



PERFORMANCE ANALYSIS OF APPLICATIONS

Document Filename: **BG-DNA3.8-PerformanceAnalysisApplications.doc**
Activity: **NA3**
Partner(s): **VU, IFJ PAN, EENet, NICPB, IMCS UL, PSNC, ITPA**
Lead Partner: **VU**
Document classification: **PUBLIC**

Abstract: The BalticGrid's deliverable DNA3.8 presents the application performance tools, installed and integrated in to Grid infrastructure The usage of application performance tools in the BalticGrid project is caused by the difficulty in adapting parallel application to the grid environment experienced by project's partners. Also the analysis of efficiency of some applications (like HEP application LHCb, especially designed for test purposes) running in the BalticGrid environment is presented.





Document review and moderation

	Name	Partner	Date	Signature
Released for moderation to				
Approved for delivery by				



Document Log

Version	Date	Summary of changes	Author
0.1	13/02/2007	Draft version	Linas Bukauskas
0.2	15/04/2007	Text of the deliverable, representing the content and analysis, done for applications	Tomas Szepieniec, Mariusz Witek
0.3	25/04/2007	Integrating parts	Linas Bukauskas, Tomas Szepieniec
0.4	30/04/2007	Final version	Algimantas Juozapavicius

CONTENTS

1. INTRODUCTION.....	5
1.1. PURPOSE OF THE DOCUMENT	5
1.2. ABBREVIATIONS.....	5
2. EXECUTIVE SUMMARY.....	6
3. DESCRIPTION OF PERFORMANCE ANALYSIS	7
3.1. APPLICATION PERFORMANCE MONITORING SYSTEM	7
<i>Aim of the OCM-G.....</i>	7
<i>Feature Lists of OCM-G.....</i>	7
<i>Specific Requirements for Applications.....</i>	8
<i>Scope of work and progress</i>	8
3.1.1.1 Performance monitoring features	8
3.1.1.2 Monitoring support features	9
3.1.1.3 Timer triggered events.....	9
3.1.1.4 Command execution.....	10
3.1.1.5 Extended portability	10
3.1.1.6 OCM-G and Java.....	10
3.1.1.7 OCMG Java API layers	10
3.1.1.8 Parallel Pipe.....	11
3.1.1.8.1 Parallel Pipe features	11
3.1.1.9 New OCM-G execution methods	11
3.1.1.10 Deregistration of a Local Monitor	11
<i>Status of accessibility</i>	11
3.1.1.11 Web page.....	12
3.1.1.12 Project page.....	12
3.1.1.13 BalticGrid webpage changes	12
3.1.1.14 Presentations and Tutorial events	12
3.1.1.15 Installing using SW_DIR.....	13
3.1.1.16 Tutorials	13
3.1.1.16.1 Self installing scripts.....	13
3.1.1.17 Others	14
3.2. PERFORMANCE ANALYSIS TOOL	14
<i>Aim of G-PM Tool.....</i>	14
<i>Specific requirements for BalticGrid.....</i>	14
<i>List of features of G-PM.....</i>	14
<i>Scope of Work and Status.....</i>	15
3.3. INTEGRATION SCENARIOS	15
<i>MPI Applications.....</i>	15
3.3.1.1 Step 1: Application instrumentation	16
3.3.1.2 Step 2: Application submission	16
3.3.1.3 Step 3: Performance monitoring with G-PM.....	16
<i>GAMESS.....</i>	16
3.3.1.4 Problem	17
3.3.1.5 The goal.....	17
3.3.1.6 Solution	17
<i>Text Analysis in Distributed Prolog</i>	17
3.3.1.7 Problem	17
3.3.1.8 The goal.....	17
3.3.1.9 Solution	17
3.4. LHCb ANALYSIS.....	18

1. INTRODUCTION

The Baltic Grid project aims i) to develop and integrate the research and education computing and communication infrastructure of the Baltic States into emerging European Grid infrastructure, ii) to bring the knowledge in Grid technologies and use of Grids in the Baltic States to a level comparable to that in EU member states with a longer experience in the development, deployment and operation of Grids, and iii) to further engage the Baltic States in policy and standards setting activities. The integration of The Baltic States into the European Grid infrastructure will primarily focus on extending the EGEE to the Baltic States (with which four partners are already engaged).

