computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

MONALISA: A MONITORING FRAMEWORK FOR LARGE SCALE COMPUTING SYSTEMS

Ciprian Dobre [1], Ramiro Voicu [2], Iosif C. Legrand [3]

[1] University POLITEHNICA of Bucharest, Spl. Independentei 313, Romania, ciprian.dobre@cs.pub.ro
[2] California Institute of Technology, Pasadena, CA 91125, USA, Ramiro.Voicu@cern.ch
[3] CERN, European Organization for Nuclear Research, CH-1211, Geneve 23, Switzerland, Iosif.Legrand@cern.ch

Abstract: *The MonALISA (Monitoring Agents in A Large Integrated Services Architecture) framework provides a set of distributed services for monitoring, control, management and global optimization for large scale distributed systems. It is based on an ensemble of autonomous, multi-threaded, agent-based subsystems which are registered as dynamic services. They can be automatically discovered and used by other services or clients. The distributed agents can collaborate and cooperate in performing a wide range of management, control and global optimization tasks (such as network monitoring, resource accounting) using real time monitoring information. MonALISA includes a coherent set of network management services to collect in near real-time information about the network topology, the main data flows, traffic volume and the quality of connectivity. A set of dedicated modules were developed in the MonALISA framework to periodically perform network measurements tests between all sites. We developed global services to present in near real-time the entire network topology used by a community. The time evolution of global network topology is shown in a dedicated GUI. Changes in the global topology at this level occur quite frequently and even small modifications in the connectivity map may significantly affect the network performance. The global topology graphs are correlated with active end-to-end network performance measurements, done using the Fast Data Transfer application, between all sites. Access to both real-time and historical data, as provided by MonALISA, is also important for developing services able to predict the usage pattern, to aid in efficiently allocating resources globally. For resource accounting, MonALISA collects information regarding the amounts of resources consumed by the users, which represent virtual organizations in a large scale distributed system. Besides providing statistical information, an accounting system can also be the base for managing distributed resources upon an economic model. In the MonALISA monitoring framework we developed modules that provide accounting facilities, collecting information from cluster managers like Condor, PBS, LSF and SGE. The usage statistics is used for an intelligent management of the resources.*

Keywords: *monitoring, large scale networks, topology, accounting, MonALISA.*

1. INTRODUCTION

An important part of managing global-scale systems is a monitoring system that is able to monitor and track in real time many site facilities, networks, and tasks in progress. The monitoring information gathered is essential for developing the required higher level services, the components that provide decision support and some degree of automated decisions and for maintaining and optimizing workflow in large scale distributed systems (LSDS). These management and global optimization functions are performed by higher level agent-based services.

MonALISA, which stands for **Monitoring Agents using a Large Integrated Services Architecture**, is a monitoring framework designed as an ensemble of dynamic services, able to collaborate and cooperate in performing a wide range of information gathering and processing tasks. Current applications of MonALISA's higher level services include resource accounting, optimized dynamic routing, control and optimization for data transfers, distributed job scheduling and automated management of remote services among a large set of distributed facilities. MonALISA is currently used around the clock in several major projects and has proven to be both highly scalable and reliable.

The main aim for developing the MonALISA system was to provide a flexible framework capable to use in near real-time the complete monitoring information from large number of jobs, computing facilities and wide area networks to control and optimize complex workflows in distributed systems.

Compared with other existing monitoring tools for LSDS, MonALISA is more generic and provides real-time, scalability, and dependability guarantees. Currently existing monitoring frameworks tend to be

too dedicated to specific activities. For example, Ganglia, and Lemon are mainly used to monitor computing clusters while tools like MRTG, PerfSonar, Nagios and Spectum are used to provide monitor information for Wide Area Networks. In MonALISA we provide the functionality to easily collect any type of information and in this way to offer a more synergetic approach, necessary to control and optimize the execution of data intensive applications on large scale distributed systems.

In this paper we present the system architecture and its applications to monitor and control real-world LSDSs. In particular, we present details for two important monitored services: the accounting and networking components. We acknowledge that an important part of managing any global-scale distributed systems is the monitoring system that should be able to monitor and track in real time many site facilities, networks, and tasks in progress. The monitoring information gathered is essential for developing the required higher level services, the components that provide decision support and some degree of automated decisions and for maintaining and optimizing workflow in LSDSs.

In LSDSs an accounting component is used to records the resource consumption for each user, and may have other functionalities like enabling the administration of the storage of this information, and interacting with other related services. One of the functions of the accounting system is to enable an economically self-sustained distributed system. Such a system should provide the possibility to charge the users for the resources consumed, or the possibility to trade resources among organizations. Another function of an accounting system is to provide statistical information that can be further used to develop intelligent algorithms for scheduling and resource management.

As the importance of the accounting component is widely recognized (i.e., as a fundamental pillar of Cloud Computing), several projects have been initiated in this domain. Still, there are significant challenges in developing an accounting system, related mostly to the complexity and heterogeneity of LSDS environments. We distinguish between accounting systems and monitoring systems that include accounting features [1]: while an accounting system stores detailed information about single jobs/users, and can provide usage records for a particular job, a monitoring system usually collects statistical information such as the total number of jobs run by each user, or per-VO resource consumption (*VO* stands for *virtual organization*).

In MonALISA we concentrated on the latter approach, and developed a set of dedicated monitoring and accounting modules for LSDSs. The accounting modules collect information from job managers such as Condor, PBS, LSF and SGE, and the accounting data is further stored in the MonALISA databases.

The monitoring framework has to intelligently collect, in a LSDS environment, a large number of monitoring events that are generated by the system components during the execution or interaction with external objects (such as users or processes). Monitoring such events is necessary for observing the run-time behavior of the large scale distributed system and for providing status information required for debugging, tuning and managing processes. However, correlated events are generated concurrently and can be distributed in various locations, which complicates the management decisions process.

To illustrate this, we also present a set of services developed in the context of the MonALISA framework for monitoring and controlling large scale networks, as an extension of the work previously presented in [2].

The rest of the paper is structured as follows. Section 2 presents the MonALISA monitoring framework. In Section 3 we present the monitoring services for large scale networks, together with solutions for the representation of network topologies at different OSI layers. This is followed by a real-world use-case for monitoring network topology in case of one of the largest network supporting the LHC experiments at CERN. Section 4 describes the accounting modules, and several results obtained using the modules.Finally, in Section 5 we give conclusions and present future work.

2. SYSTEM DESIGN

MonALISA (Monitoring Agents in A Large Integrated Services Architecture) [3,15] is a globally scalable framework of services jointly developed by California Institute of Technology (Caltech) and University Politehnica of Bucharest (UPB). MonALISA is currently used in several large scale High-Energy Physics communities and grid systems including CMS [4], ALICE [5], ATLAS [6], the Open Science Grid (OSG) [7], and the Russian LCG sites. It actively monitors the USLHCNet production network, as well as the UltraLight R&D network [4].

As of this writing, more than 360 MonALISA sites are being monitored 24/7 throughout the world. The services monitor more than 60,000 computing servers, and thousands of concurrent jobs. More than 3.5 million parameters are currently monitored in near-real time with an aggregate update rate of approximately 50,000 parameters per second.

