Expanding scalability of LCG CE

A. Kiryanov, PNPI.

globus@pnpi.nw.ru

Why LCG CE?

- Probably the most ancient piece of LCG/gLite software which is still there
- Based on Globus/VDT with some extra bits like LCAS/LCMAPS, DGAS accounting, LCG jobmanagers
- GRAM interface, could be used both directly via Globus API/CLI and via RB/WMS or Condor-G
- Has visible scalability problems for quite a long time

Scalability problems

- Was not initially designed to deal with so many users and jobs at once
  - It's not LCG fault, it's initial Globus architecture
  - Situation became even worse with pool accounts and VOMS roles
- "One process per job" or "one process per user" approach
  - All processes run in parallel, no queues or other machinery to control this
- As a consequence more than 10 users working in parallel can draw LCG CE completely irresponsive (load average over 100)
  - Linux kernel is still far from perfect in handling such a heavy load
  - Takes a light run to CC and back just to press a reset button

Solutions (so far)

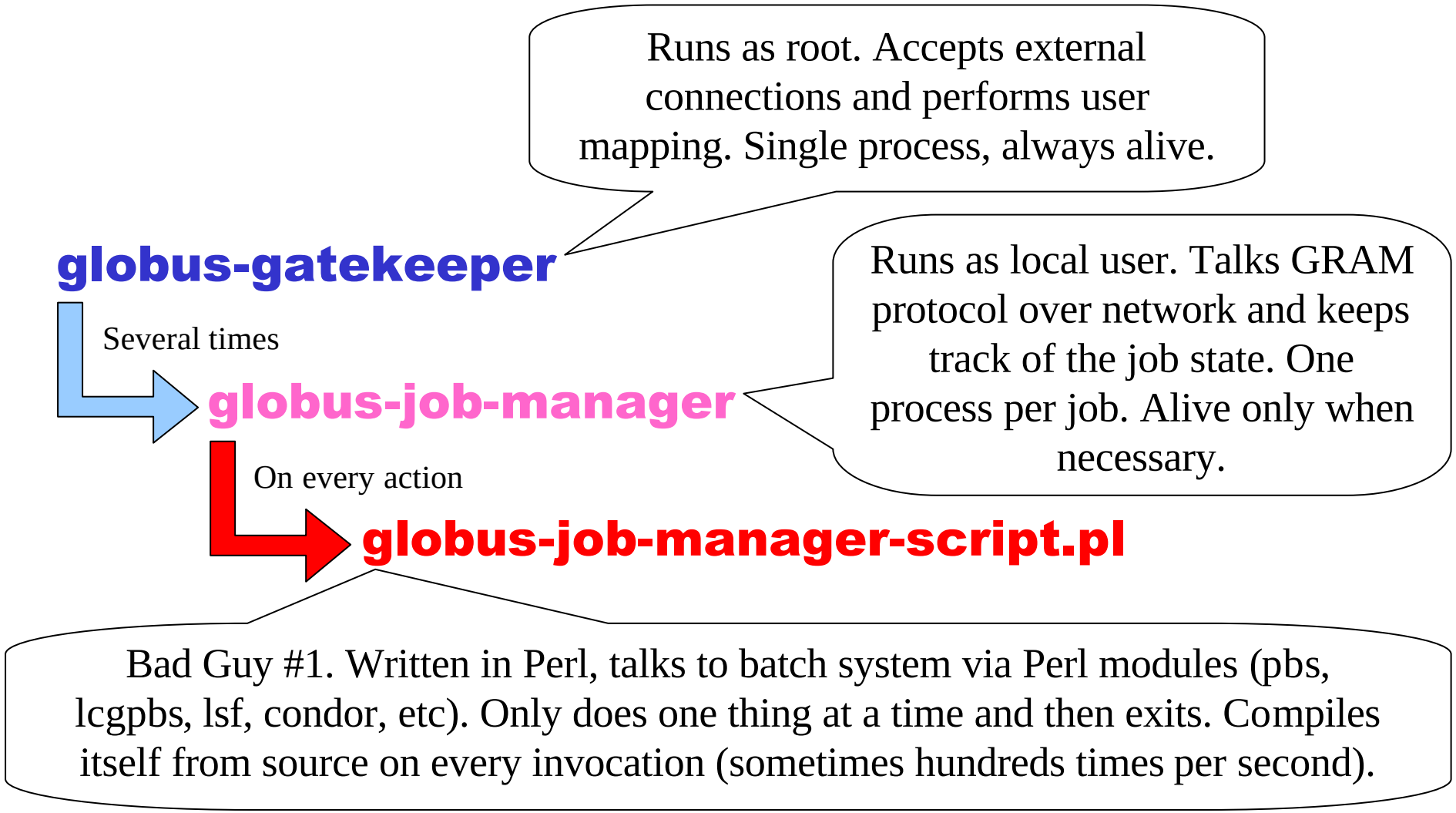
- Use of monitoring tools to keep track of system load on LCG CEs and reboot them as necessary
- Deploy a cluster of CEs to serve a relatively large fabric (LCG CEs cannot be used in fault-tolerant or load-balancing manner)
- Wait for gLite CE (RIP) or Cream CE (pre-beta quality)

