic changes to ATLAS TDAQ; second, how to provide a proper servicing of the application on the Grid, taking into account its real-time requirements.

In order to be able to delegate some of the filtering work to remote computers, we propose changes relating to PT tasks only, i.e. the tasks explicitly involved in processing. It is not important to induce changes to the EFD daemon along with communication with SFI i SFO. We propose to split the PT task into two parts: proxyPT – the part being responsible for the data flow, and remotePT – the part performing data processing. The proxyPT runs on the same machine as EFD and implements the PT I/O library to communicate with the EFD.

In case a proper responsiveness of PT tasks is ensured, a challenge is to design and implement appropriate facilities (grid middleware) that realize the idea of pilot jobs. These are: Virtual Organization for HEP (HEP VO), monitoring system (MS), and real-time dispatcher (RTD).

HEP VO supports the preparation and management of a Grid environment in a two-stage manner: the goal of the first stage is to appropriately certify the sites on which computations will run, while the second stage will focus on monitoring a correct functioning of the environment. The control of appropriate operation of the environment is based on the defined relevant Service Level Agreement (SLA) metrics obtained with MS providing the monitoring of the environment infrastructure and the monitoring of the running application.

The monitoring of the infrastructure realized by the JIMS monitoring facility aims at obtaining performance data on network load needed to assess where and via which path event data is going to be transferred. The monitoring of the application supplying the data on the processor load and memory usage is based on the functionality of the OCM-G monitoring system.

A key role in the support of the HEP application is played by RTD. It is aimed at re-distributing the events coming from CERN to the jobs launched by HEP VO on the Grid. This is a realization of the above-mentioned pilot jobs idea.

In spite of the above challenges, it should be emphasized that delegating computations to the Grid is a solution, which enables in the long run the utilization of Grid computation power which is required when conducting such demanding experiments like HEP applications. The experience and the middleware developed are capable to provide portability to the Grid for many interactive or real-time applications.

In the following we are presenting in detail the design and implementation of the above-mentioned facilities, supporting the execution of the HEP application in the grid environment.

4 HEP VO

HEP VO is responsible for providing the necessary environment to run the HEP application on the Grid. Available grid resources are carefully checked during the certification procedure. Next, SLA [7] is signed and resources are included into a HEP VO. The HEP VO is endowed with a portal [8] which simplifies the usage and management of VO.

4.1 Certification Procedure and SLA

Since the HEP application requirements are difficult to fulfil each site which is intended to support the VO has to pass the certification procedure. During this procedure the site is checked whether it is able to support the application within a long term. Legal issues are also resolved during the certification, especially, w.r.t. data privacy.

When being under the certification procedure, the site configures its middleware to support HEP VO and the application software together with monitoring components are installed. On successful software installation, VO staff can start performance and stability tests. Site stability and availability is tested by SAM tests during the certification and the site's regular operation in HEP VO.

Once the performance tests are done, an SLA document is prepared. This document includes a description of services provided by the site and assures data privacy. With the SLA signed, the site can be certified and its resources can be used during the experiment.

The parties of the SLA document are an experiment owner and a service provider. The signed SLA assures a proper level of services provided by the site, whose

fulfilment can be controlled by specially drawn up metrics. The metrics cover the following areas: performance, availability and connectivity, support and expertise.

Performance metrics characterize the site's performance w.r.t. computational power and network bandwidth. The *availability* and *connectivity* metrics describe the site's ability to faultlessly run the application on the long-term basis and describe the amount of computational resources which can be consumed by the application. The *support* and *expertise* metrics characterize the site's staff responsiveness to reported problems and the site's administrators effectiveness in problem solving. HEP VO Management System monitors the realization of SLA metrics.

4.2 Management System

HEP VO Management System has been created to simplify the VO usage and management. The system is designed to support three different user groups during their interaction with HEP VO. These groups are: VO Managers, Experiment Operators, and Site Managers. Each group has its own profile with a different view on the system, limited to relevant functionalities.

