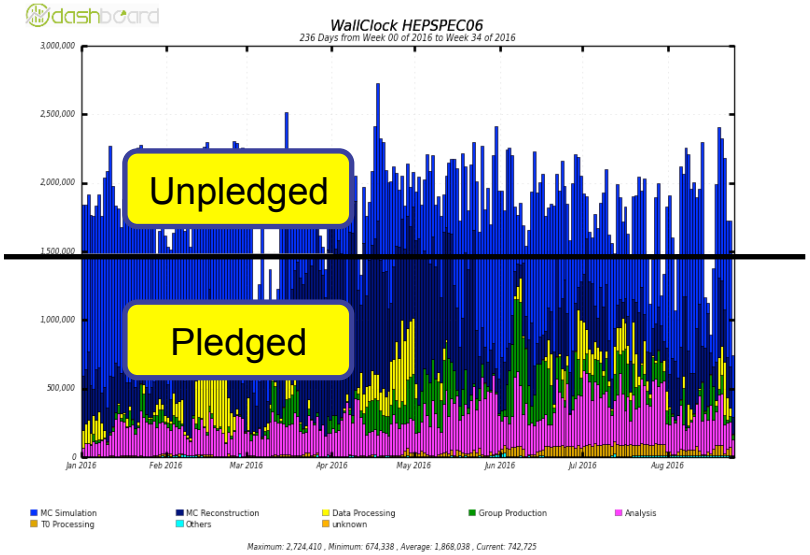
UiO : University of Oslo

Exploiting Opportunistic Resources for ATLAS with ARC CE and the Event Service

David Cameron
Andrej Filipcic
Wen Guan
Vakhtang Tsulaia
Rod Walker
Torre Wenaus
on behalf of the ATLAS Collaboration

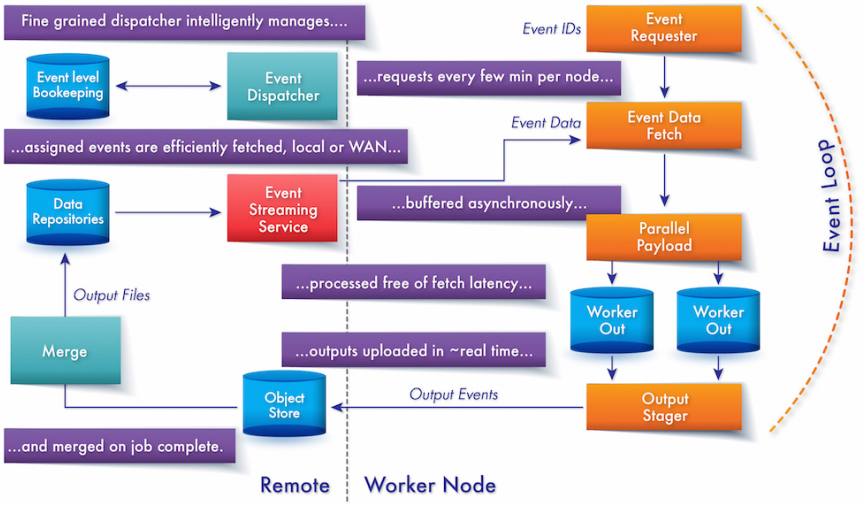
Opportunistic Resources

- A vital contributor to ATLAS computing
  - HPC backfilling, cloud spot market, volunteer computing, HLT farm
- A common factor is that resources can disappear at any time, even in the middle of a job
- Other potential issues:
  - No network connectivity from worker nodes
  - No root-mounted CVMFS for software
  - No persistent edge services on eg login nodes