- What about now? LHC data will come this year $\Rightarrow$ no time left to start from scratch

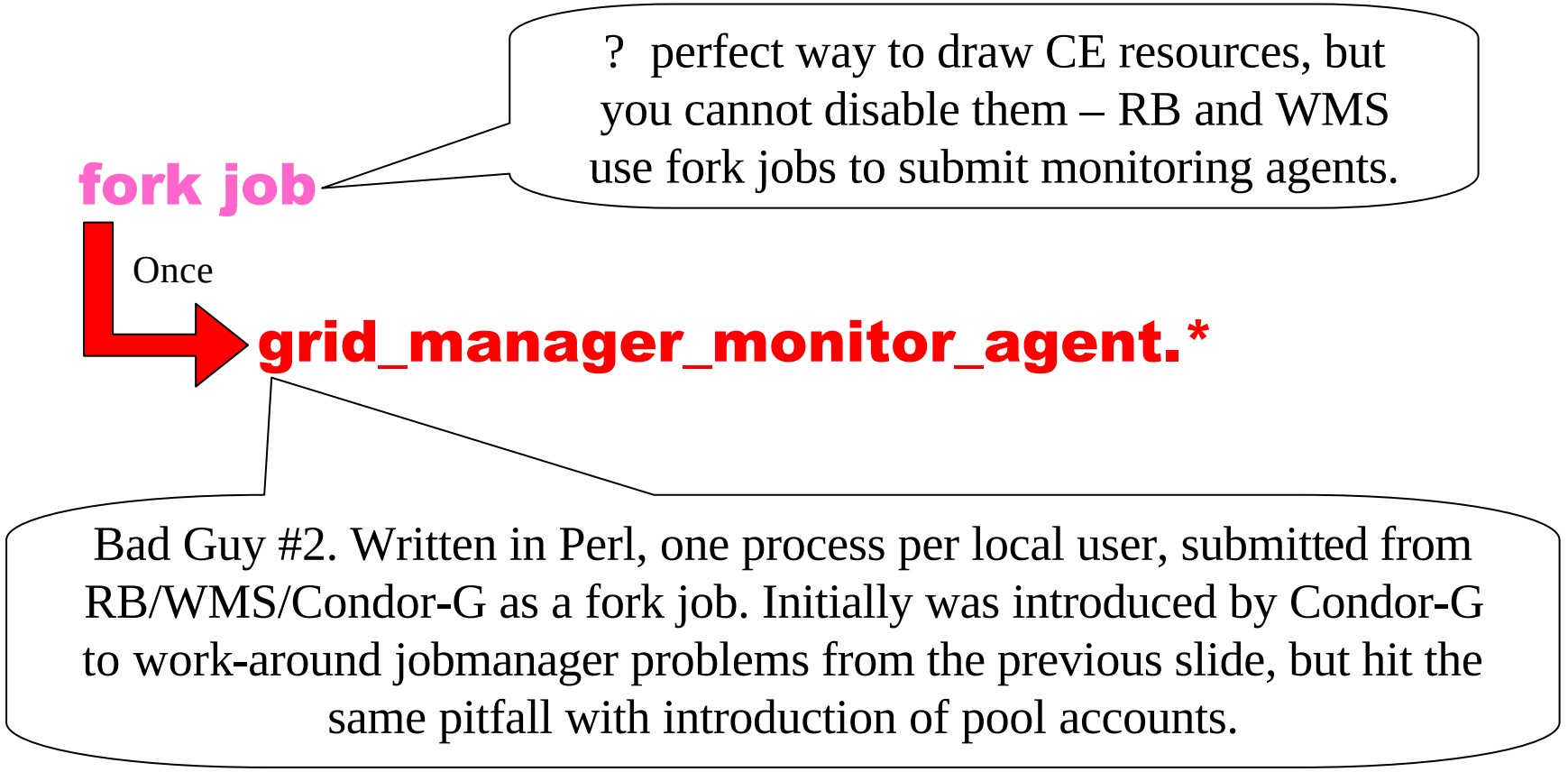
Task description

1. Try to find a problem source
2. Try to get rid of it (partially or completely if possible)
3. Try to make changes as transparent as possible to not ruin existing applications and to avoid a long-term testing and certification
4. Evaluate the impact of changes
5. goto 1

