Presentation and Analysis of Grid Performance Data*

Norbert Podhorszki and Peter Kacsuk

MTA SZTAKI, Budapest, H-1528 P.O.Box 63, Hungary
{pnorbert,kacsuk}@sztaki.hu

Abstract. In this paper, data presentation and analysis tools, which use R-GMA, the information and monitoring system of the EU DataGrid project, are presented. The information browser is used to quickly browse among large amount of data. Pulse, a flexibly configurable tool is used to analyse and visualise performance data of grid resources and services. PROVE, a performance visualisation tool is used to analyse the performance of parallel grid applications.

1 Introduction

A fundamental requirement in the case of any computer system is that its behaviour must be observed and possibly controlled. Therefore, information must be acquired about its operation but it is a crucial issue what sort of data should be gathered and what is the way they can be obtained. For instance, to check its performance, archived data from the efficiently working system and new data about its current behaviour are needed as well as the right methods to compare them. The bigger and the more complex a system is the more data should be collected. A grid is a large system of computational resources and thus, huge amount of information is generated all the time.

Yet, there is a debate on what type of information and what data should be collected for grids at all. What status information should be recorded, what frequency of data collection is needed, what performance metrics can be defined and what performance data should be collected are questions yet to be discussed. However, for observing that high amount of information arriving from the many many sensors, the right tools should be used for the right purpose.

In this paper, three different tools are presented. All of them are developed in the EU DataGrid project [1] and as such, they get data from the monitoring and information system of this project. First, the information and monitoring system used in this project and its information browser, the R-GMA Browser are shortly introduced. Pulse, a new flexible tool for analysing and visualising information about grid resources and services is presented in the third section. It provides a framework to represent data in a general form and it consists of analysis and visualisation components. Finally, PROVE, a trace visualisation tool is presented that is used to analyse the behaviour and performance of parallel applications executed in the grid.

* The work described in this paper has been supported by the following grants: EU DataGrid IST-2000-25182, Hungarian DemoGrid OMFB-01549/2001 and OTKA T042459.

H. Kosch, L. Böszörményi, H. Hellwagner (Eds.): Euro-Par 2003, LNCS 2790, pp. 119–126, 2003.

2 R-GMA, a Relational Grid Monitoring Architecture

R-GMA, [2]) is being developed as a Grid Information and Monitoring System for both the grid itself and for use by applications. It is based on the GMA concept [3] from Global Grid Forum and on the relational data model. It offers a global view of the information as if each Virtual Organisation had one large relational database.

It provides a number of different Producer types with different characteristics; for example some of them support streaming of information. It also provides combined Consumer/Producers, which are able to combine information and republish it. At the heart of the system is the mediator, which for any query is able to find and connect to the best Producers to do the job.

R-GMA provides a way of using the relational data model in a Grid environment [4] but it is not a general distributed RDBMS. All the producers of information are quite independent. It is relational in the sense that Producers announce what they have to publish via an SQL CREATE TABLE statement and publish with an SQL INSERT and that Consumers use an SQL SELECT to collect the information they need. R-GMA is built using servlet technology and is being migrated rapidly to web services and specifically to fit into an OGSA (Open Grid Services Architecture, [5]) framework.

2.1 Information Browsing

The information in R-GMA can be browsed by web pages and servlets provided with the R-GMA package. The `R-GMA Browser` provides the list of available tables within R-GMA and detailed information about each table. A query can be composed for a selected table and `R-GMA Browser` returns a table of the result. It is a Java servlet that generates web pages about the rarely changing content of R-GMA, i.e., list of tables and definitions of tables. For dynamic information, the servlet provides functions. If the user is interested in the current information available within R-GMA for a selected table, a servlet function creates a Consumer that reads data from R-GMA and the servlet function generates a dynamic web page containing the table as a result.

