MonALISA
MONitoring Agents using a Large Integrated Services Architecture

An Agent Based, Dynamic Service System to Monitor, Control and Optimize Distributed Systems

June 2007

Iosif Legrand
California Institute of Technology
The MonALISA Framework

An Agent Based, Dynamic Service System to Monitor, Control and Optimize Distributed Systems

- MonALISA is a Dynamic, Distributed Service System capable to collect any type of information from different systems, to analyze it in near real time and to provide support for automated control decisions and global optimization of workflows in complex grid systems.

- The MonALISA system is designed as an ensemble of autonomous multi-threaded, self-describing agent-based subsystems which are registered as dynamic services, and are able to collaborate and cooperate in performing a wide range of monitoring tasks. These agents can analyze and process the information, in a distributed way, and to provide optimization decisions in large scale distributed applications.

- An agent-based architecture provides the ability to invest the system with increasing degrees of intelligence; to reduce complexity and make global systems manageable in real time. For an effective use of distributed resources, these services provide adaptability and self-organization.
The MonALISA Architecture

Regional or Global High Level Services, Repositories & Clients

Secure and reliable communication
Dynamic load balancing
Scalability & Replication
AAA for Clients

Distributed System for gathering and analyzing information based on mobile agents:
Customized aggregation, Triggers, Actions

Distributed Dynamic Registration and Discovery-based on a lease mechanism and remote events

Fully Distributed System with no Single Point of Failure
MonALISA service & Data Handling

- **Data Store**
- **Data Cache**
- **Service & DB**
- **Configuration Control (SSL)**
- **Predicates & Agents**
- **Data (via ML Proxy)**
- **Applications**
- **Agents**
- **Filters / Triggers**
- **Monitoring Modules**
- **WS Clients and service**
- **Postgres**
- **Web Service**
- **WSDL SOAP**
- **Lookup Service**
- **Registration**
- **Discovery**
- **Clients or Higher Level Services**
- **Collects any type of information**
- **Push and Pull**

Dynamic Loading
Monitoring Grid sites, Running Jobs, Network Traffic, and Connectivity
ApMon – Application Monitoring

- Lightweight library of APIs (C, C++, Java, Perl, Python) that can be used to send any information to MonALISA Services
- High comm. performance
- Flexible
- Complete Sys. Monitoring

**No Lost Packages**

- Lightweight library of APIs (C, C++, Java, Perl, Python) that can be used to send any information to MonALISA Services
- High comm. performance
- Flexible
- Complete Sys. Monitoring

**Monitoring Data**

- UDP/XDR
- 

**ApMon configuration generated automatically by a servlet / CGI script**

**System Monitoring**

- load1: 0.24
- processes: 97
- pages_in: 83

**ApMon**

- Time;IP;procID
- parameter1: value
- parameter2: value
- ...

**MonALISA Service**

- Config Servlet
- dynamic reloading

**ApMon – Application Monitoring**

- UDP/XDR Monitoring Data

**MonALISA hosts**

- MonALISA Service

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading

**APPLICATION**

- App. Monitoring
- MonALISA Service

**UDP/XDR Monitoring Data**

- MonALISA Service

- Config Servlet
- dynamic reloading
Monitoring the Execution of Jobs and the Time Evolution
End User / Client Agent
LISA- Localhost Information Service Agent

- Authorization
- Service discovery
- Local detection of the hardware and software configuration
- Complete end-system monitoring: Per-process load, I/O and network throughputs,
- End-to-end performance measurements
- Will act as an active listener for all events related with the requests generated by its local applications.
- Can execute agents to re-configure the system and rollback
LISA – Network Monitoring

Network monitoring module – monitors the network performance of the local workstation (network interfaces, traffic patterns, TCP/IP running stack parameters) and it also able to configure TCP/IP parameters for optimizing the networking performances. It can use IPERF, WEB 100 and other network monitoring tools.
LISA- Provides an Efficient Integration for Distributed Systems and Applications

