Use ATLAS@home jobs to backfill the cluster

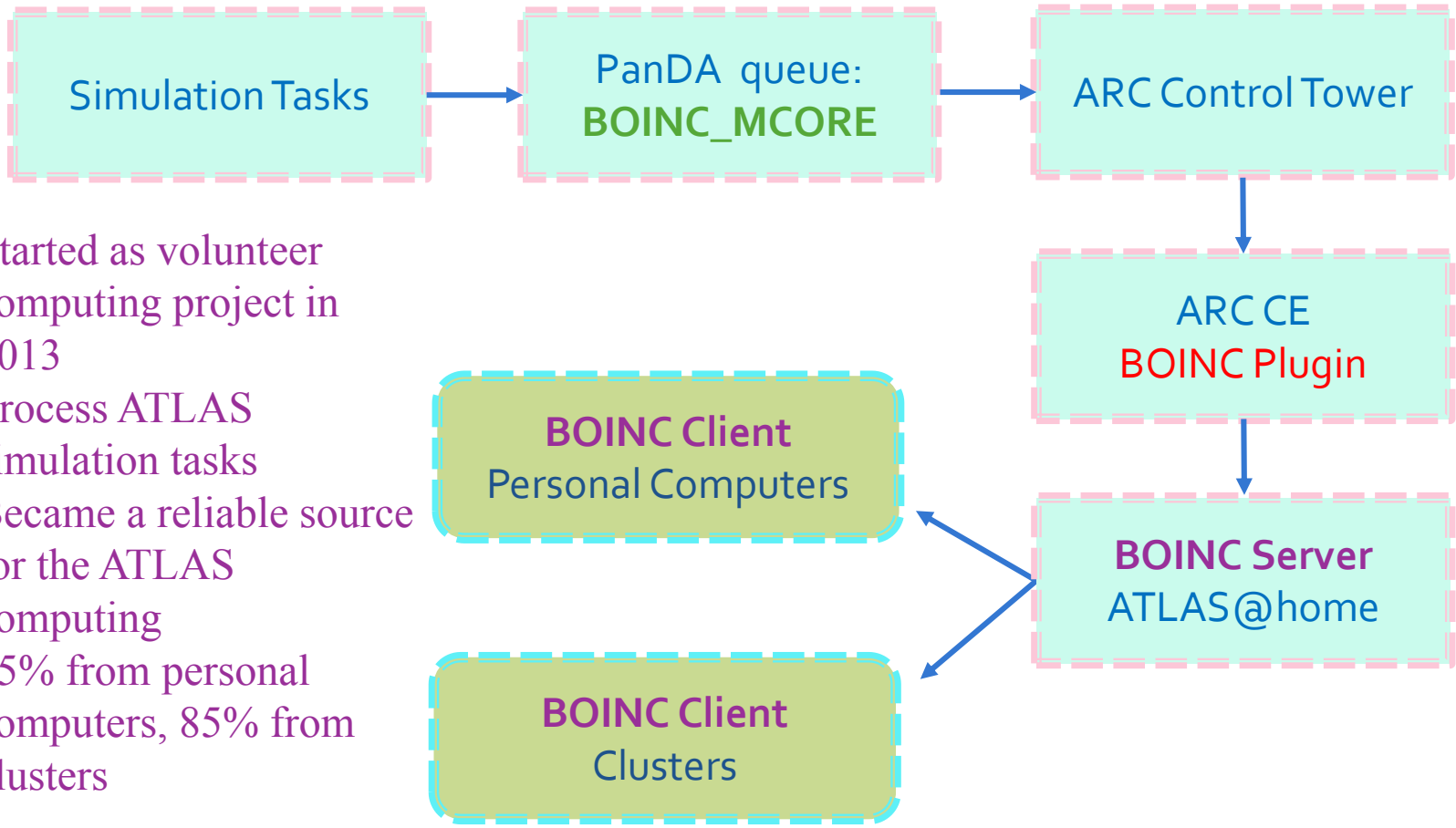
Wenjing Wu

AGLT2

Outline

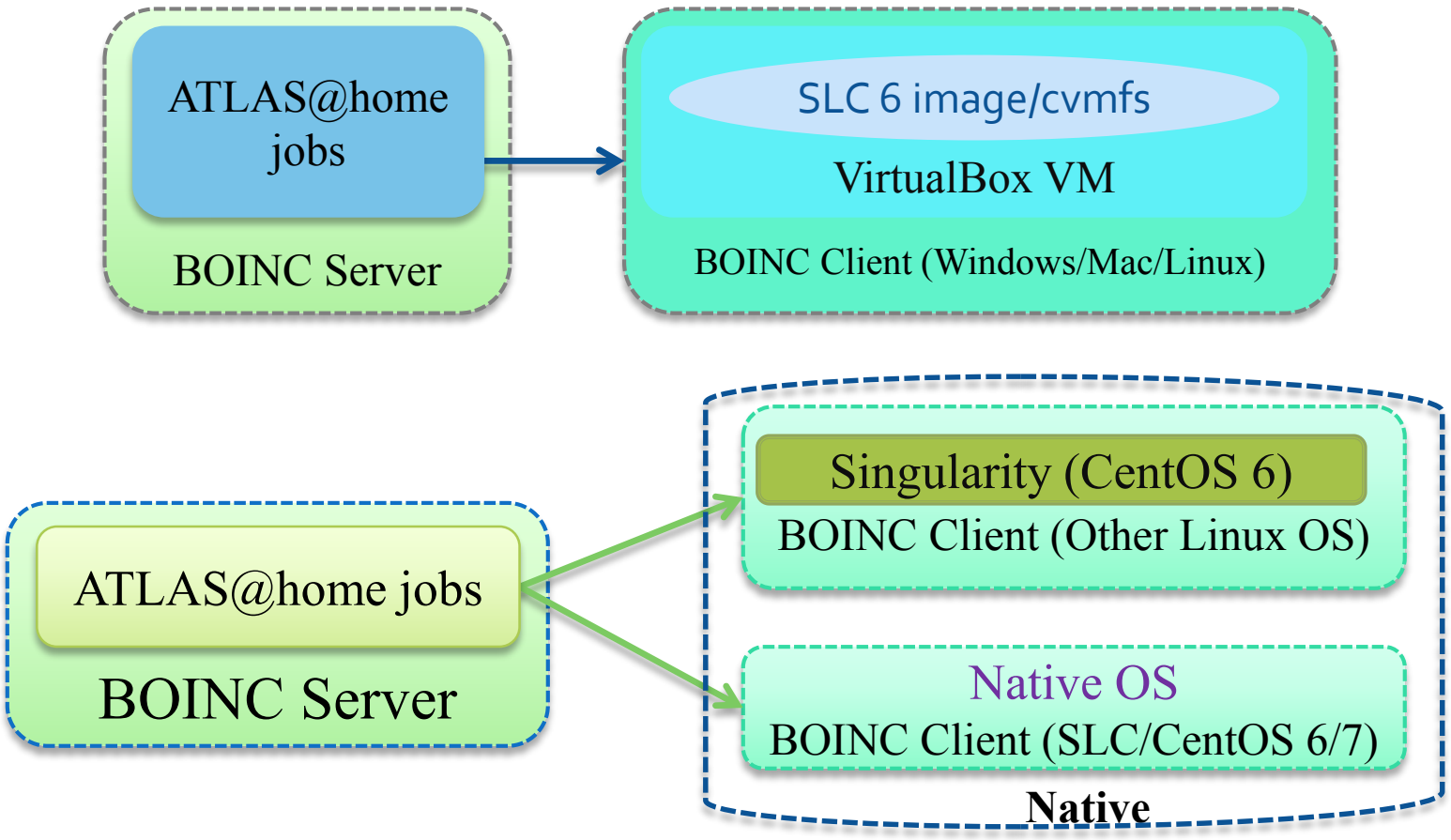
- ATLAS@home : Virtualization vs. Containerization
- Performance measurement
- Use cases
  - Backfilling the grid sites
  - CERN IT cloud cluster
- Summary

ATLAS@home



- Started as volunteer computing project in 2013
- Process ATLAS simulation tasks
- Became a reliable source for the ATLAS computing
- 15% from personal computers, 85% from clusters

