Grid Middleware for WLCG

Computing in High Energy and Nuclear Physics
Prague | Czech Republic | 21 - 27 March 2009

Where are we now – and where do we go from here?

Prague, 24th March 200

Ian Bird, James Casey,
Oliver Keeble, Markus Schu
& thanks to Pere Mato

CERN

Introduction

This is a talk about middleware/software

- But WLCG is quite a lot more – it is a collaboration and contains many other pieces that allow us to provide a global computing environment:
  - Policies and frameworks for collaboration
  - Operations coordination and service procedures (to service providers: sites, networks, etc)
  - GGUS, user support, ...
  - User and VO management
  - Tools for monitoring, reporting, accounting, etc.
  - Security coordination and follow up
- Complex middleware can make much of these more difficult to manage
- The question arises:
  - Within this framework can we simplify the services in order to make the overall environment more sustainable and easy to use?
  - Today there are new technologies and ideas that may allow (some of) this to happen

Introduction

- We should also remember what our real goal is ...

- To provide the computing resources and environment to enable the LHC experiments
  - Using the most appropriate technology ... no matter what its label

  - Is what we are doing still the most appropriate for the long term?
  - We have to consider issues of maintainability and ownership (risk management)

- And although we use gLite as an example here, the lessons apply elsewhere too

Evolution has been

- Simplifying grid services
- Experiment software has absorbed some of the complexity,
- Computing models have removed some of the complexity,

- Grid developments have not delivered:
  - All the functionality asked for
  - Reliable, fault tolerant services
  - Ease of use

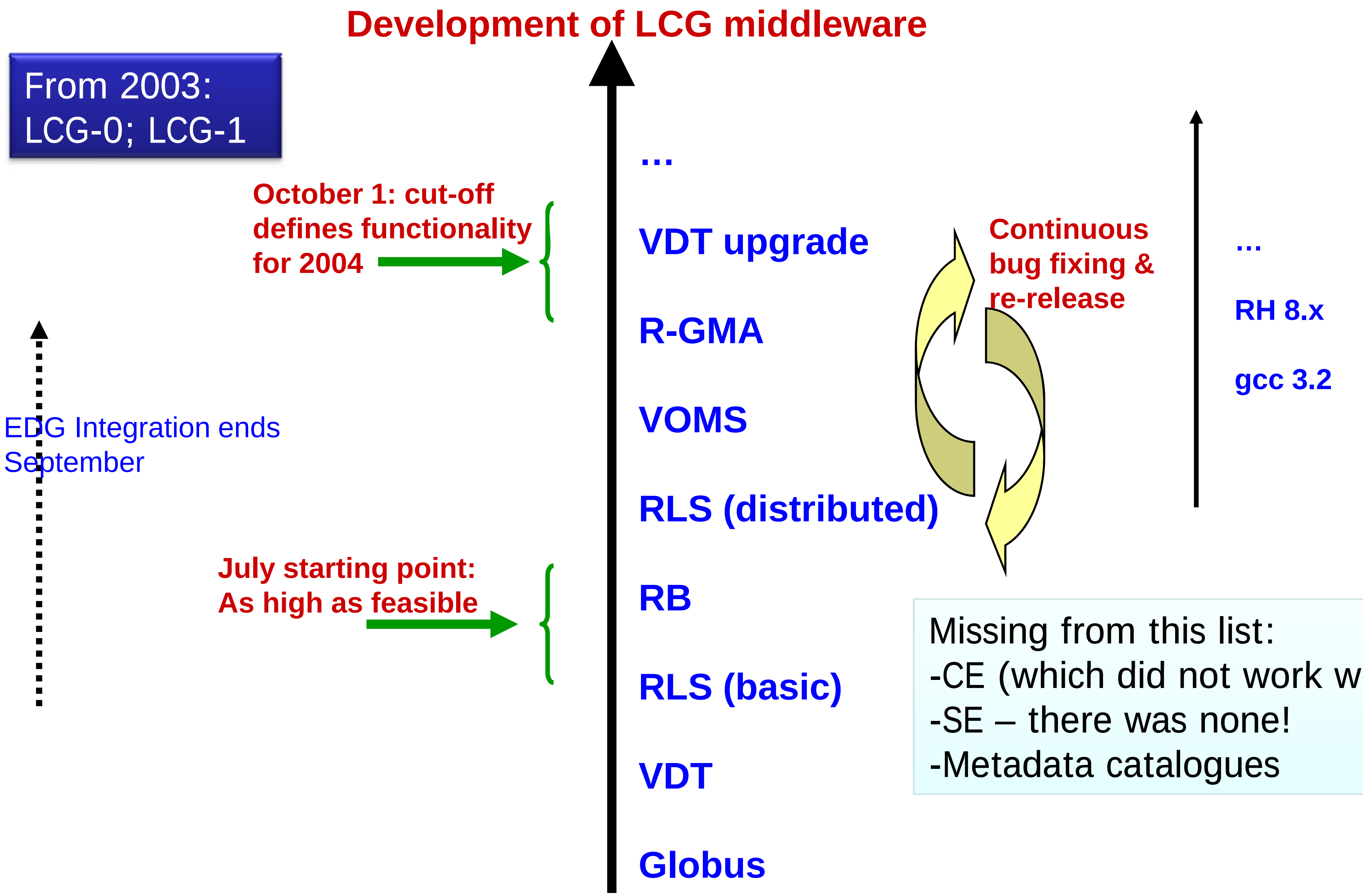
- But requirements surely were overstated in the beginning