The main motivation of the user adapting their applications to grid environment is possibility to have more efficient computing procedure and to have more computing resources. Thus, the performance of the application adapted to the new environment is crucial for user's satisfaction. Unfortunately, due to heterogeneity of resources, high efficiency of application is not guaranteed by the environment. Even in case of contemporary solutions when the overall performance metrics are available to machine, the execution time of the application could not be estimated or forecast. Execution time and the efficiency of such execution depend on many factors.

Integration of performance analysis tools was based on the analysis of identification of applications as well as requirement for extending its functionality to meet their demands. In this deliverable the installation of tools with basic functionality in BalticGrid infrastructure and instruction of using it with applications are presented. The program of integration also describes on how to use such tools. Also a suitable application (coming from HEP) performance analysis is given.

1.1. PURPOSE OF THE DOCUMENT

The purpose of this document is to present the results of conceptual and technical analysis done to select, design and implement tools for application performance measurement, as well as to present some technical texts, like description of selected tools usage and technical requirements.

1.2. ABBREVIATIONS

BG – BalticGrid

BI – Bioinformatics

HEP – High Energy Physics

G-PM – Grid Performance Measurement tool

MS – Material Sciences

OCM-G – OMIS-Compliant Monitor for the Grid

2. EXECUTIVE SUMMARY

The usage of application performance tools in the BalticGrid project is caused by the difficulty in adapting parallel application to the grid environment experienced by project's partners. The root of the problem is that grid is batch oriented and provides no access to the system where application is running. The other hand, application would perform differently on grid/cluster sites, so there is a need of evaluation of application performance to find the bottlenecks.

Results of identification of application tools lead to the project and implementation of monitoring system - OCM-G and of a tool for performance monitoring - G-PM. These products were selected as meeting many of requirements coming from applications.

This document presents the sort characteristics of OCM-G, requirements coming from application, as well as efforts and improvements done in OCM-G to meet these requirement. The description is given also to several typical integrations of these tools with applications presenting the real scenarios that are under development. Namely, application performance monitoring system OCM-G is analysed and presented by the feature list, specific requirements for applications, scope of work, status of accessibility. The performance analysis tool G-PM was analysed for compliance with specific requirements for BalticGrid, the list of features of G-PM is presented, as well as the scope of work. Such analysis done enabled to produce integration scenarios, especially for the case of MPI, GAMESS and Text Analysis in distributed Prolog applications. Finally the performance analysis of HEP application of LHCb is presented.

3. DESCRIPTION OF PERFORMANCE ANALYSIS

The usage of application performance tools in the BalticGrid project is caused by the difficulty in adapting parallel application to the grid environment experienced by project's partners. The root of the problem is that grid is batch oriented and provides no access to the system where application is running. The other hand, application would perform differently on grid/cluster sites, so there is a need of evaluation of application performance to find the bottlenecks.

3.1. APPLICATION PERFORMANCE MONITORING SYSTEM

The result of identification of application tools completed until PM6 (to meet the milestone MNA3.6), was the establishment of monitoring system - OCM-G and of a tool for performance monitoring - G-PM. These products meet many of the requirement coming from applications. However some extensions and improvements for these tools need to be done.

This section presents the sort characteristics of OCM-G, requirements coming from application, as well as efforts and improvements done in OCM-G to meet these requirement. We describe also several typical integrations of these tools with applications presenting the real scenarios that are under development.

Aim of the OCM-G

The main target for OCM-G is to provide infrastructure for application performance monitoring in grid environment. While applications on the grid would be distributed, OCM-G should be also distributed and provide monitoring data from different application's parts. Moreover, the long-running of applications on the grid and lack of mechanisms to direct control during the run, demand on-line mode operations.

