

# WP4: lessons learned

GridPP Middleware Workshop  
4 March 2004

**Alexander Holt**

School of Informatics  
University of Edinburgh

`lex@fixedpoint.org`

## Achievements

### LCFG:

- ▶ Almost universal use across testbeds
- ▶ Not user-friendly, but gets job done
- ▶ Reliable enough *not* to be talked about...

### Quattor:

- ▶ Written from scratch (CERN, INFN, Liverpool, Edinburgh)
- ▶ Retains LCFG architecture
- ▶ Better language, less invasive
- ▶ Already managing 1000 nodes at CERN
- ▶ CERN taking over development & maintenance

## Challenges 1: central compilation bottlenecks

- ▶ CERN & Edinburgh Informatics: 1000 nodes
- ▶ Central compiler  $\Rightarrow$  changes take 30 minutes
- ▶ Debugging painful
- ▶ Unacceptable for fault tolerance feedback...
- ▶ ... and server/compiler machine a single point of failure
- ▶ Manageable now, but won't scale to 10,000 nodes (or to greater complexity/node diversity)

## Challenges 2: cross-fabric config data

Virtually all fabrics involve 'client-server' relationships: DHCP, NFS, firewalls, . . .

- ▶ Not sufficient to consider each node's config in isolation
- ▶ Need to gather data across the fabric, and republish to nodes as appropriate
- ▶ LCFG has 'spanning maps': limited scope, non-robust implementation
- ▶ Quattor (Pan) has nothing built in: can retrieve data indirectly via SQL server module
- ▶ Neither approach satisfactory

### Challenges 3: devolved management and specification conflicts

As scale increases, single points of failure (or responsibility) increasingly unacceptable. So devolve configuration process:

- ▶ Different administrators work on different *aspects* of the site's configuration
- ▶ But existing config languages force overspecification (e.g., absolute list positions for boot services) . . .
- ▶ . . . and rely on linear ordering of config statements to resolve default conflicts:

linux  $\Rightarrow$  xdm

server  $\Rightarrow$  no xdm

server with display  $\Rightarrow$  xdm

## Challenges 4: fault tolerance and dynamic reconfig

WP4 has novel fault tolerance system: *correlation engine* examines monitoring data; fires *actuators* to tweak the configuration. But not yet integrated with config system. Unclear whether can handle issues on systematic scale. Architectural conflict with current language design?

But need dynamic reconfiguration:

- ▶ Flexible application environments: job determines config (plug: <http://groups.inf.ed.ac.uk/ogsconfig/>)
- ▶ User control: desktops (packages, services, firewall...)
- ▶ Autonomics: node gets to fix things itself

## Challenges 5: usability

Robustness:

- ▶ Hardware autodetect, package dependency checking, etc.
- ▶ Language: error checking, types, validation

Dealing with extra layer of indirection:

- ▶ Tracing
- ▶ Explanation
- ▶ Integration with monitoring

## Future directions

- ▶ Constraints & declarative template composition. Good for handling aspect composition. Intuition: express underlying administrator intentions more explicitly.
- ▶ Autonomic/peer-to-peer architectures. 'Asymptotic configuration'. Allow nodes to determine intended state for themselves, and reconfigure to match.

Quattor may well see increasing use in LCG, but not as part of core middleware.

Lack of integrated fabric management in LCG/EGEE/GridPP2. Risk that developers & sites end up duplicating work? —and not getting the results they need.