omii europe

open middleware infrastructure institute

Benchmarking of integrated OGSA-BES with the Grid middleware

Hedman, Riedel, Mucci, Netzer, Gholami, M. Memon, A. Memon, Shah

Outline

- **Batch Job Execution Interfaces**
- **Benchmark details**
- **Results**
  - System Level
  - Component Level
- **Conclusions**

Introduction

- **Most Grids offer services to execute batch jobs.**
- **Middleware traditionally used proprietary protocols to provide these services.**
- **OGF standardized job submission in the BES recommendation.**
- **The OMII-Europe project developed BES implementations for three middleware stacks (Globus, gLite and UNICORE).**
- **This benchmarking effort targets this new services and tries to provide information about the performance of the developed solutions.**

UNICORE Atomic Services (UAS)

- **Exposes core functionality via Web-Service interfaces**
- **Specific to the UNICORE 6 stack**
- **Follows OASIS WS-RF pattern**
- **Provides job control, data management and file transfer**
- **We only consider job submission and control**
  - Target System Service for job submission
  - Job service to manage job

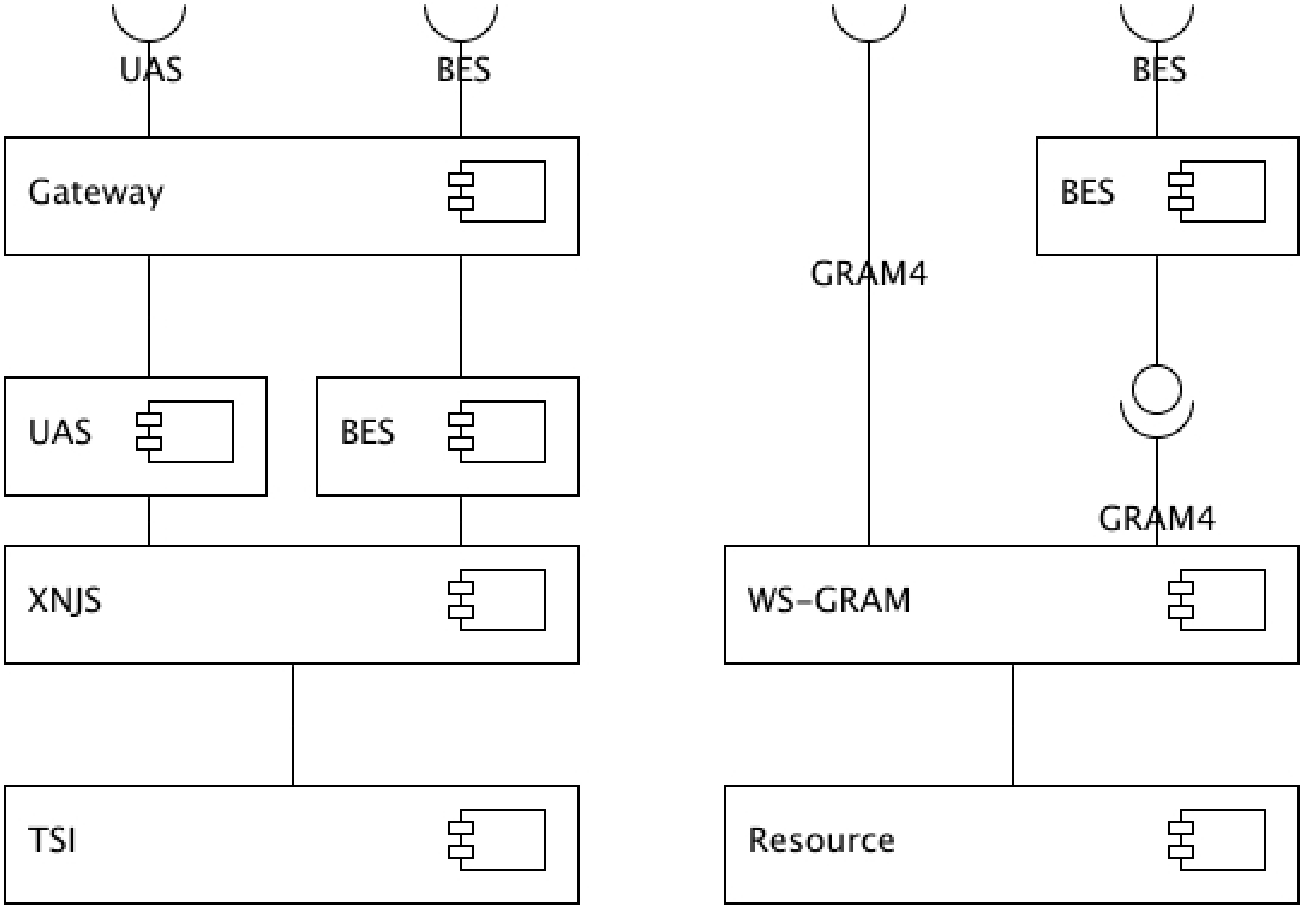
Basic Execution Service (BES)

- **Web-Services based interface for job submission**
- **Standardized by the Open Grid Forum**
- **Consists of three port types:**
  - BES Factory for job creation and bulk job management
  - BES Activity for management of a single job
  - BES Management for service management tasks
- **Excludes security solution**
  - UNICORE 6 uses SAML and optionally VOMS
  - Globus Toolkit uses proxy certificates
  - gLite uses VOMS proxy certificates