- Virtualization for Windows
- Containerization for Linux

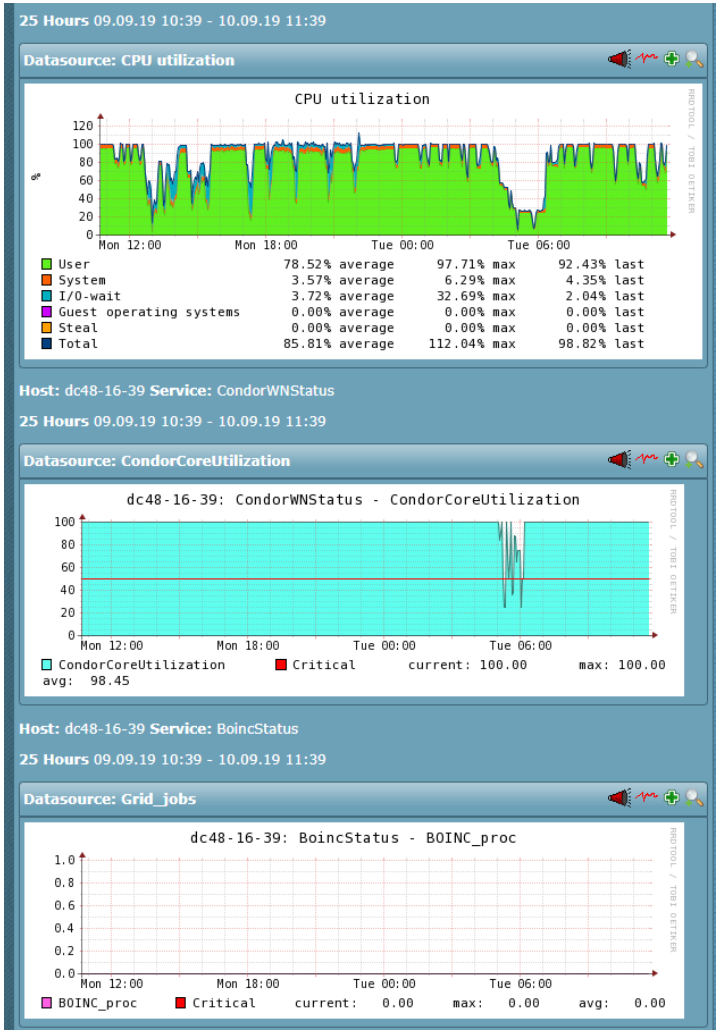


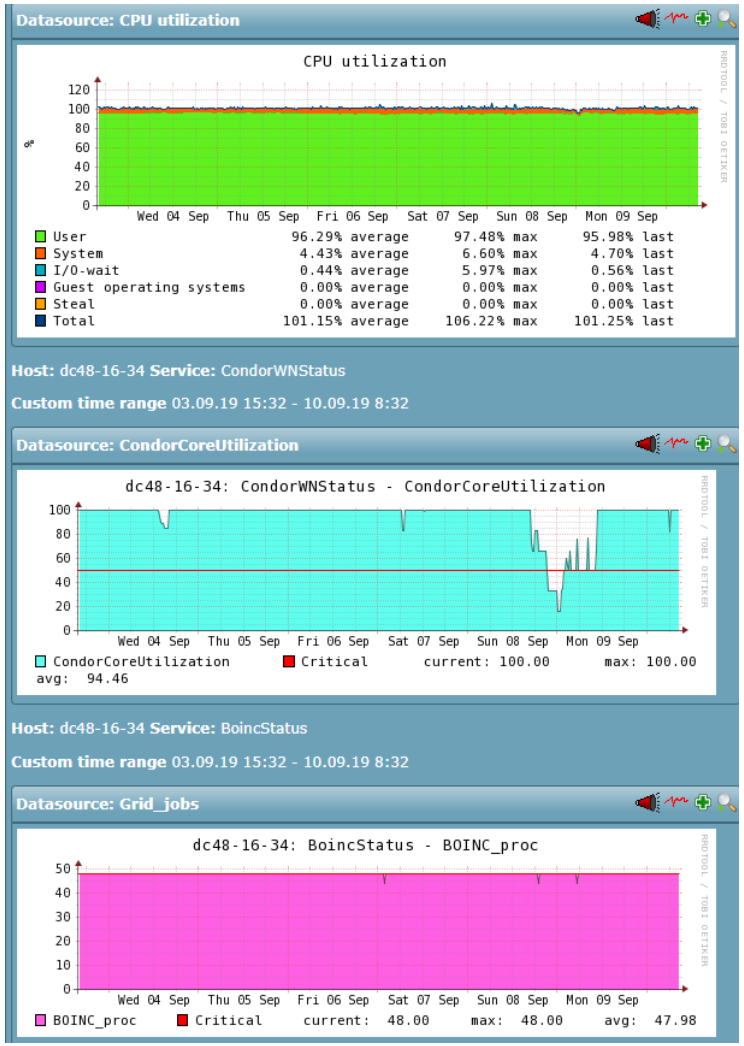
Workflow and Job Features

- Every work node runs the BOINC client (installed in cvmfs) in a daemon model
- BOINC client request jobs from the ATLAS@home BOINC server which are pilot code
- Pilot pulls job from PanDA queue BOINC_MCORE
- Jobs can be suspended/resumed depending on the available resource on the node, but will be kept in memory, since Athena does not support checkpoint and restart.
- Jobs are very standard, CPU intensive, simulation jobs with 200 events, each event takes 200-400 seconds, produces about 300MB output file.
- Jobs are memory efficient, 800MB/core for 4 core jobs. And the core number per job is dynamical (between 1 and 12) depending on the available cores.
- BOINC jobs have nice value of 19, so it should have the lowest priority among all processes.
- If the process scheduler policy does not “Horner” the nice value, like sched_other, can also use cgroup to control how much cpu/memory can be used by BOINC jobs.

Why do we run

- Site batch system can hardly use 100% of the cores. AGLT2 did a lot of optimization, the average utilization of HTCondor is 99.5% during no downtime.
  - Site downtown; retire work node; Fairness to different accounting groups; defragment from dynamical partitioning.
- ATLAS jobs have different CPU Efficiency, ranging from 12% to 96%.
- Overall, according to Fred's recent study , the CPU Utilization by calendar time for USATLAS site is at most 66%. There is plenty of room to backfill.