Feature Lists of OCM-G

1. Provide on-line monitoring data - all monitored data could be obtained immediately by the user;
2. Support for interactive monitoring - all monitoring features in OCM-G could be enable and disable in the runtime;
3. Access to the working application - extension provides access to application environment during application runtime.
4. Reflects dynamic of application - in case of applications in which new processes are spawn, OCM-G will add them to monitoring system;
5. Wide range of available services including: information services, manipulation services and events services;
6. Possibility to determine actions of monitoring system in case of occurrence of a specific event (so called: conditional requests);

7. Full support of performance metrics in MPI - instrumentation method tested with several MPI implementation provide performance data about communication in MPI;
8. Monitoring interfaces are build upon OMIS standard;
9. Good scalability of distributed monitoring - tree layers of OCM-G components makes it very scalable, as well as easily manageable;
10. Integration with gLite middleware - information about gLite job IDs are recognized by OCM-G and service for translated it to the process tokens are provided;
11. Running commands through the monitoring system - possibility to spawn a command on the node then application is running;
12. Actions triggered by timers - possibility to trigger actions based on steerable timer;
13. Full support for IA32, most of features are available on IA64 as well;
14. Support for either communication using Globus and pure sockets

Specific Requirements for Applications

Long considerations and analysis made by Balticgrid Application Support team led to many conclusions - the most important fact was to provide a monitoring tool for the most well known parallel programming library MPI. The message-passing nature of library makes it a great solution for grid-based calculations, but at the same time really hard to monitor. The other aspects of the monitoring tool needed in Balticgrid were:

- It should make a *good view on the grid job* - by allowing multi-dimensional, multi-metric access to the application components
- It should allow making a data (input and output) stream, what makes application more error resistant, and helps user deploy more the data when it is needed
- It should be able to scale - while monitoring multiple applications through multiple sites

Having in mind these considerations we found that most of the requirements are fulfilled by OCM-G, and the rest is really easy to implement.

Scope of work and progress

All the OCM-G features described in this section was identified as important to meet the defined application's requirements or to complete the set of functionality in OCM-G. All the extensions were developed and included in official release of OCM-G.

3.1.1.1 Performance monitoring features

- **node load** - useful metric of overall load of machine caused by all processes on it. Let's the user check the bottlenecks caused by unexpected CPU overwork. Implemented as a remote call to the Local Monitor.

- **free space** on disks - metric showing how much space is available for process, when it is necessary to write to the local files. Implementation based on remote call of the df command executed by Local Monitor.
- **CPU utilization** - metric letting the user check time spent by CPU running particular application; this knowledge is especially useful when it comes to performance of CPU usage; made as a remote call to Local Monitor which checks particular process' properties.
- an amount of **used memory** - metric useful for checking the memory usage performance - shows how much of memory we have used in particular memory segments, such as stacks, or heaps. Made as a remote call to the Local Monitor.

3.1.1.2 Monitoring support features

- getting opened **files list** - metrics used when we want to check how much files our application opens, and what kind of files these are. This feature is especially useful for monitoring precompiled software. It was implemented as a remote call to Local Monitor which seeks this information in a /proc filesystem.
- spawning **simple commands** - this feature is useful to overcome problems with unpredicted informations that can appear on a node - especially when there is special software on CE, and we have no OCM-G call for it. Implementation is based on a popen method which makes it simple and powerful solution, but we can't write to input [which was solved by command tokens mentioned below].
- getting gLite **JobID** of the process - special call to remote Local Monitor which provides us the gLite JobID of the Local Monitor within it's working. Lack of that feature makes it impossible to write any OCM-G based framework, as the identification of the desired job is crucial.
- file **upload** - this call is to improve the overall performance of calculations by supplying the input data while the job is running. Such behavior diminish the overhead for the task submitting, and therefore makes the computations more efficient. It was implemented as a call transporting some part of data - and deploying file results in a set of subsequent calls.
- file tail (as well as **download**) - this feature makes it visible what happens with all the accessible files in the files system. Particularly user is able to see how much of work has been done by looking into the output files of the application. Moreover it's is possible to download the file to, for example, keep the live mirror on the local machine. The file tail was implemented as a remote call to the Local Monitor.