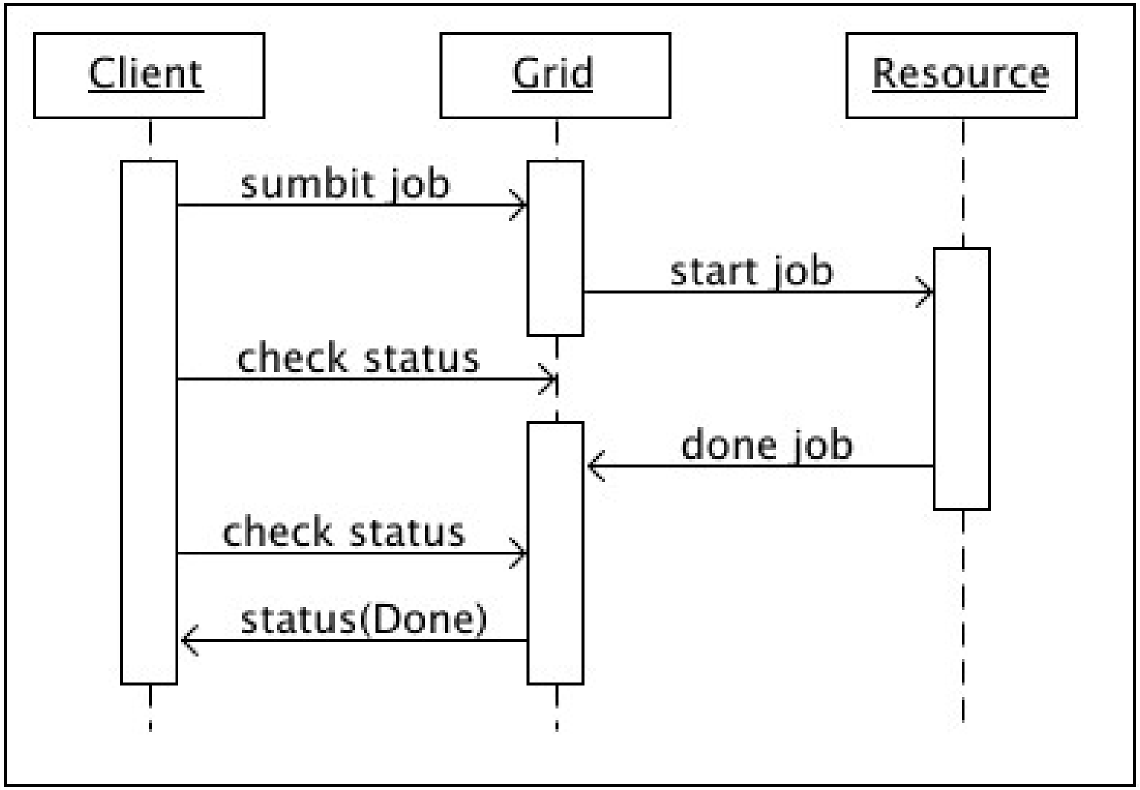
BES Implementations

- **Three Implementations were provided by OMII-Europe**
- **Independent Services**
  - UNICORE BES above XNJS backend
  - gLite CREAM-BES as plugin for the CREAM-CE
- **Wrapper/Adapter approach**
  - Globus BES as a wrapper for a WS-GRAM service
- **CROWN Metascheduler**
  - Can submit jobs to multiple BES instances.
  - Provides its own BES interface.

BES Implementations



The Benchmark



Measure the overhead that the Grid middleware adds to job execution.

Benchmark Variants

- **System Level Approach**
  - Use command line clients provided by MW.
  - Provides performance from end-user perspective.
  - Includes client side startup overhead.
  - High load on client machine limiting factor.
  - Utilizes bulk submission capabilities if available.
- **Component Level Approach**
  - Directly use the web service interface of the MW.
  - More appropriate for server performance measurements.
  - Only overhead for making the WS call are included.
  - More complicated to adopt to new MW stack.

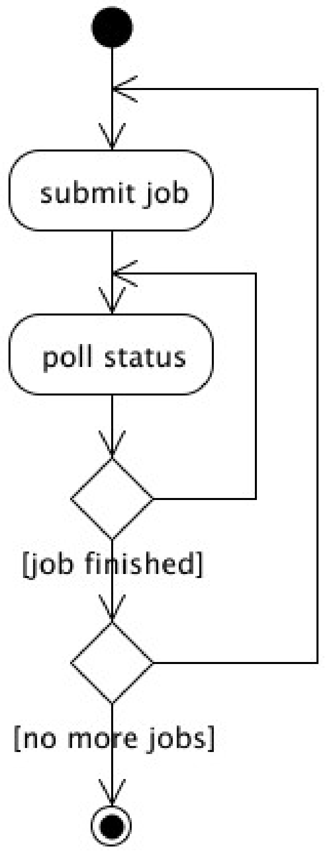
System Level Benchmark

- **Advantages**
  - Measures end-user performance
  - Relatively simple to add new MW stack
  - Shows client side performance
- **Disadvantages**
  - Includes client start up costs
  - Limited by client side performance
  - Complicated to simulate “real” life usage scenario
  - Problems in obtaining compareable results

Component Level Benchmark

- **Advantages**
  - Measures service performance
  - Allows direct comparison between different MW stacks and interfaces.
  - Lower client side load
- **Disadvantages**
  - High development effort to add new MW or interface
  - Cannot benchmark client behavior (e.g. CLIQ)
  - Faces interoperability problems (e.g. security)