- And grid software was less real than we had thought ...

- And as Les Robertson said, technology has moved on

Incremental Deployment

From 2003:
LCG-0; LCG-1



Development of LCG middleware

October 1: cut-off defines functionality for 2004

July starting point: As high as feasible

EDG Integration ends September

…

VDT upgrade

R-GMA

VOMS

RLS (distributed)

RB

RLS (basic)

VDT

Globus

Continuous bug fixing & re-release

…

RH 8.x

gcc 3.2

Missing from this list:
-CE (which did not work well)
-SE – there was none!
-Metadata catalogues

Original Baseline Services Report, May 2005

Baseline services

- We have reached the following initial understanding on what should be regarded as baseline services

- Storage management services
  - Based on SRM as the interface
- gridftp
- Reliable file transfer service

X File placement service – perhaps later

- Grid catalogue services
- Workload management
  - CE and batch systems seen as essential baseline services,

  ? WMS not necessarily by all
- Grid monitoring tools and services
  - Focussed on job monitoring – basic level in common, WLM dependent part

- VO management services
  - Clear need for VOMS – limited set of roles, subgroups
- Applications software installation service
- From discussions added:
  - Posix-like I/O service → local files, and include links to catalogues
  - VO agent framework
  - Reliable messaging service

Middleware: Baseline Services

Baseline services today

The *Basic* Baseline Services – from the TDR (2005)

- Storage Element
  - Castor, dCache, DPM
  - Storm added in 2007
  - SRM 2.2 – deployed in production – Dec 2007
- Basic transfe
- File Transfer Service (FTS)
- LCG File Catalog (LFC)
- LCG data mgt tools - lcg-utils
- “Posix” I/O –
  - Grid File Access Library (GFAL)
- Synchronised databases T0←→T1s
  - 3D project

Some services have not evolved and are not adequate (e.g. Software installation)

- Information System
  - BDII, GLUE
- Compute Elements
  - Globus/Condor-C
  - web services (CREAM)
  - ulti-user pilot jobs
- Workload Management
  - WMS, LB
- VO Management System (VOMS), MyProxy
- VO Boxes
- Application software installation
- Job Monitoring Tools
- APEL etc.

What works?

- Single sign-on – everyone has a certificate, we have a world-wide network of trust
  - VO membership management (VOMS), also tied to trust networks
- Data transfer – gridftp, FTS, + experiment layers;
  - Demonstrate full end-end bandwidths well in excess of what is required, sustained for extended periods
- Simple catalogues – LFC
  - Central model – sometimes with distributed read-only copies (ATLAS has a distributed model)
- Observation: The network – probably the most reliable service – fears about needing remote services in case of network failure probably add to complexity
  - i.e. Using reliable central services may be more reliable than distributed services