3.1.1.3 Timer triggered events

This feature gives user the possibility to trigger time based events such as sending particular system's average load with given frequency. With this possibility user can easily collect the data asynchronously from the monitored objects, what helps him a lot in **performance monitoring**, visualizations and preparing reports.

3.1.1.4 Command execution

It is possible now to execute a command as a child of the running Local Monitor. With such command you can:

1. **write** to its input
2. **read** its standard output and error output
3. **check** its existence
4. **stop** it

The main purpose of it was to give the user more availability to control the application which is being run on the CE. By forking from Local Monitor user have full control on input/output of the desired application as well as on the lifetime of the process. Having this possibility user can easily stop the task which has, for example, made something wrong. The command token was implemented with fork and pipe UNIX calls, what makes it highly portable.

3.1.1.5 Extended portability

As the Grid middleware develops user was provided with the ability to use other versions than the standard GLOBUS 2.4.

- full **GLOBUS 2 I/O** support
- full **GLOBUS 3 I/O** support
- full **GLOBUS 4 I/O** support
- partial support of 64 bit architectures (**IA64**)

As an additional work we made it possible to run OCM-G without GLOBUS, in a **simple cluster** environment.

3.1.1.6 OCM-G and Java

Nowadays it seems that Java is the most commonly used programming language in the world. There are more and more Java web services, portals and others. Unfortunately OCM-G did not provide any libraries for use in this language. This is the place where OCM-G Java API shows up.

OCM_G Java API was designed to satisfy the needs of many developers. Therefore it consists of multi-layered architecture.

1. simple applications (like cg-ocmg-tool) may use low layer and operate directly on the requests/replies
2. complicated applications may use high layer which provides many useful facilities

3.1.1.7 OCMG Java API layers

- layer 1: simple GSI/MCI connection to the MainSM and request/reply mechanism

- layer 2: stateless wrappers which provides common operations on various tokens (ie. attaching to a site)
- layer 3: stateful objects which directly maps OCM-G tokens to Java objects
 - thread-safe
- caches and buffers for static informations (ie. site information)
- full support of various conditional requests
- conditional requests' parameters verification
- support of expansions/localisation mechanism
- automatic monitoring of OCM-G objects

3.1.1.8 Parallel Pipe

Many grid users executes typical unix-like commands joined with pipes (ie. `cat /etc/passwd | grep a`). Unfortunately they could not exploit grid paralelism with such command in a simple way. As a result a new simple tool - Parallel Pipe - was proposed. It allows commands to be run in a grid/cluster environment with execution distributed among available nodes.

3.1.1.8.1 Parallel Pipe features

1. allows piped commands to be executed in parallel thus reducing total time needed for executing the commands
2. supports both grid and cluster environment
3. does not require any additional user skills to operate

3.1.1.9 New OCM-G execution methods

In previous versions there was only one way to execute OCM-G. The method is based on instrumentation of MPI library and then linking it with our applications. It provides us many interesting features (such as ab ility to breakpoint the application or to profile it's MPI calls), but is uncomfortable, and when it comes to external applications impossible.

To overcome these problems, two completely new methods were introduced:

1. executing OCM-G Local Monitor **in parallel** with application - It means that each application is run by special application helper which keeps Local Monitor running as long as monitored application is alive
2. executing OCM-G Local Monitor **instead of a particular job** - Local Monitor running as a job can be easily run by performing an OCM-G call

3.1.1.10 Deregistration of a Local Monitor

To fulfill the needs of Local Monitor run as a grid job there was provided an ability for it to deregister itself from OCM-G monitoring system.

