

Development and use of MonALISA high level monitoring services for the STAR Unified Meta-Scheduler (SUMS).

STAR/ITD & US-CMS MonALISA team
May/June 2004

1. Introduction

While many success stories can be told as a result of Grid middleware developments, most of the existing systems relying on work-flow and job execution are based on integrated or self-contained production systems interfacing with a specific scheduling component or portal. This approach hardly satisfies the Grid ideal demanding the use or re-use of inter-operable components. In fact, such systems are often either non-Grid enabled (using "a" local resource manager system or LRMS) or fully-grid enabled, they are rarely, if ever, a hybrid solution taking advantage of both worlds. This is often due to the lack of a consistent set of monitoring information that is usable and shareable across heterogeneous resource management systems. Such information, and a consistent monitoring framework, is required to allow a meta-scheduler to evaluate the resources and negotiate with heterogeneous workload execution software, in order to utilize and schedule the appropriate resources to fulfil a high level user-described computing request.

We propose to evaluate, study, implement and demonstrate within the MonALISA monitoring framework and the STAR Unified Meta-Scheduler (SUMS), the usefulness of a kernel of tools allowing Meta-Schedulers to take advantage of a consistent set of information across RMS (local and/or Grid scheduler systems). We believe such development would highly benefit Grid laboratory efforts such as Grid3+ or the OpenScience Grid (OSG).

2. The STAR Scheduler

The SUMS project provides to the users of the STAR collaboration (and beyond¹) a way to submit jobs on a farm, at a site (multiple pools or farms) or to the Grid without the need to know or adapt to the diversity of technologies and knowledge involved while using multiple LRMS and their specificities². Additionally, the strategy was adopted to shield the users against changes in technologies inherent to the emerging Grid infrastructure and developments.

In its simplest form, SUMS presents itself to the user as a wrapper around evolving technologies on top of one or multiple queue systems negotiating between available queues and pools. To achieve this goal, the client wrapper interprets as input an XML describing the user's *intent* for accomplishing a task and its associated work-flow

-
- 1 The PHENIX collaboration has evaluated the STAR Meta-Scheduler for their needs. The Information and technology division (ITD) at BNL is deploying it on their cluster as a generic user submission tool.
 - 2 To submit a job to a queue system such as LSF, PBS, Condor, SGE, etc ... one has to either have their input to the queue system formatted in a certain manner or have detail knowledge of the queue, pool, priorities layout of the available LRMS.

rather than a traditional “shell script code” work-flow. The XML meta-job description follows a schema defined by a high level User Description Language or U-JDL³ and a single meta-job may be split into many jobs or sub-tasks which are then dispatched to diverse LRMS. The sub-division of a meta-job into sub-jobs depends entirely on the user's input and the external available resource criterion. Interfaced with the STAR Meta-data and Replica Catalog, such decisions are, in practice, made based on adaptive scheduling policies which may take into considerations conditions such as:

- the computing cycle availability across available resources (via delegation to the LRMS or based on a statically defined branching ratios for scheduling to different sites RMS or via Condor-G)
- the availability and knowledge of statically populated datasets across distributed pool (pinned datasets).
- the intended user access type and pattern for a given requested datasets: resources depend on the available access method for datasets from a given site, a given host, a given cluster.
- Resource estimator taken from user input or “guesstimate” (best use of the available queues or pools)
- ...

While simplistic, the initial implementation was planned to be enhanced by several additional modules aimed to provide to SUMS information such as the ones related to the available LRMS (status, availability, priorities, ...), available resource and foreseen resource availabilities (instantaneous or historical load information and later, possible use for predictions). In essence, such information is provided to SUMS as input necessary to make high level resource brokering decisions allowing, for example, a dynamic load balance of jobs between (and across) available sites⁴. The evolutionist architecture of the SUMS allows for an easy way to take advantage of existing information providers plugins (MDS, Ganglia, ML were evaluated). Such information will be gathered along experiment specific information (FileCatalog, MetaData or dataset and data collections information) into scheduling policies which encapsulate the logic by which the request will be satisfied. Additionally, new policies may be developed in parallel of production-level policies without disruptions of user's activities.

3. The MonALISA project

The MonALISA framework provides a distributed monitoring service system using JINI/JAVA and WSDL/SOAP technologies. Each MonALISA server acts as a dynamic service system and provides the functionality to be discovered and used by any other services or clients that require such information.

The goal of the project is to provide monitoring information from large and distributed systems to a set of loosely coupled "higher level services". This is part of a loosely coupled service architectural model to perform effective resource utilization in large, heterogeneous distributed centers. The framework can integrate existing monitoring tools and procedures to collect parameters describing computational nodes, applications

³ <http://www.star.bnl.gov/STAR/comp/Grid/scheduler/rdl/>

⁴ Usually, the distribution of jobs across sites is done via statically and pre-defined percentage of jobs to redirect to a given site.

and network performance. The scalability of the system derives from the use of a multi-threaded execution engine to host a variety of loosely-coupled self-describing dynamic services or agents, and the ability of each service to register itself and then to be discovered and used by other services, or clients that require such information.

