On-Line Monitoring of Service-Level Agreements in the Grid

Bartosz Balis[1,2], Renata Slota[1], Jacek Kitowski[1], and Marian Bubak[1,3]

[1] AGH University of Science and Technology,
Department of Computer Science, Krakow, Poland
[2] ACC Cyfronet AGH, Krakow, Poland
[3] University of Amsterdam, Institute for Informatics, Amsterdam, The Netherlands
balis@agh.edu.pl

Abstract. Monitoring of Service Level Agreements is a crucial phase of SLA management. In the most challenging case, monitoring of SLA fulfillment is required in (near) real-time and needs to combine performance data regarding multiple distributed services and resources. Currently existing Grid monitoring and information services do not provide adequate on-line monitoring capabilities to fulfill this case. We present an application of Complex Event Processing principles and technologies for on-line SLA monitoring in the Grid. The capabilities of the presented SLA monitoring framework include (1) on-demand definition of SLA metrics using a high-level query language; (2) real-time calculation of the defined SLA metrics; (3) advanced query capabilities which allow for defining high-level complex metrics derived from basic metrics. SLA monitoring of data-intensive grid jobs serves as a case study to demonstrate the capabilities of the approach.

Keywords: on-line monitoring, SLA monitoring, Grid computing, complex event processing.

1 Introduction and Motivation

Grid infrastructures federate resources from different providers [11], hence Service Level Agreements between computing centers comprising the Grid, and users running jobs, are needed to ensure the desired quality of service [10,7]. An essential phase in SLA management is the monitoring of SLA fulfillment. The prevailing approach is off-line SLA monitoring: data about resource usage and performance is periodically sampled, stored, and subsequently analyzed for SLA violations, like in the European EGI/EGEE infrastructure [13]. In on-line SLA monitoring, on the other hand, resource usage and performance are analyzed on the fly which allows for immediate alerts or corrective actions when an SLA violation is detected or predicted.

We present a framework for on-line monitoring of SLA contracts in the Grid. The solution is based on leveraging Complex Event Processing for on-line monitoring in the Grid – GEMINI2 [1]. In this approach, basic SLA performance

M. Alexander et al. (Eds.): Euro-Par 2011 Workshops, Part II, LNCS 7156, pp. 76–85, 2012.

metrics are collected on-line, while complex SLA metrics can be defined on-demand as queries in a general-purpose continuous query languages EPL, and calculated in real-time in a CEP engine. Advanced query capabilities are afforded by this approach: value aggregations, filtering, distributed correlations, joining of multiple streams of basic metrics, etc. Furthermore, client-perspective SLA monitoring is made possible. The capabilities of the solution are demonstrated in a case study: SLA monitoring of data-intensive Grid jobs.

This paper is organized as follows. Section 2 presents related work. Section 3 describes the framework for on-line SLA Monitoring in the Grid. In section 4, SLA monitoring of data intensive jobs is studied. Section 5 concludes the paper.

2 Related Work

On-line monitoring of large-scale infrastructures is essential for many purposes such as performance steering [16], system intrusion detection [12] or self-healing [3]. There are a few approaches for on-line SLA monitoring in the Grid [4,5,15].

Menychtas et al. [5] propose a QoS provisioning approach which takes into account real-time monitoring information about jobs and resources. A generic mapping mechanism is employed in order to map low-level metrics to high-level QoS parameters.

Litke et al. [4] present an execution management framework for OGSA-based Grids which, given a set of client requirements expressed in SLAs, finds candidate services satisfying these requirements, executes them, and monitors the SLA fulfillment. The monitoring service is tightly coupled with the framework and provides basic QoS metrics (such as CPU / memory / disk usage, network bandwidth). On-line monitoring boils down to periodical notifications of QoS metrics.

Truong et al [15] describe a framework for monitoring and analyzing QoS metrics of Grid services. On-line monitoring of QoS is based on the SCALEA-G framework [14]. In comparison to CEP, query capabilities of SCALEA-G are limited. A client essentially can choose which entities to monitor, select desired metrics, and optionally specify XQuery or XPath filters (data is represented in XML). Moreover, unlike CEP, XQuery/XML have not been designed for real-time queries over data streams.

