AP 2: Monitoring

Job Execution Monitor

Stefan Borovac, Joachim Clemens,
Torsten Harenberg, Matthias Hüsken,
Peter Mättig, Markus Mechtel,
David Meder-Marouelli, Peer Ueberholz

HEPCG Workshop, November 30th, 2006

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Outline

- Motivation
- Goals
- Solution
- Architecture
- Job Execution Monitor
  - Watchdog
  - Script wrapper
- Summary
- Outlook

Motivation

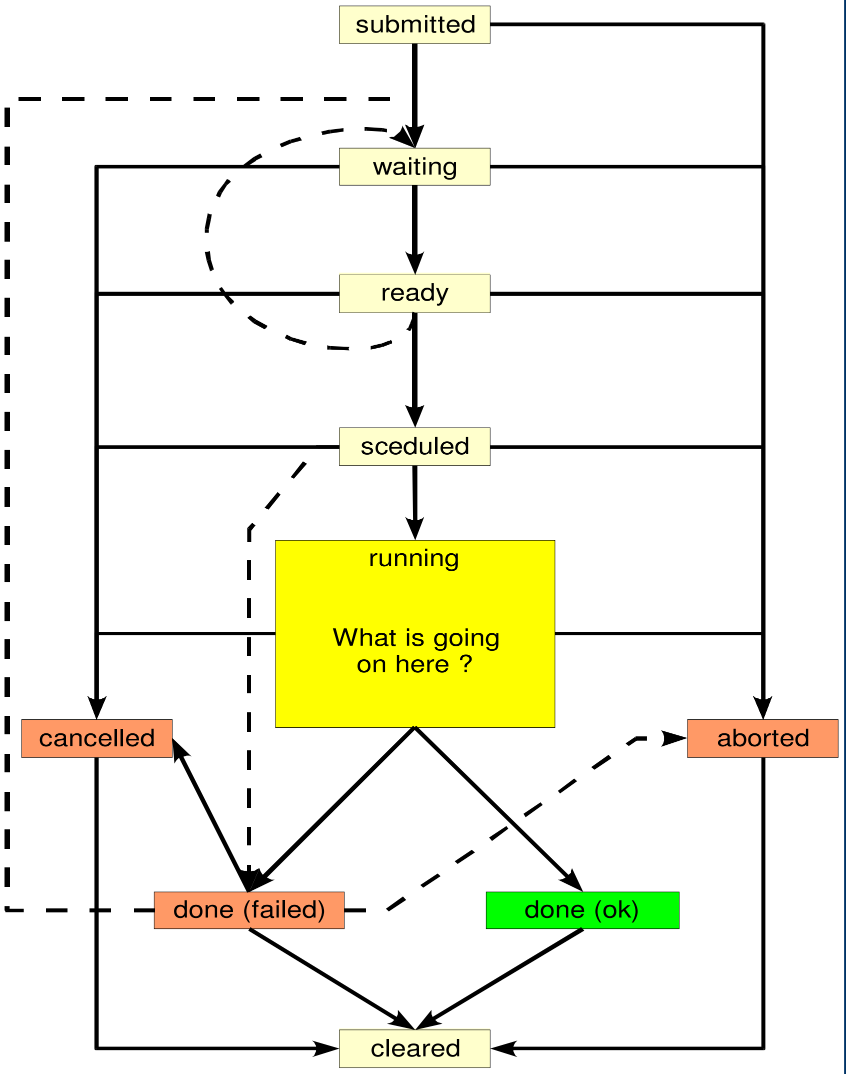
LCG jobs either

- finish **ok** or
- **fail**

user doesn't get any information about the reason of job failures

~30% of real LCG jobs fail

LCG-software evaluates function of grid-infrastructure only, not the results of jobs



sources of failures

possible sources of failures

- grid middleware configuration
- workernode configuration
- workernode problems
  - disk full
  - lost network connection
  - firewall misconfiguration
  - ...
- missing software (e.g. required libraries)
- problems in user software
- ...