Benchmark Implementation

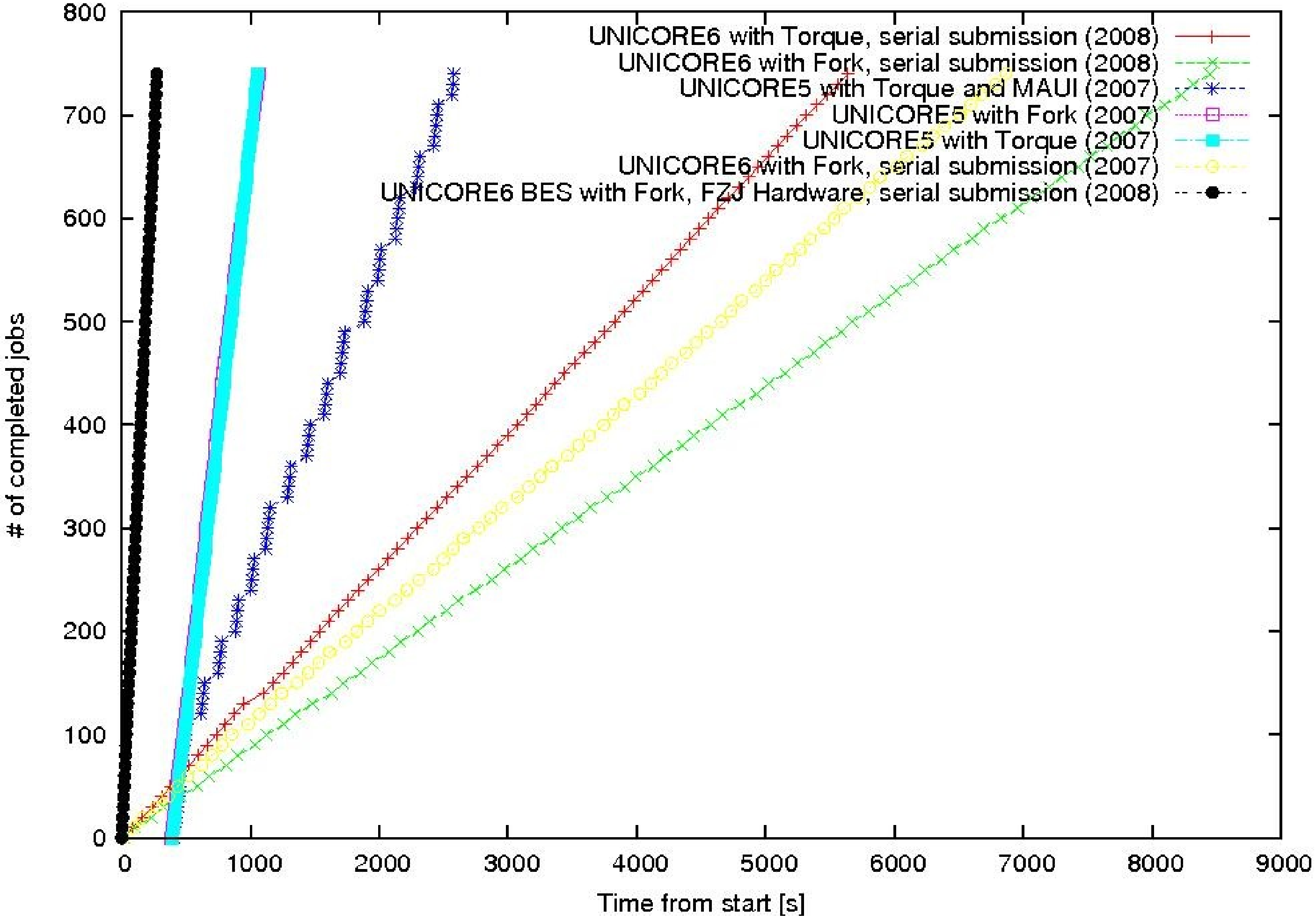


- Serial BM for UAS and BES uses only one single thread.
- System level BM uses a thread pool to first submit all jobs and then poll for status (except for CondorG and CLIQ).
- Concurrent BM for UAS and BES uses two thread pools, one for submitting jobs and one for polling status.

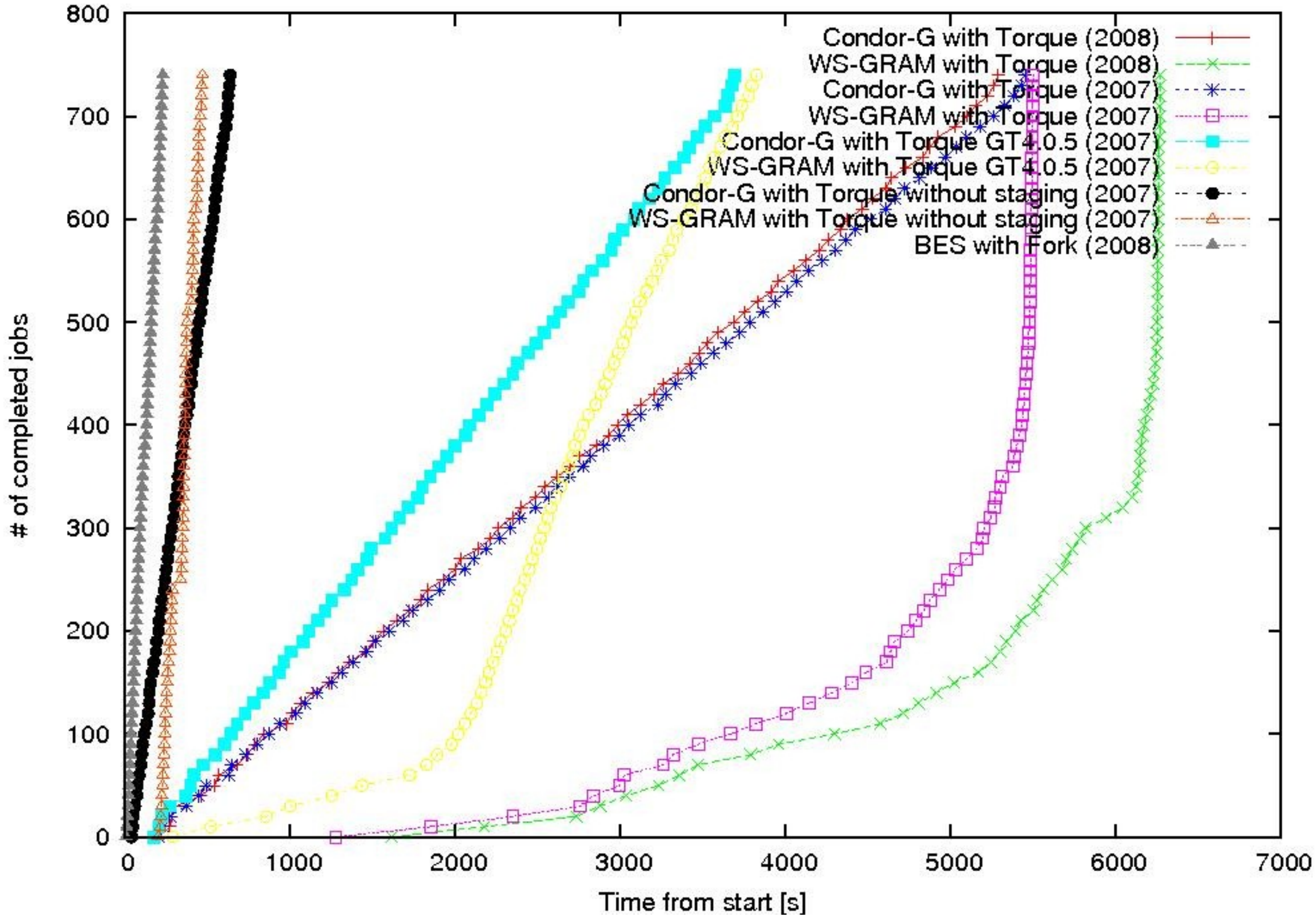
Benchmark Limitations

- **Uses only 0 length jobs**
- **All jobs are submitted in the beginning of a run**
- **Uses only polling, no notifications**
- **Needs command line client tools**
- **Code for different MW stacks not unified**

System Level Performance (UNICORE)