ATLAS Event Service

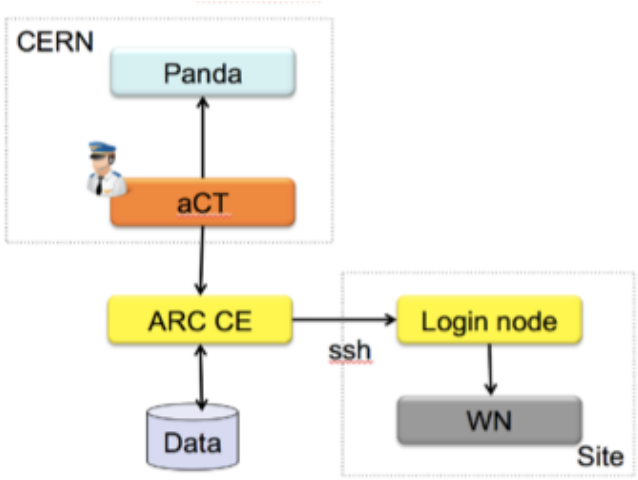
- A regular ATLAS job processes a fixed number of events
- The Event Service goes to event-level granularity
  - As a job runs, it regularly saves events produced to an Object Store
  - If a job is terminated it loses at most the last 10 mins of events
  - Unfinished events are reassigned to another job
  - Events are merged and stored on regular Grid storage

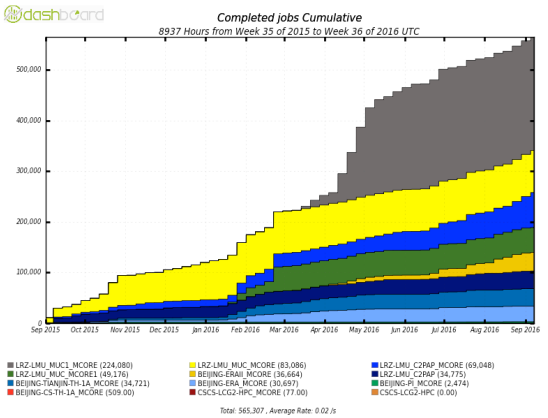


See ES talk for more info

ARC CE and ARC Control Tower

- ARC CE: Designed to be a lightweight non-intrusive resource connector
- Performs staging in and out of data
- ARC Control Tower (aCT) handles communication with Panda (ATLAS workload management system)
- No Grid software or outbound connectivity needed from worker nodes
- ssh mode for the more restrictive HPCs
- Used to integrate HPCs in Europe and China (as well as many “regular” Grid sites)





Integration of ARC CE and Event Service

- How to run jobs on a preemptable resource with no network connectivity?
  - Run Event Service jobs and do asynchronous data staging with ARC CE
  - The job writes each event + metadata to working directory
  - Once finished, the names of events are put in a special file
  - ARC CE detects the job finished in the batch system, and uploads the events listed in the file
  - ARC Control Tower tells Panda which events succeeded
- If preempted, a special ARC CE plugin runs
  - It tells ARC CE that the job succeeded
  - It finds the events that were produced and puts them in the special file

Challenges

- The number of events to upload from one ARC CE quickly became a bottleneck
  - 10k cores, 5 mins per event = 30Hz upload
- Led to zip functionality
  - After job finishes pilot creates a tarball of events
  - Reduces data transfer rate by orders of magnitude
  - Also makes merge job more efficient
- Many extensions of aCT/ARC required
  - Object store support
  - Passing event ranges back and forward
  - Plugin to emulate pilot functionality for preempted jobs

