DESIGN AND IMPLEMENTATION OF A NOTIFICATION MODEL FOR GRID MONITORING EVENTS*

S. Andreozzi, , G. L. Rubini, INFN-CNAF, Bologna, Italy
Sergio Fantinel, Lab. Naz. di Legnaro, Legnaro, Italy
N. De Bortoli, G. Tortone, INFN, Napoli, Italy

Abstract

Grid systems involve a large number of users and resources across traditional administrative and organizational domains. An aspect to consider as regards the monitoring activity of such systems is the efficient match and delivery of notification events to subscribers. In this paper, we propose a notification model for events that relies on recent evolution in XML document filtering. We also propose a design implementation in the context of the GridICE monitoring tool.

INTRODUCTION

The Grid monitoring is the activity of measuring significant Grid resource-related parameters in order to analyze usage, behavior and performance of a Grid, and detect and notify fault situations and user-defined events. In this context, we have proposed a tool called GridICE [1] designed to fulfil a number of requirements that are specific of Grid systems.

Its architecture already contains by design a notification service for monitoring events. In the current GridICE implementation (version 1.6.2), such a service is implemented with basic capabilities, i.e., the sending of e-mail notifications to persons or group can be configured for a set of predefined events.

In this paper, we present a new design of the notification service that enhances the current one by providing (1) an expressive subscription language, (2) aggregated notifications, (3) efficient matching between subscriptions and events, (4) customizable notification frequency (5) and asynchronous notification delivery. The proposed solution gains from the recent evolutions in the area of XML document filtering.

OVERVIEW OF GRIDICE

In this section, we provide an overview of GridICE [1]. This monitoring tool has been initially developed by the INFN (Istituto Nazionale di Fisica Nucleare) as part of the European DataTAG project. It has been later integrated in the LHC Grid Computing middleware version 2 [7].

The GridICE architecture is modular and structured in five layers (see Figure 1). The first layer is the Measurement Service that probes the resources for simple or composite metrics. The set of collected metrics are an extension of the GLUE Schema [3], i.e., a common definition of entities and attributes that is the result of a joint collaboration between large European and American Grid projects.

The second layer is the Publisher Service that offers the gathered data to potential consumers. Monitored data are locally collected in an edge node that have access to the Internet. The monitored data are exposed through a common interface, that in the current version is provided by the Globus Monitoring and Discovery Service (MDS) version 2 [6].

The third layer is the Data Collector Service that allows for the collection of historical monitoring data. Its main components are the New Resources Detector, the Scheduler and the Persistent Storage. The first acts periodically in order to detect new sources of data to be observed. Given a new set of sources of data, the Scheduler fires periodical observations to discover which monitored data are offered and stores the result via the Persistent Storage.

The fourth layer is the Data/Notification and Data Analyzer Services. The first refers to a flexible and configurable means for event description, detection and notification and is the focus of this paper. The second is the Data Analyzer Service that provides performance analysis, usage level and general reports and statistics.

The fifth and last layer is the Presentation Service that offers a web-based graphical user interface. The monitored data are aggregated depending on the type of the consumer (e.g., VO manager, Site manager, GOC administrator).

Figure 1: Layered Architecture

*This research was partially funded by the IST Program of the European Union under grant IST-2003-508833 (EGEE project) and by the Italian Grid.It project

NOTIFICATION SERVICE REQUIREMENTS

In the introduction, we have mentioned that the notification service currently implemented within GridICE provides basic capabilities, as it offers only the notifications of a predefined set of events (e.g., disk space depletion, i-node availability lower than a certain threshold, transition of a process from down to up and vice versa).

The growing interest for more flexible and scalable notification capabilities from LHC (Large Hadron Collider) experiments has led us to study a more suitable solution satisfying their needs. The following set of requirements were inferred after to have investigated a number of use cases: (1) enable each final user to define the precise set of events it is interested in receiving; (2) final user should be able to add and delete notification requests; (3) the service should aggregate the information about all the events occurring at Grid level and matching his specifications, in order to provide final users with the possibility to receive unified notification; (4) final users should choose the frequency with which receives notifications (e.g., daily, hourly); (5) asynchronous notification delivery.