What else works

- Databases – as long as the layer around them is not too thick
  - NB Oracle streams works – but do we see limits in performance?
- Batch systems and the CE/gateway
  - After 5 years the lcg-CE is quite robust and (is made to) scales to today's needs ... But must be replaced (scaling, maintenance, architecture, ...). Essentially a reimplentation of the Globus gateway with add-ons
- The information systems – BDII – again a reimplentation of Globus with detailed analysis of bottlenecks etc.
  - GLUE – is a full repository of experience/knowledge of 5 years of grid work – now accepted as an OGF standard
- Monitoring, accounting
  - Today provides a reasonable view of the infrastructure
- Robust messaging systems – now finally coming as a general service (used by monitoring ... Many other applications)
  - Not HEP code!

What about...

- Workload management?
  - Grand ideas of matchmaking in complex environments, finding data, optimising network transfer etc
  - Was it ever needed?
  - Now pilot jobs remove the need for most (all?) of this
  - Even today the workload management systems are not fully reliable despite huge efforts

- Data Management
  - Is complex (and has several complex implementations)
  - SRM suffered from wild requirements creep, and lack of agreement on behaviours/semantics/etc.

And ...

- Disappointment of existing m/w robustness and usability
  - Consistent logging, error messages, facilities for service management, etc....
- Providers have never been able to fully test own services – rely on certification team (seen as bottleneck)
  - Plus problems of complexity/interdependencies have taken a long time to address
- What if WLCG is forced to have its own m/w distribution – or recommended components?
  - Can we rely on a gLite consortium, “EGI” middleware development, etc?
  - How can we manage the risk that the developments diverge from what we (WLCG) need?

Other lessons

- Generic services providing complex functionality for several user communities are not easy
- Performance, scalability, and reliability of basic services are most important (and least worked on?)
- Complex functionality is almost always application specific and should be better managed at the application level
- Too many requirements and pressure to deliver NOW;
  - But lack of prototyping
  - Wrong thing produced, or too complex, or requirements had changed
- Suffered from lack of overall architecture

What could be done today?

- A lot of people mention Clouds and Grids. But they solve different problems:
  - For WLCG:
    - The resources available to us are in worldwide-distributed locations for many good reasons
    - How can we make use of those resources effectively?
  - Elsewhere:
    - More cost effective to group resources into a single cloud – economies of scale in many areas
    - Use technologies such as virtualisation to hide underlying hardware
    - Simpler interfaces – forces simple usage patterns
    - Does not address data management of the type that WLCG needs
- So, while we cannot physically pool the resources into a few “clouds” (yet!); we can use several of the ideas

A view of the future

- WLCG could become a grid of cloud-like objects:
- Still have many physical sites
- But hide the details with virtualisation –

- What else is useful?

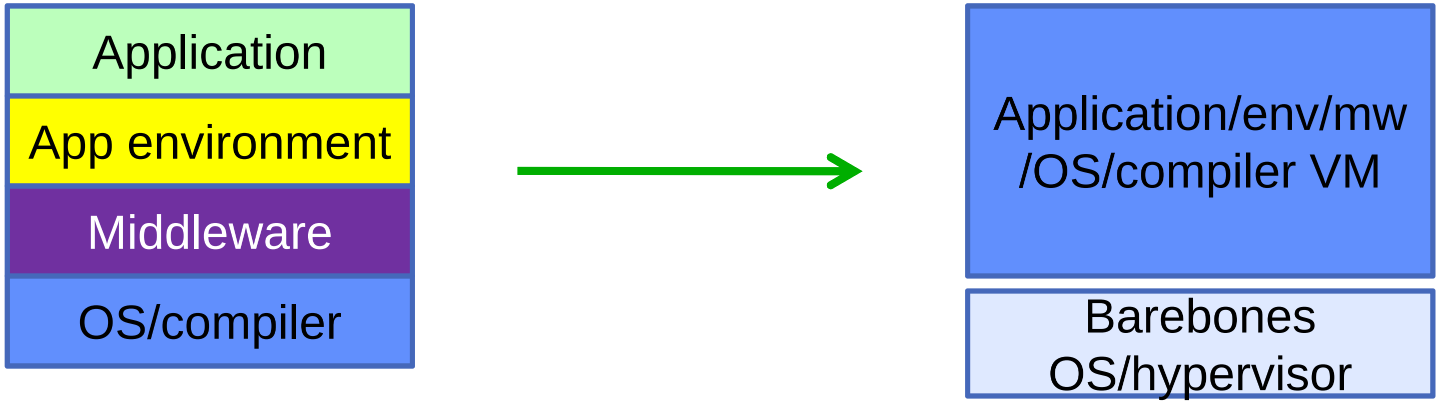
  - Virtualisation
  - Pilot jobs
  - File systems
  - Scalable/Reliable messaging services
  - Remote access to databases
  - Simplified data management interfaces (is Amazon too simple?)