Leibniz Supercomputing Centre
of the Bavarian Academy of Sciences and Humanities

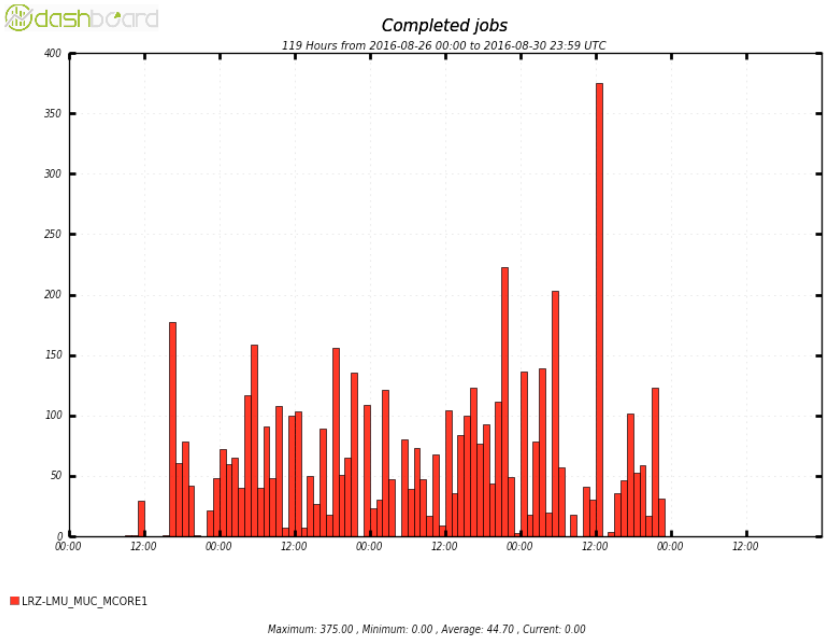
Test setup

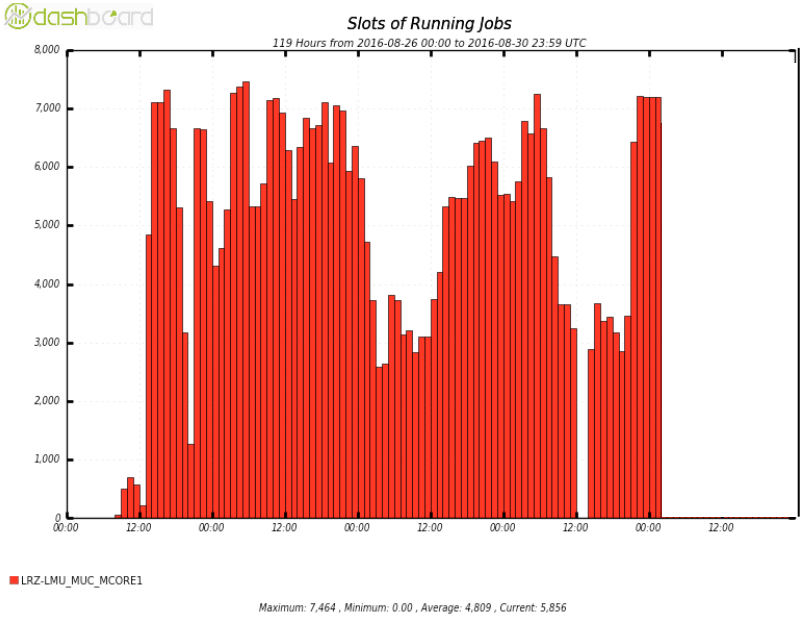
- The SuperMUC HPC in Munich is our test system
  - Phase 1 (2011-13): 155000 cores, 3.2PFlops
  - Phase 2 (2015): 86k cores, 3.6 PFlops
  - #27 on Top500 (June 16)
- Over a period of 3 days ran Event Service jobs on Phase 1
- Each job attempted to run 500 events over 24 cores (~4h walltime)
- All jobs could be preempted at any time

Results

	Finished	Preempted	Total
Jobs	3861	1014	4875
Events (M)	1.93	0.23	2.16
CPU hours (k)	196	24	220

- Jobs used up to 7000 cores (24 each)
- Out of a total of 4875 jobs 21% of jobs were preempted
- 10% of events were saved from preempted jobs





Conclusion

- Integrating ARC and Event Service provides an effective solution for using opportunistic resources in restricted environments
  - Specifically HPC but could be applicable in other areas eg volunteer computing, clouds
- In a 3-day test on SuperMUC we used 1000 CPU days which would have otherwise been lost
- Future directions:
  - Scaling up
  - Using on other HPC sites as Event Service usage grows
  - Adapting to the Event Streaming Service