Architectural Models for WLCG Monitoring

James Casey, CERN

GDB

5th December, 2007

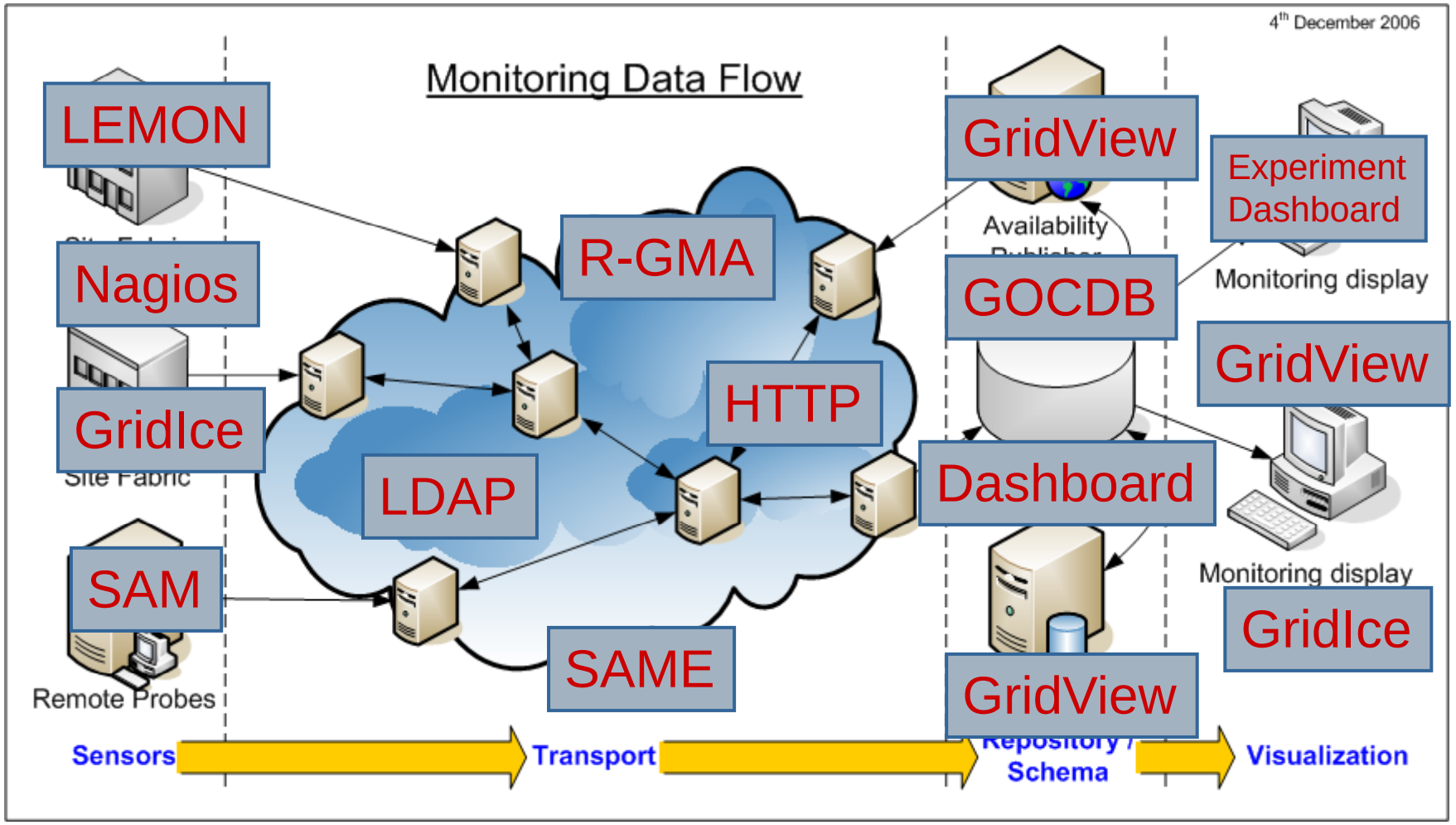
Background

- WLCG Monitoring Working Groups has now been going for 1 year
- Initial aims achieved
  - Ability to pass back information into sites on their availability
  - Addition of better site local monitoring
- Now need to look at next phase of work
  - Not as a working group
  - Just part of normal operations

Who is involved?

- Four main stakeholders
  - Site Administrators
  - Grid Operators
    - “CIC on Duty”
    - Regional Operation Centre (ROC)
  - WLCG Project Management
  - Virtual Organisations
    - WLCG Experiments
- 5th ‘Stakeholder’
  - Monitoring developers + operators

High-level Model



See https://twiki.cern.ch/twiki/pub/LCG/GridServiceMonitoringInfo/0702-WLCG_Monitoring_for_Managers.pdf for details

Architectural Principles

Why Architectural Principles?

