ATLAS Computing on CSCS HPC

A. Filipcic[1], S. Haug[2], M. Hostettler[2], R. Walker[3], M. Weber[2]

[1] Jozef Stefan Institute, Slovenia [2] AEC, University of Bern, Switzerland [3] Ludwig Maximilian University of Munich, Germany

Contact: michi.hostettler@cern.ch

Abstract

The Piz Daint Cray XC30 HPC system at CSCS, the Swiss National Supercomputing centre, was in 2014 the highest ranked European system on TOP500, also featuring GPU accelerators. Event generation and detector simulation for the ATLAS experiment have been enabled for this machine. We report on the technical solutions, performance, HPC policy challenges and possible future opportunities for HEP on extreme HPC systems. In particular a custom made integration to the ATLAS job submission system has been developed via the Advanced Resource Connector (ARC) middleware. Furthermore, some GPU acceleration of the Geant4 detector simulations has been implemented to justify the allocation request for this machine.

Challenge and Solutions

Current distributed ATLAS computing is based on dedicated x86 Linux clusters of the WLCG specifically set up to meet the needs of the ATLAS software stack, which process jobs from a global job database. This concept is quite different from common self-contained, optimized HPC applications run manually by specific users on specific machines.

The minimal prerequisites to run ATLAS jobs on a HPC system, and our respective solutions, are:

- **Access to ATLAS Software**
  ⇒ Local relocated software/CVMFS copy
- **Run ATLAS software**
  ⇒ x86-based HPC systems: Minor changes to the Software environment done by modified job submission scripts
- **Multi-Threading and Parallelization**
  ⇒ AthenaMP (multi-threaded jobs, event-level parallelism) detector simulation, parallel single-node jobs
- **Integration into the *Production ANd Distributed Analysis* (PanDA) system**
  ⇒ ARC-CE SSH submission back-end (see poster 161 „*The ATLAS ARC ssh back-end to HPC*").

GPU Code Tests

After successful integration of multi-threaded CPU workloads, means for adding GPU code to the ATLAS Geant4-based detector simulation codebase without recompiling the full ATLAS software stack were evaluated. Replacing the internal Geant4 random number generator by a cuRAND-GPU-based implementation showed a ~5% speedup.

We conclude that in the future, computationally heavy parts of key software might be dynamically replaced by equivalent GPU counterparts on hybrid HPC systems.