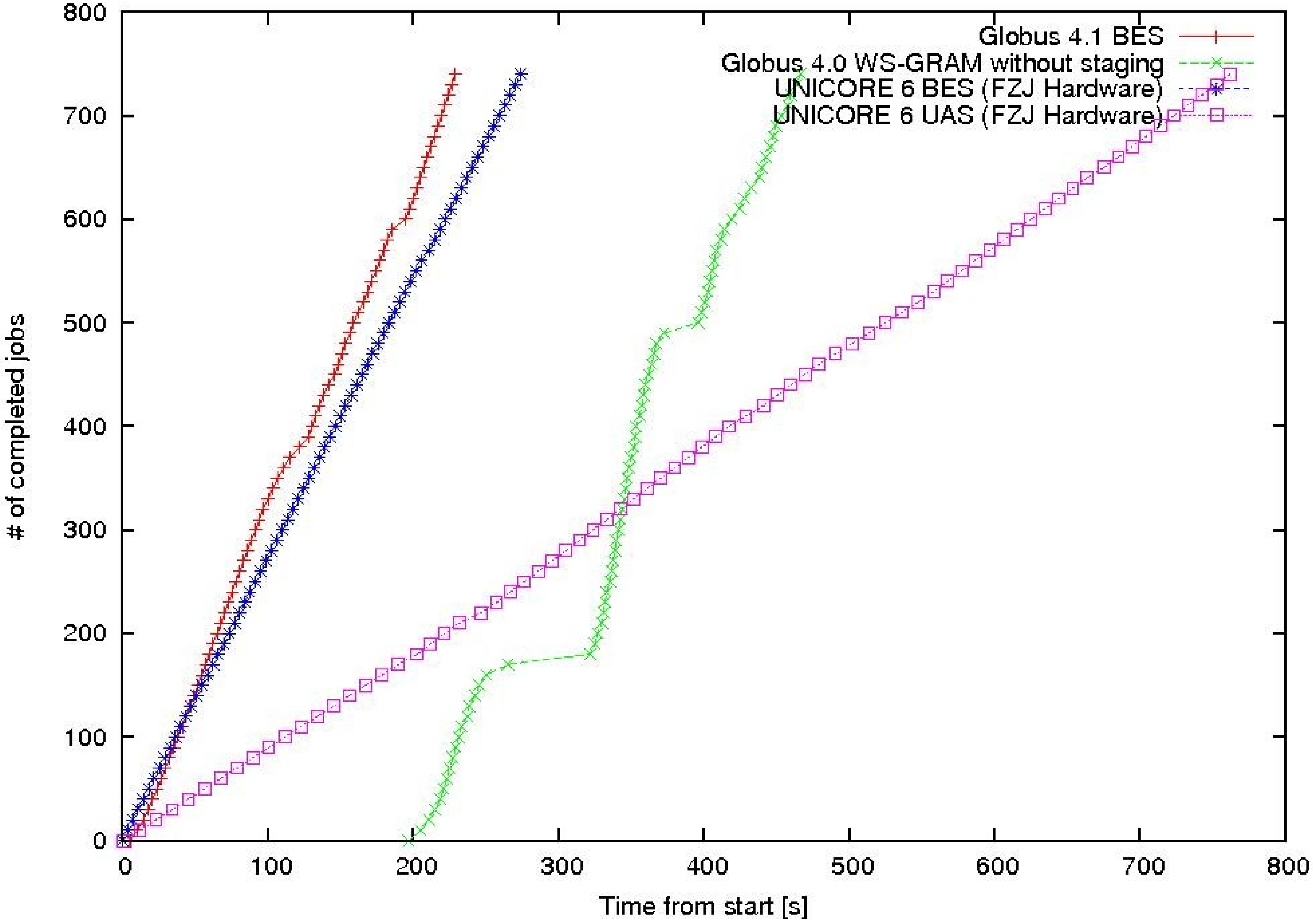
System Level Performance (Globus)