VO Manager has access to all management functionalities. He/she supervises the certification procedure and can change a site status according to its performance during the procedure. The site status can also be changed during a regular site operation. A site's operation within HEP VO can be suspended in case of problems. VO Manager is also responsible for the preparation of application software which can be run by an experiment operator. He/she prepares job submission profiles for all available application versions.

Experiment Operator can submit a bunch of grid jobs with a selected submission profile. He can also request more resources by submission of additional grid jobs or release resources by stopping running jobs. During the experiment run, information about submitted grid jobs is presented to the Experiment Operator. Each grid operation, like job submission or job cancellation requires a valid proxy certificate. Therefore each experiment operator needs to provide a valid proxy using his/her local account on HEP management node.

Using the portal, Site Manager registers the site in HEP VO and starts the certification procedure. During regular site operation he/she has access to job monitoring data.

On the one side, SLA is a document being a pre-requisite to be signed; on the other side, proper metrics related to SLA are monitored by HEP VO Management System, to notify the site and VO managers about all the improper behaviour cases occurred.

5 MONITORING FACILITIES

The monitoring facilities intended to provide RTD with monitoring data comprise two systems, each meant to serve for different elements of the whole system: JIMS targeting network load parameters and OCM-G – local resources usage.

5.1 Network Monitoring

As the main monitoring component for the HEP application we have chosen JIMS – the JMX-based Infrastructure Monitoring System. It is equipped with a specialized interface following the WS-Management standard (a standard protocol for managing resources through the web) [9]. Built on top of JMX (Java Management Extensions) [10, 11], JIMS is a flexible framework that allows deployment of many differentiated modules, like modules for monitoring grid infrastructure as well as modules facilitating the discovery of monitored nodes within the cluster and the grid. Moreover, JIMS communication layer is also composed of modules, e.g. JSR 262 connector – a Web Services connector for JMX technology [9] implementing the aforementioned WS-Management standard. In `int.eu.grid`, the JIMS system was enhanced with P2P capabilities based on Global Discovery Protocol [12]. GDS (Global Discovery Service) module allows JIMS agents to discover all the agents running within the grid. Having the list of discovered agents, another module – JIMS Gateway – allows connecting to any other agent from one arbitrarily chosen agent. In our case this is the agent that is selected and configured in Real-Time Dispatcher (RTD), which collects the monitoring information from one point of attachment with the JIMS monitoring system via WS-Management protocol. For the purposes of the HEP application, the JIMS system comprises two types of monitoring components. The first one is the *Network Monitoring module*, monitoring the available bandwidth using the algorithm of measuring dispersion between pairs of packets [13]. The concept of available bandwidth monitoring is used by the senders/receivers components of JIMS agents, communicating with each other in order to measure the available possible bandwidth between HEP application in CERN and the computing resources in clusters all over the Europe (IFCA, CYFRONET, etc.).

5.2 Local Resources Monitoring

The second monitoring component – OCM-G – is represented by a proxy module allowing the Dispatcher to obtain information about computing infrastructure utilization. The module exploits *OMIS-Compliant Monitoring System for the Grid* [14, 15] to realise its monitoring functionality. It acts as a bridge between OCM-G and JIMS system [16]. Since JIMS uses JMX MBeans to implement monitoring modules, a new MBean was created for this purpose. The MBean provides information like a load average value for a given worker node (i.e. the number of jobs in the run queue or waiting for disk I/O averaged over 1, 5, and 15 minutes), an amount of available host's main memory and an amount of remaining swap space. When the one of the MBean's operations is called, the the proper request is constructed and sent to the OCM-G system. Upon an OCM-G's reply arrival, the reply is processed, if needed, and returned as the operation result. For collecting monitoring information, OCM-G uses monitors running on worker nodes. The MBean itself runs in the JIMS agent launched on a chosen host of Grid site (probably Computing Element). With help of the JIMS infrastructure the data from all sites can be supplied to the Dispatcher.

The OCM-G's monitor on a worker node is started together with a remote processing task and stopped when the remote processing task is stopped, too. Thus we save resources and this is the main reason why we use the OCM-G system instead of JIMS agent which is capable of providing the same monitoring information but is not so lightweight.