- It is using external services to identify the real IP of the end system, its network ID and AS.
- Discovers MonALISA services and can select, based on service attributes, different applications and their parameters (location, AS, functionality, load ...)
  - Based on information such as AS number or location, it determines a list with the best possible services.
  - Registers as a listener for other service attributes (e.g., number of connected clients).
  - Continuously monitors the network connection with several selected services and provides the best one to be used from the client's perspective.
  - Measures network quality, detects faults and informs upper layer services to take appropriate decisions.
Monitoring Internet2 backbone Network

- Test for a Land Speed Record
- \( \sim 7 \text{ Gb/s} \) in a single TCP stream from Geneva to Caltech
Monitoring USLHCnet

- Operations & management assisted by agent-based software
- Used on the new CIENA equipment used for network management
USLHCNet Integrated Traffic
The UltraLight Network

BNL ESnet IN / OUT

Image of a globe with network connections and data over time.
Monitoring The GLORIAD Ring

- **wan-link Daejeon-Seattle Ultra OUT**
  - Traffic: 0.0 Mbps
  - Capacity: 1.0 Gbps
  - Utilisation: 0%

- **wan-link StarTap OUT**
  - Traffic: 0.4 Mbps
  - Capacity: 10.0 Gbps
  - Utilisation: 0%

- **wan-link Daejeon-StarTap OUT**
  - Traffic: 59.6 Mbps
  - Capacity: 1.0 Gbps
  - Utilisation: 5%

- **wan-link Daejeon-Seoul_2 IN**
  - Traffic: 27.4 Mbps
  - Capacity: 1.0 Gbps
  - Utilisation: 2%

- **wan-link StarTap IN**
  - Traffic: 3.9 Mbps
  - Capacity: 10.0 Gbps
  - Utilisation: 0%

- **wan-link Seattle-Daejeon_2 OUT**
  - Traffic: 28.2 Mbps
  - Capacity: 1.0 Gbps
  - Utilisation: 2%

- **wan-link StarTap OUT**
  - Traffic: 14.6 Mbps
  - Capacity: 155.0 Mbps
  - Utilisation: 9%

- **wan-link Seattle OUT**
  - Traffic: 9.3 Mbps
  - Capacity: 155.0 Mbps
  - Utilisation: 6%

- **wan-link Amsterdam OUT**
  - Traffic: 2.6 Mbps
  - Capacity: 155.0 Mbps
  - Utilisation: 1%

- **wan-link Amsterdam IN**
  - Traffic: 29.7 Mbps
  - Capacity: 155.0 Mbps
  - Utilisation: 13%

**node(s) under cursor: GLORIAD (192.91.246.188)**
Monitoring Network Topology
Latency, Routers

NETWORKS

AS

 ROUTERS
### Available Bandwidth between UL sites (average)

<table>
<thead>
<tr>
<th>Site (from→to)</th>
<th>CERN-UL</th>
<th>KEK-UL</th>
<th>SPRACE-UL</th>
<th>UL_CIT</th>
<th>UM</th>
</tr>
</thead>
<tbody>
<tr>
<td>CERN-UL</td>
<td>-</td>
<td>794.4 Mbps</td>
<td>97.37 Mbps</td>
<td>97.73 Mbps</td>
<td>97.85 Mbps</td>
</tr>
<tr>
<td>KEK-UL</td>
<td>750 Mbps</td>
<td>-</td>
<td>96.32 Mbps</td>
<td>993.2 Mbps</td>
<td>96.34 Mbps</td>
</tr>
<tr>
<td>StarLight-UL</td>
<td>97.4 Mbps</td>
<td>97.53 Mbps</td>
<td>-</td>
<td>875 Mbps</td>
<td>97.5 Mbps</td>
</tr>
<tr>
<td>UL_CIT</td>
<td>97.5 Mbps</td>
<td>993.2 Mbps</td>
<td>876 Mbps</td>
<td>-</td>
<td>96.63 Mbps</td>
</tr>
<tr>
<td>UM</td>
<td>97.48 Mbps</td>
<td>96.84 Mbps</td>
<td>96.55 Mbps</td>
<td>96.85 Mbps</td>
<td>-</td>
</tr>
<tr>
<td>Min</td>
<td>97.4 Mbps</td>
<td>97.53 Mbps</td>
<td>96.32 Mbps</td>
<td>96.85 Mbps</td>
<td>96.34 Mbps</td>
</tr>
<tr>
<td>Max</td>
<td>750 Mbps</td>
<td>993.2 Mbps</td>
<td>876 Mbps</td>
<td>993.2 Mbps</td>
<td>97.85 Mbps</td>
</tr>
</tbody>
</table>