Message-oriented systems represent an appealing solution as the communication paradigm of the notification service. Moreover, the flavor of publish/subscribe family includes event driven mechanisms and distribution of data functionality regardless who or where the recipients are.

Publish/Subscribe Systems: Overview

In publish/subscribe systems, components interact by publishing messages and subscribing to classes of messages. The main components are: publisher(s), subscriber(s), broker. Publishers and subscribers exchange messages trough the broker without having reciprocal knowledge: publishers connect to the broker in order to publish events into the network, while subscribers connect to the broker in order to register their preferences, in which they specify, via selection criteria, the set of messages they are interested to receive. Published messages are named events, while subscribers preferences are named subscription.

The broker should match events with subscriptions, delivering the message of interest to all (and only) the subscribers. Two important considerations regard the type of subscriptions supported by a publish/subscribe system and how such system matches events against subscription. A publish/subscribe system can be classified as:

- topic-based: a subscription can selects events choosing a specific topic from a predefined set of available topics.
- content-based: a subscription can specify predicate over the content of the events.

The matching between events and registered subscriptions is based on a filter algorithm implemented by a component named filter engine: due to the filter capabilities, a subscriber will only receive messages satisfying his expressed criteria.

A filter engine is required of (1) execute matching events with subscriptions; (2) expressivity for language used to define subscription criteria; (3) handle very large number of subscriptions; (4) to be able to add and remove subscriptions.

Filter Engine: XML filtering and YFilter

The XML (Extensible Markup Language) is used worldwide for representing and exchanging data. In recent years, a growing interest has been directed to the filtering and content-based routing of XML data. In an XML filtering system, events are represented by XML documents while subscriptions are expressed in a language (e.g., XPath or XQuery) able at specifying constrains over both structure (path expressions) and content (value-based predicates). Matching between events and subscriptions is a two-steps process: structure matching and predicates processing. A number of XML filtering algorithms and implementations have been proposed. Within the design of the new GridICE notification service, the adopted filter engine is YFilter [5], that evaluates path expressions and predicates over streaming XML data, via a Nondeterministic Finite Automata (NFA).

YFilter represents an evolution of the XFilter project, where filtering of path expressions over streaming XML data was based on the use of indexed Finite State Machine (FSM): elements of a path expression are mapped to states; a transition from an active state succeeds when an element is found in the event that matches that transition; if an accepting state is reached, then the event satisfies the subscription. While in XFilter each path expression is related with a FSM, YFilter combines multiple subscriptions into a single NFA, merging common prefixes of the subscription paths and reducing the number of states needed to represent a set of subscriptions.

In YFilter, path expressions are expressed using a subset of XPath and are composed of a sequence of location steps. A location step consists of an axis, a node test and zero or more predicates. An axis is used to specify the hierarchical relationship between the nodes (the focus is on parent-child and descendent-or-self relationship); the node test is typically a name test that could be an element name or a wildcard operator '*'; the predicates are filters over node-test. Path expressions specified by XPath can be transformed into regular expressions for which exists an FSM that accepts the language described by the expression. The combined NFA representing the set of subscriptions should (1) identify the exact language defined by all path expressions; (2) when an accepting state is reached, it outputs all subscriptions accepted at this state. Figure 2 depicts an example of an NFA referred to eight queries. A circle denotes a simple state, grey circle denotes a shared state and crossed circle denotes an accepting state, marked by the IDs of accepted subscriptions.

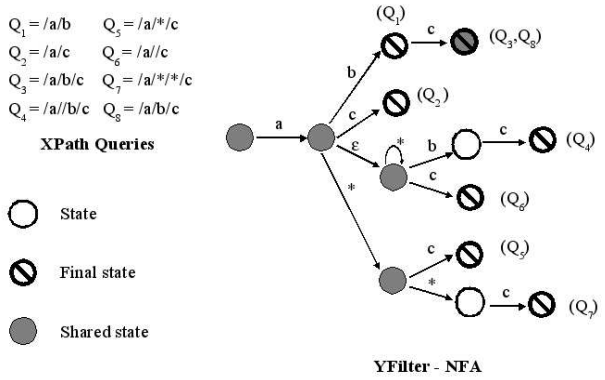

Figure 2: YFilter NFA and subscription examples

Each edge represents a transition and is labeled with the symbol that fires that transition. The symbol '*' matches any element; the symbol 'ε' is used to mark a transition that requires no input (empty input transition). The NFA construction in YFilter is an incremental process and new subscriptions can easily be added to an existing system.