6 REAL-TIME DISPATCHER

6.1 The Concept of RTD

In order to facilitate the process of delegating computations from a particular application to available grid resources, RTD [17, 18] uses the master-worker paradigm. In our case the role of *master* is played by the HEP application.

Via HEP VO Management System the experiment operator creates a pool of jobs called *worker processes* (on the sites that have passed the certification procedure; their number is subject to changes depending on meeting the SLA metrics). These processes register within RTD and wait until RTD passes them a request from the application. When the application needs additional computational power it sends a request for worker(s) to RTD. This additional computational power is chosen from those workers which are currently registered in RTD. The pool of jobs is created with a safety margin but if there is a need of more resources, the operator can launch additional jobs which can register in RTD, thus increasing the amount of available resources.

The application request for additional computational power contains the application hostname and the port on which it is listening for a TCP connection from a worker process, which will be assigned by the RTD. When the RTD receives an application's request, it uses the monitoring facilities (see Section 5) to find the most optimal worker (based on the currently available bandwidth and computational power of currently available free workers) and passes to it the hostname and port on which the application is listening. Then the worker connects to the application and is ready to serve its requests. At this point, the role of RTD is finished. Passing data, carrying out computation and returning the results is accomplished directly via the created connection between the master and the workers.

6.2 RTD Architecture

The RTD architecture assumes its modular construction (cf. Figure 1), with the following modules:

Frontend Interface – the module responsible for the gateway interface to the system.

Backend Interface – the module responsible for communication with the worker running on the computing nodes on the grid.

Resource Manager with Resource Registry – the module responsible for managing available workers.

Engine – the module that implements task dispatching.

Monitor – the module responsible for gathering data from the monitoring facilities concerning the state of computing nodes within sites in order to supply RTD with information required for optimal selection of the available worker.

In order to increase the reliability and efficiency of the RTD, its modules are constructed in a way that allows placing them on separate cluster nodes. Such a construction of the system is useful for the multiplication of module instances and balancing their load, as well as taking over the assignments of a damaged instance.

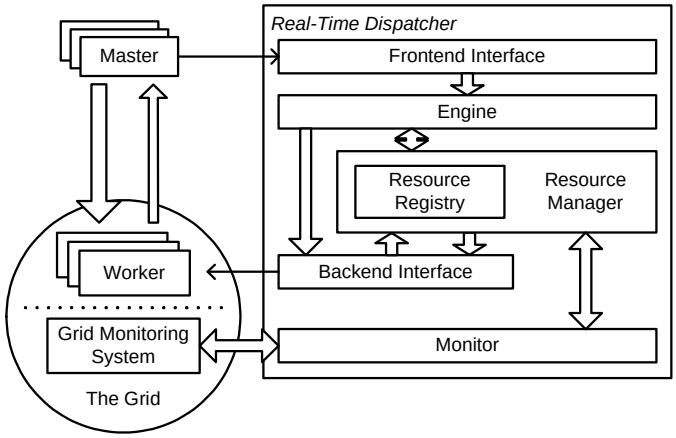


Fig. 1. High-Level Design of the Real-Time Dispatcher architecture

6.3 RTD Use Cases

Two main actors and one secondary actor interact with the systems. They are:

Master – represents the application that requests a need for computing power and delegates the computation to the grid environment.

Worker – a worker process running on the computational node in the grid environment, containing the computing algorithm and expecting input data.

Monitoring (secondary) – a grid monitoring system, provides data concerning the state of grid nodes.

Two main use cases have been identified as presented below.