The user can select a table, can compose a query for the table in the SQL language and also can choose among the available producers that gives information according to the selected table. If a general query is to be issued, the Mediator of R-GMA can be used to find and connect all producers and deliver all available information. The `R-GMA Browser` can be used to discover available information in R-GMA, sites in the grid, services and resources at the sites and status information on selected services or resources. A working `R-GMA Browser` can be accessed from the web page [6].

3 Pulse: Online Presentation of Multiple Streams of Data

The static nature of web technology, even using servlets, disables the user to stream data from R-GMA to the browser. The other shortcoming is that the user is not able to perform analysis or create new ways of presentations. For establishing streaming connection and performing specific analysis, another tool is available. Pulse is an efficient tool to compose an analysing chain for performance data about services or resources.

Basically, it provides a framework to represent data in a general form and it provides data analysis/presentation components to work on that data.

Pulse is able to subscribe to different data sources (not only to R-GMA producers, but to anything for which an appropriate input component is implemented), drive the incoming data through analysis/filtering components and present the result in a tabular or in a graphical representation. The computations and screen updates can be driven in two different ways. In the case of a source that is streaming data, the incoming data rate determines the updates in Pulse. In the case of a source that has to be polled for information, Pulse can poll it by a given interval.

In Fig. 1 a configuration for displaying load average and memory usage information as histograms can be seen. The top components are sensors that read the status information of the machine (under Linux). Each sensor is connected to a buffered sensor (same name with an asterix) that stores the values. The output of the buffered sensors are displayed by histogram plotters that are put together into one window frame. The window appearing in Pulse can be seen in Fig. 2.

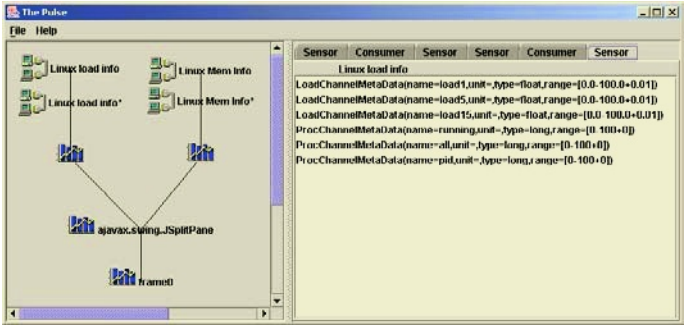

Fig. 1. Configuration of components in Pulse

3.1 Data Representation in Pulse

Pulse uses the concept of data channels. One channel has a data type so a row of data with the same type (e.g., integers) is delivered through it. There is one piece of data at a time in one channel thus, the row can be considered as data produced in time. For example, a service reporting the number of connected clients regularly generates a row of integers in time and can be considered as one channel. If a source of information provides more complex data than the basic data types (integers, floats, strings etc.), several channels has to be defined and the data has to be split into those channels. For example, a query to a relational database returns relational records. A record consists of columns with basic data type so the record can be split into channels, one column to one channel. Components in one hierarchy are connected with channels and data is flowing from the bottom to the top of the hierarchy through the channels.

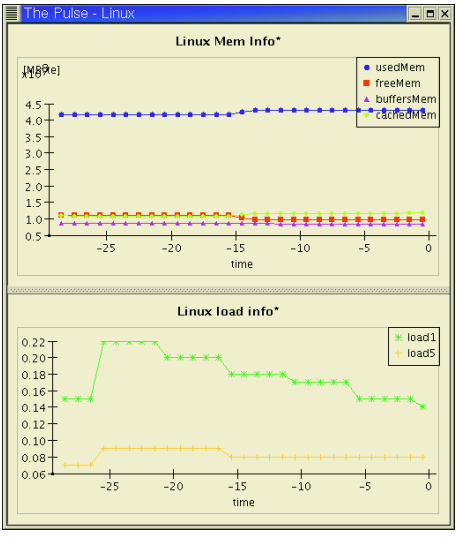

Fig. 2. Load average and memory usage histograms

3.2 Input Components

One of the basic components is the *Sensor*, the input component in Pulse. The task of such a component is to give an interface to any kind of data source. Pulse needs different channels (each typed) for different elements of source data. The sensor has to translate data coming from outside into the channel based representation of data.