There have been some efforts to support on-line SLA monitoring for Web Services [6,9]. Michlmayr et al. [6] present an approach similar to our work in that it leverages event-based monitoring and Complex Event Processing. However, the way CEP is used has some restrictions in comparison to our approach. Basically only three event streams exist which represent QoS properties at the level of service, service revision, and service operation, respectively. The use of CEP query constructs is limited to sliding windows, aggregations and filtering. Overall the approach is strictly oriented to Web Services. In contrast, we propose a generic framework in which event streams represent individual performance metrics which can be combined into high-level composite metrics. The way these metrics are mapped into SLA obligations is out of scope of this paper.

In [9], the authors propose the timed automata formalism to express SLA violations, and automatically generate monitors for these violations. Exactly the same can be achieved with Complex Event Processing: continuous query language enables one to express SLA violations, while installing a query in a generic CEP engines is equivalent to creating a new monitor. However, CEP has the advantage of availability of mature and efficient technologies. Moreover, a continuous query engine is more user-friendly and arguably no less expressive than a timed automata. In fact, automata are formalisms often used in the implementation of CEP engines [8].

3 On-Line SLA Monitoring Framework

3.1 Architecture

Fig. 1 presents a high-level view over the architecture of the on-line SLA monitoring framework. The *SLA Monitoring Service* and the *Resource Information Registry* are the core components of the framework. Also shown are Resources of the Grid Infrastructure (computers, storage devices, software services), a Resource Provider, and a Service-Level Management Service which uses the SLA Monitoring Service to define SLA Metrics, and makes corrective actions when an SLA violation takes place or is predicted.

The resources of the Grid infrastructure provide event streams of basic SLA metrics, such as current CPU load, current memory consumption, current data transfer rate, response time to the latest client service request, etc. Additional metrics can also be provided by the client side (response times, transfer rates measured by the client, etc.).

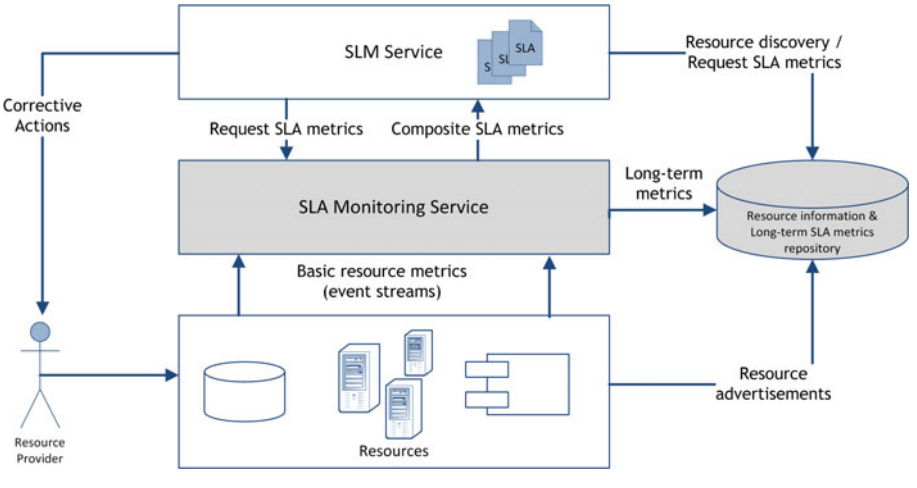

Fig. 1. Architecture of Grid On-Line SLA Monitoring Framework

The streams of basic metrics are consumed by the SLA Monitoring Service wherein they can be transformed into *composite metrics* derived from one or more basic streams. The composite metrics are defined on demand using the *continuous query language*, and calculated in real-time in the CEP engine. Examples of composite metrics include:

- *Value aggregations*: average / minimum / maximum values in a specified time window, etc. For example: *Average CPU load on every monitored host within last 5 minutes.*
- *Stream joining*: combination of values from different streams joined by a common attribute value. Example: *return all host names whose average CPU load over last 5 minutes exceeds 90% AND top 10 processes on those hosts in terms of CPU consumption.*
- *Distributed correlations*: event patterns, such as event not followed by another one within a specified time, occurrence of any of two events, etc.