The MonALISA system is designed as an ensemble of autonomous self-describing agent-based

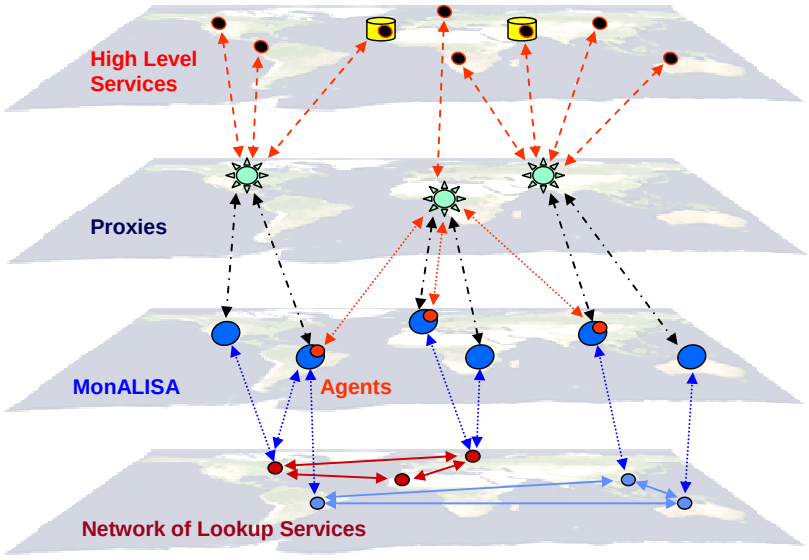

Fig. 1 – The four layers, main services and components of the MonALISA framework

subsystems which are registered as dynamic services. These services are able to collaborate and cooperate in performing a wide range of distributed information-gathering and processing tasks.

An agent-based architecture of this kind is well-adapted to the operation and management of large scale distributed systems, by providing global optimization services capable of orchestrating computing, storage and network resources to support complex workflows. By monitoring the state of LSDS-sites and their network connections end-to-end in real time, the MonALISA services are able to rapidly detect, help diagnose and in many cases mitigate problem conditions, thereby increasing the overall reliability and manageability of the distributed computing systems. The MonALISA architecture, presented in Fig. , is based on four layers of global services. The entire system is developed based on the Java technology.

The network of Lookup Discovery Services (LUS) provides dynamic registration and discovery for all other services and agents. MonALISA services are able to discover each other in the distributed environment, and be discovered by interested clients. The registration uses a lease mechanism. If a service fails to renew its lease, it is removed from the LUSs and a notification is sent to all services or applications that subscribed for such events. Remote event notification is used in this way to get a real overview of this dynamic system.

The second layer represents the network of MonALISA services that host many monitoring tasks through the use of a multithreaded execution engine. The network also hosts a variety of loosely coupled agents that analyse the collected information in real time. These agents are able to process the information locally, and to communicate with other services or agents to perform various global optimization tasks. A service in MonALISA is a component that interacts autonomously with other services, either through dynamic proxies or via agents that use self-describing protocols. By using the network of lookup services, a distributed services registry, and the discovery and notification mechanisms, the services are able to access each other seamlessly. The use of dynamic remote event subscription allows a service to register an interest in a selected set of event types, even in the absence of a notification provider at registration time.

On the third layer MonALISA hosts a series of Proxy services. The layer provides an intelligent multiplexing mechanism for the information requested by the clients or other services, and ensures a reliable communication between agents. It also provides an Access Control Enforcement layer to provide secures access to the collected information.

Higher level services and client access the collected information using the proxy layer of services. A load balancing mechanism is used to allocate these services dynamically to the best proxy service. The clients, other services or agents can get any real-time or historical data by using a predicate mechanism for requesting or subscribing to selected measured values. These predicates are based on regular expressions to match the attribute description of the measured values a client is interested in. They may also be used to impose additional conditions or constraints for selecting the values. The subscription requests create a dedicated priority queue for messages. The communication with the clients is served by a pool of threads. The allocated thread performs the matching tests for all the predicates submitted by a client with the monitoring values in the data flow. The same thread is responsible to send the selected results back to the client as compressed

serialized objects. Having an independent thread for clients allows sending the information they need, in a fast and reliable way, and it is not affected by communication errors which may occur with other clients. In case of communication problems these threads will try to re-establish the connection or to clean-up the subscriptions for a client or a service which is no longer active.

3. NETWORK MONITORING AND MANAGEMENT

A large set of MonALISA monitoring modules has been developed to collect specific network information or to interface it with existing monitoring tools, including: SNMP modules for passive traffic measurements and link status; Active network measurements using simple ping-like measurements; Tracepath-like measurements to generate the global topology of a wide area network; Interfaces with the well-known monitoring tools MRTG, RRD [8]; Available Bandwidth measurements using tools like pathload; Active bandwidth measurements using Fast Data Transfer (FDT) [9]; Dedicated modules for TL1 [10] interfaces with CIENA's CD/CIs, optical switches (Glimmerglass and Calient) and GMPLS controllers (Calient) [11, 12].

In the MonALISA framework the overall status of the dispersed systems being monitored is provided by either a GUI client or through specialized web portals. For the dedicated modules and agents used to monitor and control Optical Switches the GUI client of MonALISA provides a dedicated panel. This panel facilitates the interaction between users and the monitored Optical Switches. It offers to the end user a number of features such as complete perspective over the topology of the monitored optical networks or the possibility to monitor the state of the Optical Switches or the possibility to dynamically create new optical paths.

The tremendous interest in optical networks led the Internet Engineering Task Force (IETF) to investigate the use of Generalized MPLS (GMPLS) and related signaling protocols to set up and tear down lightpaths. GMPLS is an extension of MPLS that supports multiple types of switching, including switching based on wavelengths usually referred to as Multi-Protocol Lambda Switching (MPλS). In order to meet the expectations of future network technologies in the prototype system we made the first step towards integrating emerging light path technologies. We implemented the monitoring module and control agent that provide an interface between MonALISA and the Calient's GMPLS-based control plane.The described system, part of MonALISA, is currently used in production to monitor and control a CALIENT Optical Switch located at California Institute of

Technology in USA and another GLIMMERGLASS Optical Switch located at the European Center for Nuclear Research, in Switzerland. The dedicated monitoring modules use the TL1 language to communicate with the switch and they are used to collect specific monitoring information. The state of each link and any change in the system is reported to MonALISA agents. The system is integrated in a reliable and secure way with the end user applications and provides simple shell-like commands to map global connections and to create an optical path / tree on demand for any data transfer application. A schematic view of how the entire system works is shown in Figure 2.

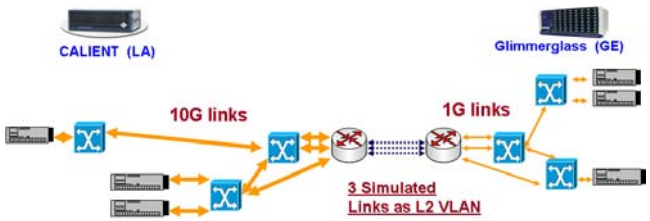

Fig. 2 – The system used to monitor and control Optical Switches and to create on demand optical path used in production

The implemented prototype system is able to create dynamically an end to end light path in less than one second independent of the number of switches involved and their location. It monitors and supervises all the created connections and is able to automatically generate an alternative path in case of connectivity errors. The alternative path is set up rapidly enough to avoid a TCP timeout, and thus to allow the transfer to continue uninterrupted.

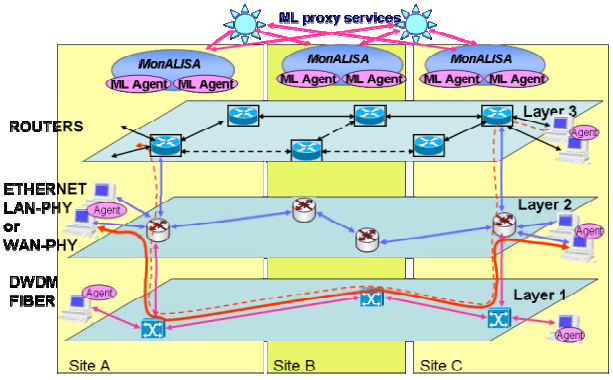

Fig. 3 – A schematic view of the functionality to provide dynamically an efficient end to end path to data intensive applications. The VINCI system is optimizing the path allocation using as much as possible Layer 1 or Layer 2 segments

To satisfy the demands of data intensive applications it is necessary to move to far more synergetic relationships between applications and networks. Currently, even the most complex

scientific applications are simply passive users of the existing network infrastructure. The main objective of the VINCI (Virtual Intelligent Networks for Computing Infrastructures) project is to enable users' applications, at the LHC and in other fields of data-intensive science, to effectively use and coordinate shared, hybrid network resources, to correlate them with available processing power in order to dynamically generate optimized workflows in complex distributed system (Figure 3).