The facility …

- Goal to decouple the complexities and interdependencies:
- Ability to run virtual machines
  - Still need the batch systems – fairshares etc
  - Need to be able to manage VMs (LSF, VMWare, ...)
    - Tools for debugging (e.g. Halt and retrieve image?)
- Entry point:
  - CE? – but can now be very simple
    - Mainly needs to be able to launch pilot factories (may even go away?)
    - Need to be able to communicate fully with the batch system – express requirements and allow correct scheduling
    - Information published by site directly to a messaging system (rather than via 3rd party service)
  - Probably need caching for delivery of software environments etc

- The complexities of OS/compiler vs middleware vs application environment vs application interdependencies goes away from the site (to the experiment!)

Virtual machines at a site



Site installs and maintains:
-OS,compiler
- Middleware
VO at every site installs:
-App environment
Complex dependencies between all layers

Site installs and maintains:
- bare OS
Experiment installs (~once!):
-pilotVM
-Imagine that sw env installed in pilot via cache at site
-Almost no dependencies for site

Site could also provide VM for apps that want a “normal” OS environment, need tools to manage this. This is like Amazon – the app picks the VM it needs, either a standard one, or its own

Data management

- Mass storage systems:
  - Still need gridftp, FTS
    - FTS can benefit from a messaging system
    - Is gridftp still the best thing?
  - Management interface to storage systems
    - Today SRM → what lessons can be learned – to simplify and make more robust?
  - Would like to be able to access data in the same way no matter where:
  - Can/should we decouple the tape backends from the disk pools
  - Can we move to a single access protocol?
    - E.g. Xrootd
  - But still missing ACLs, (grid-wide) quotas, ...

- Filesystems
  - Today large scale mountable filesystems are becoming usable at the scales we need
    - Lustre, Hadoop, gpfs, NFS4, etc.