Solution

- job monitoring on the workernode
  - stepwise execution of script files
  - on internal errors → run job without monitoring
  - process monitoring
- realtime information
  - user knows current state of his jobs
  - stdout/stderr access even in case of failures (in LCG only available on successful jobs)
- graphical user interface for easy access

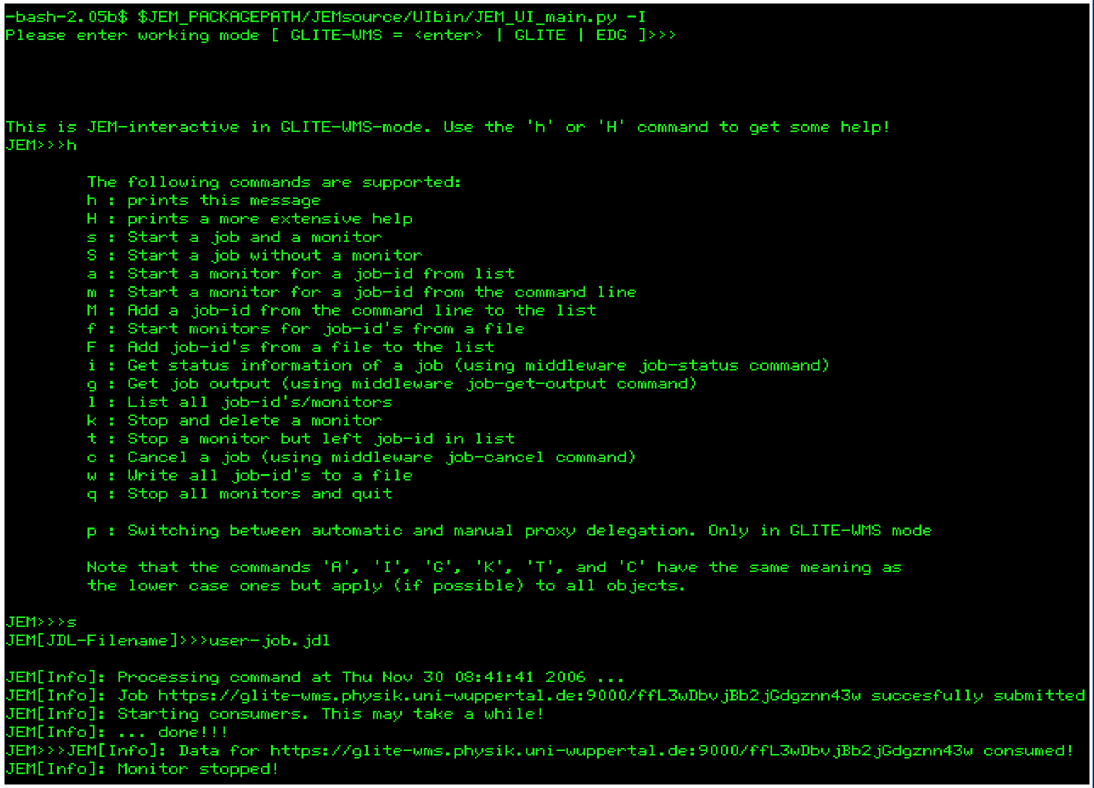
  → Dresden

Architecture