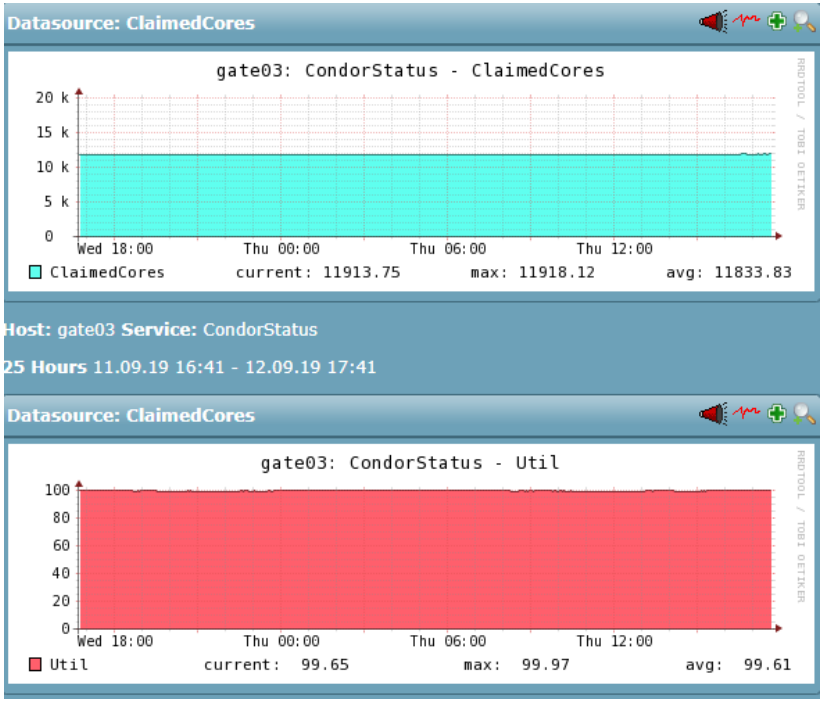


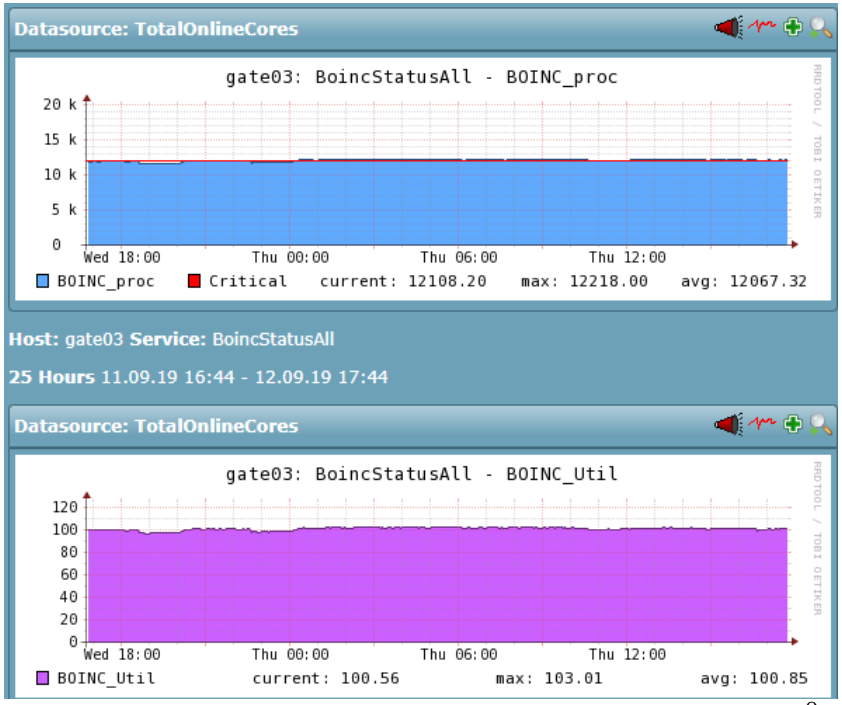


- Work Node 1 has BOINC backfilling jobs, Work Node 2 does not
  - Node 1 Condor Utilization is 98.45%, CPU Efficiency is 78.52%
  - Node 2 Condor Utilization is 94,46%, CPU Efficiency is 96.29%
- Over 20% of CPU time was scavenged from the node in (nice time)
- Condor Utilization varies depending on available idle jobs and their resource requirements, BOINC backfilling does not affect it

How backfilling works

- BOINC and HTCondor clients co-exist on the work node, receiving jobs from different PanDA queues.
- BOINC and HTCondor does not know the existence of each other, and does not communicate with each other.
- Each of them sees the all the cores of the cluster to be dedicated to them, so they schedule jobs to work nodes as they were the solo scheduler.
- HTCondor Utilization is 99.97%, BOINC is nearly 100% (We have 4 nodes not running HTCondor but BOINC, the utilization is based on the size of Condor Cluster)
  - Util. for Condor=total_claimed_cores/total_online_cores_of_condor
  - Util. for BOINC=total_cores_by_boinc/total_online_cores_of_condor