- Many different architectures could solve our problems
  - We've deployed many already !
- Need to keep consistency when making choices
  - Use the principles to guide concrete choices
- Approach already was used to design the site-local monitoring prototype
  - This is an attempt to extend them more globally

Improve reliability by reducing time to respond

- "Site administrators are closest to the problems, and need to know about them first"
- Our focus has been on site monitoring
  - not been deployed widely by sites
- Implications
  - Improved understanding of how to monitor services
    - ✓ "Service Cards" being developed by EGEE SA3
  - Need to deploy components to sites
    - ✗ Sometimes an entire monitoring system
    - ✗ Needs active participation of site admins

Tell others what you know

- "If you're monitoring a site remotely, it's only polite to give the data to the site" *Chris Brew.*
  - Remote systems should feed back information to sites
- Implications
  - ✓ Common publication mechanisms
  - ✓ Integration into fabric monitoring
  - ✗ Discovery of data ?
  - ✗ Site trust of data – Is it a "backdoor" communications mechanism?

Authority for data…

- Currently repositories have direct DB connections to (all) other repositories
  - E.g. SAM, Gridview, Gstat, GOCDB, CIC
- And they cache and merge and process the data
- Implications
  - 💣 We have a “Interlinked distributed schema”
  - ✓ And tools should take responsibility for contents of parts of it

No monolithic systems

- Different systems should specialize in their areas of expertise
  - And not have to also invent all the common infrastructure

- Implications
  - ✓ Less overlap and duplication of work
  - ✗ Someone needs to manage some common infrastructure
  - 💣 We need to agree on the common infrastructure

Don't have central bottlenecks

- "Local problems detected locally shouldn't require remote services to work out what the problem is"
  - Still a role for central detection of problem
    - Just they're reported locally too
- Lots of central processing done now in SAM/Gridview
- Implications
  - ✓ Do as much processing locally (or regionally)
  - ✓ Helps scaling – improves robustness
  - ✓ enables automation - reduces manpower
  - ✗ Harder to deploy

Visualization for each community

- “user-targeted” visualization
  - But all should use the same underlying data
    - Extract information processing out of visualization tools
    - Provide same processed info to all visualizations
  - Interface with community specific information, e.g. names
- Implications
  - 💣 Many “similar” dashboards
  - ✓ Everyone sees the same data
  - ? Common frameworks/widgets needed here ?

Locality issues

- Our operations model is changing
  - From central to regional/national/local
- Architecture should reflect this
  - ✓ Distribution will help scaling
- Still need to pass information upstream
  - For reporting, central debugging, …
- Implications
  - Guidelines needed to help developers to decide what to pass back
  - Possibility of more info going across the site boundary than the site admin wants

Tasks

Things to do… independent of architecture

- Messaging System
  - messaging seen as vital component for reliability, robustness and scaling
- Operations Dashboard
  - “New” SAM Portal
- Deploy the site monitoring
  - Make it easier for sites (provide yaim, …)
- Reporting
  - Automate our management reporting

Summary

- Have some principles to guide us
- Should come back in January with concrete workplans for tasks
- Remember that the goal is to improve reliability of the monitored system
  - And every decision should reflect that

Questions ?

Other Principles (Secondary)

Make it less brittle

- Remove SPOFs
  - Push reliability into the fabric
  - Less point-to-point connections over the WAN
- Asynchronous communication works for monitoring
  - Latency is ok (to a point)
- Implications
  - ✓ Reliable transport fabric using messaging systems
  - ✓ Less central (unique) repositories improves availability

Re-use, don't re-invent

- What do we do?
  - Collect some information, Move it around
  - Store it, View it, Report on it
- This is pretty common ☺
  - We should look at existing systems
- Already happening for site fabric…
  - Nagios, LEMON, …
- Implication
  - ✓ Less code to develop and maintain
  - 💣 Integration nightmare?

Don’t impose systems on sites

- We can’t dictate a monitoring system
  - Many (big?) sites already have a deployed system
  - We have to be pluggable into them
- Implications
  - ✓ Modular approach
  - ✓ Specifications to define interfaces between existing systems and new components

Common names

- Infrastructures has identifiers for things
  - GOCDB name is really an ID, not a name – used as a primary key for linkage
- Implication
  - ? (GOCDB ?) ID used for mapping between names
  - ✓ A community should only see their name
    - And never the ID

Responsibility for data

- Certain system clean up data
  - FTM for log files
  - Gstat for BDII
- And should re-publish the “validated” version
  - Everyone shouldn’t do the processing (differently)
    - E.g. gridmap/SAM/gstat for CPU counts
- Implications
  - ✓ Tools are simpler if they use cleaned up data
  - ✗ At the cost of latency

Flexible topology

- Communities also have structure
  - Distributed tier-2s, split between CAF and production, disk and tape, …
- We need to be able to map sites, services and metrics into the community structure
- Implications
  - ✓ Common ontology/schemas
  - Flexible and distributed
    - Integrate many data sources

Necessary (and complete) security

- Security has an overhead
  - Don't kill your system for "needless" security e.g. R-GMA
- Encryption of specific elements
  - E.g User DN
- Signing for provenance
- Implication
  - ✓ Security is available when needed depending on application use-case
  - ✓ It's not used when not needed

A (technical) architecture?

- Scalable underlying fabric
  - Messaging Systems, HTTP REST — ActiveMQ
- Use (simple) standards — HTTP, SSL
  - Web technology
- Publish metadata — RDF
  - Common grid topology information
- Republish “raw” bulk data — “standard” XML (?)
  - metrics, usage records, …
- Visualization “toolkits” — Dashboard, Yahoo UI, …
- Reporting — JasperReports