The execution of the NFA is event-driven: when a new XML document is received for filtering, firstly it is parsed. The start or the end of an XML element triggers a transition in the NFA. As introduced, subscription specifications could include value-based predicates on the elements of path expressions: they are applied to address attributes or text data of those elements. YFilter proposed two alternative methods for value-based predicate processing: Inline and Selection Postponed (SP). With the first method, the value-based predicates are processed as soon as the relevant state is reached during structure matching. This approach requires predicates to be stored with their corresponding states. In the SP approach, only when an accepting state is reached all the value-based predicates are applied for the corresponding matched subscriptions. In this last case, predicates can be stored on a query-by-query basis.

A NOTIFICATION SERVICE ARCHITECTURE

This section provides a general overview of the notification service architecture that we propose (see Figure 3). The general architecture is depicted in figure.

The notification service architecture is based on the publish/subscribe system model and consists of four modules: Publisher, Subscriber, Filter Engine, Notification Manager. Events are related to information stored in the GridICE central database. Subscriptions, as described for publish/subscribe systems, contain user selection criteria. Notification messages about events satisfying one or more subscriptions related to a user, can be aggregated and a document containing details about events and matched subscription is produced. Such a document is named report.

Publisher Module

The purpose of this module is to provide events that drive the filtering algorithm as described in YFilter. According to the different levels of abstraction in a Grid system (e.g., VO level, Site level and operations domain level), several kind of XML views are periodically generated. Each XML view is an abstraction of the information present in the monitoring data collection. This is decomposed in a set of elementary XML documents that are sent to the filter engine.

Subscriber Module

This module is responsible for the management of the subscriptions and offers a web-based graphical user interface in order to enable *user registration* and *subscriptions definition*. During the user registration, information about user identity are gathered: unique ID and a default profile are assigned. For each subscriber, the service could aggregate information about events and matched subscriptions. The user profile contains the definition of the delivery frequency, that is the minimum interval between two notifications related to his subscriptions. The user is able to modify its profile. The service has no limit about the number of users supported.

The subscriptions definition is graphically driven and users have no view about the XPath representation of their subscriptions. Users are able to: (1) add and delete subscriptions: there is no limit about the number of subscription that can be defined; (2) choose to be notified with a single notification about a matched subscription: notification message contain only detailed report about events matching that subscription (such subscription has a customizable delivery frequency associated); and (3) stop multiple notifications about the same events by using an acknowledgement message.

The service notifies users about the change of state of one or more subscriptions; we say that a subscription change its state from unmatched to matched status when one or more events satisfying its selection criteria are generated. Vice versa, a subscription change its status from matched to unmatched when the current set of generated events does not contain anymore events satisfying its selection criteria. Notifications for the last kind of transition can be enabled by users that want to relate subscriptions to a critical state and enable notifications of restored states.

Filter Engine Module

The Filter Engine is based on YFilter implementation; its input are: (1) XML events generated by Publisher module and driving the filter mechanism (each event is processed only one time); (2) Subscriptions, expressed by XPath and used to build the NFA. The Filter Engine outputs a set of filtered events matching some subscriptions. Each events is linked to matching subscription(s) and subscriber identity.

