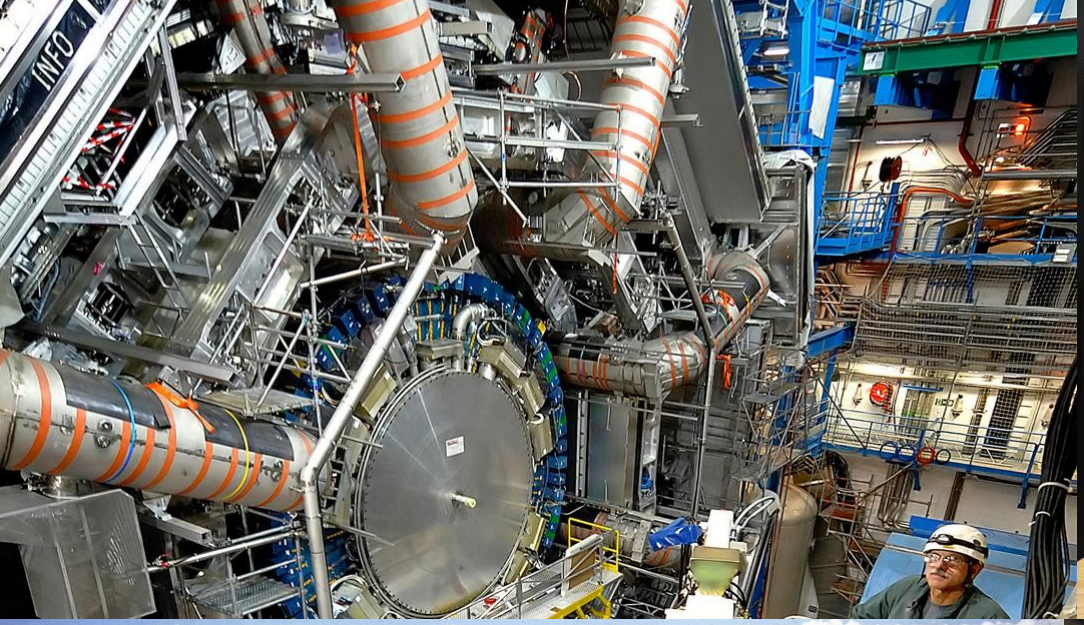



Controlling and utilizing global job centric monitoring for PanDA

Raphael Ahrens, Thomas Beermann, Torsten Harenberg,
Peter Mättig, Tim dos Santos, Joachim Schultes*,
Frank Volkmer, Benjamin Wolff
Bergische Universität Wuppertal

Agenda

- Motivation and Current Status:
  The Job Execution Monitor in PanDA
- Controlling the Monitor:
  Central "Controller" Web-Service
- Utilisation of the monitoring data:
  Django-based user interface
- Outlook

Ph.D. Thesis:
Frank Volkmer,
M.Sc. Thesis:
Benjamin Wolff

Follow up projects:
Thomas Beermann, Raphael Ahrens

Motivation and current status
JEM and PanDA

Motivation

- Available monitoring solutions offer
  - “Post-Mortem“-overviews using the final job state (succeeded, failed)
  - Overviews of the current situation (running jobs) based on some-minutes-old data
- Classification between site- or taskdef-problem is in the responsibility of the shifter
- Real-time status (“now“) can be difficult to see
- Trends (“more and more failures at XYZ”), too