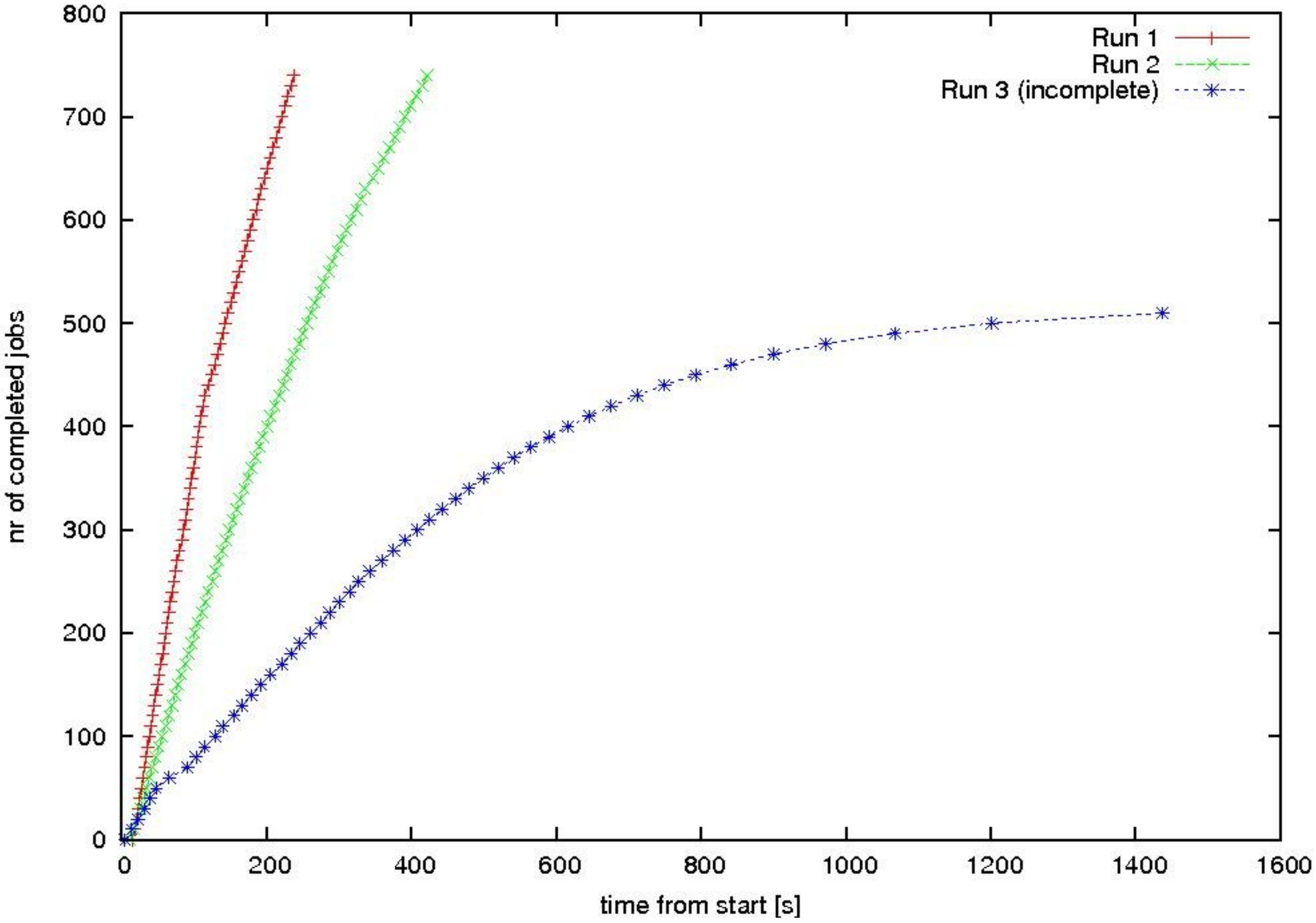
System Level Performance

- Data staging has biggest influence.
- Bulk submission modes (CLIQ, CondorG) better that many single job submissions.
- Polling leads to congestion in the end of Globus experiments, possibly caused by client start-up costs.
- Relatively big spread between different runs of the same experiment. Carefully controlled environment necessary.
- UNICORE seems to be better adopted to the presented benchmark.

Component Level Performance (Serial)