Additional query mechanisms available for defining composite metrics include value filtering, results grouping and ordering, etc.

The functionality of the SLA Monitoring Service is complemented by the Resource Information Registry. While the SLA Monitoring Service deals with dynamic metrics of the resources, the Registry stores their static attributes (OS info, total memory, CPU type, total storage capacity, etc.), as well as long-term metrics (monthly, yearly, all-time average, etc.). Information about static attributes is published by the resources using *advertisements* – special messages sent periodically to the SLA Monitoring Service which in turn updates the Registry if necessary. If the advertisement is not received for a certain period of time, the resource is considered unavailable and its corresponding entry is marked inactive. The SLA Monitoring Service can also be configured to update the long-term metrics in the Registry based on the values from the event streams.

Such an architecture enables one to monitor long-term SLA metrics (e.g. monthly availability), and to define even more complex composite metrics which combine calculations based on dynamic metrics and constraints imposed on static attributes. The values from the Registry can be joined in a continuous query with other real-time streams. Example: *return all host names whose average CPU load exceeds 90% within the last 5 minutes, and whose operating system is a Linux distribution.*

3.2 Design and Implementation

The SLA Monitoring Service is designed and implemented on the basis of the GEMINI2 monitoring system [1]. GEMINI2 provides a framework for on-line monitoring which encompasses a CEP-based monitoring server (GEMINI2 Monitor) and local sensors (GEMINI2 Sensors). Monitoring data is represented as events (collections of name – value pairs) which typically contain at least a unique resource identifier (e.g. a host name), and a set of associated metrics (e.g. current CPU load on the host).

Sensors are responsible for measuring the metrics and publishing the associated events to a Monitor. The Monitor contains a CEP engine (Esper [2]) and exposes a service to formulate queries in the Event Processing Language (EPL). The event streams from Sensors are processed against the queries in the CEP engine which results in derived complex metrics returned to the requester.

Besides monitoring event streams, Sensors also periodically publish *Advertisement events* in the Monitor. These events register a resource with the Monitor, along with their static attributes.

3.3 Defining Composite SLA Metrics Using EPL

Composite SLA metrics are defined as EPL queries over streams of basic metrics. Let us consider a relatively complex composite metric which demonstrates the query capabilities of the EPL language: *return host names whose average CPU load exceeds 95% in the last 5 minutes, and top 10 processes on those hosts in terms of CPU usage*. Expressed in EPL:

```
select host.hostName, avg(host.cpuLoad),
       proc.pid, avg(proc.cpuUsage)
from HostMs.win:time(5min) as host,
     ProcessMs.win:time(5min) as proc
where proc.hostName = host.name  /* join 2 streams */
group by host.name, pid
having avg(host.cpuLoad) > 95
output all every 2 minutes
order by avg(proc.cpuUsage) desc  /* sort results */
limit 10 /* display top 10 results */
```

This request selects attributes from two streams: *HostMs* (which contains host name and host metrics such as the current CPU load), and *ProcessMs* (which contains a process identifier, host name on which the process is running, and metrics, such as the CPU usage). The streams are joined with the value of the common attribute: the host name.

3.4 Registry

Registry is a database associated with a Monitor which contains information about resources, specifically their static attributes (metadata) which are not published in the monitoring event streams.