In fact, an essential part of managing a global system, like the Grids, is a monitoring system that is able to monitor and track the many site facilities, networks, and the many task in progress, in real time. The monitoring information gathered also is essential for developing the required higher level services, and components of the Grid system that provide decision support, and eventually some degree of automated decisions, to help maintain and optimize work-flow through the Grid. MonALISA is an ensemble of autonomous multi-threaded, agent-based subsystems which are registered as dynamic services and are able to collaborate and cooperate in performing a wide range of monitoring tasks in large scale distributed applications, and to be discovered and used by other services or clients that require such information. MonALISA is designed to easily integrate existing monitoring tools and procedures and to provide this information in a dynamic, self describing way to any other services or clients. MonALISA services are organized in groups and this attribute is used for registration and discovery.

4. Queue Monitoring MonALISA Module

The STAR/ITD Grid team have developed a Queue Monitoring Module for the MonALISA monitoring Framework. This custom module provides aggregate status info for the most popular queuing systems (CONDOR, LSF, PBS, SGE) using the **same** attributes making the use of this information easy and self-contained into a defined schema. We initially aimed to make the provided info compatible with the [GLUE Schema](#). In particular, the Computing Element (CE) of the GLUE Schema represents the entry point to a Queue. There is one Computing Element per Queue.

In developing the ML Queue Monitoring Module, work done by others have been taken into account. In particular, the EDG have developed an MDS [information provider](#) that calculates the same attributes. In a way, our goal is to convert the MDS IP into a MonALISA Monitoring Module. See ⁽⁵⁾ for more information on this work.

The following table lists the attributes of the Status object of the CE Element:

Attribute	Description
RunningJobs	Number of currently running Jobs
WaitingJobs	Number of jobs that are in a state other than running
TotalJobs	Total Number of Jobs in the CE (Running + Waiting)
Status	States a Queue can be in (Production, Closed, Queuing, ...)
WorstResponseTime	Worst time between job submission till when job starts its execution (in secs)

5 <http://www.star.bnl.gov/STAR/comp/Grid/Monitoring/MonaLisa/MLQueueMon.html>

EstimatedResponse Time	Estimated time between job submission till when job starts its execution (in secs)
FreeCPUs	Number of Free CPUs available to the Scheduler

A presentation of the initial ideas on Queue Monitoring was given at the 2003 PPDG collaboration Meeting and is available [here](#).

5. Current state of development

As SUMS needs to reliably access to monitoring data for more complex decision making mechanisms, the MonALISA framework appeared to be a natural match for its primary goal. We are currently deploying three separate MonALISA monitoring services for the Grid needs of the STAR collaboration: one for the resources at the RHIC Computing Facility (RCF) at BNL, one at the Information Technology Division, also at BNL, and a third one at PDSF, at NERSC/LBNL. A separate ML group (“STAR”) has been setup by the ML developers and all three monitoring services join into it. The group automatically discovers the monitoring services that provide relevant monitoring data. A Web Repository has also been set up and it is currently being configured to provide real time and historical aggregate monitoring data for our group of monitoring services.

The average load per CPU per cluster is, for example, valuable input information that can be used to decide where jobs will be submitted and whether or not a request can be satisfied. Currently, the ML WS Client has been integrated into SUMS. SUMS can then pull data from individual ML monitoring sites that belong to the STAR/ITD group and have been configured to provide access to monitoring data as a Web Service. The STAR/ITD Grid team is however looking into having the integrated Client pull data from the Repository rather than each individual site for a better and more scalable solution.

6. Common project development Goals

The STAR/ITD and ML team therefore propose to work together on the elaboration and deployment of a set of high level services and solutions aimed to enhance scheduling capabilities of resource brokers, Grid enabled schedulers or Meta-Schedulers.

We propose to work along the following roadmap, the terminology of “client program” is meant to identify tools such SUMS.

- Integrate the Queue monitoring module into the MonALISA package for general use (other experiments and user community outside NP, Grid3+ exercise, ...)
- Define the requirements and schema by which one can consistently provide attributes for the most common batch systems (LSF, PBS, Condor, SGE). Those attribute may be outside of the scope of existing schema (GLUE). Attributes include information about the queue characteristics and access mode.
- Implement and integrate a solution for providing queue attributes for the most common batch systems in the ML framework
- Study and define the best approach (scalable and light weight) for accessing,

from a client perspective, the ML information. Particular attention will be given to the feasibility of accessing real-time and/or aggregate information and study of the delays inherent to each approach and solution. Large number of sites will be tested as part of a scalability study.

- The designed solution should be fault tolerant from the client program perspective. No single point of failures.
- Client program are envisioned to function in a non-centralized fashion (many user may execute “a” client session). Such architecture may raise issue and impose limitations as per the adopted solution.
- Implement and test the approach within the STAR Meta-Scheduler context
- Incorporate and support the solution in the ML framework.

7. Expected milestones

We plan and milestones are as follow:

- July 5th 2004: deploy and/or ensure consistent MonALISA monitoring services on STAR/ITD sites and configure STAR Web Repository.
- July 15th 2004: Complete development of Cluster and Queue Monitoring Modules and deploy them to ML service sites.
- August (mid) 2004: Integrate ML Web Service Client with SUMS. Test scalability and performance of various mechanisms accessing ML Web Services (individual monitoring services versus Web Repository service). Test integrity of provided data using different mechanisms.
- August (end) 2004: Deploy ML pseudo-client and run performance and scalability tests is accessing locally stored monitoring data (second technology choice). Develop and implement policies in SUMS that will use the retrieved monitoring data.
- September 2004: Investigate usage of provided real time or historical data and algorithm benefits. Develop matrix to monitor enhancement/benefits of approach.
- November 2004: deploy on Grid3+ test-bed (as available) and complete scalability testing.