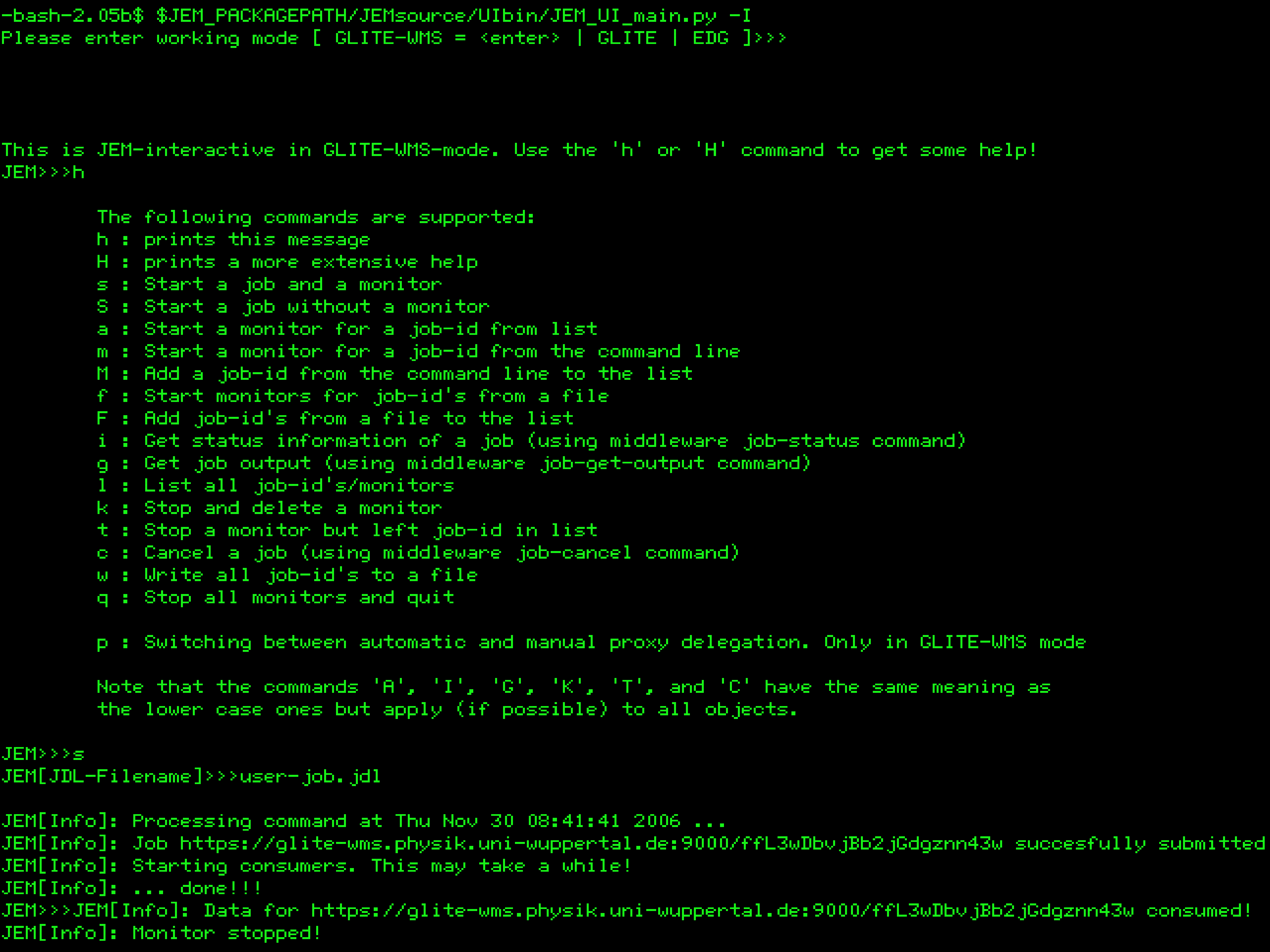
- Python
  - installed on every LCG machine
  - platform-independant
    (programs run on 64bit CPUs without modification)
- information exchange via R-GMA only
  - preserves security context
  - no firewall problems

Features

- stepwise execution of bash and python scripts
- regular output to R-GMA (job status, system resources)
- resource usage graphics on UI
- command line interface/menu
- detailed logfile