VINCI is a multi-agent system for secure light path provisioning based on dynamic discovery of the topology in distributed networks. For this project we are working to provide integrated network services capable to efficiently use and coordinate shared, hybrid networks and to improve the performance and throughput for data intensive applications. This includes services able to dynamically configure routers and to aggregate local traffic on dynamically created optical connections.

The system dynamically estimates and monitors the achievable performance along a set of candidate (shared or dedicated) network paths, and correlates these results with the CPU power and storage available at various sites, to generate optimized workflows for LSDS tasks. The VINCI system is implemented as a dynamic set of collaborating Agents in the MonALISA framework, exploiting MonALISA's ability to access and analyze in-depth monitoring information from a large number of network links and LSDS sites in real-time.

3.1. MONITORING AND REPRESENTATION OF NETWORK TOPOLOGIES AT DIFFERENT OSI LAYERS

We present monitoring and representational services developed considering various network topologies and the differences posed by network equipments operating at various OSI levels. In large-scale networks, such as USLHCNet and UltraLight, we found devices at ever OSI layer.

A. The Physical Network Layer Topology

A set of specialized TL1 modules are used to monitor optical switches (Layer 1 devices) from two major vendors: Glimmerglass and Calient. We were able to monitor the optical power on ports and the state of the cross-connects inside these switches.

Based on the monitoring information an agent is able to detect and to take informed decisions in case of eventual problems with the cross connections inside the switch or loss of light on the connections. The MonALISA framework allows one to securely configure many such devices from a single GUI, to see the state of each link in real time, and to have historical plots for the state and activity on each link. It also allows to easily manually create a path using the GUI. In Figure 4 we present the MonALISA GUI that is used to monitor the topology on the Layer 1 connections and the state and optical power of the links. The same GUI can be used to request an optical path between any two points in the topology. All the topology related information are kept distributed, every MonALISA service having its own view of the network. Every agent computes a shortest path tree based on Dijkstra's algorithm. The convergence in case of problem is very fast, as every agent has the view of the entire topology.

B. Layer 2 Network Topology / Circuits

The USLHCNet transatlantic network has evolved from DOE-funded support and management of international networking between the US and CERN. USLHCNet today consists of a backbone of eight 10 Gbps links interconnecting CERN, MANLAN in New York, and Starlight in Chicago. The core of the USLHCNet network is based on Ciena Core Director CD/CI multiplexers which

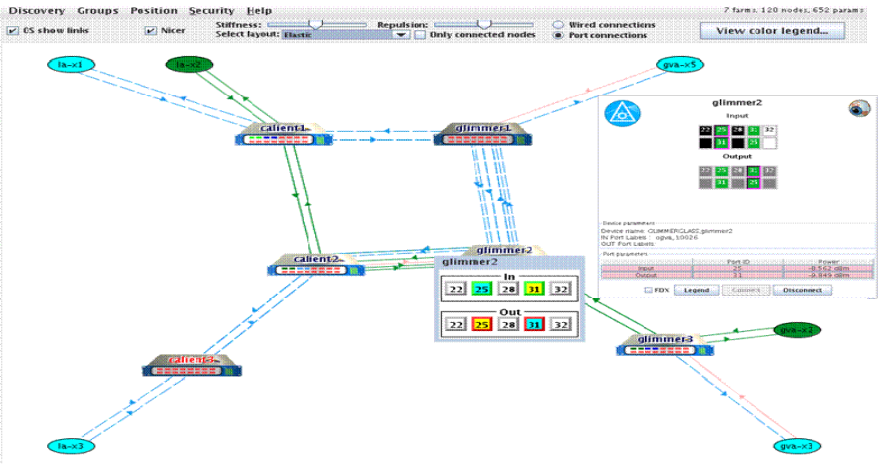

Fig. 4 – Layer 1 topology: Monitoring and autonomous controlling optical switches

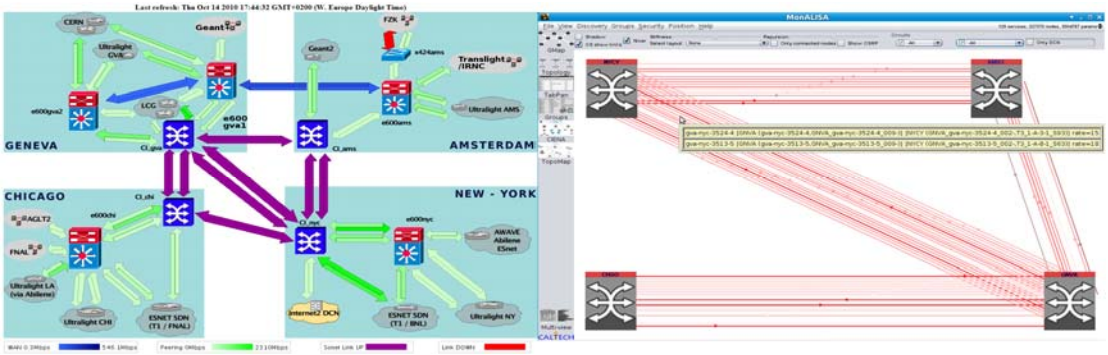

Fig. 5 – A network weathermap (left) and the layer 2 topology for the dynamic circuits (right)

provide stable fallback in case of link outages at Layer 1 (the optical layer), and full support for the GFP/VCAT/LCAS [13] protocol suite.

For the Core Director (CD/CI) we developed modules which monitor the routing protocol (OSRP) which allows us to reconstruct the topology inside the agents, the circuits (VCGs), the state of cross connects, the Ethernet (ETTP/EFLOW) traffic, the allocated time slots on the SONET interfaces and the alarms raised by the CD/CI (see Figure 5).

The operational status for the Force10 ports and all the Ciena CD/CI alarms are recorded by the MonALISA services. They are analyzed and corresponding email notifications can be generated based on different error conditions. We also monitor the services used to collect monitoring information. A global repository for all these alarms is available on the MonALISA servers, which allows one to select and sort the alarms based on different conditions. The link status information is very sensitive information for the SLA (Service Level Agreement) with both the experiments and the link providers. Because of this very strict monitoring requirement the monitoring had to have almost 100% availability. We achieved this monitoring each link at both ends from two different points. The NOCs in Europe, Geneva and Amsterdam, are cross-monitored from both locations, and the same in US. In this way we monitor each link in four points and with special filters this information is directly aggregated in the repository. For redundancy and reliable monitoring we keep at least two instances of repositories running, one in Europe and one in US. For the past two years we manage to have 100% monitoring availability inside USLHCNet.

C. Layer 3 Routed Network Topology

For the routed networks, MonALISA is able to construct the overall topology of a complex wide area network, based on the delay on each network segment determined by tracepath-like measurements from each site to all other sites, as illustrated in Figure 6.

For any LHC experiment such a network topology includes several hundred of routers and tens of Autonomous Systems. Any changes in the global topology are recorded and this information can be easily correlated with traffic patterns. The time evolution of global network topology is shown a dedicated GUI. Changes in the global topology at this level occur quite frequently and even small modifications in the connectivity map may significantly affect the network performance.