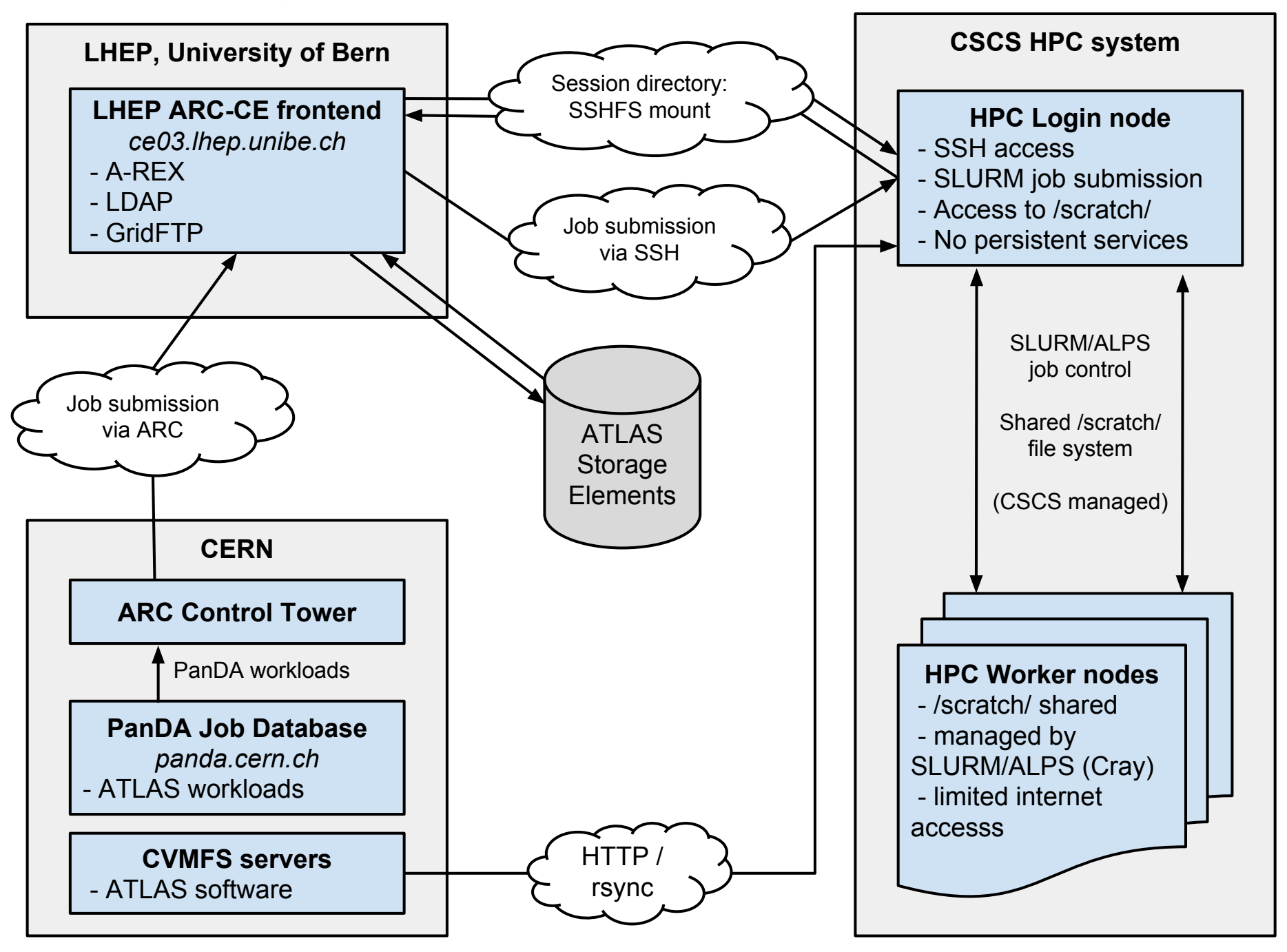
Conclusion and Outlook

We have developed a solution for running the detector simulation part of ATLAS Production on general-purpose shared HPC systems. Apart from CSCS, a similar approach is also used on SuperMUC (Germany), Pi (China) and Hydra (Germany, see poster 153).

In the future, running bulk ATLAS Production jobs on a few big HPC machines might be a cost efficient, complementary approach to running on many distributed WLCG clusters.

ATLAS Job Handling

For full details, see poster 161.



