



IST-2001-32133

GridLab - A Grid Application Toolkit and Testbed

D11.1 REQUIREMENTS ANALYSIS REPORT

Author(s):	Zoltán Balaton, Gábor Gombás
Document Filename:	GridLab-11-D11.1-01-Requirement
Work package:	WP11
Partner(s):	GridWare, MU, SZTAKI, VU
Lead Partner:	SZTAKI
Config ID:	GridLab-11-D11.1-01-v1.2
Document classification:	Internal

Abstract: In this document we describe the basic requirements of monitoring in the GridLab project. We present some use cases and describe the arising problems and requirements. We also describe support needed from other Work Packages.



Contents

1	Introduction	2
2	Information in the Grid	2
3	Use Cases	2
3.1	Scenario 1: Fault Detection and Analysis, Heartbeats	2
3.1.1	Requirements	3
3.2	Scenario 2: Job Status and Progress Monitoring	3
3.2.1	Requirements	3
3.3	Scenario 3: Application Performance Monitoring	3
3.3.1	Requirements	3
3.4	Scenario 4: Performance Analysis of Distributed Systems	3
3.4.1	Requirements	4
3.5	Scenario 5: Scheduling and Data Replication Services, Self Tuning Applications	4
3.5.1	Requirements	4
3.6	Scenario 6: Accounting and Auditing	4
3.6.1	Requirements	4
3.7	Summary of Requirements	5
4	Requirements Analysis	5
4.1	Push and Pull Data Delivery Models	5
4.2	Performance, Intrusiveness, Scalability	5
4.3	Data representation, Freshness, Accuracy	6
4.4	Monitoring Grid Entities	6
4.5	Reliability	7
5	Requirements from Other Work Packages	7
	References	7

1 Introduction

The Grid is a vast array of distributed resources. In essence, the aim of the Grid is to utilise resources in the most effective way giving the applications access to new capabilities and, furthermore, can be used to distribute an application's behaviour in a way which has not been possible before. Monitoring is necessary for understanding the complex behaviour of the system, for fault detection and to discover performance problems. To achieve this, data about the system must be gathered and processed to reveal important information. Then, according to the results, the system may need to be controlled. The monitoring system provides facilities for this: it supports collecting data from the components of the system and transforms this representation into an appropriate form for the end users.

2 Information in the Grid

There are many types of information available within the Grid, each with different properties. Some data are only available triggered by certain events (e.g. changes in the status of a job) while others can be measured continuously. In this context, event-like and continuous information can be distinguished. Continuous information can be made event-like by periodically taking a measurement of it (sampling). Since the Grid is a dynamic collection of resources, information changes in time. Some kinds of information can change quite often, while others change rarely. Even though information can be thought of as being dynamic, it is customary to speak about static and dynamic information. However, this classification is ambiguous because there is no clear line separating them, and so the difficulty of handling data representing these two classes of information is different. Static information (that changes very rarely) can be handled easily by existing systems such as LDAP [2]. Monitoring information on the other hand is usually highly dynamic and existing systems have problems handling it.

The information used for brokering also has different properties than monitoring information. The latter can usually be represented with simple data types. Another common property of monitoring information is that the recipients (who requested the measurement) are known when the data becomes available. Moreover, the number of recipients is small, usually consisting of one or at most a few. In contrast, information needed by a resource broker is very different. It has complex data types and the number of its recipients is large (ideally everybody) and not known when the data becomes available. Therefore, although current grid monitoring and information systems aim to handle both monitoring and brokering information, we believe that it is advantageous to separate the two since they have different requirements. Thus, we exclude brokering information from monitoring. It is the task of the information system to handle this information, possibly based on data provided by the monitoring system.

3 Use Cases

In this section we present typical usage scenarios of monitoring data and summarise their requirements they pose to the monitoring system. These use cases are based on those specified by the Performance Working Group of the Global Grid Forum [1], although they are grouped differently and we look at them from a monitoring point of view.

3.1 Scenario 1: Fault Detection and Analysis, Heartbeats

The corresponding scenario in [1] is scenario 5.

- Monitoring data is used to detect faults in system components and applications.
- Monitoring data could also be used to find the cause of the faults.

3.1.1 Requirements

- Monitoring of processes using heartbeats require push model data delivery.
- Heartbeat events are valid only for a short time interval and should be delivered within this time constraint.

3.2 Scenario 2: Job Status and Progress Monitoring

The corresponding scenario in [1] is scenario 6.

- Monitoring data is used to determine if a job is running, has died or hung.
- Following the status of long running jobs by monitoring.

3.2.1 Requirements

- This scenario implies both pull and push model data delivery (to query job status and to follow long running jobs respectively).
- Monitoring of jobs running in the Grid requires per job data scope (i.e. monitoring of grid job entities must be supported).

3.3 Scenario 3: Application Performance Monitoring

The corresponding scenario in [1] is scenario 1 and 13.