Status of accessibility

3.1.1.11 Web page

For better understanding of the project and more widespread usage, new OCM-G website was provided. It includes:

- Download - containing **GSI**, as well as **MCI** versions of OCM-G compiled on x86 and IA64 architectures
- Publications - page which enlists publications connected directly or indirectly to the OCM-G system
- Tutorials - page with **tutorials and screencasts** about OCM-G usage
- Specifications - specifications used in OCM-G implementation (useful for future development)
- Links - many **useful links** which could be helpful for the user

Web page is here <http://www.icsr.agh.edu.pl/ocmg/>

3.1.1.12 Project page

Project page was established to make project and issue management easier. It also provides:

1. internal project informations
2. latest sources
3. subversion repository with latest sources package download feature
4. documentation management
5. improves developers communication

Project page is here <http://gforge.cyfronet.pl/projects/ocm-g/>

3.1.1.13 BalticGrid webpage changes

BalticGrid homepage was modified to show all the key benefits of OCM-G, and encourage its use in performance monitoring in BalticGrid.

OCM-G on BalticGrid page is here http://www.balticgrid.org/Applications_on_BalticGrid/Documents/ocmg.html

3.1.1.14 Presentations and Tutorial events

10. [Tutorial](#) given on Summer School at the Tartu University, Estonia, on 8th July 2006 - more info in tutorials subsection
11. [Poster](#) shown on CGW 06 Cracov Grid Workshop - M. Bubak, B. Balis, W. Funika, M. Radecki, T. Szepieniec, R. Wismueller, P. Marchewka, J. Janczak, T. Duszka and R. Chojnacki. Enhancing the functionality of OCM-G and G-PM for further grid applications. Poster on Cracov Grid Workshop, Poland, November 2006
12. [Presentation \(in Polish\)](#) shown on Internal Cracov Grid Meeting on 19 March 2007 at The University of Science and Technology AGH by Tomasz Szepieniec

3.1.1.15 *Installing using SW_DIR*

There are two ways of installing OCM-G in SW_DIR

From tarballs:

On the website there are some binary tarballs provided, to let the user avoid the compilation step. All the user should do is:

- download tarball
- execute:

```
cd $SW_DIR  
tar -xzf <package_name>.tar.gz
```

From sources:

To install application from sources, user should download the source package, and then:

1. unpack it by executing:

```
tar -xzf <package_name>.tar.gz
```

- run ./configure
- run

```
make
```

- change the DESTDIR in Makefile to \$SW_DIR
- run

```
make install
```

Additional configuration (needed for both cases):

Users wanting to utilize OCM-G should set the CG_LOCATION, by executing:

```
export CG_LOCATION=$SW_DIR/<where_ocmg_is>
```

and then configure it by running supplied script

```
.$CG_LOCATION/etc/profile.d/cg-ocmg.sh
```

The best solution is to add these two lines to the shell rc file (~/.bashrc for example)

3.1.1.16 *Tutorials*

The tutorial given in Tartu resulted in a document useful for teaching the concepts of OCM-G monitoring system. The tutorial handout given there is accessible on site: http://www.balticgrid.org/Applications_on_BalticGrid/Documents/OCM-G_tutorial_handout-v1.1.doc

3.1.1.16.1 Self installing scripts

Fully automatic OCM-G installation script was provided to ease the process of deploying. It covers all the needed work to fulfill the work, including:

- downloading the latest OCM-G tutorial package
- setting the environment
- providing its own GLOBUS distribution if needed

On the other hand, another script was provided to give the user ability to use OCM-G on CE's where it's not installed, making run of OCM-G enabled MPI application much more easier.

3.1.1.17 Others

OCM-G is successfully **used at The University of Science and Technology AGH**, as an example of performance monitoring tool for grid applications.

3.2. PERFORMANCE ANALYSIS TOOL

With Grid computing, a new level of complexity has been introduced to the development process of distributed applications. Main emphasis here is to compute as effective as possible. This leads to the requirements for the applications to utilize the available resources in the optimal way. Many typical conditions which disturb efficiency (like imbalance in computation distribution, flaws in the schema of synchronization, suboptimal allocation of resources) can be fully observed only during an application run in an environment that closely resembles the one in which the application is to be deployed. The best approach would be to observe the application directly on the Grid, which results in new set of tools being developed. These tools should work in on-line mode, and do not introduce a noticeable overhead as this might disturb the application's execution. A performance monitoring tool for Grid which fulfill these requirements is the G-PM tool.