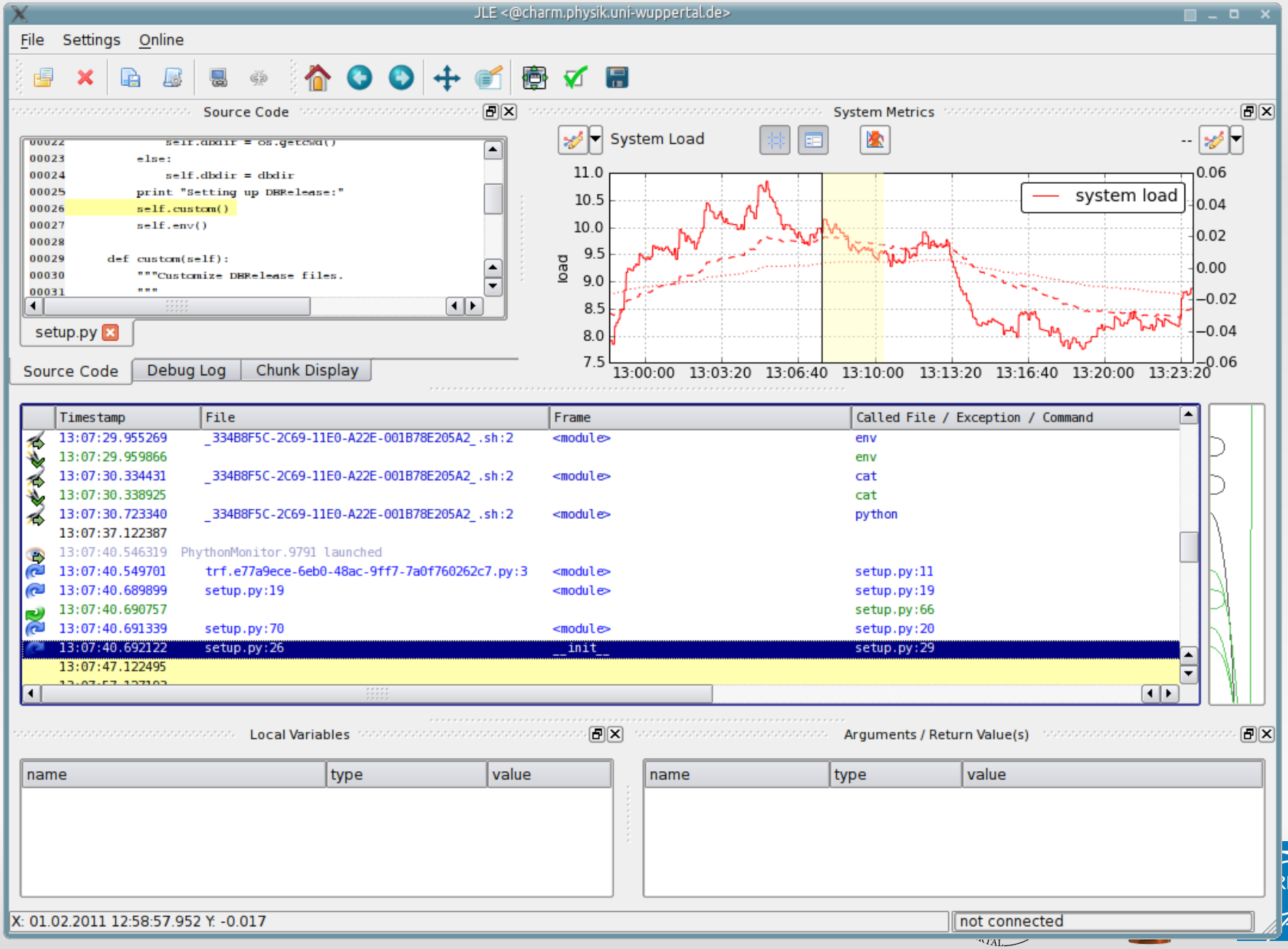
Application of JEM

- JEM (The Job Execution Monitor) can be applied to provide a real time view
- Hence: Provide data in higher detail, at small delay / time granularity, at negligible overhead
  - We performed scaling tests - preliminary results:
    - Job-Efficiency: comparable to „without JEM“
      - difficult to measure because of much noise, even between consecutive job runs on same site *without* JEM
      - JEM vs. non-JEM differences dominated by inherent noise: good sign!
    - Job-Success rate: unchanged

Current status of JEM in PanDA

- JEM is integrated into PanDA for end-user application
- Using the PanDA client (CLI or Python-API), JEM can be enabled and configured on a per-job basis easily
- Pilots only fetch and apply JEM when enabled
- Results are available as log files (after job terminated) or in real-time, using JEM-specific (non-web) tools:

Current status of JEM in PanDA



Follow-up project #1:
Central “Controller“ Web-Service

Central “Controller“ Web-Service

- Firstly, it must be decided:

  Who may activate JEM-monitoring for which jobs, at what sites, in what verbosity?

- Because there are numerous groups of interest
  - Shifters may want to supervise a certain set of jobs (on some site...) in more detail
  - End users may want to monitor their jobs in great detail
  - Site-Admins may want to temporarily lessen / disable / increase the monitoring on their site
- Real time monitoring can be overdone...

Central “Controller“ Web-Service

Our proposed solution:

- A central “Controller” Web-Service will be established
- Pilots ask the Controller whether – and if so, in what verbosity – to monitor the job they fetched
- The Controller decides on the requests based on a job→monitoring mapping
- This way, overrides in both direction (“more” or “less” monitoring) becomes possible

Central “Controller“ Web-Service

It must be

- decided,
- designed
- and implemented,

who may modify this mapping, and how – e.g.:

- shifters and/or site admins may need to add temporary entries in the mapping
- end users must **not** be able to finally override the mapping, to prevent misuse
- etc

Follow-up project #2:
Django-based user interface

Django-based user interface

All real time JEM data is transferred via an ActiveMQ-broker

- Obviously it's worthwhile to aggregate and further process the data there, centrally

  → summaries / history plots / trends
- For this purpose, we feed a database directly at broker level and develop a Django-based view onto it
- Django was chosen for easy prototyping

Django-based user interface

- Prototyping in Django allows easy addition of
  - New analyses, correlations, filters, sorting...
  - Visualisation of the data (plots, histograms)
  - Machine-to-machine (M2M) Interface (JSON)
- Also, the planned end user interface (in-depth visualisation and interactive examination of all monitoring data of one job) can be done this way
  - Similar to the shown QT GUI
  - Interactive elements: jQuery

Benjamin Wolff

Outlook

Outlook

- Short-Term: Provide (via Django. web-based)
  - Live Job Health, Live Site Health pages
  - Trend analyses (Mem-leaks, Resource-Hogs)
  - Outlier detection
- Mid-Term: using M2M-data (JSON) – e.g. by multivariate tools – trigger automatic measures:
  - “Red Light“ for shifters
  - Mail-notifications for site admins
  - …

Outlook

- Further follow-up projects in planning state:
  - Thomas Beermann
    - Performance Tuning of the MQ and data aggregation
    - Data transfer monitoring using JEM
  - Raphael Ahrens
    - Real time back channel to a single job
    - Based on secure (→ Grid-Cert.) MQ communication
    - Allows end user to send commands to the running job (e.g. control monitoring verbosity, request logfile contents)

Project page:

→ https://svn.grid.uni-wuppertal.de/trac/JEM

Questions? Comments?