- Monitoring data is used to determine the performance characteristics of an application.
- The monitored events might be processed (e.g. creating resource usage profile of the application for scheduling) or visualised in real-time.
- Measurement data can be archived for later off-line analysis.

3.3.1 Requirements

- This scenario requires push model data delivery.
- Possibly large volume of events must be delivered in real-time.
- For general application monitoring, user defined event types must be supported.
- Remote data processing (statistics, remote data reduction) should be supported to handle the large number of events generated and to reduce network traffic. Level of remote data processing supported could range from the application of predefined algorithms, scripts, through to plug-ins.
- The monitoring system must support the creation of data objects containing archived measurements, likely to be stored on a passive storage element.

3.4 Scenario 4: Performance Analysis of Distributed Systems

The corresponding scenario in [1] is scenario 7.

- Monitoring data is used to locate performance bottlenecks in complex distributed systems.

3.4.1 Requirements

In addition to the requirements discussed in 3.3.1:

- Very large volume of data must be gathered from many different sources in real-time.
- Measurement data must be accurate and consistent.
 - Time stamps from different sites must be accurate.
 - Measurements from different sources must be comparable.

3.5 Scenario 5: Scheduling and Data Replication Services, Self Tuning Applications

The corresponding scenarios in [1] are scenarios 8, 9, 10 and 14. Data replication can be regarded as scheduling of data instances so it is included in this scenario. Self tuning applications have very similar needs to those of the schedulers, so they are also discussed here. Although we excluded brokering information from monitoring, schedulers could use monitoring directly to get additional data not handled by the information system.

- Schedulers or data replication services can use monitoring data to determine the optimal resources for a job or where to migrate or replicate data to optimise access time.
- Applications could also use monitoring data to adapt themselves to the current situation in the Grid (select suitable algorithms or change runtime parameters).

3.5.1 Requirements

- Accurate current usage and availability information is needed from all resources (including the network) in the Grid.
- Freshness of supplied data is critical.
- Availability of accurate forecasted data is desirable.

3.6 Scenario 6: Accounting and Auditing

The corresponding scenario in [1] is scenario 11.

- Monitoring data is used to account for resource utilisation.
- Verifying resource utilisation and level of service also requires monitoring data.

3.6.1 Requirements

- This scenario requires per user or per bank account data scope.
- Accuracy of measurements is important.
- Privacy of data is very important.

3.7 Summary of Requirements

In the following section, the requirements are summarised. Numbers in parenthesis denote scenarios that demand the requirement in question.

- Both push and pull data delivery model must be supported (all).
- The monitoring system must be scalable to be able to handle a large number of data producers and serve a lot of users (all).
- The monitoring system must be able to handle very large volume of data in real-time (3, 4).
- Monitoring must be unintrusive not to interfere with the system being monitored to ensure accuracy of measurements (3, 4, 5, 6).
- Accuracy and freshness of data must be identifiable (all).
- Data representation should be generic and easily extensible (3,4).
- A consistent set of standard measurements must be supported to provide comparable data (4, 5).
- Support for monitoring all grid entities (resources, network, jobs, users) is required (2, 5, 6).
- Remote data reduction and forecasting capability is desirable (3, 4, 5).
- The monitoring system must be reliable which includes fault tolerance and security (all).

4 Requirements Analysis

4.1 Push and Pull Data Delivery Models

Users can either access measured data via queries or alternatively, they may want to monitor a specific kind of information by subscribing to it. Subscription could be implemented by periodic queries (known as polling). However, this makes it very hard for the monitoring system to exploit knowledge about expected consumer behaviour. Therefore, it is useful to allow users to explicitly tell the monitoring system if they also want to get subsequent measurements (subscribe) or if they are only interested in getting one answer (query). On the other hand, querying is not equivalent to subscribing for only one measurement because the frequency of measurements is subject to a local policy. Further, the monitored information can be event-like, so the next measurement is not guaranteed to follow the subscription within a short time.

The semantics of query and subscription are different. A query implies pull model data delivery, whereas subscription corresponds to the push model. Both models have advantages and limitations. The pull model cannot handle event-like information. This may be circumvented by the use of a buffer i.e. measurements can be put into a buffer from where users can later retrieve them. However, this is not an ultimate solution because the size of the buffer is limited and so a notification needs to be sent using the push model when the buffer is full. The push model on the other hand cannot handle continuous information, thus the monitoring system must support both data delivery models and their combination.

4.2 Performance, Intrusiveness, Scalability

It is important that the intrusiveness of monitoring should be as low as possible. If monitoring interferes with the normal operation of the system being monitored, then measurements can show different behaviour than it would without monitoring. To avoid this, sensors should be able to limit their impact

on the monitored systems to control their intrusiveness. If more than one sensor is operating at the same time on the same system then they might need to coordinate their activity to control their impact on other sensors. Dynamic sensor management is desirable to control intrusiveness on the monitored system. There are various levels of sensor control than can be supported. These include enabling/disabling sensors, setting predefined sensor parameters or the use of more complex algorithms using past or forecasted information. This will be investigated later in the third year of the project.