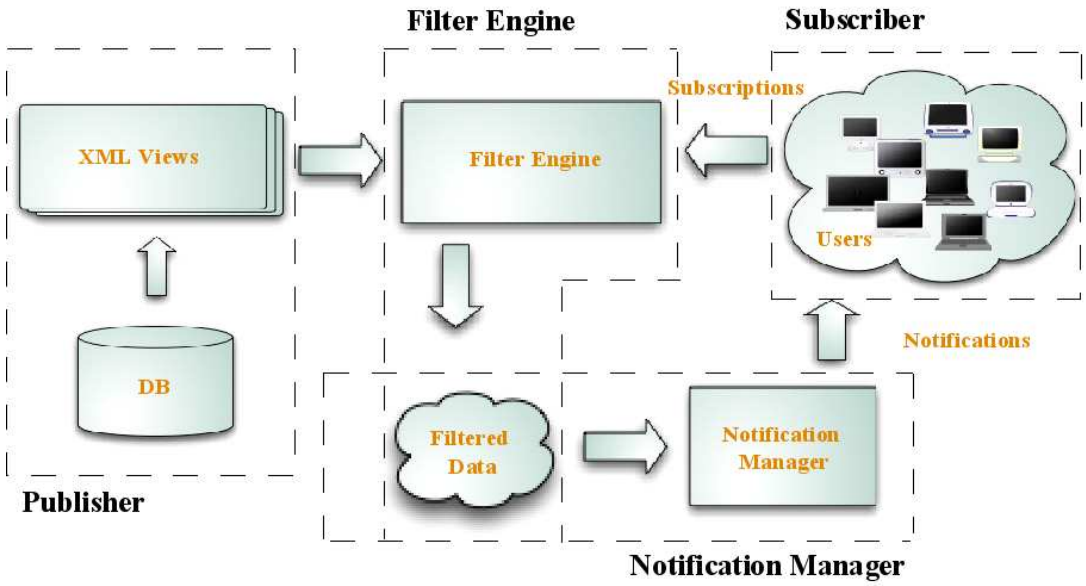

Figure 3: Architecture

Notification Manager Module

This module is responsible to send notifications to the subscribers. It is based on a synchronous process that operates on the set of Filtered Events from the Filter Engine module and (1) identifies involved users, (2) aggregates matched events and subscriptions, (3) composes notification reports accordingly to users profile and subscriptions properties, (4) sends notification messages to involved users.

DESIGN OF A PROTOTYPE IMPLEMENTATION

The design of the new notification service is modular and easy to integrate in the GridICE monitoring tool. In particular, recent evolution in the upper layers of GridICE envision the generation of XML-based views of monitoring data following the Presentation-Abstraction-Control design pattern [4]. This means that a small effort is needed in order to have the publisher service (see Figure 3).

The filter engine, the subscriber and the notification manager have to be developed. As regards the first one, a complete description of the algorithm is given in the YFilter literature. We plan to develop all these components ourselves.

The new notification service will interface with GridICE only in the XML views of data and in the presentation layer, for the management interface.

CONCLUSION

In this paper, we have described the design of a more flexible and scalable notification service for the GridICE Monitoring Service. The proposed work improves the current notification component by satisfying a wider number of identified requirements, adding new features and taking advantage of the rich semantic and structural information offered by XML data representation. It is important to underline that the proposed solution is suitable for all scenarios in which a message Broker plays a key role in the exchange of information (e.g., broker matchmaking and content-based routing).

REFERENCES

[1] S. Andreozzi, N. De Bortoli, S. Fantinel, A. Ghiselli, G.L. Rubini, G. Tortone, and C. Vistoli. GridICE: a Monitoring Service for Grid Systems. To appear in Future Generations Computer Systems (FGCS) Journal. Elsevier.

[2] S. Andreozzi. GLUE Schema Implementation for the LDAP Model. INFN Technical Report INFN/TC-04/16. 30 Sep 2004. http://www.lnf.infn.it/sis/preprint/pdf/INFN-TC-04-16.pdf

[3] GLUE Schema - Resources http://www.cnaf.infn.it/∼sergio/glue

[4] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. Pattern-Oriented Software Architecture, Volume 1: a System of Patterns. Wiley. 1996.

[5] Y. Diao, M. Altinel, M. J. Franklin, H. Zhang, and P. Fischer. Path sharing and predicate evaluation for high-performance XML filtering. ACM Transactions on Database Systems (TODS). Volume 28, Issue 4. December 2003.

[6] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid Information Services for Distributed Resource Sharing. Proceedings of the 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), San Francisco, CA, USA. Aug 2001.

[7] LHC Computing Grid Web Site. http://lcg.web.cern.ch/LCG/