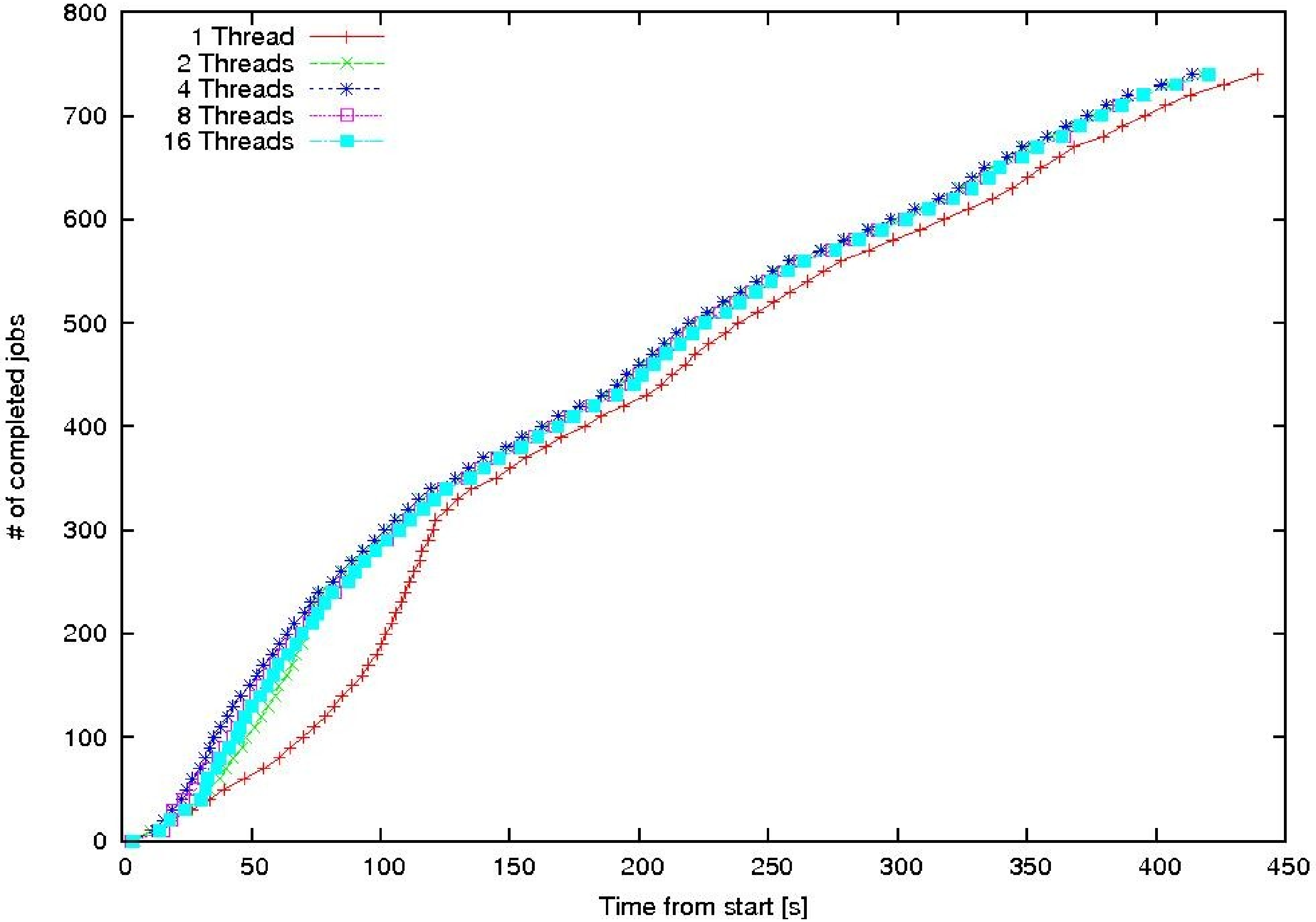
BM as Stress Test Tool



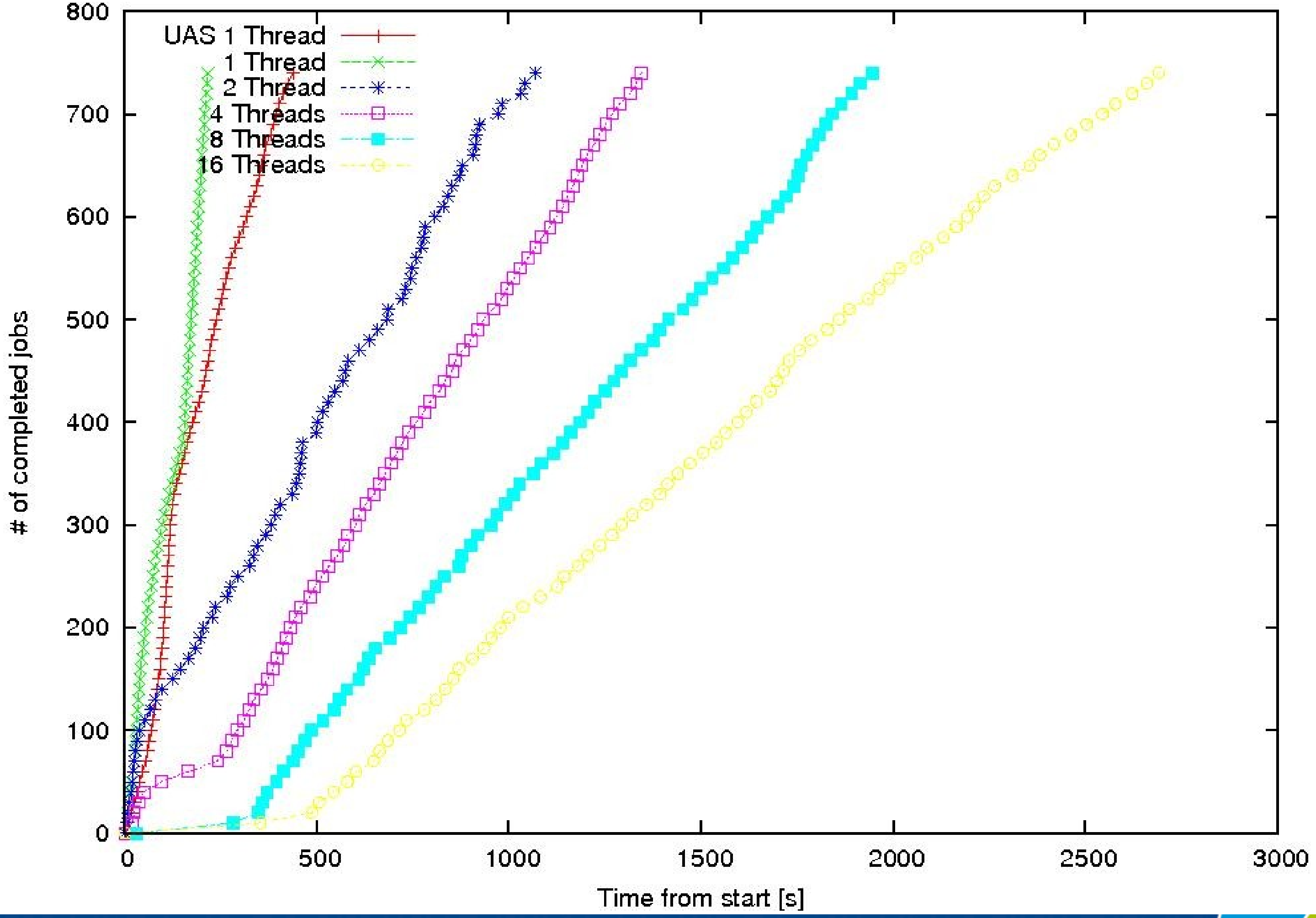
Component Level Performance

- **Serial submission does not suffer from polling congestion.**
- **However experiments show resource leaks and memory problems.**
- **Sensitive to latency of submission and polling interval.**
- **BES components compareable to legacy interfaces.**

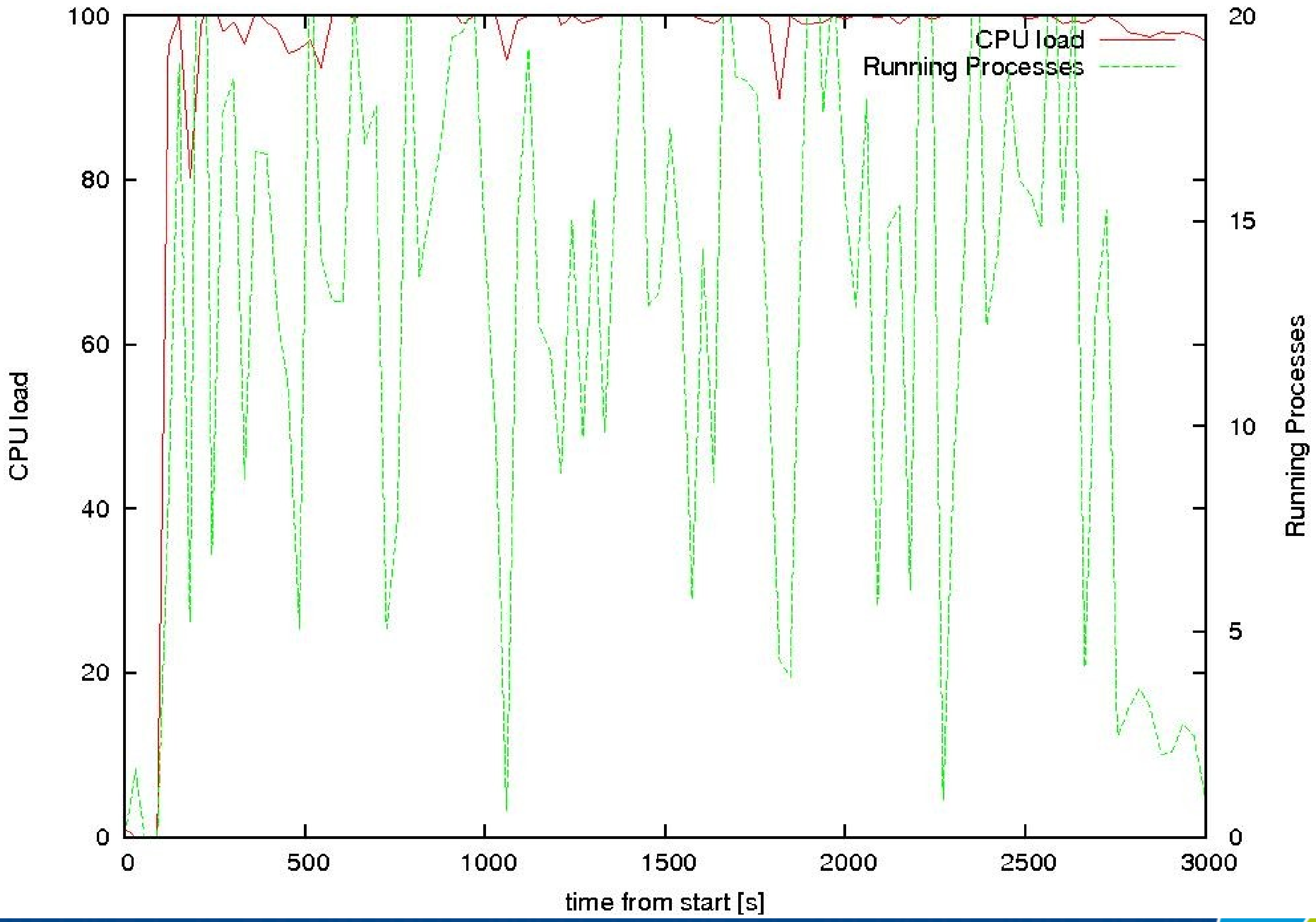
UAS Performance (Concurrent)