**Available Bandwidth History**

![Available Bandwidth History Chart](chart.png)

---

**MonALISA Repository UltraLight**

**Available Bandwidth Measurements**

**Embedded Pathload module.**

---

**Available bandwidth measurements using pathload**

---

**MonALISA**

**HIT-Coordination Services**
Monitoring and Controlling Optical Switches

Port power monitoring

Controlling
Monitoring Optical Switches
ALICE: Job status & traffic - real-time map
Local and Global Decision Framework

- Based on monitoring information, actions can be taken in:
  - ML Service
  - ML Global Services / Repository

- Actions can be triggered by:
  - Values above/below given thresholds
  - Absence/presence of values
  - Correlation between multiple values

- Decisions types:
  - Alerts
    - e-mail
    - Instant messaging
    - RSS Feeds
  - External commands
  - Event logging
  - Global Optimization Services

- Actions based on global information:
  - Traffic
  - Connectivity
  - Jobs
  - Hosts
  - Apps

- Actions based on local information:
  - Temperature
  - Humidity
  - A/C Power
  - ...

- Sensors
- Local decisions
- Global decisions
Alerts and actions

**MySQL daemon** is automatically restarted when it runs out of memory
*Trigger: threshold on VSZ memory usage*

**ALICE Production jobs queue** is automatically kept full by the automatic resubmission
*Trigger: threshold on the number of *aliprod* waiting jobs*

**Administrators** are kept up-to-date on the services’ status
*Trigger: presence/absence of monitored information*
Monitoring Video Conference System: Reflectors and Communication Topology
Creating a Dynamic, Global, Minimum Spanning Tree to optimize the connectivity

A weighted connected graph $G = (V,E)$ with $n$ vertices and $m$ edges. The quality of connectivity between any two reflectors is measured every 2s. Building in near real time a minimum-spanning tree $T$

$$w(T) = \sum_{(v,u) \in T} w((v,u))$$
- Dynamic Discovery of Reflectors
- Creates and maintains, in real-time, the optimal connectivity between reflectors (MST) based on periodic network measurements.
- Detects and monitors the User configuration, its hardware, the connectivity and its performance.
- Dynamically connects the client to the best reflector
- Provides secure administration.
- It is using alarm triggers to notify unexpected events.
“On-Demand”, Dynamic Path Allocation

CREATE AN END TO END PATH < 1s

APPLICATION

LISA AGENT
LISA sets up
- Network Interfaces
- TCP stack
- Kernel parameters
- Routes
LISA $\rightarrow$ APPLICATION
“use eth1.2, …”

INTERNET

MonALISA Distributed Service System

Control Monitor

MonALISA Service

Regular IP path

Real time monitoring

APPLICATION

LISA AGENT

OPTICAL SWITCH

Real time monitoring

OS Agent

DATA

Active light path

Detects errors and automatically recreate the path in less than the TCP timeout

APPLICATION

LISA AGENT

LISA sets up
- Network Interfaces
- TCP stack
- Kernel parameters
- Routes
LISA $\rightarrow$ APPLICATION
“use eth1.2, …”

INTERNET

MonALISA Distributed Service System

Control Monitor

MonALISA Service

Regular IP path

Real time monitoring

OS Agent

DATA

Active light path

Detects errors and automatically recreate the path in less than the TCP timeout

APPLICATION

LISA AGENT

LISA sets up
- Network Interfaces
- TCP stack
- Kernel parameters
- Routes
LISA $\rightarrow$ APPLICATION
“use eth1.2, …”

INTERNET

MonALISA Distributed Service System

Control Monitor

MonALISA Service

Regular IP path

Real time monitoring

OS Agent

DATA

Active light path

Detects errors and automatically recreate the path in less than the TCP timeout
FDT – Fast Data Transfer