Quest for a seat of the trouble:
internal saboteurs



Quest for a seat of the trouble:
external saboteurs



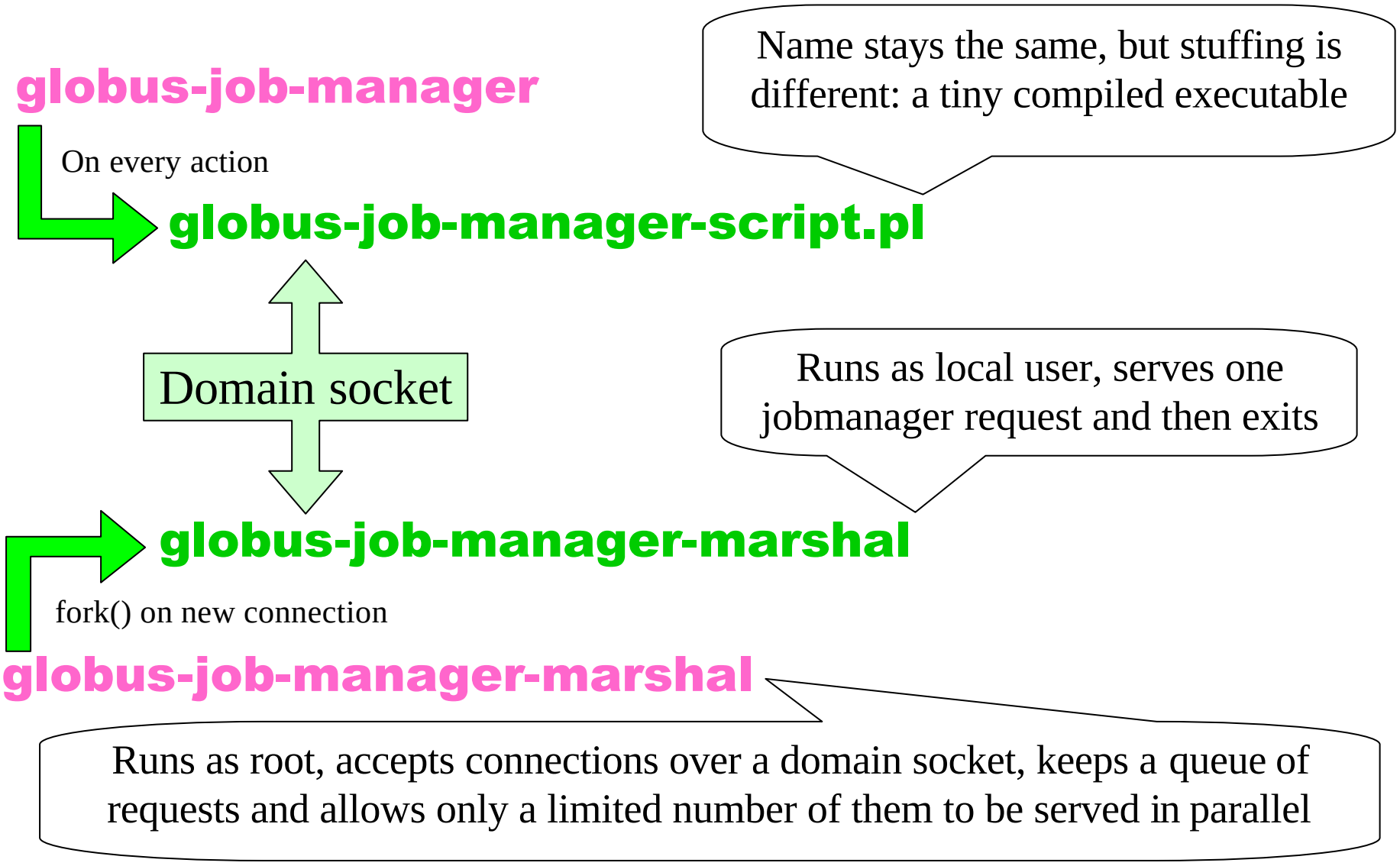
Problem summary

- All these processes running in parallel induce a very high load on a system
- Doing everything in parallel is actually much slower than doing it in series
- Frequent recompilation of a Perl code, that never changes, is a waste of resources