An example for such sensors in Pulse is the *LinuxLoadInfo* (see Fig. 1 top left icon), which simply reads the content of the file `\proc\loadavg` under Linux and creates three data channels for the load average values: 1, 5 and 15 minutes load averages, plus three other data channels giving the current number of the running processes, all processes and the value of the largest process id (see the list of channels on the right side of the figure).

The next component in the hierarchy should be a *buffered sensor* which provides buffering of data between a sensor and a processing component. Thus, the processing components need not to care about the sensor's capability of buffering. E.g. a histogram plotter needs data back for a given time interval. An implemented instance of this component for general use is the *SamplerBuffer*. However, the buffered sensor is just an interface and some sensors can also implement the interface and behave as a buffered sensor. Thus, there is no need to connect them to a SamplerBuffer but directly to other processing components. The *RGMAConsumer* sensor that makes the connection between R-GMA and Pulse is, for example, a buffered sensor.

3.3 Analysis and Presentation Components

A presentation component of data is needed to show anything in Pulse. The most basic presentation component is a tabular view of data. One row in the table shows one data set

read from a sensor. The columns represent the different channels defined by the sensor. It can be considered as a relational record. The rows contain data read from the sensor regularly.

Another presentation component is able to draw histograms of data values using the Java Analysis Studio's plotters [7], see Fig. 2. Data arriving on one channel are considered as the values for one histogram line. Channels containing numbered values should be selected to use a histogram plotter.

The regularity of update is determined by the sensor to which the presentation component is connected. The processing elements subscribe to the connected sensor and they receive a notification when the sensor's status is changed, i.e. new data arrived or some failure happened.

Presentation components are the other end of the chain of the components. They produce an output on the screen to the user. Analysis components in-between the chain can be used to modify, filter or analyse data going through the chain. The simplest (and implicit component) one is a channel selector for the histogram plotter. It simply keeps the channels selected for drawing and drops the others. It is implicit as it is created automatically below the histogram plotter in the hierarchy.

Another basic analysis components can merge data coming from different sensors (provided they have the same type), filter one channel or compute new value from several channels. Binding such components together in a hierarchy, a dataflow graph can be created that the input sensors pour data into, and one or more presentation modules provide output to the user.

It must be mentioned here that in most cases a performance parameter taken on one machine can be compared to the same parameter on another machine. There is an ongoing research in our laboratory about defining grid metrics and procedures for transforming local information into globally interpretable grid information.

3.4 Configuration of Pulse

The graph of the components should be described in a configuration file using XML. The simplest component configuration consist of one data producer component, one sampler buffer component, one tabular viewer of data and one window frame. The use of independent components allows a flexible configuration of Pulse for actual data processing and analysis needs.

New components can be implemented as Java classes and they can be referred by their names in the configuration file. A new component should implement the interfaces of Pulse according to its role. Input components should implement the Sensor API while presentation components should implement the data consumer API. Analysis components should implement both of them.

4 Application Performance Monitoring

GRM [8] is a monitoring tool for performance monitoring of parallel applications running in the grid. PROVE is a visualisation tool for GRM traces. GRM has been developed for the P-GRADE [9] graphical parallel programming environment and it has been used

for on-line monitoring of P-GRADE applications in cluster environment. GRM collects trace data from all machines where the application is running any time when requested and the trace is visualised by the PROVE performance visualisation tool of P-GRADE.