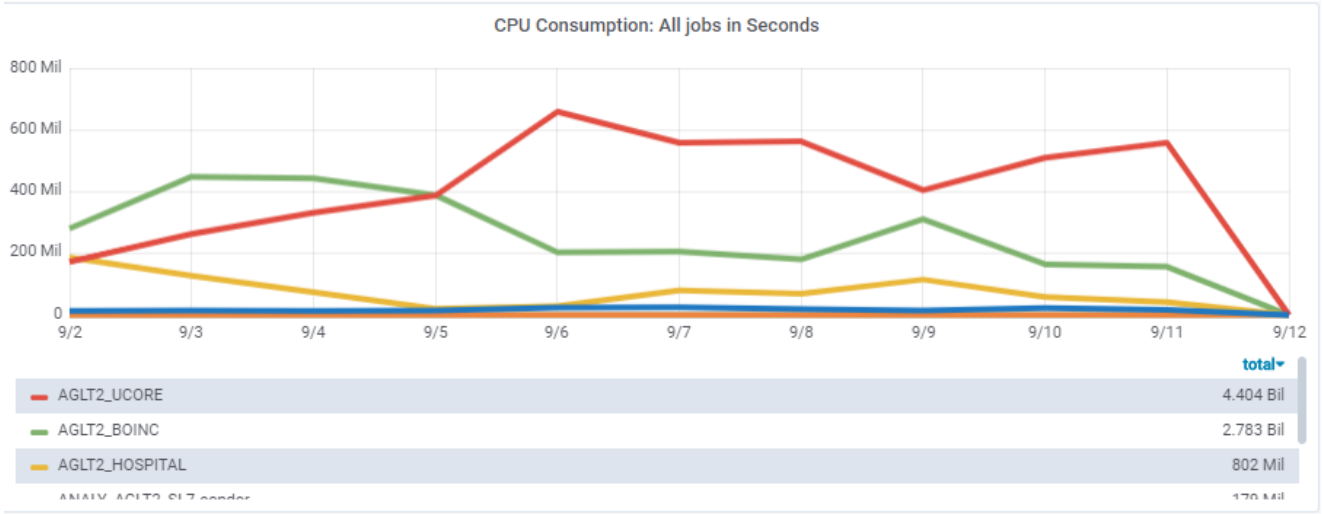


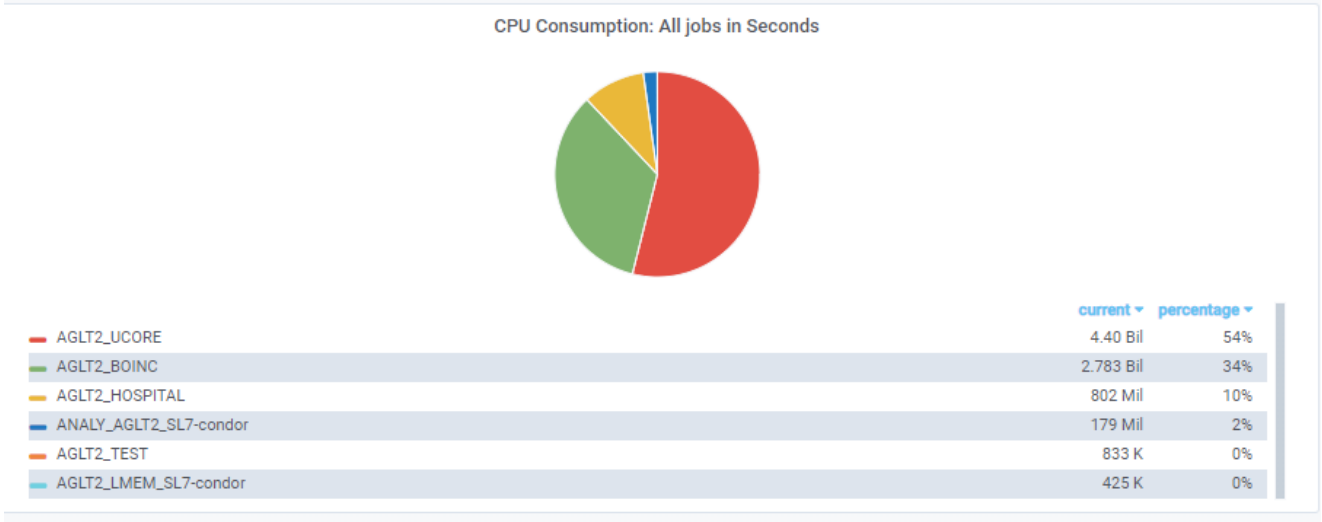


How does a site start

- Create a BOINC account on LHC@home， use the site name as the user name, i.e. AGLT2, so ATLAS accounting can credit these jobs to the site's contribution
- Create a local account on the cluster to run the BOINC jobs
- Configure BOINC on the work node, just a few commands can be put into one script to run over the cluster.
  - This creates the work directory for BOINC jobs
  - Creates a few cron jobs to monitor/manage the BOINC jobs, to make sure it can recover itself.
- Currently the BOINC jobs are accounted to the site in the ATLAS accounting, but not to the EGI accounting.

BOINC jobs are accredited be a pseudo PanDA queue AGLT2_BOINC belonging to the site





Experiences from some sites

- In early 2017 it was first experimented in BEIJING Tier2, a site with 400 cores
- Then TRIUMF Tier1 started it on their cluster, with +10K cores in late 2017.