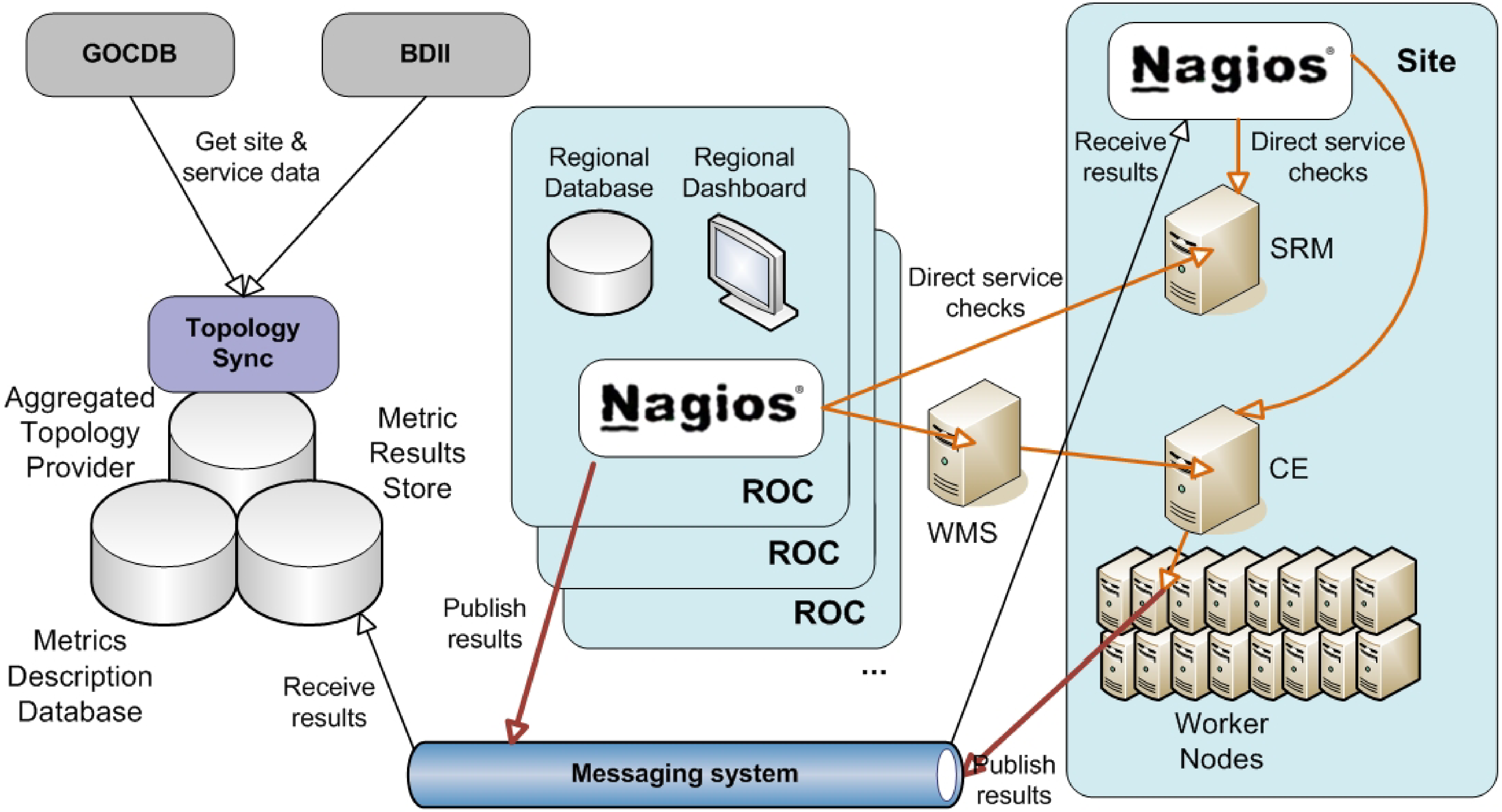
What else is needed?

- AAA:
  - Full environment that we have today (VOMS, Myproxy, etc), support for multi-user pilot jobs, etc. must be kept ... and developed
  - No/less need for fine grained policy negotiation with sites (e.g. Changing shares)?
  - But should probably address cost of using authn/authz – e.g. Session re-use etc.
- Information system
  - But becomes thinner and no need for 2min updates as no longer needed for matchmaking etc.
  - Needed mainly for service discovery and reporting/accounting
  - GLUE is good description based on 5 years experience
- Monitoring systems
  - Are needed – and need to continue to be well developed
  - Using industrial messaging systems as transport layer
  - Nagios v. good example of use of Open Source
- Databases – with good grid/remote interfaces
  - Could mean that e.g. LFC could be directly Oracle etc

Building distributed systems

- Web services have not really delivered what was promised
  - Unless you use only Microsoft, or only Java, etc.
  - Tools to support our environment (like gsoap) have not matured (or available)
- Interconnecting systems in the real world is done today with messaging systems
  - Allows to decouple distributed services in a real way
- The work done in monitoring with ActiveMQ shows that this is a realistic mechanism
  - ... And there are many potential applications of such a system
  - And we don't have to develop it
  - It is asynchronous ... But so is most of what we do

What are the limitations & possible solutions?