In order to combine data from the event streams and the Registry, the EPL request can contain an SQL query. For example the request from section 3.1 is expressed in EPL as follows:

```
select rh.host_name, avg(host.cpuLoad)
from HostMs.win:time(5min) as host,
     sql:Registry [''select host_name from Host
                   where host_name = ${host.hostName}
             and host_os = 'Linux'''] as rh
     having avg(host.cpuLoad) > 0.9
```

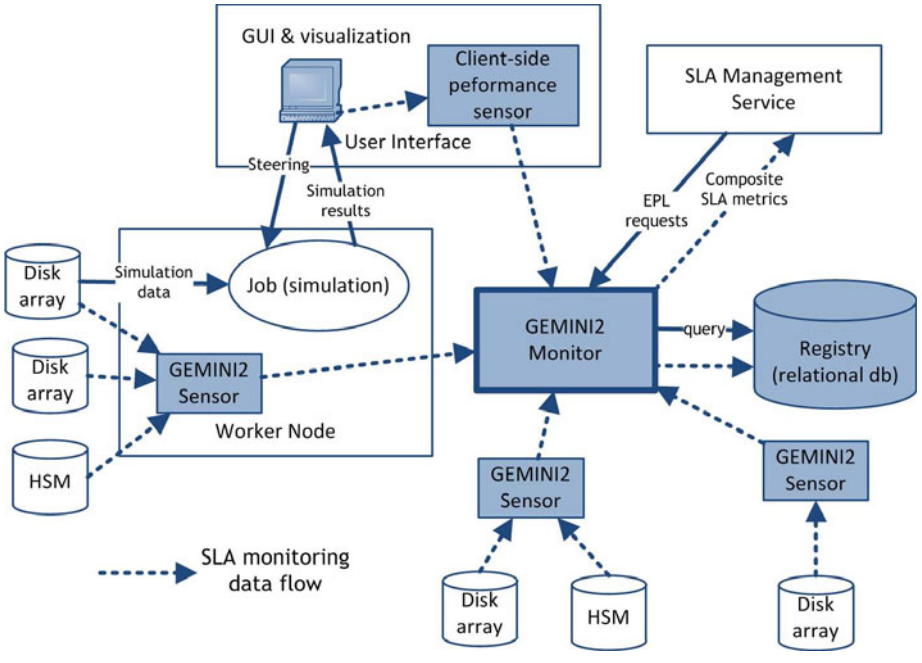

Fig. 2. Example deployment of resources and SLA Monitoring system components for the monitoring of data-intensive jobs scenario

4 SLA Monitoring for Data-Intensive Computations

The capabilities of the presented SLA Monitoring solution will be demonstrated in a case study which involves data storage and data-intensive computations. The main entities involved in this case study are: (1) storage resources (local disks, disk arrays and hierarchical storage management (HSM) devices); (2) jobs processing large volumes of data, running on worker nodes of the Grid infrastructure; (3) user interface.

An example deployment of these entities is shown in Fig. 2. In this case, a job running on a worker node retrieves data from a disk array in order to run a simulation, whose results are visualized on a graphical user interface. Furthermore, the simulation is interactive: the user can steer it on the fly.

For the purpose of SLA monitoring, storage resources publish streams of performance metrics. However, the client host and GUI application are also instrumented in order to publish client-side performance metrics, such as response times and inbound data transfer rates. This allows for SLA monitoring also from the client perspective. The monitored entities, their attributes (only applicable for storage resources), and SLA metrics are summarized in Table 1.

Table 1. Monitored entities, their attributes and basic SLA metrics

Entity / *Event stream name*	**Static attributes & long-time metrics**	**Basic SLA metrics**
Local disk *LDMs*	average read/write transfer rate total capacity	current read/write transfer rate free capacity
Disk array *DAMs*	average read/write transfer rate total capacity raid level strip size	current read/write transfer rate free capacity
Hierarchical storage management (HSM) device *HSMMs*	average read/write transfer rate total capacity average mount time average load time average position time number of libraries, drivers and tapes	current read/write transfer rate free capacity
Client GUI *ClientPerfMs*	N/A	response time of steering requests
Client host *DataTransferPerfMs*	N/A	inbound data transfer rate outbound data transfer rate

Let us consider a number of examples of composite SLA metrics formulated in the EPL query language. The first three metrics rely on storage resource performance metrics.