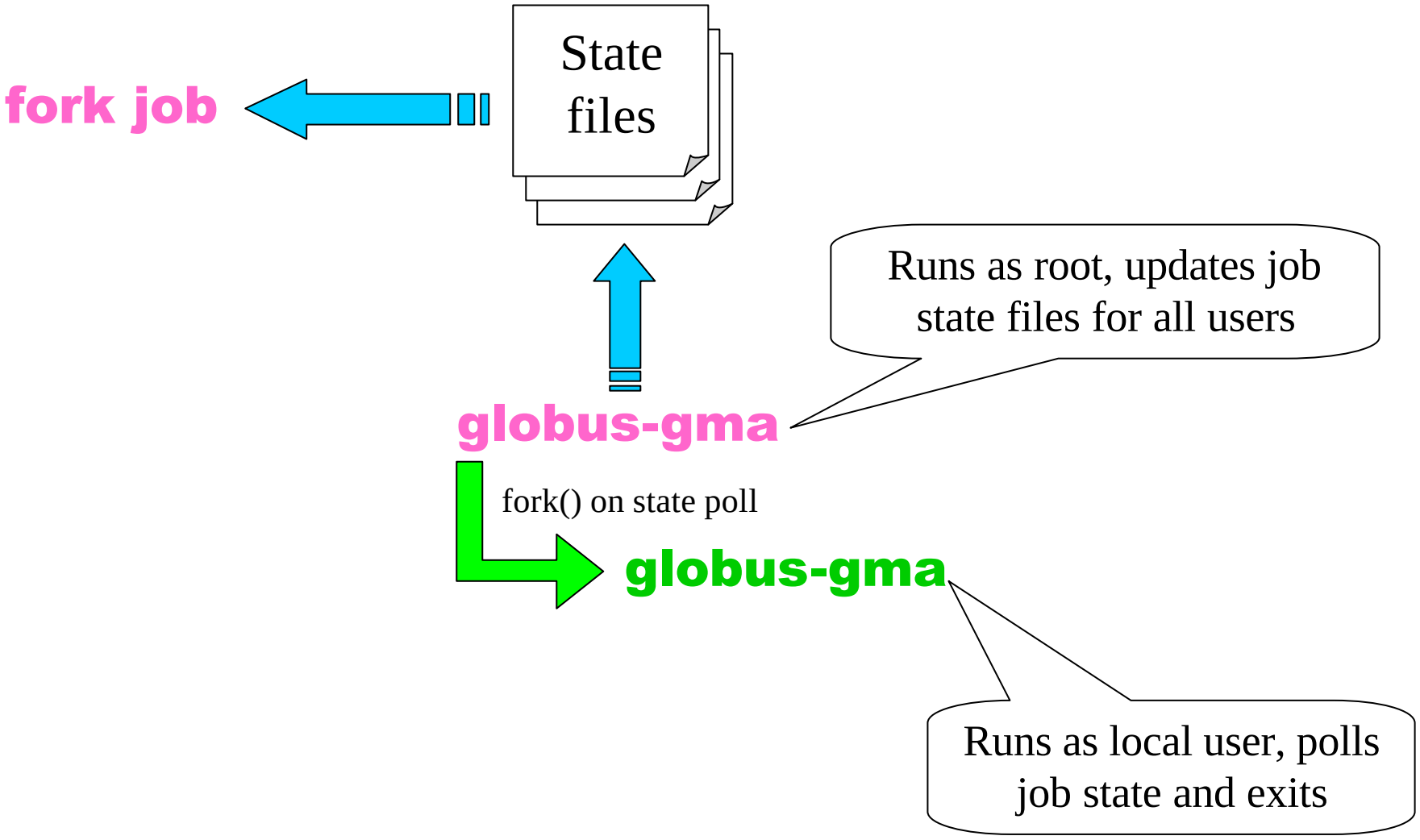
Modifications

- Perl Jobmanagers
  - A memory-persistent daemon was developed to avoid Perl code recompilation and to provide queue-based control over parallel requests from binary jobmanagers
  - Perl script was replaced by a tiny client written in C, that communicates with daemon via domain socket
- Grid Manager Monitor Agents
  - An existing machinery inside “fork” jobmanager was used to detect incoming agents and replace them with modified “light” ones
  - A monitoring daemon was developed to supply job state information to “light” monitoring agents