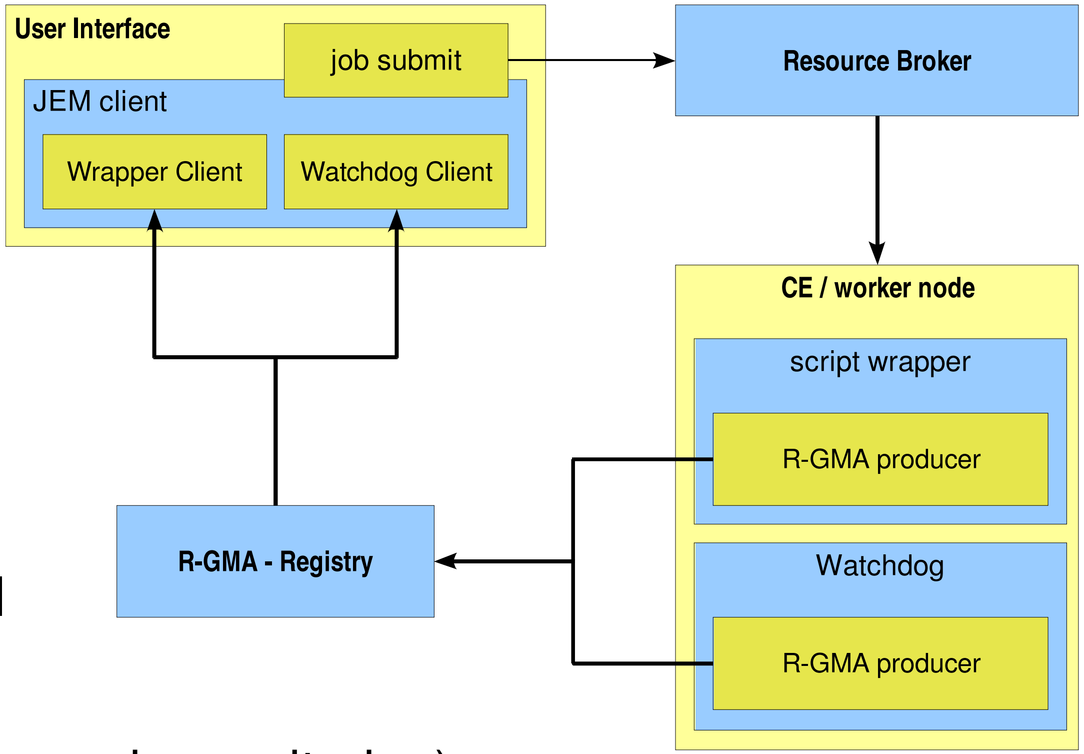


```
-bash-2.05b$ $JEM_PACKAGEPATH/JEMsource/UIbin/JEM_UI_main.py -I
Please enter working mode [ GLITE-WMS = <enter> | GLITE | EDG ]>>>



This is JEM-interactive in GLITE-WMS-mode. Use the 'h' or 'H' command to get some help!
JEM>>>h

        The following commands are supported:
        h : prints this message
        H : prints a more extensive help
        s : Start a job and a monitor
        S : Start a job without a monitor
        a : Start a monitor for a job-id from list
        m : Start a monitor for a job-id from the command line
        M : Add a job-id from the command line to the list
        f : Start monitors for job-id's from a file
        F : Add job-id's from a file to the list
        i : Get status information of a job (using middleware job-status command)
        g : Get job output (using middleware job-get-output command)
        l : List all job-id's/monitors
        k : Stop and delete a monitor
        t : Stop a monitor but left job-id in list
        c : Cancel a job (using middleware job-cancel command)
        w : Write all job-id's to a file
        q : Stop all monitors and quit

        p : Switching between automatic and manual proxy delegation. Only in GLITE-WMS mode

        Note that the commands 'A', 'I', 'G', 'K', 'T', and 'C' have the same meaning as
        the lower case ones but apply (if possible) to all objects.

JEM>>>s
JEM[JDL-Filename]>>>user-job.jdl

JEM[Info]: Processing command at Thu Nov 30 08:41:41 2006 ...
JEM[Info]: Job https://glite-wms.physik.uni-wuppertal.de:9000/ffL3wDbvjBb2jGdgznn43w succesfully submitted
JEM[Info]: Starting consumers. This may take a while!
JEM[Info]: ... done!!!
JEM>>>JEM[Info]: Data for https://glite-wms.physik.uni-wuppertal.de:9000/ffL3wDbvjBb2jGdgznn43w consumed!
JEM[Info]: Monitor stopped!
```