1. *Return average read transfer rate for a disk array with particular ID for the last 80 minutes.*

```
select avg(currentReadTransferRate)
from DAMs(id = 'IP:mountDir').win:time(80 min);
```

2. *Every 5 minutes return average read transfer rate for those disk arrays for which it exceeded 100MB/s within the last 40 minutes.*

```
select serverName, id, avg(currentReadTransferRate)
from DAMs.win:time(40 min) group by serverName, id
having avg(currentReadTransferRate) > 100
output all every 5 minutes;
```

3. *Return current free capacity and average write transfer rate for all disk arrays managed by server zeus.cyfronet.pl.* This request may be useful, e.g., to predict the running out of the disk space.

```
select id, freeCapacity, avg(currentWriteTransferRate)
from DAMs(serverName='zeus.cyfronet.pl').win:time(5 min),
group by id
output all every 5 minutes;
```

The next example shows a metric which combines data from event streams and the Registry. The request selects HSM devices which currently undergo high write transfer rates. In addition, the historical average for the device is returned.

```
select hsm.id, avg(hsm.currentWriteTransferRate), hsmreg.avgWriteTransferRate
from HSMMs.win:time(5 min) as hsm,
     sql:Registry [ ''select avg_write_transfer_rate as avgWriteTransferRate
                      from HSM
                      where res_id = ${hsm.id}''']
     having avg(hsm.currentWriteTransferRate) > 60
```

Finally, the following example demonstrates SLA Monitoring that includes client-side metrics. Let us assume that the user running and steering the simulation would like that two requirements are satisfied:

- The simulation is sufficiently responsive to user steering actions.
- The simulation results are delivered to GUI with transfer rate large enough for real-time visualization.

Consequently, the following SLA could be requested: (a) average response time of user interactions does not exceed 100ms, AND (b) average data transfer rate from the processing job to the GUI does not drop below 128KB/s. Expressed in EPL:

```
select avg(a.responseTime, 90), avg(b.inTransferRate)
from pattern [ every (a=ClientPerfMs(appId='app1') or
(b=DataTransferPerfMs(port='1111')))
].win:time(5 min)
having avg(a.responseTime, 90) > 100 or
       avg(b.inTransferRate) < 128
```

This request consumes two event streams mentioned earlier: *ClientPerfMs*, which contains, among others, response time of the latest simulation steering request; *DataTransferPerfMs* which contains performance metrics of data transfers to/from a host. The first stream also contains attribute *appId* which identifies the particular simulation session, and which is used to filter the stream. The second stream is also filtered against port number 1111 on which the GUI receives the simulation results. The request defines an event pattern '*AorB*' – fulfilled if either of two event happens.

5 Conclusion

This paper presents a novel and generic solution for efficient, near real time monitoring of Service Level Agreements in the Grid. This solution is based on the application of Complex Event Processing principles and supporting technologies. We have elaborated a generic framework in which event streams represent individual performance metrics which, in turn, can be combined into high-level composite metrics. The main features of the monitoring framework are: on-demand definition of SLA metrics using a high-level query language, real-time calculation of the defined SLA metrics and advanced query capabilities which allow for defining high-level complex metrics derived from basic metrics. Resource information registry complements the functionality of the framework by providing a space for storing historical or long-term metrics, as well as resource metadata. The information from the Registry can also be used in continuous queries, further enhancing the capabilities of the framework in terms of definition of complex SLA metrics. The case study of the data-intensive application have demonstrated the feasibility of this approach.

Future work involves the investigation of an efficent way of mapping of high-level metrics into SLA obligations, improvement of performance of the framework, and investigation of other on-line SLA monitoring use cases.

Acknowledgments. This work is partially supported by the European Union Regional Development Fund, POIG.02.03.00-00-007/08-00 as part of the PL-Grid Project.