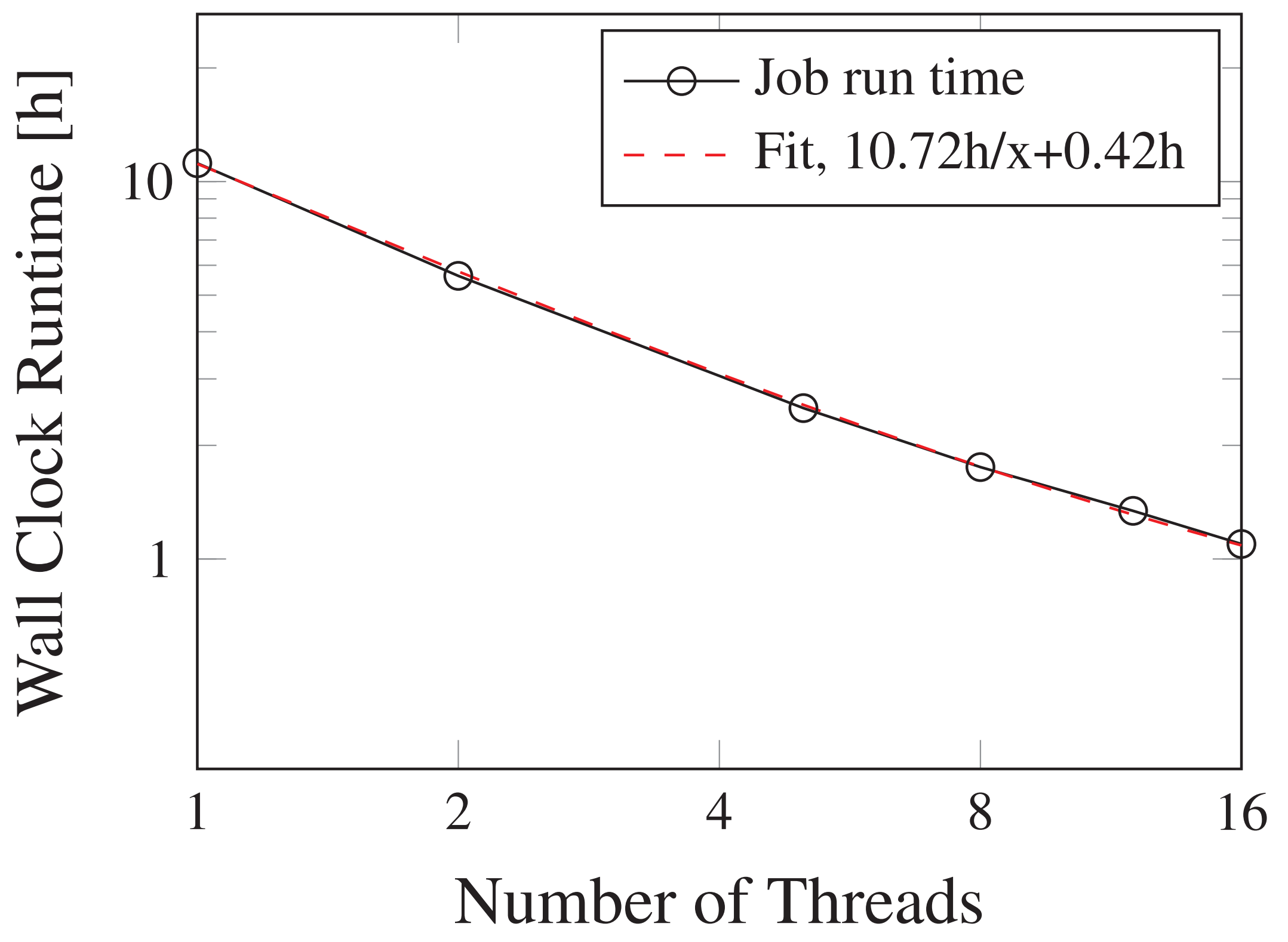
Compiler Performance

We compared the wall clock time of detector simulation jobs with the Geant4 libraries compiled by different compilers for different RNG seeds.

Precompiled	Optimized gcc	CrayCC
880 s	834 s	1219 s
879 s	833 s	1208 s
887 s	840 s	1178 s