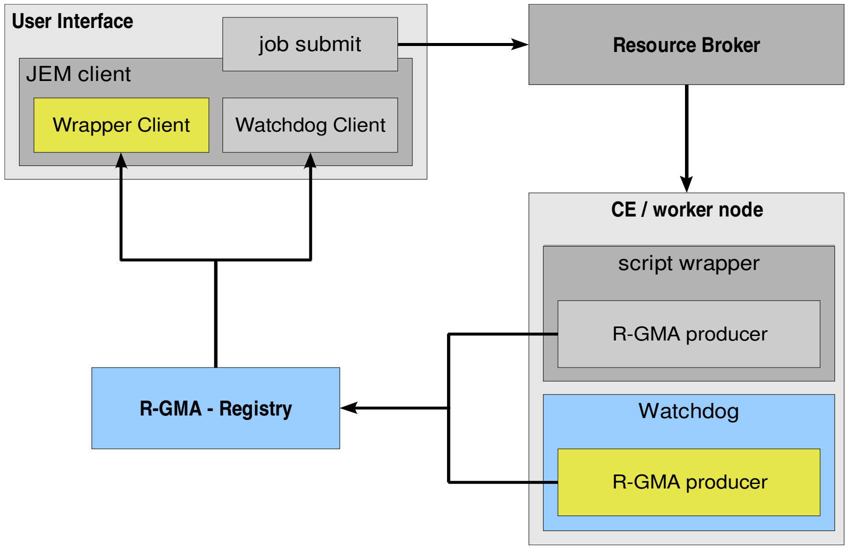
```
-bash-2.05b$ $JEM_PACKAGEPATH/JEMsource/UIbin/JEM_UI_main.py -I
Please enter working mode [ GLITE-WMS = <enter> | GLITE | EDG ]>>>




This is JEM-interactive in GLITE-WMS-mode. Use the 'h' or 'H' command to get some help!
JEM>>>h

        The following commands are supported:
        h : prints this message
        H : prints a more extensive help
        s : Start a job and a monitor
        S : Start a job without a monitor
        a : Start a monitor for a job-id from list
        m : Start a monitor for a job-id from the command line
        M : Add a job-id from the command line to the list
        f : Start monitors for job-id's from a file
        F : Add job-id's from a file to the list
        i : Get status information of a job (using middleware job-status command)
        g : Get job output (using middleware job-get-output command)
        l : List all job-id's/monitors
        k : Stop and delete a monitor
        t : Stop a monitor but left job-id in list
        c : Cancel a job (using middleware job-cancel command)
        w : Write all job-id's to a file
        q : Stop all monitors and quit

        p : Switching between automatic and manual proxy delegation. Only in GLITE-WMS mode

        Note that the commands 'A', 'I', 'G', 'K', 'T', and 'C' have the same meaning as
        the lower case ones but apply (if possible) to all objects.

JEM>>>s
JEM[JDL-Filename]>>>user-job.jdl

JEM[Info]: Processing command at Thu Nov 30 08:41:41 2006 ...
JEM[Info]: Job https://glite-wms.physik.uni-wuppertal.de:9000/ffL3wDbvjBb2jGdgznn43w succesfully submitted
JEM[Info]: Starting consumers. This may take a while!
JEM[Info]: ... done!!!
JEM>>>JEM[Info]: Data for https://glite-wms.physik.uni-wuppertal.de:9000/ffL3wDbvjBb2jGdgznn43w consumed!
JEM[Info]: Monitor stopped!
```