Aim of G-PM Tool

G-PM is a on-line tool for visualization the performance measurements on grid application. Therefore, it is very useful tool to study performance bottle-neck in the application. G-PM uses OMIS standard to obtain the monitored data, that makes is compliant with OCM-G.

Specific requirements for BalticGrid

The main requirement for BalticGrid users was to provide simple application which would derive benefits from OCM-G monitoring system, and adding new including:

- simple usage - GUI
- visualisating the data
- collecting, comparing and aggregating data from multiple monitoring subsystems
- saving the data for later processing

List of features of G-PM

1. Provides a wide range of measured quantities related to the performance of the analysed application;
2. Provides some quantities related to the sites and hosts (Grid infrastructure) as well as the network connecting them (although the tool's main focus is on Grid applications);
3. Application performance can be examined at multiple levels of detail, allowing to measure performance characteristics for:
 4. the whole program
 5. specific functions inside the code
 6. other, user specified code regions
7. Performance analysis can be narrowed down to parts of the executed application (e.g. a single process, or a set of running processes);
8. G-PM uses well-known OCM-G monitoring service for gathering of the raw performance data;
9. After data gathering, the metrics can be displayed using a set of visualization graphs (e.g. bar graphs, multi-curve plots);
10. Support of user-defined metrics (which are evaluated on-line in a distributed, scalable way) specified in the PMSL language.

Scope of Work and Status

G-PM was developed in CrossGrid Project and released with them. No major changes were required to adapt the tool to BalticGrid environment. Direct use with BalticGrid applications have been possible since OCM-G was enabled in BalticGrid environment. However, while G-PM focus primarily on visualization of performance in MPI application its usability were limited to sites where submission of MPI jobs is possible. This will improve in near future, while work to enable MPI on most of sites is in progress in SA1.

G-PM was presented in the tutorial given in Tartu. The tutorial handouts that includes user manual are accessible on site: http://www.balticgrid.org/Applications_on_BalticGrid/Documents/OCM-G_tutorial_handout-v1.1.doc

For the purpose of integration with Migrating Desktop and other java tools that are using in the project the Candle framework was planned to develop. Candle is a tool having the same functionality as G-PM but implemented in Java using OCM-G JavaAPI.

3.3. INTEGRATION SCENARIOS

In this section we described three main use cases of OCM-G with BalticGrid applications. As a result of close cooperation with BalticGrid users we managed to describe their needs and problems, and help them with our software.

MPI Applications

The main integration scenario is related to the application performance study in MPI application running on the grid. The application need to be prepared to enable possibility of monitoring, spawned with OCM-G environment and finally monitored. The monitoring process could be managed by G-PM user, who can observe different performance measurements in on-line mode. Below we describe shortly how this process is seen from user perspective.

3.3.1.1 Step 1: Application instrumentation

To enable performance monitoring to the application we need to link them against instrumented version of communication library. OCM-G provides special support for instrumenting MPI library installed on the machine, namely when PMPI libraries are available it uses them if not automatic instrumentation of binary version of MPI library is done.

Typically no changes in source code of the application are required, until user wants to add some special function to the source code - probes - that would act as 'markers' in the monitoring process. The only step that user need to do is relinking the application with instrumented version of MPI libraries.

3.3.1.2 Step 2: Application submission

Before the application is submitted, the user needs to run the main components of OCM-G and obtain its address, that need to be pass to the application as additional argument. Due to various methods of job submission used in BalticGrid this action was left to users as manual operation.

Application submission is done in the same way as without OCM-G. The only changes that are needed are two additional parameters that user needs to add to the application. Arguments are added to JDL and provides information about localization of OCM-G main component.

When application successfully reach the working node, application register in the OCM-G automatically and hierarchy of distributed OCM-G components are extended to support the newly added process.

3.3.1.3 Step 3: Performance monitoring with G-PM

When G-PM tool is started the registered processes are available for performance monitoring. Users could define measurements and observe sophisticated the visualization of them on-line. Also redefinition, adding new measurements as well as new types of visualizations are possible in runtime.

GAMESS

GAMESS is a well known, and widespread chemical calculation application. It's main purpose is to develop molecules by optimizing it's structure - it's a really tough and heavy job for CE's.