3.2. A REAL USE-CASE FOR TOPOLOGY INFORMATION

The Alice Grid infrastructure uses MonALISA framework for both monitoring and controlling. All the resources used by AliEn [14] services: computing and storage resources, central services, networks, jobs are monitored by MonALISA services at every site.

A. Bandwidth measurements between Alice sites

The data transfer service is used by the ALICE experiment to perform bandwidth measurements between all sites, by instructing pairs of site MonALISA instances to perform FDT memory-to-memory data transfers with one or more TCP streams.

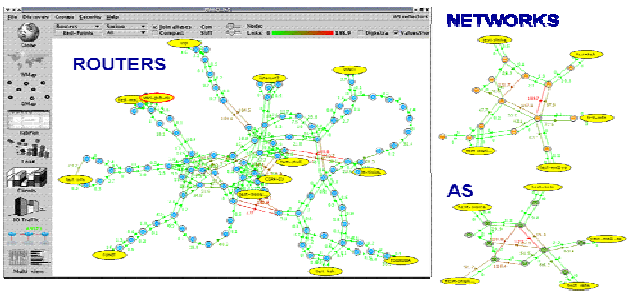

Fig. 6 – MonALISA real time view of a topology. A view of all the routers, or just the network or "autonomous system" identifiers can be shown

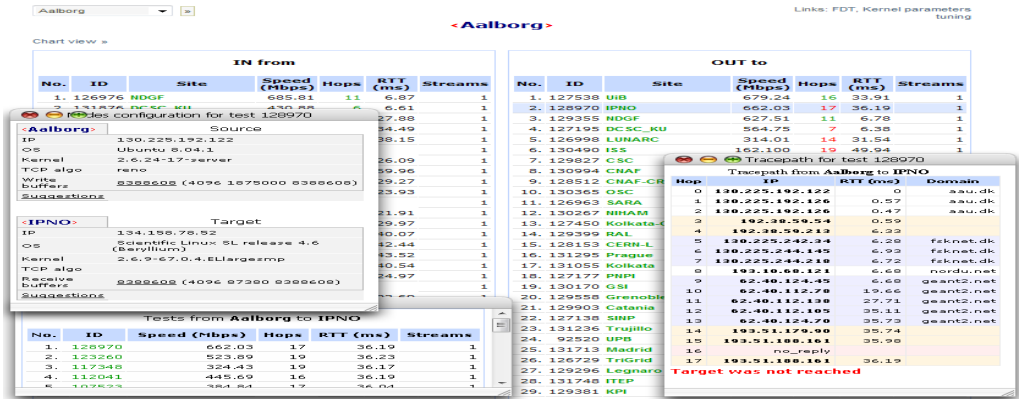

Fig. 7 – Inter-site bandwidth test results. Tracepath is also recorded.

The results are used for detecting network or configuration problems, since with each test the relevant system configuration and the tracepath between the two hosts are recorded as well. The MonALISA services are also used to monitor the end system configuration and automatically notify the user when these systems are not properly configured to support effective data transfers in WAN. In Figure 7 we present the results recorded from one site to all the others.

B. Automatic storage discovery for Alice

Using the monitoring information from trace-like measurements, derived information is computed in the repository, associating the Autonomous System (AS) number to each of the nodes in a network path. The repository also runs other monitoring modules that provide global values and one of them periodically queries AliEn for the list of defined storage elements and their size and usage according to the file catalog. Then periodic functional tests are performed from the central machine to check whether the basic file operations (add, get, remove) are successful. The entire software and network stacks are checked through these tests, thus the outcome should be identical for all clients trying to access the storages.

Aggregating the monitoring and test results, a client-to-storage distance metric is computed and used to sort the list of available storage elements to a particular client. Then the closest working storage elements is selected either to save the data or, in case of reading, sorting the available locations based on this metric, trying to read from the closest location. The algorithm associates to each storage element a list of IP addresses representing known machines from its vicinity.

C. Monitoring modules for dynamic light path provisioning

Given the monitoring part, we present solutions adopted for dynamic and automatic provision of light path based on the monitoring information. For that MonALISA has two monitoring modules that provide information about the optical power on ports and the state of the cross-connect links inside the switch in near real-time. The modules use Transaction Language 1 (TL1) commands to retrieve monitoring information from the optical switch. Based on the monitoring information the agent is able to detect and to take informed decisions in case of eventual problems with the cross connections inside the switch or loss of light on the connections.

For control the Optical Switch Agent is a software agent that is dynamically deployed and runs embedded in a MonALISA service. Its role is to monitor and control an optical switch, to publish and to continuously update its configuration. The configuration consists of the port map, which specifies the devices attached to the switch, state of the ports, optical cross-connects inside the switch and the necessary routing information. The agents use the MonALISA framework to discover each other, publish their configuration, and collaborate to create on-demand and end-to-end optical paths.

The algorithm for dynamic path provisioning is able to establish an end-to-end connection in the shortest possible time. In order to achieve this, every agent in the system has the exact view of the network and adapts very quickly to changes, using the previously described solution. The network topology, implemented as a network graph, has agents as vertices and optical links between switches as edges, every edge having a cost associated with it. The system is modeled using a directed graph. Such an approach makes it possible to have both full-duplex and simplex links between optical switches. Each agent in the graph computes a shortest path tree using a variant of Dijkstra's algorithm. The agents system uses a two-phase commit strategy for creating the optical cross-connects and a lease mechanism to guarantee the reliability in case of partial failures.

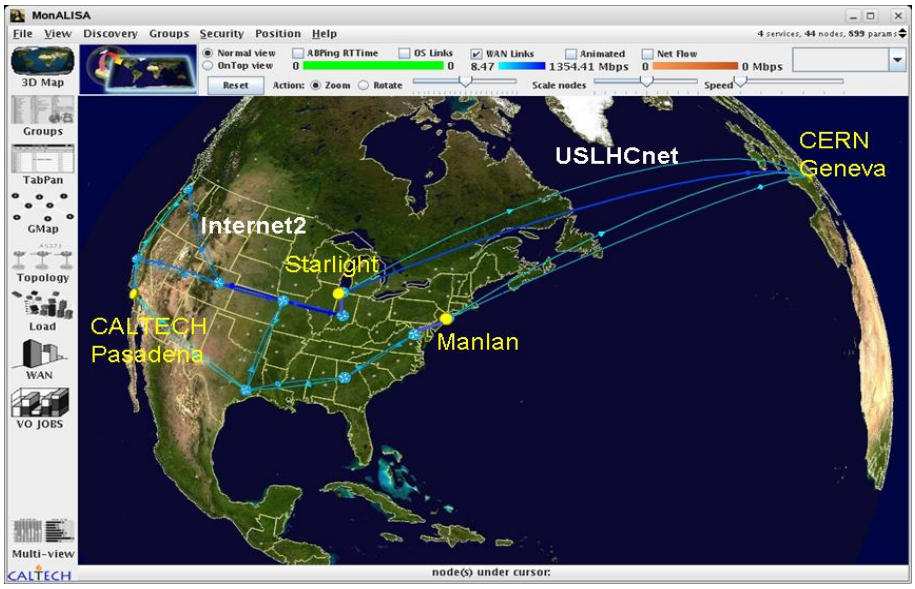

Fig. 8 – The network topology used for creating dynamically, on demand an end to end optical path

An agent that receives a lightpath request determines, based on the local tree that is already built, if the request can be fulfilled or not. If it is possible it also initiates transactions with both the local and remote ports involved in the path. Once the transaction is started, the agent assigns a unique ID for the path, sends the remote cross-connect commands and after that it tries to establish the cross-connects on the local switch. An independent thread is waiting for acknowledgements from the remote agents. Any remote agent which receives such a cross-connect request starts a local transaction only with the ports involved in the cross-connect. If it succeeds in creating the cross-connect, it commits the local transaction and it sends back an "acknowledged" message, otherwise the transaction is rolled-back and a "not acknowledged" message is sent. Based on the received messages the local agent takes the decision whether or not the transaction can be committed or it has to be rolled back. The algorithm described above is reliable and guarantees that the system remains in a consistent state even if a network problem occurs. The newly created lightpath has a lease assigned which must be renewed by all the involved agents and in this way it can provide a viable mechanism for the system to recover from partial failures.