Job Execution Monitor



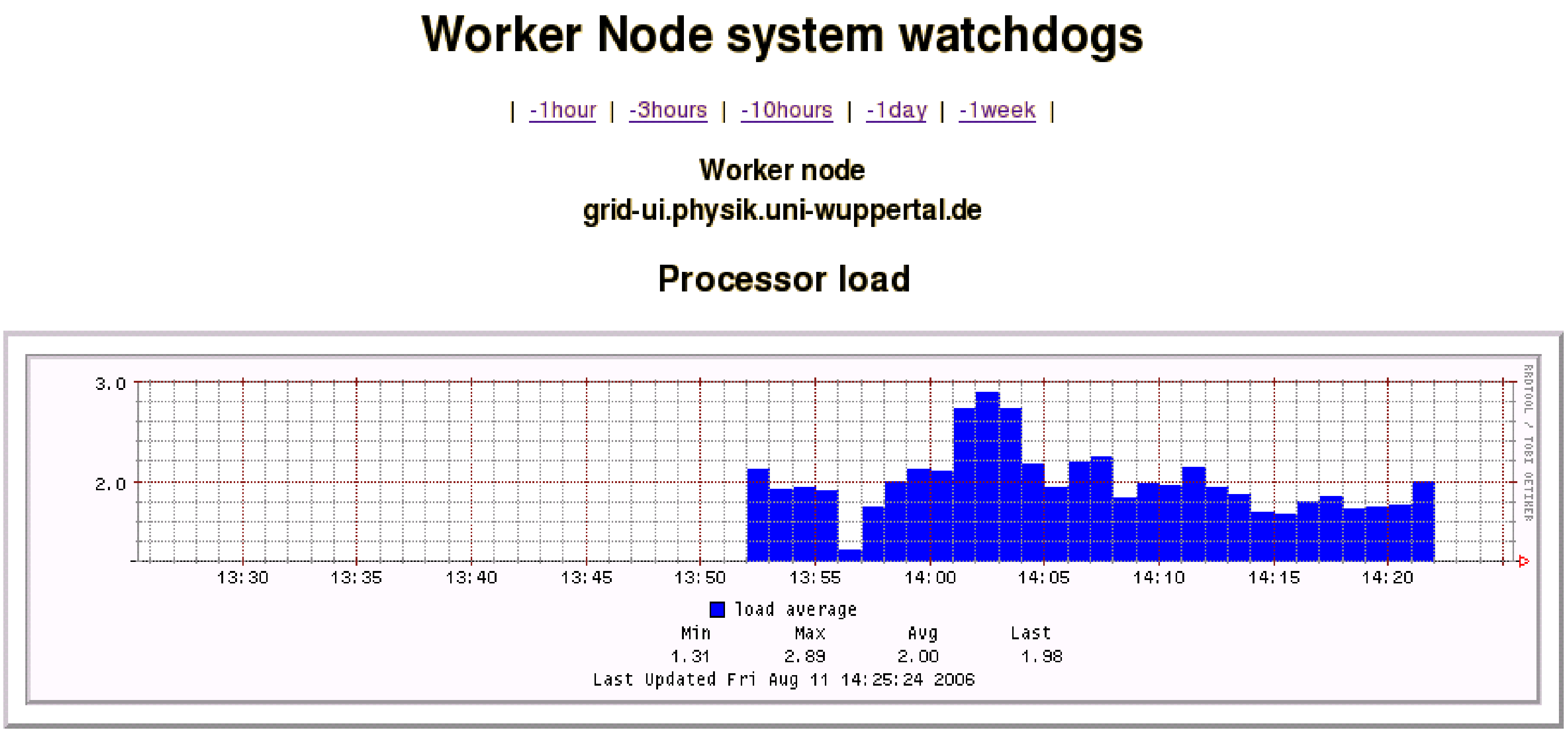
- client/server modell
- 2 components
  - watchdog (workernode monitoring)
  - script wrapper (stepwise execution of script files)
- monitoring system is automatically added to the user's job submission, no additional work for user

Watchdog



- monitors system resources:
  - free memory
  - free disk space
  - network I/O
  - processor load
- provides graphical representation using rrdtool
- data published regularly via R-GMA

Watchdog



Worker Node system watchdogs

| -1hour | -3hours | -10hours | -1day | -1week |

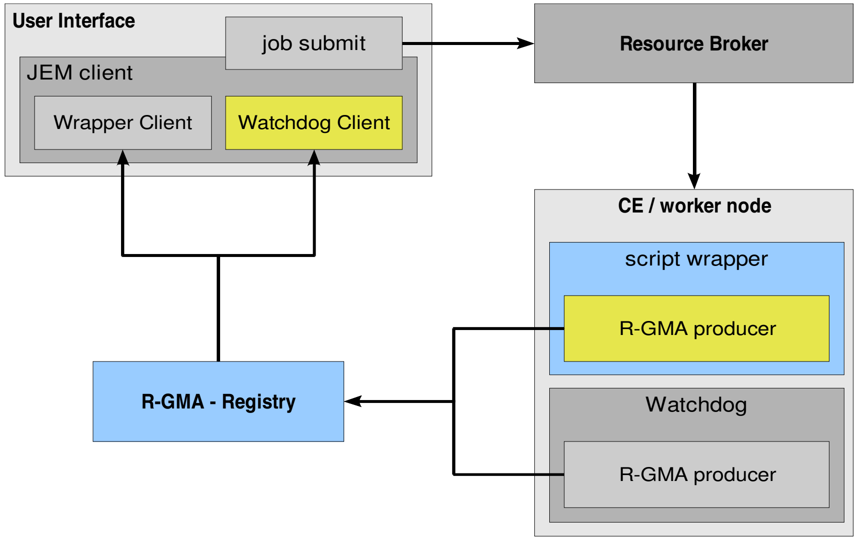
Worker node
grid-ui.physik.uni-wuppertal.de