3.3.1.4 Problem

As a result - the usual calculations take a long time which makes them really error and fault prone. Every network failure or cluster halt causes the calculation to restart from the beginning which is uncomfortable for user and decreases overall calculation performance.

Apparently - GAMESS comes with restart procedure, but it's crucial to have a real-time file backup to perform it.

3.3.1.5 The goal

Our goal was to provide a framework which simply will mirror all the needed files to the safe place while the calculations are being performed. The secondary objective was to make it as simple as possible.

3.3.1.6 Solution

We have used a OCM-G monitoring system in this configuration:

▪

User who is planning to submit a GAMESS job, just executes a simple tool which do all the work instead of him/her and the calculation begin, it "keeps an eye" on mirrored files.

Text Analysis in Distributed Prolog

DNLP is a Latvian language processing application which utilizes Prolog extended with MPI especially for this purpose. Each task computed by DNLP consists of small amount of data which is quickly computed, but when we take into an account a sample input data-set it becomes a really serious and long computation.

3.3.1.7 Problem

As a really fine-grained and light-weight computation DNLP, user is able to simply submit job with a bunch of data, and collect the output quickly after it. But when it comes to a real huge input sets, result delivery delay problem becomes real. In a consequence such a big computation becomes more prone to the external factors.

3.3.1.8 The goal

The goal was to distribute a fine grained computation between a bunch of job-matching CE's in order to save time, and make it much more error resistant.

3.3.1.9 Solution

Solution comes as a tool which:

- spawns OCM-G Local Monitors through all the matching nodes
- each monitor waits for tasks submitted by the tool
- for each line of input data, Local Monitor executes DNLP and after a computation, returns it's results to the tool

- arrived data are being saved on the local file system (one file per input line)

3.4. LHCb ANALYSIS

The applications developed by the LHCb experiment were selected as pilot applications to test the EGEE compatible GRID middleware implemented in BalticGrid sites. There are three basic types:

1. Various applications of individual LHCb users.
2. Regular production of simulated data managed by the central production team of LHCb.
3. The distributed analysis of the data (only simulated data are available since experiment will start end of 2007).

The applications mentioned in the first two points were used for middleware validation. The application Nr. 3 needs full functionality of LHCb Tier-2 center and therefore was not employed.

The LHCb virtual organization was enabled on twelve BalticGrid sites. On 10 sites the applications of type 1 were successfully running.

The application Nr. 2 has been in development for many years and it is the most matured LHCb application employing many functionalities of the GRID middleware. The main purpose of the application is to simulate the real events of proton-proton collision and the detection of the collision products. The details of the application were described in the deliverable DNA3.3.

Before running the production jobs, the validation of the configuration is performed. The regular SAM tests of LHCb were used to check whether the sites are properly configured. They concern proper settings of the environment variables and the parameters of queues, checking the existence and access to dedicated directories, testing of network connections etc.

The application requires relatively modern hardware: CPU faster than 1.5 GHz and at least 1 GB of RAM memory. The main platform for software development is i386 compatible processors and Linux operating system. Currently supported Linux flavour is Scientific Linux CERN version 3 or 4 (SLC3 or SLC4). In case of SLC4, both 32 bit and 64 bit kernels are supported. Unfortunately, the majority of BalticGrid sites did not match these requirements. In particular a few sites contain Itanium64 based clusters. The LHCb software is not supported on this platform. Some sites do not have enough RAM per processor/core. In such cases the extensive use of system page file during the execution blocks the node completely. In effect the elapsed time of the execution becomes too long and the job gets aborted.

So far the conclusion is that on 10 BalticGrid sites which enabled LHCb VO, the simple jobs are running successfully. However, the application selected for extensive tests of the middleware - the production of simulated LHCb data - turned out to be so demanding that out of 12 BalticGrid sites willing to participate in the tests, only 3 fulfill the minimum hardware

requirements. The first results show that these 3 sites are able to produce and transfer data to the central repository. The further tests are being carried out to estimate the rate of successful jobs and the quality of produced data.