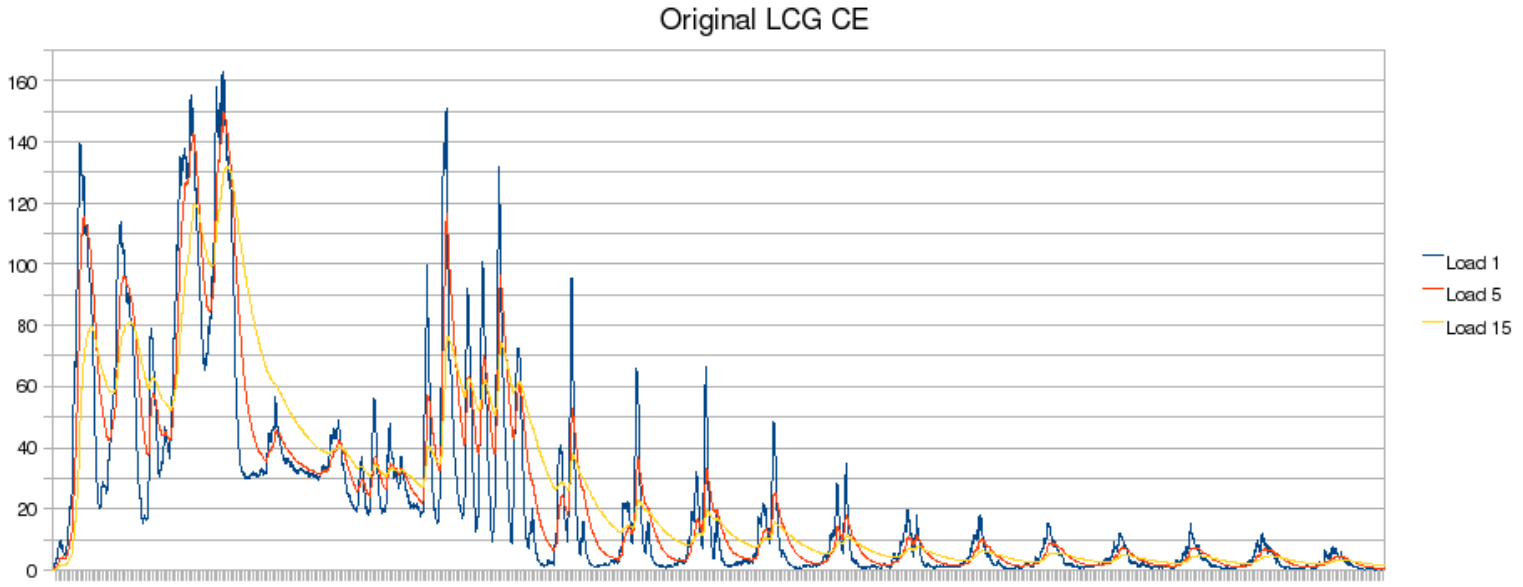
Perl jobmanagers reinvented



Grid Monitor Agents