Processor load

```
■ load average
      Min        Max        Avg        Last
      1.31       2.89       2.00       1.98
Last Updated Fri Aug 11 14:25:24 2006
```

- temporal behaviour of system resources
- planed to merge watchdogs with resource monitoring → Dresden

Script-wrapper



characteristics:

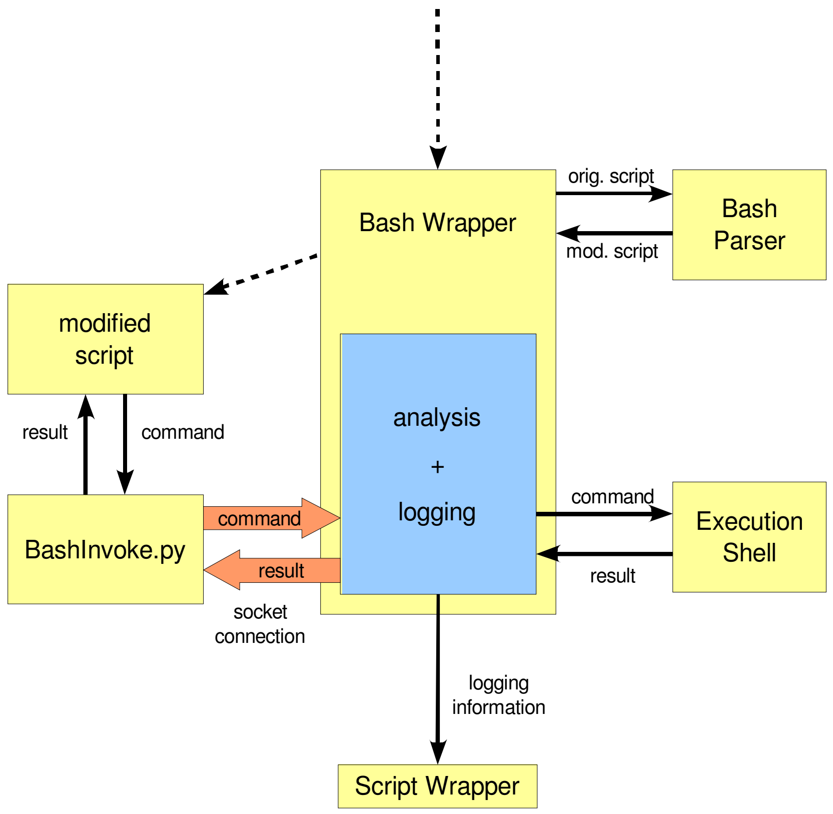
- stepwise execution

  →traceback of script execution in case of errors
- lexical and semantical analysis of script files
- data published regularly via R-GMA
- critical actions are made more robust, e.g. file transfers
- Languages: bash, python (more can be easily added)

Bash-wrapper and -parser

operating principle:

- parser identifies commands in shell-script
- wrapper starts isolated shell
- wrapper starts modified shell-script
- modified script sends single commands to subshell
- subshell executes commands separately
- wrapper monitors and logs results of commands



Python-wrapper

- Python provides mechanisms for monitoring execution
- operating priciple:
  - get environment from execution shell
  - run python script with monitoring
  - set (new) environment in execution shell

Summary

reached goals:

- stepwise execution of bash- and python-scripts
- more languages can be easily added
- traceback of failures possible
- monitoring of system resources
- much more information about job execution

http://www.grid.uni-wuppertal.de/jms

Outlook

work in progress:

- identification of job failures
- classification of errors
- expert system to automatically recover error conditions
- harden commands, known to be critical

timetable:

- job wrapper could enter LCG/gLite release soon