CPU Scaling

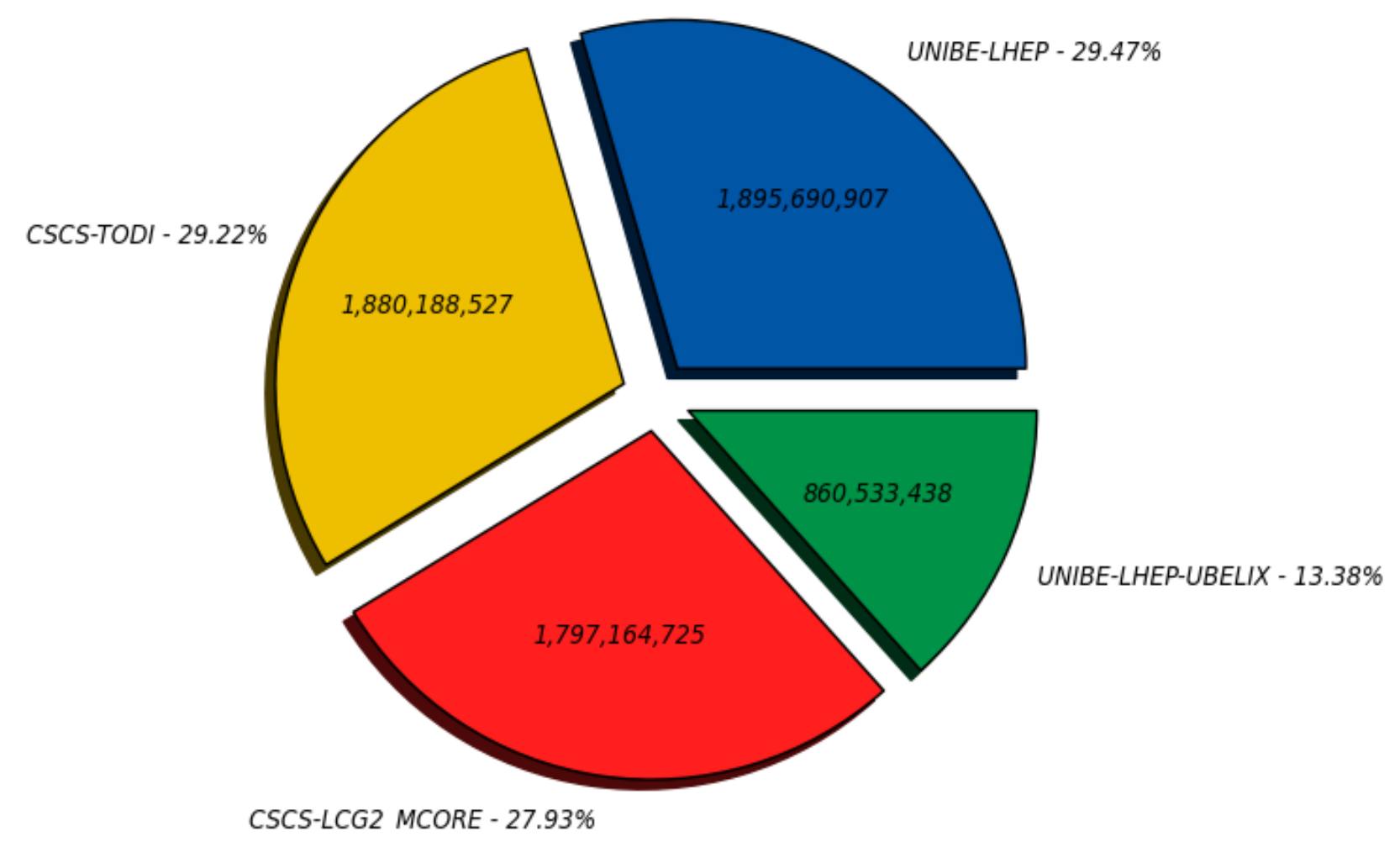
ATLAS Simulation can use **event-level parallelism (using AthenaMP)** to efficiently use all the 16 available CPU cores of a compute node with a memory footprint significantly smaller than 16 parallel single-threaded jobs.
We observed a **near-perfect linear scaling** with an offset of ~30 min due to initialization and finalization.