The monitoring system must be scalable i.e. it must be prepared to serve a large number of users using many producers as well as well as being able to handle a high volume of data efficiently. The scalability requirement naturally implies that centralised solutions are not adequate, therefore only distributed systems can be considered. It is also important that data should only be sent where it is needed and only when requested.

4.3 Data representation, Freshness, Accuracy

In the monitoring system measurable quantities are represented by metrics. A metric is defined by the following properties:

- Name of the metric (e.g. CPU usage).
- Scope. This can be used to differentiate between similar metrics operating at different levels (e.g. CPU utilisation can be measured on a per-job or per-host level) as well as grouping metrics.
- Parameters (e.g. hostname). Many metrics can be measured at different places simultaneously. For example, CPU utilisation can be measured on all hosts in the resource. The metric parameters can be used to distinguish between these different metric instances.
- Type of measurement. This can be continuous, meaning the data is always available, or event-like, meaning data only becomes available when some external event happens. For continuous metrics the application can request the monitoring system to do periodic measurements automatically.
- Unit. The unit in which the quantity is measured.
- Data type. Most monitoring data has simple data types (integer, string or float) but there are cases when this is not enough so simple aggregate types (records and arrays) also have to be supported.

The accuracy of measurements must be known. This can be ensured by providing error bounds for measurements. It is also important that stale data can be identified, thus measurements must contain a timestamp.

The measurements must be comparable. This problem does not arise in the case of a local cluster since the system is either homogeneous or the properties of the machines needed for the comparison are known by the user. In a grid environment, these assumptions do not hold since it is typical that resources are inhomogeneous and their properties are not known by grid users. Thus a consistent set of possible measurements must be agreed upon and provided by all parts of the distributed system. These *grid metrics* must be standardised by the grid community (or a standardisation body accepted by the grid community) and the monitoring system must be able to convert local metrics to grid metrics. Additionally, the monitoring system must also support an ad hoc set of “local standards” where only a smaller community of grid users (e.g. within a virtual organisation) agrees on the metric semantics.

4.4 Monitoring Grid Entities

Monitoring every grid entity, such as jobs or resources, should be supported. This raises the issue of handling entities that are not tied to a host. For example, the monitoring system must be able to find the

processes that belong to a job. However, since it can only monitor processes and jobs can consist of more than one process (possibly running on several resources) then either it should be possible to identify the processes as being part of a job or an interface is necessary to the scheduler to get this information.

To monitor the whole lifetime of a job it is important to start measurements no later than the processes start. Since in the Grid the hosts executing the processes of the application are determined by the scheduling services and influenced by the local policies of the resources it is not known in advance where to start measurements. Thus, buffering of possibly large volume of data in the monitoring system would be needed. To avoid this or the loss of measurements, a mechanism is required to set up monitoring before the job starts.

4.5 Reliability

The monitoring system must be reliable. This comprises fault tolerance and security. It must ensure that only authorised users have access to published information and that a local fault in the system can only cause problems in that part and does not influence other parts of the system. The fault tolerance requirement is met by the use of a completely distributed decentralised system where data is only sent where it is requested. These are the same requirements that are needed for scalability.

For accessing job related metrics, the basic rule of thumb is that whoever has the right to cancel a job also has the right to monitor it. Other metrics can be either public or limited to a small group of administrators.

5 Requirements from Other Work Packages

For timestamps to be useful a reasonable level of time synchronisation should be provided. Since this does not belong to a monitoring system, it must be handled by other tools (e.g. the monitored sites should use a time synchronisation protocol such as NTP to synchronise their clocks).

To support job level monitoring the relation between process IDs and grid job IDs must be known. There are two basic methods to solve this problem:

- The jobmanager must provide an interface that can be used to get the mapping between process IDs and grid job IDs.
- Every job must run under a unique dynamically allocated user ID, so that the mapping can be determined by looking at which user ID the processes run under.

The jobmanager interface must also provide mapping between grid users and running jobs.

An interface to the local resource manager is also needed to ensure that the monitoring is set up before the job starts running so that the whole lifetime of the job can be monitored. The monitoring system needs information about what measurements the user requested and it must be able to start sensors on the hosts that will run the job before the processes are started.

We will rely on the security recommendations and mechanisms provided by the Security Work Package.

References

- [1] Ruth Aydt and Darcy Quesnel: Performance Data Usage Scenarios
Technical Report, Global Grid Forum, October 2, 2000
<http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-3-1.pdf>
- [2] M. Wahl, T. Howes, S. Kille: Lightweight Directory Access Protocol (v3),
IETF RFC 2251, December 1997.