A Prototype for a Monitoring Service

Iosif Legrand

California Institute of Technology
Design Considerations / Requirements

- Act as a true dynamic service and provide the necessary functionally to be used by any other services that require such information.
- Allow dynamic configuration and the list of monitor parameters.
- It should provide:
  - single-farm values and details for each node
  - network aspects
  - real time information
  - historical data and extracted trend information
  - active filters and algorithms for prediction and decision-support
- It should easily integrate existing monitoring tools
- Scalable and reliable
A Service Registers with at least one Lookup Service using the same ID.

It provides information about its functionality and the URL addressed from where interested clients may get the dynamic code to use it.

The Service must ask each Lookup Service for a lease and periodically renew it.

If a Service fails to renew the lease, it is removed from the Lookup Service Directory. When problems are solved, it can re-register.

The lease mechanism allows the Lookup Service to keep an up to date directory of services and correctly handle network problems. LDAP does not have this functionality.
Monitoring Data Collection

PULL

SNMP get & walk
rsh | ssh remote scripts
End-To-End measurements

PUSH

snmp trap

Dynamic loading of modules or agents

Trap Agent
(ucd – snmp) perl

WEB Server

Dynamic Thread Pool

Farm Monitor

Trap Listener

Configuration Control

Int. to Other tools
The Multi-Threaded Pull Architecture

• Each request is done in an independent thread

• A slow agent / busy node does not perturb the measurements of an entire system

• Ex: Monitor 300 nodes @ 30 seconds interval → 10-15 Threads are running in parallel
Farm Monitor UNIT & Data Handling

- Config
- Status
- Data
- Data Cache Service
- Predicates & Filter Agents
- Lookup Service
- Discovery
- Registration
- Configuration Control
- Configuration Control
- MySQL
- Oracle
- Cloudscape
- WEB Server
- LDAP
- MDS
- Other tools
- Web client
- Client (other service)
- JAVA
- Discovery
- WEB Service
- WSDL
- SOAP
- Client (other service)
- Client (other service)
Results objects are store in a persistent space
(RMDB or automatic object mapping to RDB systems)

Each object Data Store is registered as an independent JINI / WEB services. Its content can be accessed using a JINI proxy to the service or a SOAP - XML gateway in a WEB Server.

Subscription to results objects matching a template / predicate.

Clients can load filter objects into the Data Cache service and generate any derived (or aggregate) data structures and register to receive them.

Monitored parameters may have a life time and summary data should be kept after a certain time
Registration with several Lookup discovery services

- Component Factory
- GUI marshaling
- Code Transport as "service attribute"
- RMI data access
Example: GUI Clients
Example: Real Time Best Replication Path
Peer-to-Peer system of monitoring units
The monitoring system will be a vital component for development of the LHC Computing Model, as well as the development of realistic "optimal" strategies for resource management and user interactions with LHC and other Grids which are both resource constrained and truly global in scope.

Monitored information should be used along with the modeling and simulation tools, and a series of working prototype Tier1, Tier2 and Tier3 facilities, to drive development of the strategies.

New optimization methods (such as the SONN) will be used to find practical ways to maximize workflow through the system.