To improve the performance and the response time all the functions executed by an agent are performed in asynchronous sessions using a pool of threads. A task can be a request for a lightpath from a client, or a cross connect request coming from another agent, or a rerouting task triggered by a topology change. The only sequential part of the algorithm described above is in the "pre-commit" phase, and this involves only the ports that are supposed to be in the lightpath. Any request submitted during this phase, which do not involve these ports can be fulfilled in parallel.

Using this information, an agent is able to detect the loss-of-light on fiber, and take specific decisions if the port is part of a lightpath. The agent who detects the problem notifies the initiator, which is responsible to try to reroute the traffic through another path, if this is possible. When the initiator detects a change in topology that affects the lightpath, it forces the shortest path tree to be recalculated. Based on the new tree, the agent is able to take the decision if the light can be rerouted using other path, or it can tear down the entire path. This is very useful, because in case of successful rerouting, the problem will not disturb already established sockets, upper network layers, like TCP, not being able to detect the problem.

The routing algorithm used to establish an end to end lightpath is similar with link-state routing protocols. The work presented here uses an algorithm similar to link-state routing algorithms because they converge faster than distance-vector algorithms. In order to guarantee consistency and reliability of the entire system, a two-phase commit strategy and a lease mechanism were also developed.

We developed dedicated modules for several types of optical switches. The system is currently used to create dynamically on demand optical connections between computers located at CERN (Geneva) and California Institute of Technology (CALTECH) located in Pasadena, CA, using the networking infrastructure of USLHCNet and Internet2 [16, 17].

USLHCNet provides two transatlantic 10 Gb/s optical links between CERN and Starlight (Chicago) and MANLAN (New York). On the Internet2 network, the pure optical connections are simulated using several VLANs to provide direct connections from Chicago and New York to CALTECH. The topology of the network infrastructure used is shown in Figure 8.

The system is able to create dynamically an end to end lightpath in less than one second independent of the number of switches involved and their location. It monitors and supervises all the created connections and is able to automatically generate an alternative path in case of connectivity errors. The alternative path is set up rapidly enough to avoid a TCP timeout, and thus to allow the transfer to continue uninterrupted.

The optical fibers are simulated through two VLANs between the two optical switches. One VLAN is routed through New York and the other one through Chicago. The monitoring module is able to simulate a fiber cut. The optical agent will detect this as a real loss-of-light and will try to reroute the path.

In the example above a disk to disk transfer is presented, using two 4-disk servers, one at Caltech and the other one at CERN in Geneva. During the transfer four fiber cuts were simulated, corresponding to the four drops in the Figure 9.

These fibers cuts simulations were done on the Geneva – Starlight and Geneva – Manlan links and the transfer was rerouted four times between these two links. The "fiber cut" and the reroute are done quick enough that the TCP does not sense the loss in connectivity and the transfer continues. The recovery time differs for various TCP stacks and the round trip time between end points.

Fig. 9 – The total disk to disk throughput between a server at CERN and one at CALTECH. Four "fiber cuts" were simulated during the transfer. The throughput drops when a rerouting is done, but it recovers quickly

In the MonALISA framework the overall status of the dispersed systems being monitored is provided by either a GUI client or through specialized web portals. For the dedicated modules and agents used to monitor and control Optical Switches the GUI client of MonALISA provides a dedicated panel. This panel facilitates the interaction between users and the monitored Optical Switches. It offers to the end user a number of features such as complete perspective over the topology of the monitored optical networks or the possibility to monitor the state of the Optical Switches or the possibility to dynamically create new optical paths.

The main panel is presented in Figure 10. The main aspect of this panel is that it displays in an intuitive way the current topology of the monitored Optical Switches and the links between. For the Optical Switches we use different colors to represent the state of their internal ports and the state of the links between the represented entities. In the panel, besides the Optical Switches a number of other devices (the blue ovals) can also be represented. These devices, equipped with optical network cards, are connected by optical links to the Optical Switches being monitored.

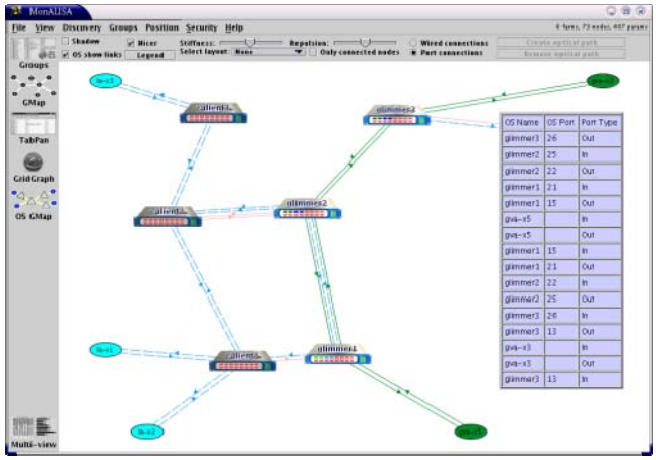

Fig. 10 – Same topology and status on the 3D Map panel

The Optical Switches and the links connecting them are also represented in the 3D Map panel (see Figure 8). This panel locates the MonALISA monitoring services on a 3D view of the world geographical map. It also shows the monitoring WAN links, real-time traffic on them, the capacity of the links, the connectivity between sites, the optical switches controlled and other parameters like sites Load, CPU usage, IO parameters, etc. In this way the user is presented with an easy to use complete visualization tool which represents the global state of the entire monitored systems.

For the optical paths created from this panel (represented in green color in Figure 4) the user is presented with the details of which devices and ports one particular path is crossing (from end to end) together with the components involved in that path (the highlighted devices in Figure 4).

The panel can be also used to create new optical

paths or delete existing ones. In order to gain access to these operations the user must present though the necessary credentials. The security layer is implemented using RMI over SSL.

4. MONITORING ALICE DISTRIBUTED COMPUTING ENVIRONMENT

One of the major challenges in designing the monitoring and accounting system in case of LSDS is the heterogeneity of the sites. They may use various resource managers, such as Condor, PBS, Sun Grid Engine, LSF, etc. Such resource managers provide accounting information, but the set of provided parameters (and their measurements units for that matter) differ from one manager to another. While some resource managers record detailed information (such as the amount of memory and disk space used by the jobs, and the amount of network traffic they produce), others only record basic information (such as runtime and CPU time consumed). Therefore, the common set of parameters that can be collected depends on the set of available resource managers.

Another challenge encountered when defining the format of a resource usage record is that there are some differences between LSDS jobs (e.g., between Grid jobs and batch jobs). Therefore, a decision has to be taken regarding which one of the job models should be used for accounting information. For example, when recording the start time for a Grid job, this may have different meanings: the time when the job got submitted to the Grid broker, the time when it entered the batch system queue, or the time when it actually started executing on a computing node. Though the last variant is usually considered at this moment (corresponding to a batch job model), it might be better to use the Grid job model and consider the time needed for brokering, data staging, etc.