Register Worker. This use case covers the registration of a worker running on a computing node. Its goal is to register the worker in the *Resource Manager*

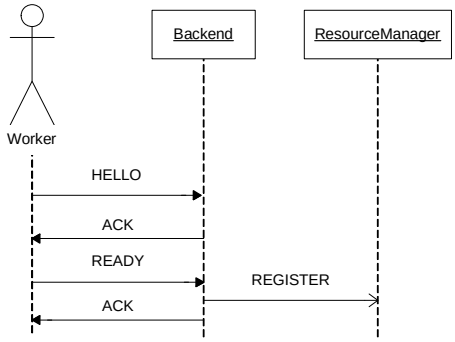


Fig. 2. Register Worker sequence diagram

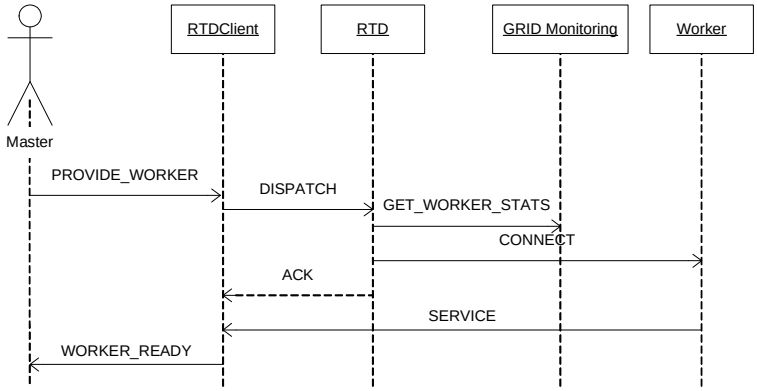


Fig. 3. Require Worker sequence diagram

module and make it available for the RTD for further assignment to the master. The initiator is the worker itself, the registration procedure is part of worker initialization. The sequence diagram with the scenario of this use case is shown in Figure 2.

Require Worker. This use case covers the request to assign a worker to a master. The goal is to select the optimal worker and pass to it some information necessary to establish a connection to the master in order to serve the client application requests. The initiator is the master. The sequence diagram with the scenario of this use case is shown in Figure 3.

7 CONCLUSIONS

The solution 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. We find the achieved solution very flexible and adaptable for other problems, not only those described in the paper.

In a natural consequence, the developed architectural approach is capable of solving all the problems connected with generation of large amounts of short data processing tasks having no relations to each other. In such a scenario, the time of going through the grid infrastructure might be much longer than the processing itself. Moreover, contemporary grid infrastructures, especially resource brokers, are not prepared for processing thousands of jobs per second. They were designed for long-run jobs. The facilities (middleware) described in this paper are able to work with any typical grid infrastructure and provide computer power being able to process thousands of small jobs without flooding the whole grid environment. This is enabled owing to the intelligent pre-allocation of computer processing resources in advance and continuous management of currently available resources, supported by advanced monitoring information coming just from the processing nodes. We find similar problems in many domains including medicine, astrophysics, flood crisis decision team support, and many others.

Apart from it, we find this solution applicable to other problems including those exploiting the so called "wasted computer power". By this term we understand free CPU cycles lost by improperly written grid jobs. Owing to the advanced monitoring sensors installed together with jobs allocated to worker nodes across the grid we can provide the architecture intelligently exploiting available free processor cycles on the grid. By simple reconfiguration of job queues at grid sites capable to provide wasted computer power to handle a particular computational problem, we can pre-allocate the free CPU cycles to some other jobs, controlled by the dispatcher. It is able to intelligently inject a very short processing request to those processors which are unoccupied at the moment because they are not processing any grid job or the executing job is waiting for some external results.

We find the problems of effective exploitation of "wasted computer power" still up to date and we see many applications especially in the medical domain being able to appropriately consume available resources which cannot be exploited in the traditional way by most of the grid infrastructure.

Acknowledgement

The research is partly supported by the EU IST 031857 `int.eu.grid` project with the related SPUB-M grant, the AGH grant 11.11.120.777, and the Cyfronet grant 500-08.

REFERENCES

[1] The EU IST int.eu.grid web page `http://www.interactive-grid.eu/`.

[2] Bubak, M..–Malawski, M.–Zajac, K.: The CrossGrid Architecture: Applications, Tools, and Grid Services. In: Rivera, F. F., et al. (Eds.), Proceedings of

Grid Computing, First European Across Grids Conference, Santiago de Compostela, Spain, February 2003, Lecture Notes in Computer Science, No. 2970, Springer, 2004, pp. 309–316, http://www.eu-crossgrid.org.