reinvented as well



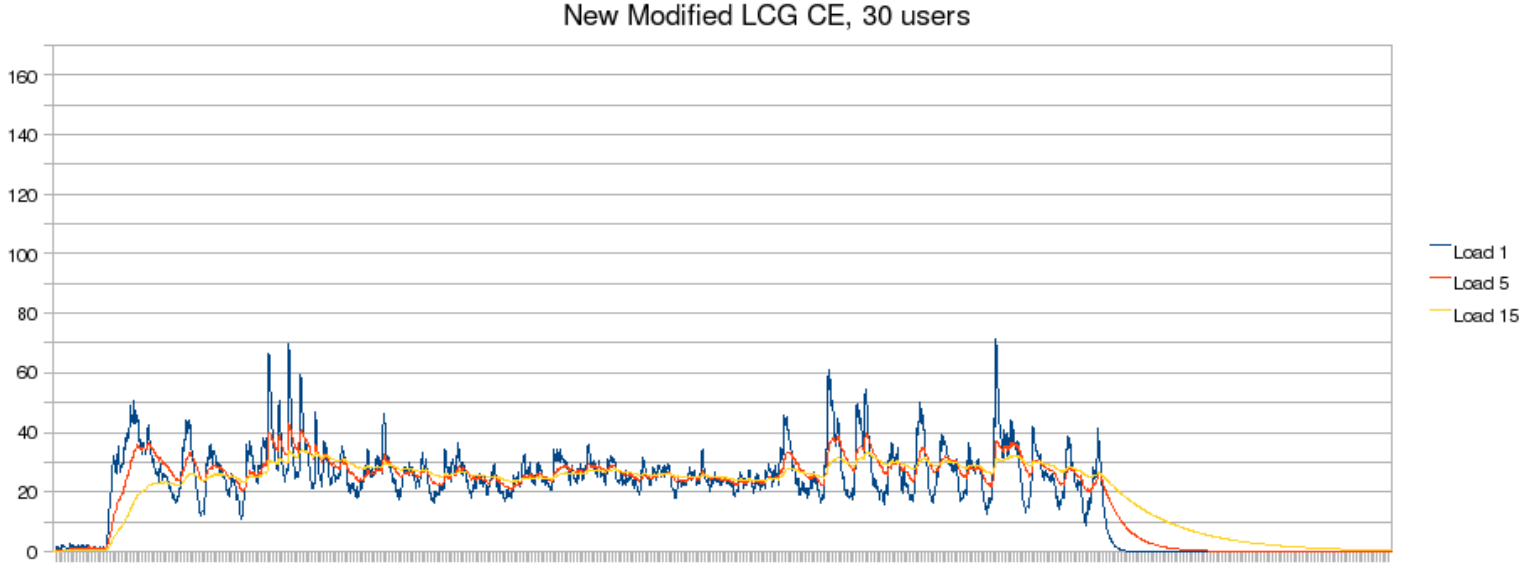
Test results (1)



Original version:

10 users

1000 jobs

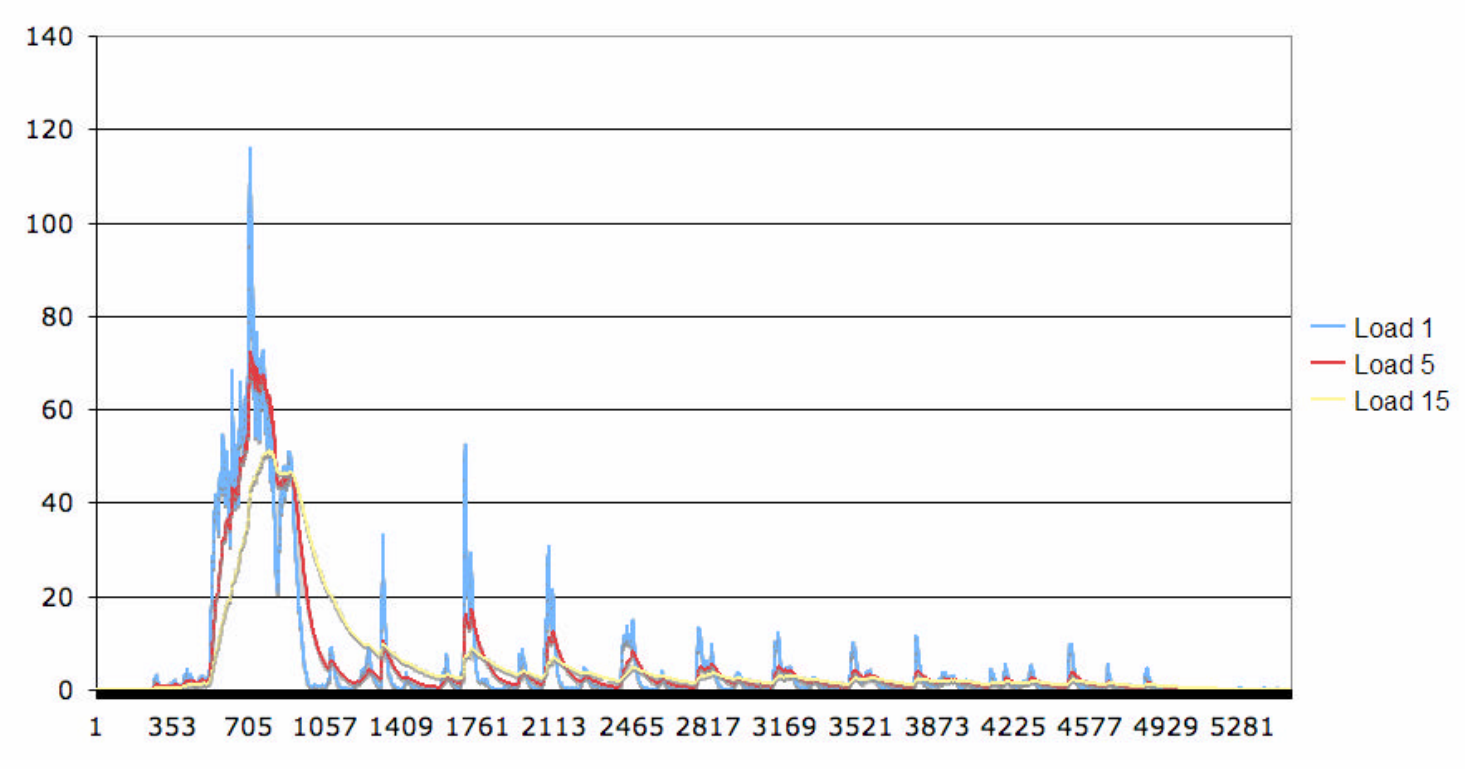


Modified version:

30 users

4000 jobs

Test results (2)