- FDT is an application for efficient data transfers.
- Easy to use. Written in java and runs on all major platforms.
- It is based on an asynchronous, multithreaded system which is using the NIO library and is able to:
  - stream continuously a list of files
  - use independent threads to read and write on each physical device
  - transfer data in parallel on multiple TCP streams, when necessary
  - use appropriate size of buffers for disk IO and networking
  - resume a file transfer session
FDT – Fast Data Transfer

Control connection / authorization

Pool of buffers
Kernel Space

Data Transfer Sockets / Channels

Pool of buffers
Kernel Space

Independent
threads per device

Restore the files from buffers
FDT – Memory to Memory Tests in WAN

CPUs Dual Core Intel Xenon @ 3.00 GHz, 4 GB RAM, 4 x 320 GB SATA Disks
Connected with 10Gb/s Myricom
FDT – Memory to Memory Tests in WAN

C1-NY -> C1-GVA

eth2_out
FDT – Memory to Memory Tests in WAN

C1-NY <-> C1-GVA

eth2_IN

eth2_OUT
Memory-to-memory in transfers in WAN in both directions with two pairs of servers:

C1-NY -> C1-GVA
C2-NY <- C2-GVA
Disk-to-Disk transfers in WAN

Page_IN
C1-NY -> C1-GVA

Read and writes on 4 SATA disks in parallel on each server

Mean traffic ~ 210 MB/s
~ 0.75 TB per hour

CERN -> CALTECH

Read and writes on 2 RAID Controllers in parallel on each server

Mean traffic ~ 545 MB/s
~ 2 TB per hour
Controlling Optical Planes
Automatic Path Recovery

“Fiber cut” simulations
The traffic moves from one
transatlantic line to the other one
FDT transfer (CERN – CALTECH)
continues uninterrupted
TCP fully recovers in ~ 20s
Bandwidth Challenge at SC2005

151 Gbs

~ 500 TB Total in 4h
FDT Used at SC 2006
Entire management was done with LISA & MonALISA
MonALISA is interfaced with many monitoring tools and is capable to collect information from different applications:

- Computing Nodes / Farms (system information, network traffic…)
  - SNMP, Ganglia, dedicated scripts
- Routers, Switches, Optical Switches
  - SNMP, NetFlow, SFlow, TL1, MRTG, WS
- End to End Network performance
  - Pathload, IPERF, Pipes, Abing, ABping …
- Batch Queuing Systems
  - LSF, PBS, Condor, NQS, Grid Job Manager
- Applications
  - Root, Xrootd, CRAB, RRD, VRVS /EVO, …
Communities using MonALISA

Major Communities
- ALICE
- OSG
- CMS
- STAR
- VRVS
- LGC RUSSIA
- SE Europe GRID
- APAC Grid
- UNAM Grid (Mx)
- ITU
- ABILENE
- ULTRALIGHT
- GLORIAD
- LHC Net
- RoEduNET
- Enlightened

MonALISA Today
Running 24 X 7 at ~340 Sites
- Collecting ~ 1 000 000 parameters in near real-time
- Update rate of 20,000 parameter updates per second
- Monitoring
  - 12,000 computers
  - > 100 WAN Links
- Thousands of Grid jobs running concurrently

Demonstrated at:
- Telecom World
- WSIS 2003
- SC 2004
- Internet2 2005
- TERENA 2005
- IGrid 2005
- SC 2005
- CHEP 2006
- CENIC 2006

Innovation Award for High-Performance Applications
The MonALISA Architecture Provides:

- Distributed **Registration and Discovery** for Services and Applications.
- Monitoring all aspects of complex systems:
  - System information for computer nodes and clusters
  - Network monitoring, topology, end to end performance
  - Monitoring the performance of Applications, Jobs or services
  - The End User Systems, its performance
  - Environment; Video streaming
- Can interact with any other services to provide in near real-time customized information based on monitoring data.
- Secure, remote administration for services and applications.
- Agents to supervise applications, trigger alarms, restart or reconfigure them, and to notify other services when certain conditions are detected.
- The MonALISA framework is used to develop higher level decision services, implemented as a distributed network of communicating agents, to perform global optimization tasks.
- **Graphical User Interfaces** to visualize complex information

[http://monalisa.caltech.edu](http://monalisa.caltech.edu)