There are two ways in which an accounting system can collect the information: 1) in a real-time mode, while the jobs are being executed, or 2) by gathering data from the resource manager logs, after a job completes. The second approach has the advantages of simplicity, and provides more information – so it is sufficient for the accounting purposes. However, a monitoring system is not of much help if it provides information about a job only after it is finished. Users need real-time information, so that they know the status of their jobs while they are being executed – especially since a job can typically run for several hours. To obtain real-time information, a resource manager command that provides the jobs' status must be run periodically, and its output interpreted. This approach has some disadvantages: a negative impact on performance (the commands sometimes take a long time to run, especially when there are thousands of jobs running on the site), and the format of the commands output can change from one version of the resource manager to another. The accounting modules implemented in MonALISA combine both the real-time and the log-based approaches, to deliver accurate information regarding the resource usage.

Resource manager failures are another important aspect that must be addressed by an accounting system. When, for various reasons, the resource manager fails to respond and the accounting system cannot obtain information, the situation should not be interpreted as zero resource usage. There are cases when, even though the queries to the resource manager fail, the jobs are still running correct.

4.1. COLLECTING ACCOUNTING INFORMATION WITH MONALISA

MonALISA provides distributed registration and discovery for services and applications, secured remote administration, and also an agent execution framework that is used to: supervise applications, restart or reconfigure them, and notify other services when certain conditions are detected. Information is provided for: system information for computer nodes and clusters, network information (traffic, flows, connectivity, topology) for WAN and LAN, information regarding the performance of Applications, Jobs or services, and end to end performance measurements. Users may access real time information using a graphical user interfaces, developed with the Java WebStart technology. They can also access history information stored in MonALISA repositories [15].

The accounting modules are a part of the set of MonALISA's monitoring modules, and have the role of collecting information from various available resource managers. These modules are able to work with Condor, PBS, LSF and SGE; if there are multiple queue managers in a cluster, the values obtained from them are combined.

Condor is a workload management system specialized for compute intensive jobs. One of its advantages over other batch queuing systems is the ability to perform opportunistic computing (harnessing CPU power from idle desktop workstations). Also, its ClassAds is a flexible mechanism for matching resource requests with resource offers. PBS, which stands for Portable Batch System, is based on the client-server model, with a client making job execution requests to a batch server and the server handling the execution of the jobs in a cluster by placing them in queues. Among the features that PBS provides are the possibility to set priorities for jobs and to specify

interdependencies between them, automatic file staging and multiple scheduling algorithms. SGE (Sun Grid Engine) is a resource management system able to schedule the allocation of various distributed resources, like processors, memory, disk space, and software licenses. Its features include resource reservation, job checkpointing, the implementation of the DRMAA job API and multiple scheduling algorithms. LSF (Load Sharing Facility), a commercial resource management system, has as its core the Platform Enterprise Grid Orchestrator (EGO), which by virtualization and automation provides a way to orchestrate all the enterprise applications into a single cohesive system.

Table 1. Resource managers and their metrics.

Job manager	CPU Time	Run Time	Job Size	Disk Space
CON	Yes	Yes	Yes	Yes
PBS	Yes	Yes	No	No
SGE	Yes	No	Yes	No
LSF	Yes	Yes	Yes	No

The monitoring modules parse the output of specific commands that provide information about the current jobs' status, and provide summarized results for each job. In Table 1 we present the metrics used for each resource manager (where CON means Condor). In case of Condor and PBS, the log files are also considered to obtain additional information. The single-job results are then added to the statistics made per user and per VO; the association between the Unix account from which a job is run and the VO is done on the base of a map file which specifies the corresponding VO for each account.

There are two categories of VO parameters provided by the monitoring module: parameters that represent values obtained in the last time interval (between the previous run of the module and the current one), and parameters that represent rates (calculated as the difference between the current value of a parameter and the value obtained at the previous run, divided by the length of the time interval between runs). Among the parameters in the first category are the number of running/idle/held jobs, the number of submitted and finished jobs, the CPU time consumed, and the total size of the jobs. The parameters in the second category are the rates of submitted jobs, finished jobs, CPU time and wall clock time. The values of these parameters can be viewed with the aid of the MonALISA graphical client, and can also be retrieved by accessing the MonALISA web service. However, a MonALISA service only stores the parameter values for a limited amount of time. For longer periods of time, the values are stored in MonALISA repositories.

As mentioned above, for Condor and PBS the modules can be configured to collect information from the history logs, besides running the job manager commands. Even though this information is not useful for real time monitoring (because the record for a job is written to the log only when the job is finished), there are several reasons why the log information is helpful:

- the log file usually contains the exit status for jobs; knowing whether the jobs were finished successfully is important both for users and site administrators
- there may be some very short jobs which start and end between two consecutive runs of the module; by examining the logs, we can add these jobs to the VO statistics
- with the aid of the logs we can double-check the values for CPU time and runtime that the module collected (the value from the log should be greater than or equal to the value obtained in the last run of the module).

A. Collecting Information from Remote Sites

Normally, the accounting modules collect the information from the local cluster on which MonALISA is running. However, in some situations it is desired to obtain accounting information from remote sites. There are two possibilities of doing that. First, we use the job manager's features. For example, Condor provides an option to query the jobs' status from a remote pool. The MonALISA module can be configured to use this option and to collect information from a remote site (and also from multiple sites). The disadvantage is that in this way, we cannot benefit of the history information as the log files are not on the local machine anymore.

Another possibility to obtain information from remote sites is to run the job manager commands through a SSH connection on the remote hosts. This can be done with any job manager, but it is necessary to configure the remote hosts so that a SSH connection can be opened with the public key authentication method. Another disadvantage is the overhead caused by the SSH communication, especially when there are problems with the network connectivity. This solution has been implemented in a version of the Grid modules that is used in the SEE-Grid project, on LCG middleware.

B. Failure Handling

As mentioned, the accounting system must consider situations when the resource manager fails and stops providing information. A distinction should be made between the case when the manager returns an error message, the case when there is no answer from the job manager, and the case when the job manager works correctly, but there are no

running jobs. Otherwise, the accounting system may mistakenly report zero current jobs while there actually are jobs still executing. To avoid this situation, we introduced a set of error codes for the accounting module. When the job manager command returns with error or it fails to answer, we set the appropriate error code, which is also visible from the graphical client.

Another type of failure is the one that affects a single job. Sometimes the resource manager may decide to restart the failed job, and the CPU time counter of the job is reset to zero. We took this case into consideration and the accounting module can be configured for one of the following behaviours: adding the old value of the CPU counter to the VO statistic or neglecting the old value.

C. Processing and Storing Accounting Information in the MonALISA Repositories

MonALISA provides an easy mechanism to create clients able to use the discovery mechanism in JINI and to find all the active services running in a set of targeted communities (groups). Such clients can subscribe to a set of parameters or filter agents to receive selected information from all the services. This offers the possibility to present global views from the dynamic set of services running in a distributed environment to higher level services.

The received values are further stored locally into a relational database, optimized for space and time. The collected monitoring information is further used to create web repositories able to present a synthetic view of how large distributed systems perform. The system allows the development of the required higher level services and components necessary to provide decision support, and eventually some degree of automated decisions, and to help maintain and optimize work-flows through the LSDS.

The repository registers with a set of predicates and stores the received values in the local database. A predicate has the following pattern: *Farm* / *Cluster* / *Node* / *start_time* / *end_time* / *function_list*. These parameters can be dynamically plotted into a large variety of graphical charts, statistics tables, and interactive map views, following the configuration files describing the needed views, and thus offering customized global or specific perspectives. The same mechanism is used to offer access to this information from mobile phones using the Wireless Access Protocol (WAP). The WSDL/SOAP interface is also available so that clients can access information received from several farms in LSDS.