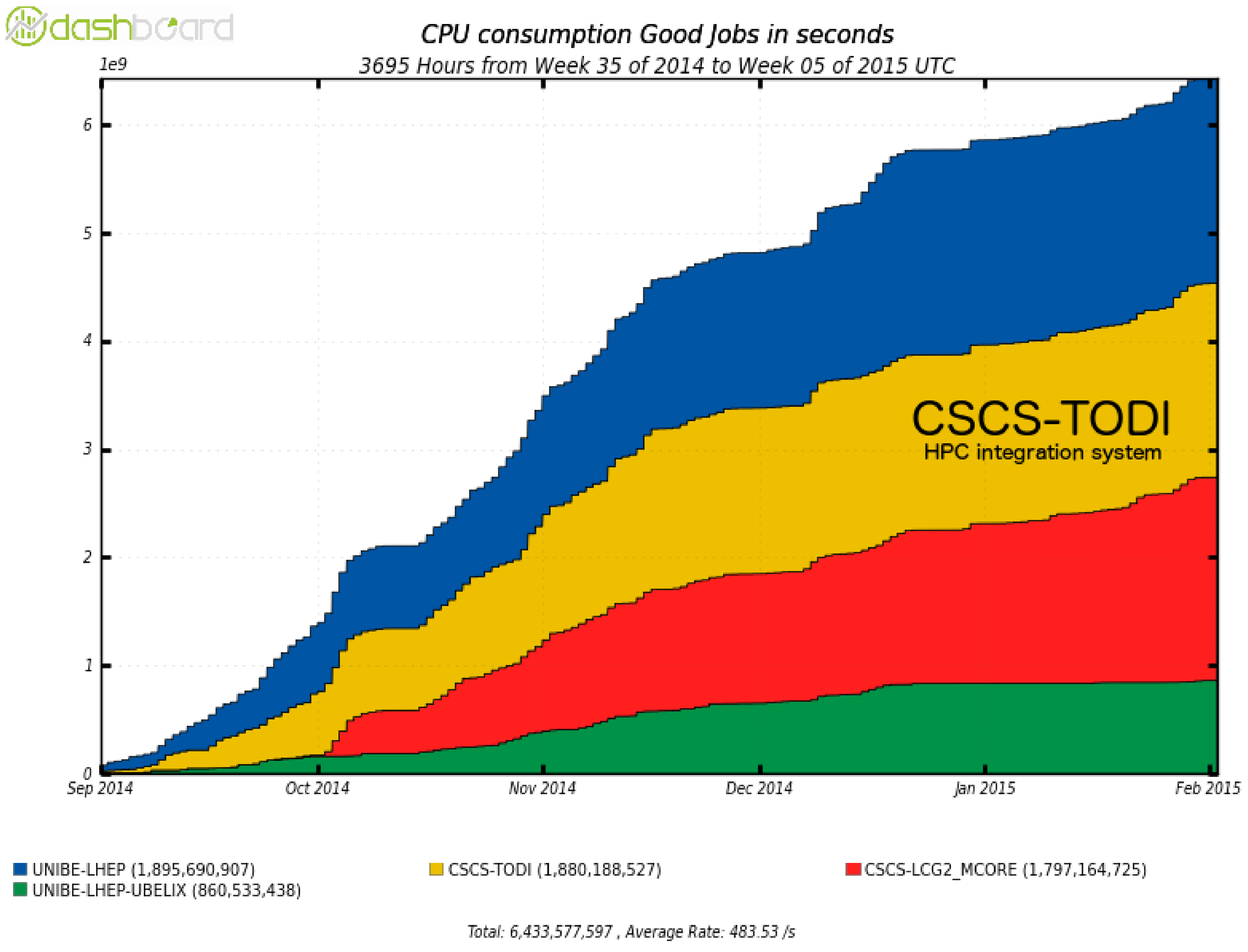


The peak memory usage when processing 100 events was ~6 GiB, which is significantly smaller than the available 32 GiB per node.

Results

We have successfully run ATLAS detector simulation production jobs on the „Piz Daint" integration system „Todi" at CSCS. The test on 50 nodes (800 CPU cores) over ~6 months **contributed ~500000 CPU-hours** towards ATLAS Production. This is roughly the equivalent of one of the dedicated Tier-2 ATLAS clusters in Switzerland (CSCS-LCG2 at CSCS and UNIBE-LHEP at the University of Bern).


UNIBE-LHEP - 29.47%
CSCS-TODI - 29.22%
UNIBE-LHEP-UBELIX - 13.38%
CSCS-LCG2 MCORE - 27.93%


dashboard
CPU consumption Good Jobs in seconds
3695 Hours from Week 35 of 2014 to Week 05 of 2015 UTC
1e9
Sep 2014
Oct 2014
Nov 2014
Dec 2014
Jan 2015
Feb 2015
CSCS-TODI
HPC integration system
UNIBE-LHEP (1,895,690,907)
CSCS-TODI (1,880,188,527)
CSCS-LCG2_MCORE (1,797,164,725)
UNIBE-LHEP-UBELIX (860,533,438)
Total: 6,433,577,597 , Average Rate: 483.53 /s