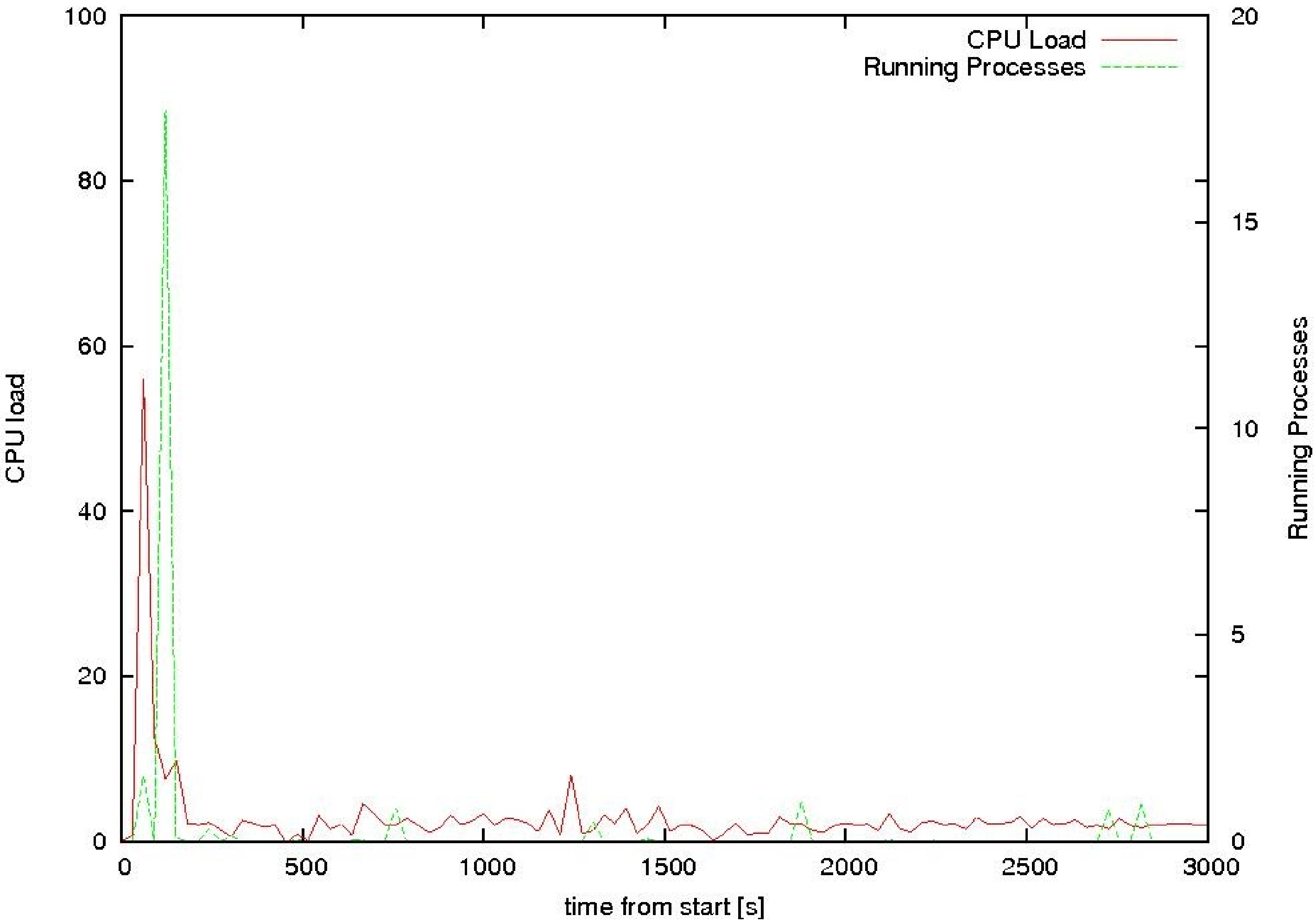
BES Performance (Concurrent)



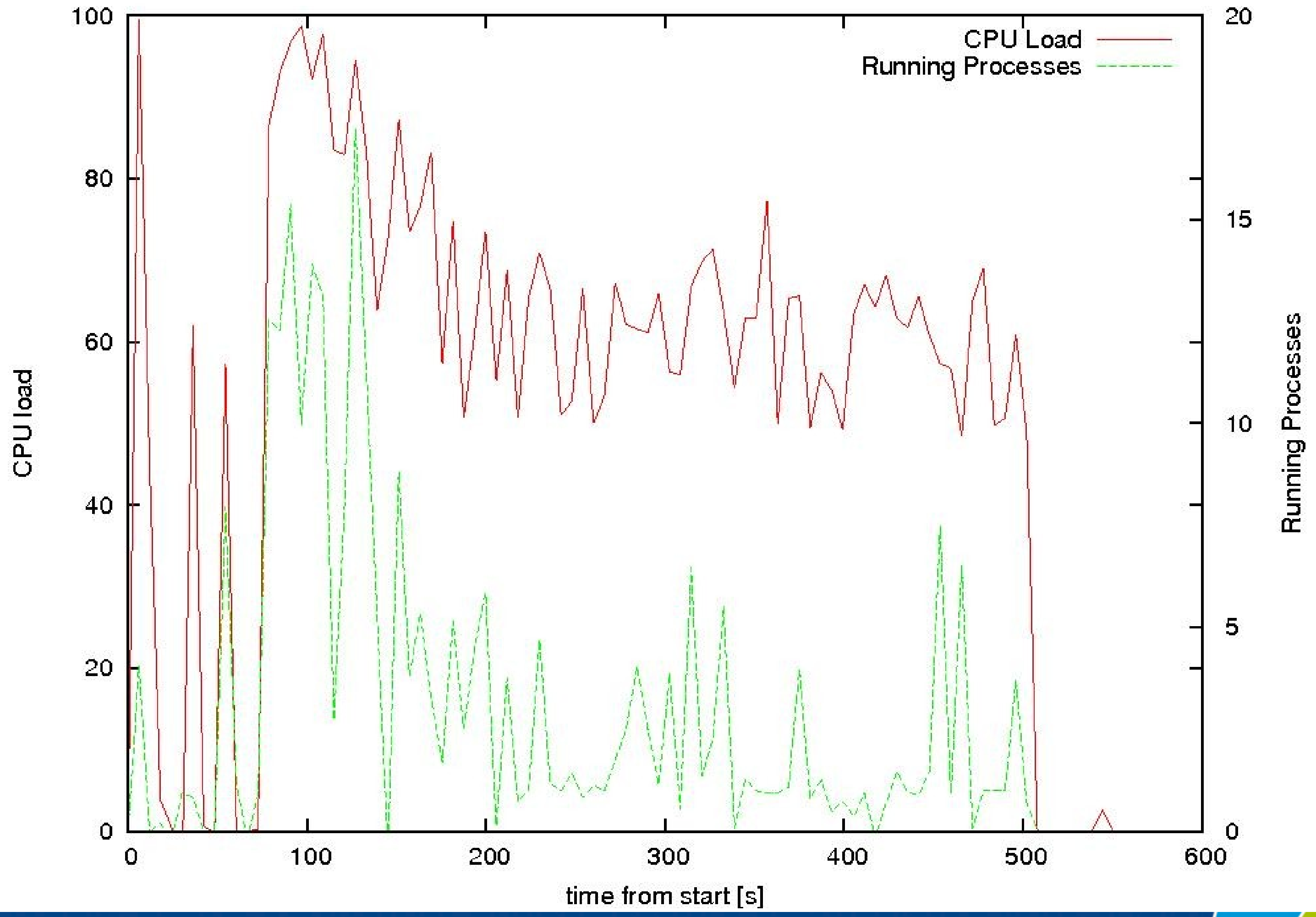
Server Load (BES 16 Threads)