The main components of the repository system are the storage client, responsible for data collection and storage, and the servlet engine, which ensures the translation of user's customized requests from the interface into appropriate queries for the storage client (according to the predicate pattern previously presented). Furthermore, the repository can plot the results in a flexible manner, according to properties set in configuration files. Each chart has a corresponding configuration file with a simple structure that ensures flexibility. Consequently, site administrators can specify custom properties of the plot: the type (bar, series, spider, double axis series, pie, histogram, table, interactive map etc.), the information to be displayed – the predicate(s), the time interval (real time information or maximum length of the history interval), the metrics, the scale, statistics generation, graphical enhancements (series colours, size, names etc.). The same configuration file is used to specify which of these options should be accessible to users from the web interface and the options default values.

This feature stresses two levels of customization permitted by the repository system: user level, at which the user can customize a default view through the interface, and site administration level, at which the super user can decide which monitoring information is made public and how this information is displayed. Furthermore, the flexibility at this level can be increased through integration of new filters and servlets, written by site's administrators and performing specific tasks for the targeted community. We developed such filters for accounting resource usage in several repositories[1]. The MonALISA repository is a Web client that is able to present a synthetic view of how large distributed systems perform. A servlet engine is used to present historical and real time values, statistics and graphical charts in a flexible way. For that, a dedicated module adds a new level of aggregation to raw data stored in the database. The suitable aggregation method and parameters can then be selected from the interface or from the configuration files: sum, minimum/maximum, average, integration over a specified interval, etc.

An important issue in resource usage accounting is the length of the monitored time interval. Usually, this time frame spans over a period of a few years which in conjunction with high parameter collection rates (~2700 parameters/minute for an average Grid/VO community) results in large amounts of monitored data. Such factor size for the repository database (~ 250 GB the average per VO) raises space and access time issues. We therefore optimized the system to achieve consistency, fault tolerance and reliable response times.

First, we allow a precise selection of relevant collected data, as not all received information

[1] For a collection of currently available MonALISA monitoring repositories the user can also consult http://monalisa.cacr.caltech.edu/monalisa__Repositories.htm.

presents interest for a certain community. The system administrator can choose from repository's configuration file which received predicates should be stored in the database, and which should not, and which should only be stored in a temporary memory buffer, but not written to disk (ex: real-time information without history relevance). In the same concern for the database's proportions, the time frame covered is adjustable using a sliding window mechanism.

Also, performing accounting queries for long history periods is a time consuming process. Therefore, an optimized data storage model was needed. We used multiple parameters / series tables for a fast access. Additionally, we designed tables with multiple sampling intervals, so different queries may be served from the appropriate sampling table. For instance, an accounting query for a short history period (1 day, 1 week, 1 month) will use a table with a higher resolution (ex: 1 minute sampling), while queries for long periods (6 months, 1 year) use lower resolution tables (ex: 15 minutes sampling). Also, the sampling method can be customized: write average values for the sampling interval or write directly received data. Further, we use an adaptive memory buffer in order to speed up the response.

Besides being written to the database, the accounting monitoring information is stored in a buffer, so recent data (used for real-time charts) can be quickly accessible without time consuming readings from the disk. The size of this buffer can be set by the system administrator, otherwise is automatically adjusted according to existing memory resources. The time frame of the buffer usually spans from a few hours to a few days according to its memory size and received results rate (~3 hours and ~1.000.000 results for an average-size repository). Besides this buffer, a data cache is used, indexed after servlet queries, which stores the most requested queries and their responses. We designed this complementary cache observing that there are certain requests with the same parameters (all information in the configuration files is translated in servlet parameters), returning the same charts, so a new plot of the same chart was both time consuming and unnecessary.

4.2. CASE STUDY: USING MONALISA FOR ACCOUNTING

The Open Science Grid (OSG) is a consortium formed as a continuation of the Grid3 project, with the purpose of enabling multiple scientist communities to access a common Grid infrastructure. The infrastructure is administered by a set of U.S. universities and national laboratories. One of the main project domains in OSG is nuclear physics, as many of the current OSG applications regard the experiments at the Large Hadron Collider from CERN, Switzerland. Other projects developed within OSG are in astrophysics, biology and gravitational-wave science. The OSG includes an Integration Grid, used for testing of new technologies and applications, and Production Grid, which is a stable environment for executing applications. The OSG middleware is packaged with the Virtual Data Toolkit (VDT), including Globus and Condor as main components, and also the MonALISA framework.

We have been monitoring the OSG group using a global repository for several years, tracking 155.000 parameters from 54 deployed MonALISA services on 26.300 nodes (see Figure 11). The number of finished jobs in this interval was ~ 9.000.000 in 50 Virtual Organizations. The repository has been serving ~ 2.100.000 requests at an average rate of 120 requests/hour, with peaks of 1.500 requests/hour. The average collection rate is ~2.300 results/minute.

Fig. 11 – The OSG Repository

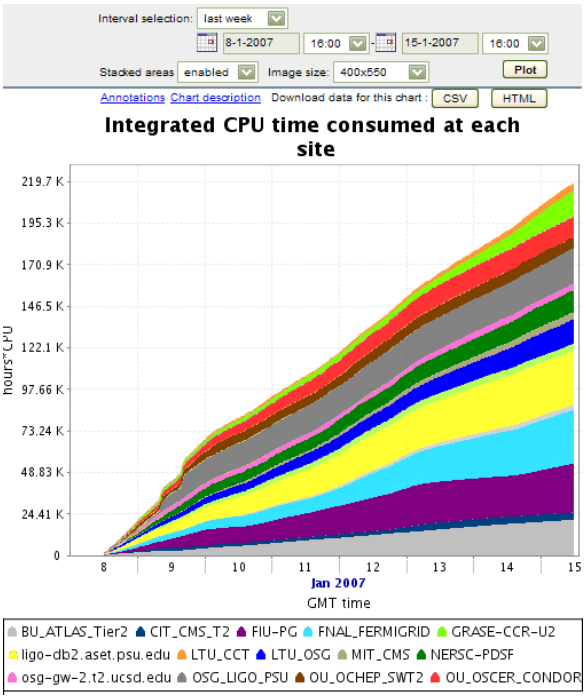

Fig. 12 – Integrated CPU time statistic from the MonALISA repository

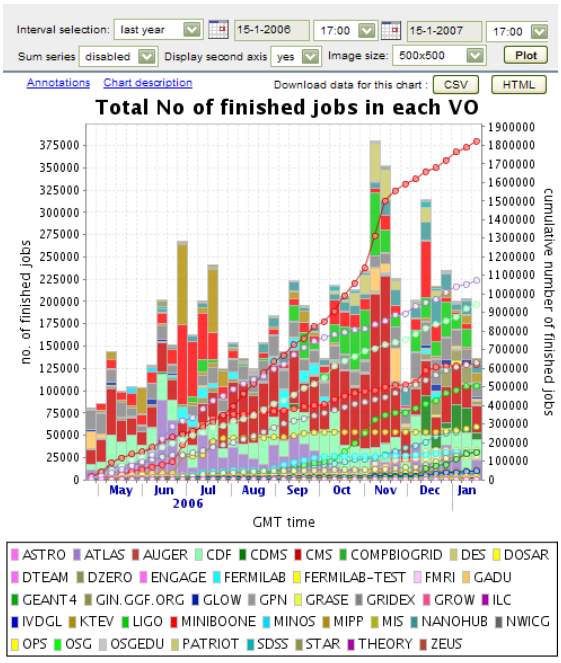

Fig. 13 – Finished jobs statistic from the MonALISA repository

Figure 12 shows an example of accounting resource usage in OSG group with the MonALISA repository. The chart presents the total integrated CPU time consumed at each site in the last week, measured as hours (of CPU time) x number of CPUs. The user can select from the interface the farms which present interest, the time interval for the plot (with predefined periods – last hour, day, week, month, year etc. or specific periods), the representation model (stacked area, series etc.) and size. The chart also has a description and annotations and is available for download in different formats: CSV, HTML.