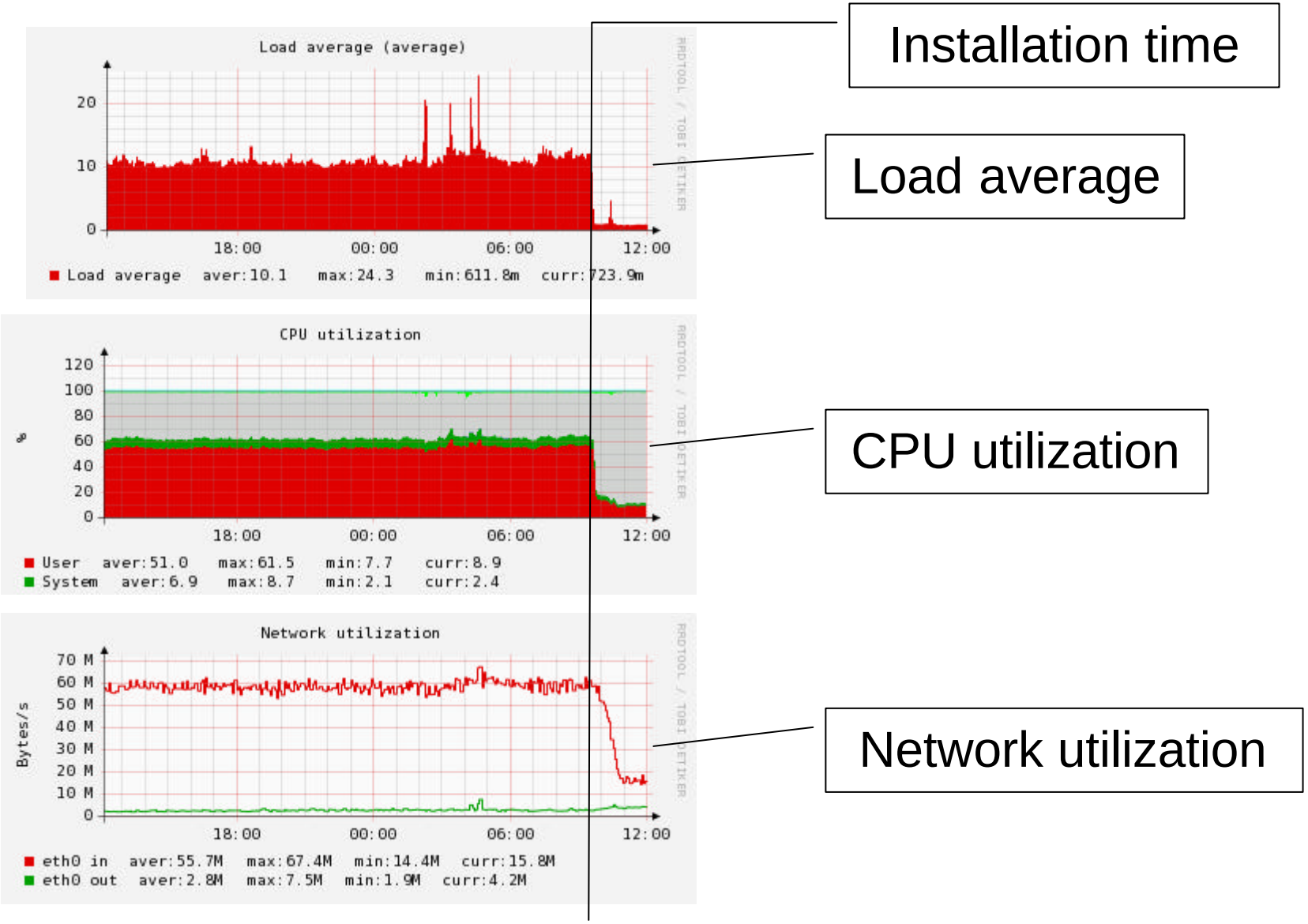
Last version:

50 users

4000 jobs

Stress test of the last version with both globus-gma and globus-job-manager-marshal packages installed on Certification TestBed

Installation on CERN CE cluster



Achievements

- System load on CE is decreased by factor of 3 to 5
- Jobs start and finish faster (especially visible with lots of short jobs)
- CE can handle significantly larger number of jobs and different users
- No interface changes, no need to modify existing software
- Software can be tuned for CEs with different CPU/disk performance
- Modified LCG CEs are in LCG/gLite production since April 2008, already installed on many sites
- Modifications made are actually not LCG-dependant. Software can be used by any Grid infrastructure that relies on Globus 2.x style CE

Thank you!