[3] Sloot, P. M. A.—Boucher, C.—Bubak, M.—Hoekstra, A.—Plaszczak, P.—Posthumus, A.—van de Vijver, D.: VIROLAB – A Virtual Laboratory for Decision Support in Viral Diseases Treatment. In: M. Bubak, M. Tura3a, K. Wiatr (Eds.), Proceedings of Cracow Grid Workshop – CGW'05, November 20–23, 2005, ACC-Cyfronet UST, 2006, Kraków, pp. 33, http://www.virolab.org.

[4] Bee, C.—Bold, T. et al.: On the Potential Use of Remote Computing Farms in the ATLAS TDAQ System. In: Real Time Conference, 2005, 14th IEEE-NPSS, June 4–10, 2005, Stockholm.

[5] Forti, A.—Garitaonandia, H.—Masik, J.—Wheeler, S.—Wengler, T.: Using the Grid to Test the ATLAS Trigger and Data Acquisition System at Large Scale. In: IEEE TNS on Nuclear Science, Vol. 54, October 2007, No. 5, pp. 1767–1772.

[6] Skital, L.—Dutka, L.—Korcyl, K.—Janusz, M.—Słota, R.—Kitowski, J.: Virtual Organization approach for running HEP applications in Grid Environment. In: Proc. Cracow Grid Workshop (CGW'06), October 15–18, 2006, Cracow (Poland), 2007.

[7] Skital, L.—Janusz, M.—Slota, R.—Kitowski, J.: Service Level Agreement Metrics for Real-Time Applications on the Grid. In: R. Wyrzykowski (ed.), Proc. PPAM 2007, Seventh International Conference on Parallel Processing and Applied Mathematics, Gdansk, Poland, September 9–12, 2007, LNCS, Springer (to appear).

[8] Skital, L.—Slota, R.—Janusz, M.—Kitowski, J.: Management of Virtual Organisation for demanding applications in the Grid Environment. In: Proc. CGW'07, October 15–17, Krakow, Poland, 2008 (to appear).

[9] Denise, J.-F.—Fuchs, D.: Java Management Extensions (JMX) Interoperation With Non Java Technologies. 2007, http://java.sun.com/javase/technologies/core/mntr-mgmt/-javamanagement/JSR262 Interop.pdf.

[10] Sun Microsystems. Java Management Extensions (JMX) Specification. version 1.4. JSR 160, Santa Clara, CA, 2006, http://jcp.org/en/jsr/detail?id=160 (jmx-14-mrel3-spec.pdf).

[11] Sun Microsystems. Java Management Extensions Instrumentation and Agent Specification, v. 1.2, JSR 003, Santa Clara, CA, 2002, http://jcp.org/en/jsr/detail?id=3 (jmx1.2-spec.pdf).

[12] Wojtas, K.—Wasilewski, L.—Balos, K.—Zielinski, K.: Discovery Service for JMX-Enabled Monitoring System. JIMS Case Study. In: Proc. CGW'05 Workshop, ACC Cyfronet AGH, Kraków, 2006, pp. 148–157.

[13] Dovrolis, C.—Ramanathan, P.—Moore, D.: What Do Packet Dispersion Techniques Measure? INFOCOM, 2001, pp. 905–914.

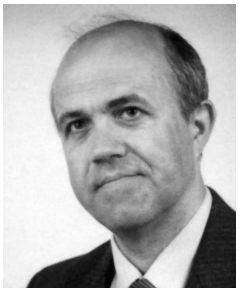
[14] Balis, B.—Bubak, B.—Funika, M.—Wisműller, R.—Radecki, M.—Szepieniec, T.—Arodz, T.—Kurdziel, M.: Grid Environment for On-Line Application Monitoring and Performance Analysis. In: Scientific Pogramming, Vol. 12, 2004, No. 4, pp. 239–251, http://grid.cyfronet.pl/ocmg.