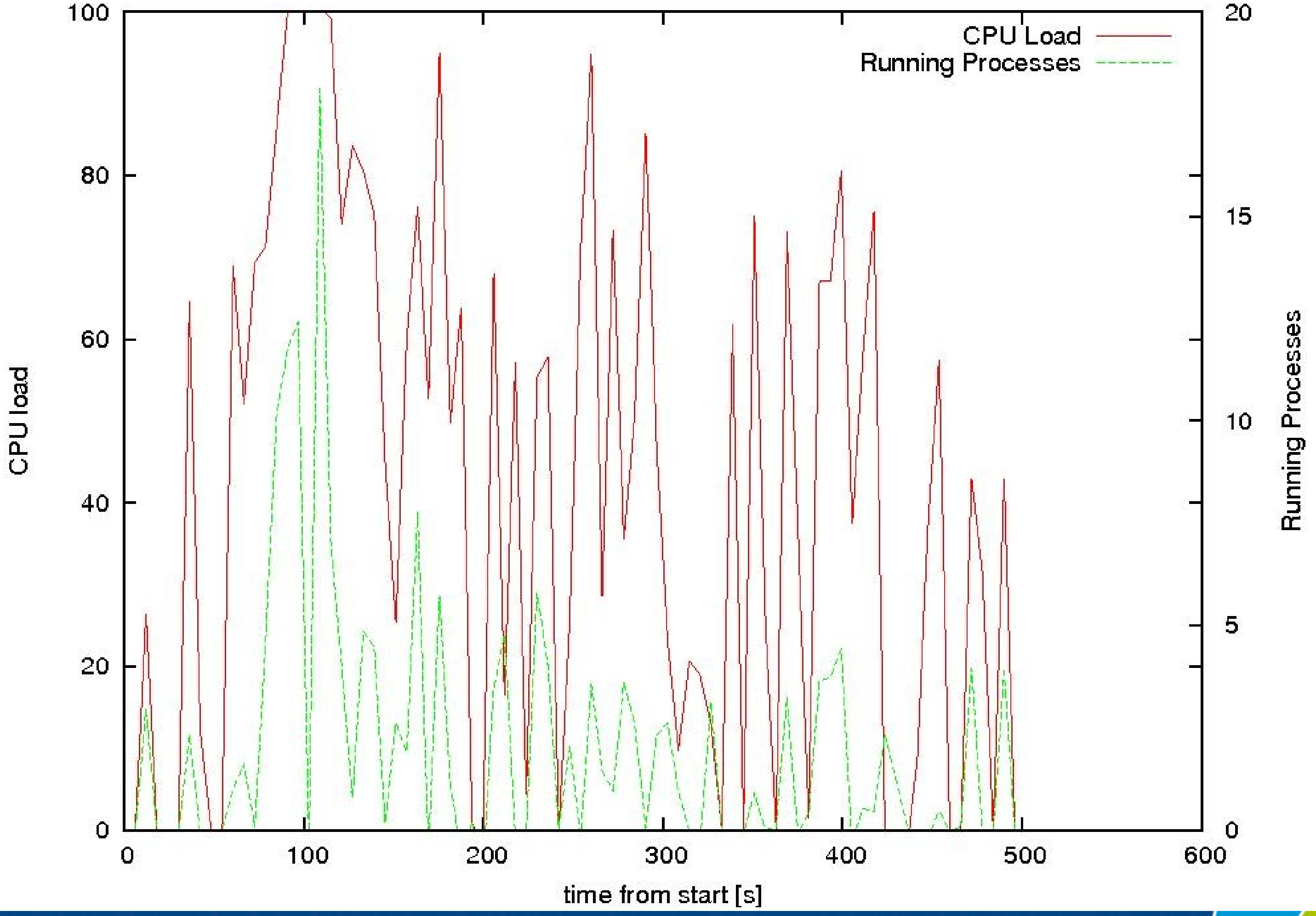
Client Load (BES 16 Threads)



Server Load (UAS 16 Threads)



Client Load (UAS 16 Threads)



omii europe
open middleware infrastructure institute

EU project: RIO31844-OMII-EUROPE

Component Performance – Concurrent Jobs

- **BES shows some concurrency problems causing performance to drop.**
- **UAS shows balanced load between client and server.**
- **UAS is slower than BES for single threads, maybe because jobs need to be started explicitly.**
- **BES drops some jobs during concurrent submission, around 2 jobs out of 750, perhaps due to server overload.**
- **More investigation needed to find out if we found a bug in BES or BM implementation.**

Conclusions

- Type of service dominates over mechanism
- Careful control of test environment is needed
- Installation and configuration of MW takes time
- Preliminary results show that BES is compareable to legacy mechanisms
- However, UNICORE BES currently cannot handle concurrent requests as good as UAS does.
- BM experiments have been able to uncover a number of implementation bugs in early BES services.

Further Work

- Use more than one client node to stress server.
- Use BES Activity instead of only BES Factory.
- Extend concurrent BM to WS-GRAM and GT BES.
- Use other than 0-length jobs.
- Allow for extended job submission simulating steady state.

Acknowledgements

This work is supported by the European Commission through the OMII-Europe project INFSO-RI-031844.

Software can be downloaded from the project repository at:

http://www.omii-europe.org

Or contact Gilbert Netzer:

noname@pdc.kth.se