Fig. 14 – CPU time consumed by the jobs of a virtual organization

In Figure 15 the plotted results represent the number of running and idle jobs for a VO, across different OSG sites. This kind of information, summarized per VO, is stored on a long term in MonALISA repositories.

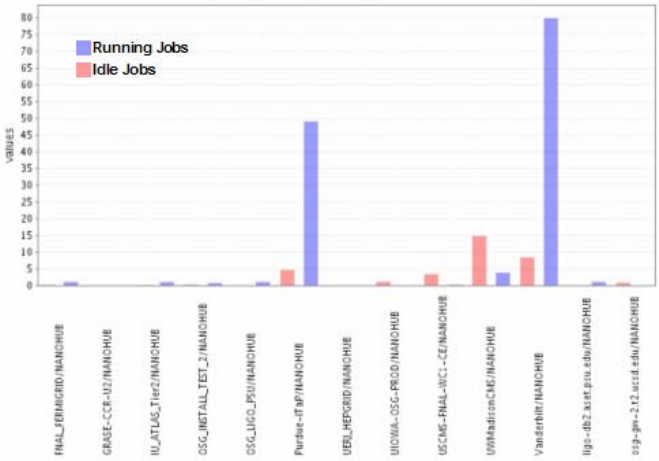

Fig. 15 – Number of running and idle jobs of a virtual organization, across different sites

Another example of accounting resource usage is shown in Figure 13 where a two axes plot presents the number of finished jobs in each Virtual Organization for the last year as well as the cumulative number of finished jobs in each VO, for the last year.

Figures 14 and 15 present information displayed by the MonALISA graphical client. The data was obtained from MonALISA services that are running on OSG sites. In Figure 15 there are the values of CPU time consumed by the jobs belonging to a VO,

on a single site. Similar charts are available for the values of wall clock time, jobs size and disk usage (depending on the resource manager running on the site). Such charts are helpful when a user wishes to learn about the status of his/her jobs.

5. CONCLUSION

We have presented a distributed framework for collecting and processing accounting information in LSDS environments. Among the strengths of the framework are: scalability, the possibility of interacting with diverse resource managers, and collecting data both in a real time manner and from logs. As of this writing, more than 360 MonALISA sites are being monitored 24/7 throughout the world. The services monitor more than 60,000 computing servers, and thousands of concurrent jobs. More than 3.5 million parameters are currently monitored in near-real time with an aggregate update rate of approximately 50,000 parameters per second. Such services are mostly deployed by the High Energy Physics community to monitor computing resources, running jobs and applications, different LSDS services and network traffic. The system is used to monitors detailed information on how the jobs are submitted to different systems, the resources consumed, and how the execution is progressing in real-time.

In this paper we presented the capabilities of MonALISA framework towards monitoring and representing large scale networks at different OSI layers. We also present a very useful use case where informed automatic decisions based on monitoring information can improve reliability and increase overall performance of the system. Network monitoring, in particular, is vital to ensure proper network operations over time, and MonALISA was successfully used to provide its monitoring services to control a vast majority of the data intensive processing tasks used by the LHC experiments. In order to build a coherent set of network management services it is very important to collect in near real-time information about the network traffic volume and its quality, and analyze the major flows and the topology of connectivity.

ACKNOWLEDGMENT

This work was supported by project "ERRIC - Empowering Romanian Research on Intelligent Information Technologies/FP7-REGPOT-2010-1", ID: 264207. The work has been cofounded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557.

6. REFERENCES

[1] L. Gaido, A. Guarise, G. Patania, R. Piro, F. Rosso, A. Werbrouck, The Distributed Grid Accounting System (DGAS), Last accessed November 22, 2012, from http://www.to.infn.it/grid/accounting/main.html.

[2] C. Dobre, R. Voicu, I. Legrand, Monitoring large scale network topologies, *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* (IDAACS 2011), Prague, Czech Republic (September 2011), pp. 218-222.

[3] MonALISA official website (2012), Last accessed November 24, 2012, from http://monalisa.caltech.edu/.

[4] CMS Experiment official website (2012), Last retrieved November 21, 2012, from http://cms.cern.ch.

[5] ALICE Experiment official website (2012), Last retrieved November 21, 2012, from http://aliweb.cern.ch.

[6] Atlas Experiment official website (2012), Last retrieved November 21, 2012, from http://atlas.web.cern.ch.

[7] OSG official website (2012), Last retrieved November 25, 2012, from http://www.opensciencegrid.org.

[8] RRD official website (2012), Last retrieved November 24, 2012, from http://www.mrtg.org/rrdtool.

[9] FDT official website (2012), Last retrieved November 24, 2012, from http://fdt.cern.ch.

[10] TL1 – Transaction Language 1 Generic Requirements Document GR-831-CORE (2012), Last retrieved November 12, 2012, from http://telecom-info.telcordia.com/site-cgi/ido/docs.cgi?ID=SEARCH&DOCUMENT=GR-831.

[11] Calient Technologies official website (2012), Last retrieved November 23, 2012, from: http://www.calient.net.

[12] GMPLS – General Multi-Protocol Label Switching Architecture RFC3945.

[13] ITU-T Rec. G.7042, Link Capacity Adjustment Scheme (LCAS) for Virtual Concatenated Signals, Feb. 2004.

[14] S. Bagnasco, L. Betev, P. Buncic, *et* al, AliEn: ALICE environment on the grid, in: *J. Phys.: Conf. Ser.* (2007), pp. 119.

[15] I.C. Legrand, H. Newman, R. Voicu, *et al*, MonALISA: An agent based, dynamic service system to monitor, control and optimize distributed systems, In: *Computer Physics Communications*, (180) Issue 12 (December 2009), pp. 2472-2498.

[16] R. Byrom, R. Cordenonsib, L. Cornwall, *et al*,

APEL: An implementation of Grid accounting using R-GMA, *in: UK e-Science All Hands Conference*, Nottingham (September 2005).

[17] A. Cooke, A.J. Gray, W. Nutt, *et al*, The relational grid monitoring architecture: mediating information about the grid, in: *Journal of Grid Computing*, (2) 4 (2004), pp. 323-339.

***Ciprian Dobre**, PhD, has scientific and scholarly contributions in the field of large scale distributed systems concerning monitoring (MonALISA), data services (PRO, DataCloud@Work), high-speed networking (VINCI, FDT), large scale application development (EGEE III, SEE-GRID-SCI), evaluation using modeling and simulation (MONARC 2, VNSim). Ciprian Dobre was awarded a PhD scholarship from California Institute of Technology (Caltech, USA), and another one from Oracle. His results received two CENIC Awards, and a Best Paper Award, and were published in 6 books, 10 articles in major international peer-reviewed journal, and over 60 articles in well-established international conferences and workshops (these articles received more than 150 citations). Currently he is local project coordinator for national projects 'CAPIM – Context-Aware Platform using Integrated Mobile Services', and 'TRANSYS – Models and Techniques for Traffic Optimizing in Urban Environments'.*

***Ramiro Voicu**, PhD, is Software Engineer at the California Institute of Technology. He has expertise in the design and development of solutions for proficient provisioning of network resources, data transfer mechanisms capable of dynamic bandwidth adjustments capabilities, extensible monitoring infrastructure development, with specific accent on the mechanisms to provide full end-to-end performance data.*

***Iosif Legrand** is Senior Software Engineer at the California Institute of Technology, Division of Physics, Mathematics and Astronomy. He worked on the simulation and modeling of the Data Grid Hierarchy concept and the globally distributed Computing Model adopted by the LHC high energy physics collaborations. He is member of the US-CMS collaboration and is working on distributed network services, monitoring systems and grid related activities.*