[15] Ludwig, T.—Wisműller, R.—Sunderam, V.—Bode, A.: OMIS – On-Line Monitoring Interface Specification (Version 2.0). Shaker Verlag, Aachen, Vol. 9, LRR-TUM Research Report Series, 1997, `http://wwwbode.in.tum.de/~omis/OMIS/Version-2.0/version-2.0.ps.gz`.

[16] Funika, W.—Guzy, K.: Integration of OCM-G into the JIMS Infrastructure for the Monitoring of HEP Application. In: M. Bubak, M. Turala, K. Wiatr (Eds.), Proc. Cracow'07 Grid Workshop, October 15–18, 2007, Cracow, Poland, ACC Cyfronet AGH, 2008 (to appear).

[17] Pieczykolan, J.—Dutka, L.—Kryza, B.—Korcyl, K.—Kitowski, J.: Data Dispatcher for Real Time Applications in Grid Environment. In: Proc. 7th Int. Conf. Computational Science – ICCS 2007, Beijing, China, May 2007, adtn'l CD, pp. 47–54.

[18] Pieczykolan, J.—Dutka, L.—Korcyl, K.—Kitowski, J.: Data Dispatcher for Interactive and Data-Streaming Applications Using Grid Environment. In: R. Tadeusiewicz, A. Ligeza, M. Szymkat (Eds.), Proc. of CMS'07 Conference (Computer Methods and Systems), November 21–23, 2007, Krakow, Oprogramowanie Naukowo-Techniczne, 2007, pp. 339–344.



Wlodzimierz Funika works at the Institute of Computer Science of the AGH University of Science and Technology in Krakow (Poland). His main research interests are in distributed programming, tools construction, performance analysis and visualization. Previously involved in the EU IST CrossGrid, CoreGRID, K-WfGrid, int.eu.grid projects, currently in the EU IST ViroLab, GREDIA projects.

Jacek Kitowski is Full Professor of computer science, Head of the Computer Systems Group at the Institute of Computer Science of the AGH-UST in Cracow (Poland), and the Academic Computer Centre CYFRONET-AGH. His topics of interest include large-scale computations, multiprocessor architectures, Grid services and Grid storage systems, knowledge engineering. Previously involved in the EU IST CrossGrid, Pellucid, KWfGrid, and int.eu.grid projects, currently in the EU IST GREDIA project.

Krzysztof Korcyl since graduation from the AGH-UST in Krakow employed at the PAN Institute of Nuclear Physics, has been developing instrumentation for high energy physics experiments. Contributed to the TDAQ system of Dephi at LEP. Currently he is involved in the design and modelling of the TDAQ system for ATLAS experiment on LHC.

Kazimierz Bałos received his Ph. D. in computer science from the AGH-UST in Krakow, Poland, in 2007. His Ph. D. thesis concerns self-confiurable and adaptable services for the monitoring of infrastructure and applications in a grid environment. Involved in national (KBN) and EU projects (CrossGrid, Int.Eu. Grid, Ambient Networks). Interests: monitoring and management of distributed systems, SOA-based systems, Web 2.0/3.0.

Jan Pieczykolan graduated from the AGH-UST in Cracow, Poland. He has been working with Grid Technologies for the last 4 years in the scope of such EU-IST projects as K-Wf Grid and int.eu.grid, especially in the scope of Grid Middleware. Worked at ACC Cyfronet AGH in Cracow as scientific researcher within the EU IST int.eu.grid project.

Renata Słota works at the Institute of Computer Science of the AGH Uni-versity of Science and Technology in Krakow, Poland. Her topics of interest include distributed systems, Grid environments, data management and storage systems, knowledge engineering. Previously involved in the EU IST CrossGrid, Pellucid, K-WfGrid, and int.eu.grid projects. Currently participates in the EU IST GREDIA project.

Krzysztof Zieliński is Full Professor and Head of the Institute of Computer Science at AGH-UST. His interests concern distributed computing, object-oriented and component distributed systems engineering, mobile technology, and networked multimedia. He is the author of over 130 papers in this area. In the years 1988-90 he worked in the Olivetti Research Lab., Cambridge (UK), and at Cambridge Computer Laboratory.