References

1. Balis, B., Kowalewski, B., Bubak, M.: Real-time Grid monitoring based on complex event processing. Future Generation Computer Systems 27(8), 1103–1112 (2011), http://www.sciencedirect.com/science/article/pii/S0167739X11000562
2. Berhardt, T., Vasseur, A.: Complex Event Processing Made Simple Using Esper (April 2008), http://www.theserverside.com/news/1363826/Complex-Event-Processing-Made-Simple-Using-Esper (last accessed June 30, 2011)
3. Gorla, A., Mariani, L., Pastore, F., Pezzè, M., Wuttke, J.: Achieving Cost-Effective Software Reliability Through Self-Healing. Computing and Informatics 29(1), 93–115 (2010)
4. Litke, A., Konstanteli, K., Andronikou, V., Chatzis, S., Varvarigou, T.: Managing service level agreement contracts in OGSA-based Grids. Future Generation Computer Systems 24(4), 245–258 (2008)
5. Menychtas, A., Kyriazis, D., Tserpes, K.: Real-time reconfiguration for guaranteeing QoS provisioning levels in Grid environments. Future Generation Computer Systems 25(7), 779–784 (2009)
6. Michlmayr, A., Rosenberg, F., Leitner, P., Dustdar, S.: Comprehensive QoS monitoring of Web services and event-based SLA violation detection. In: Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing, pp. 1–6. ACM (2009)
7. Moscicki, J., Lamanna, M., Bubak, M., Sloot, P.: Processing moldable tasks on the grid: Late job binding with lightweightuser-level overlay. Future Generation Computer Systems 27(6), 725–736 (2011), http://www.sciencedirect.com/science/article/pii/S0167739X11000057
8. Mühl, G., Fiege, L., Pietzuch, P.: Distributed Event-Based Systems. Springer (August 2006)
9. Raimondi, F., Skene, J., Emmerich, W.: Efficient online monitoring of web-service slas. In: Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering, pp. 170–180. ACM (2008)
10. Sahai, A., Graupner, S., Machiraju, V., van Moorsel, A.: Specifying and Monitoring Guarantees in Commercial Grids through SLA. In: CCGRID 2003: Proceedings of the 3st International Symposium on Cluster Computing and the Grid, p. 292. IEEE Computer Society, Washington, DC (2003)
11. Schwiegelshohn, U., Badia, R.M., Bubak, M., Danelutto, M., Dustdar, S., Gagliardi, F., Geiger, A., Hluchy, L., Kranzlmüller, D., Laure, E., Priol, T., Reinefeld, A., Resch, M., Reuter, A., Rienhoff, O., Rüter, T., Sloot, P., Talia, D., Ullmann, K., Yahyapour, R., von Voigt, G.: Perspectives on grid computing. Future Generation Computer Systems 26(8), 1104–1115 (2010), http://www.sciencedirect.com/science/article/pii/S0167739X10000907
12. Smith, M., Schwarzer, F., Harbach, M., Noll, T., Freisleben, B.: A Streaming Intrusion Detection System for Grid Computing Environments. In: HPCC 2009: Proceedings of the 2009 11th IEEE International Conference on High Performance Computing and Communications, pp. 44–51. IEEE Computer Society, Washington, DC (2009)

13. Szepieniec, T., Tomanek, M., Twaróg, T.: Grid Resource Bazaar: Efficient SLA Management. In: Proc. Cracow Grid Workshop 2009, pp. 314–319. ACC CYFRONET AGH, Krakow (2009)
14. Truong, H.L., Fahringer, T.: SCALEA-G: a Unified Monitoring and Performance Analysis System for the Grid. Scientific Programming 12(4), 225–237 (2004)
15. Truong, H., Samborski, R., Fahringer, T.: Towards a framework for monitoring and analyzing QoS metrics of grid services. In: Second IEEE International Conference on e-Science and Grid Computing, e-Science 2006, p. 65. IEEE (2006)
16. Wright, H., Crompton, R., Kharche, S., Wenisch, P.: Steering and visualization: Enabling technologies for computational science. Future Generation Computer Systems 26(3), 506–513 (2010)