Long term support

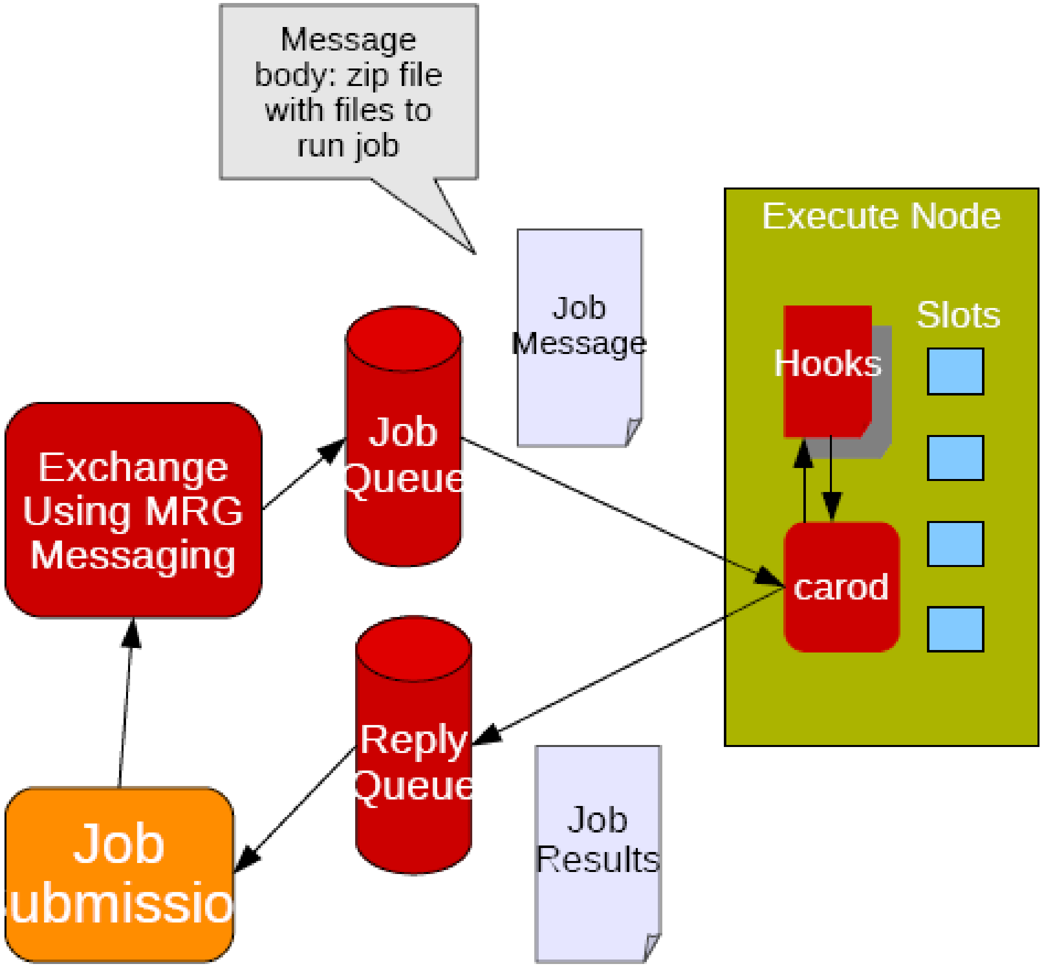
- I am not proposing changing anything now!
  - We must ensure the system we have is stable and reliable for data taking
- But we should take a good look at what we expect to need now and make use of what is available
- What is described here is not in contradiction to the needs of other e-science applications which must co-exist at many sites
  - Except perhaps the management of VMs – and there we have to think carefully
  - All this can co-exist with higher level services for other VOs, portals, etc.
- And could be deployed in parallel with existing systems

- Having less, non-general purpose middleware must be a good thing
  - Simpler to maintain, simpler to manage, available in open source etc.

- Or .... We just use RedHat??? ➔

Messaging Software Ecosystem Examples

- MRG Grid provides low latency scheduling via messaging
  - Useful pattern for other systems
- MRG/Qpid provides features people often build on top of messaging
  - XML Exchange, LVQ, Ring Queue, TTL, Federation, Management, etc.
- Open Source projects are building on AMQP Messaging
  - OpenIPA project is using AMQP Messaging for management and monitoring of Identity, Policy, Audit systems
  - LibVirt project is using AMQP messaging for management and monitoring
  - Wireshark supports AMQP



Conclusions

- We have built a working system that will be used for first data taking
  - But it has taken a lot longer than anticipated ... and was a lot harder ... and the reality does not quite match the hype ...
- We now have an opportunity to rethink how we want this to develop in the future
  - Clearer ideas of what is needed
  - And must consider the risks, maintainability, reliability, and complexity
- It was always stated that ultimately this should all come from grid providers
  - Not quite there yet, but a chance to simplify ?