These tools are the basis in the development of a grid application monitor that supports on-line monitoring and visualisation of parallel applications in the grid. The current version of GRM uses R-GMA to deliver generated trace data to the visualisation tool. The DataGrid software enables the users to execute an MPI application (as a parallel program) on one site of the grid but multi-site execution is not supported. GRM and R-GMA can be used for monitoring of MPI applications running on one site. Thus, the problem of the different timestamps of different sites is not an issue here.

PROVE is an on-line trace visualisation tool to present traces of parallel programs (see Fig. 3) collected by GRM. Its main purpose is to show a time-space diagram from the trace but it also generates several statistics from the trace that help to discover the performance problems, e.g. Gannt chart, communication statistics among the processes/hosts and detailed run-time statistics for the different blocks in the application process.

To enable monitoring of a grid application, the user should first instrument the application with trace generation functions. There is no automatic tool for instrumentation therefore, the source code of the application should be instrumented by hand. GRM provides an instrumentation API and library for tracing which are available for C/C++ and Fortran. The basic instrumentation functions are for the start and exit, send and receive, multicast, block begin and end events (giving the user the option to decide what a block means in her application code, inserting the begin and end statements by hand). However, more general tracing is possible by *user defined events*. For this purpose, first a format string of a new user event should be defined, similarly to C printf format strings. Then the predefined event format can be used for trace event generation, always passing the arguments only.

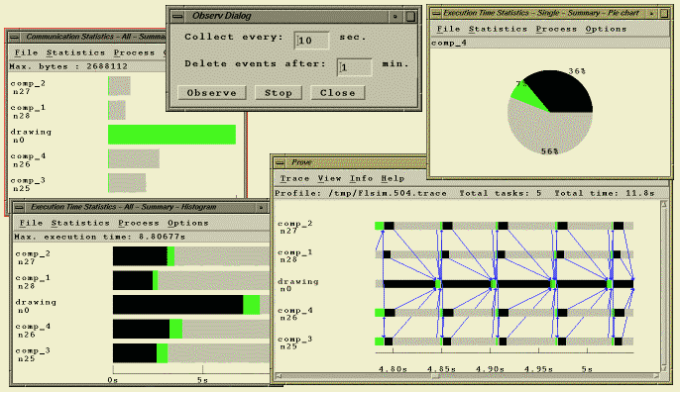

Fig. 3. Visualisation of trace and statistics in PROVE

The instrumentation functions automatically connect to R-GMA at the start of the processes and trace events are published to R-GMA. Thus, instrumented application

processes act as Producers of R-GMA. GRM's main monitor acts as a Consumer of R-GMA, looking for trace data and receiving it from R-GMA. The delivery of data from the machines of the running processes to the collection host is the task of R-GMA. R-GMA is using several servlets and buffers to deliver the trace data to the consumers. There is a local buffer in the application process itself that can be used to temporarily store data if a large amount of trace is generated fast. The processes are connected to ProducerServlets that are further connected to ConsumerServlets. Both kind of servlets create distinguished buffers for each Producer/Consumer that connect to them. GRM's main monitor receives the whole trace data in one single stream at the end.

The distinction between the traces of different applications is made by a unique id for each application. This id works as a key in the relational database schema and one instance of GRM is looking for one application with a given id/key.

After the application is submitted and GRM's main monitor is started, the main monitor connects R-GMA immediately and subscribes for traces with the id of the application. When R-GMA gives a positive response GRM starts continuously reading trace from R-GMA and the execution behaviour canbe followed by the user on the displays of PROVE.

5 Related Work

R-GMA is yet deployed within the DataGrid project only. Other grid projects are mostly based on MDS [10], the LDAP based information system of Globus but, e.g., the GridLab project is developing their own MDS information system and a new monitoring system [11].

To present the collected information, several different tools are used in different projects. Java Analysis Studio [7] has been developed to analyse and visualise large amount of scientific data stored in various formats for High Energy Physics but its features allow it to be used for more general use. Its plotting features are separated into a stand-alone package and this package is used in Pulse to create histograms. Nagios [12], a distributed host and service monitor running under Linux provides graphical plots to present monitoring information in a web browser.

Similarly to the case of GRM, the OMIS [13] on-line monitoring interface, developed for clusters and supercomuters is the basis for a grid application monitoring system within the CrossGrid project.

Netlogger is used for monitoring distributed applications in the grid rather then for parallel programs. Its time-space visualisation display concept is orthogonal to PROVE. In the vertical axis different types of events are defined while in PROVE the processes of the parallel programs are presented. Netlogger can be used for finding performance/behaviour problems in a communicating group of distributed applications/services while PROVE for a parallel program that is heavily communicating within itself. Netlogger, PROVE and other tools like Network Weather Service and Autopilot has been compared in the beginning of the DataGrid project in detail, see [14].

6 Conclusion

R-GMA is a relational Grid Monitoring Architecture delivering the information generated by the components in the grid. The large amount of monitoring information can be presented with different techniques. From simple web browser based queries to on-line monitoring of parallel applications, there are different tools available to get information from R-GMA and present it in an appropriate form. The simplest tool, the R-GMA Browser is used to search among the available information and look at the static or slowly changing behaviour of the grid infrastructure. Pulse can be used to build an analysis chain to process dynamic information about the services and the infrastructure on the fly and to create graphical plots to show the result of the analysis. GRM/PROVE is a parallel application monitoring toolset that is now connected to the grid. It can be used for on-line monitoring and performance analysis of parallel applications running in the grid environment.

References

1. EU DataGrid Project Home Page: *http://www.eu-datagrid.org*
2. S. Fisher et al. R-GMA: A Relational Grid Information and Monitoring System. 2nd Cracow Grid Workshop, Cracow, Poland, 2002
3. B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, and M. Swany. A grid monitoring architecture. GGF Informational Document, GFD-I.7, GGF, 2001, URL: *http://www.gridforum.org/Documents/GFD/GFD-I.7.pdf*
4. Steve Fisher. Relational Model for Information and Monitoring. GGF Technical Report GWDPerf-7-1, 2001. URL: *http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-7-1.pdf*
5. S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, and P. Vanderbilt. Grid service specification. GGF Draft Document, 2002, URL: *http://www.gridforum.org/meetings/ggf6/ggf6_wg_papers/draft-ggf-ogsi-gridservice-04_2002-10-04.pdf*
6. R-GMA Browser. URL: *http://hepunx.rl.ac.uk/edg/wp3/*
7. Java Analysis Studio. URL: *http://www-sldnt.slac.stanford.edu/jas/*
8. Z. Balaton, P. Kacsuk, N. Podhorszki, F. Vajda. From Cluster Monitoring to Grid Monitoring based on GRM. Proc. of EuroPar'2001, Manchester, pp. 874–881
9. P. Kacsuk. Visual Parallel Programming on SGI Machines. Proc. of the SGI Users' Conference, Cracow, Poland, pp. 37–56, 2000.
10. K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman. Grid Information Services for Distributed Resource Sharing. Proc. of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.
11. G. Gombás and Z. Balaton. A Flexible Multi-level Grid Monitoring Architecture. 1st European Across Grids Conference, Universidad de Santiago de Compostela, Spain, Feb. 2003
12. Nagios network monitoring tool. URL: *http://www.nagios.org*
13. T. Ludwig and R. Wismüller. OMIS 2.0 – A Universal Interface for Monitoring Systems. In M. Bubak, J. Dongarra, and J. Wasniewski, eds., Recent Advances in Parallel Virtual Machine and Messag Passing Interface, Proc. 4th European PVM/MPI Users' Group Meeting, LNCS vol. 1332, pp. 267–276, Cracow, Poland, 1997. Springer Verlag.
14. Information and Monitoring: Current Technology. DataGrid Deliverable DataGrid-03-D3.1. URL: *https://edms.cern.ch/document/332476/2-0*