- In March 2019, AGLT2 also started on the whole cluster, with 12K cores
- Sites all see significant improvement in the CPU utilization
  - cputime_of_alljobs/(number_of_cores*calendar time)
  - This does not exclude downtime, nodes offline with hardware issues

Improvements for sites(Beijing and TRIUMF)

TRIUMF Site CPU Utilization(4816 CPU cores in 7 days)

Mon Mar 12 00:00:00 2018 to Mon Mar 19 00:00:00 2018

	suc_rate	cpu_eff	cpu_util	wall_util
BOINC	NaN	NaN	0.00	0.00
Grid	0.9	0.8	0.69	0.88
All	0.9	0.8	0.69	0.88

Before Backfilling

TRIUMF Site CPU Utilization(4816 CPU cores in 7 days)

Thu Apr 12 00:00:00 2018 to Thu Apr 19 00:00:00 2018

	suc_rate	cpu_eff	cpu_util	wall_util
BOINC	0.97	0.29	0.27	0.91
Grid	0.95	0.50	0.65	0.97
All	0.95	0.50	0.92	1.88

After Backfilling

BEIJING Site CPU Utilization(464 CPU cores in 7 days)

Thu Apr 12 00:00:00 2018 to Thu Apr 19 00:00:00 2018

	suc_rate	cpu_eff	cpu_util	wall_util
BOINC	1.00	0.17	0.15	0.88
Grid	0.99	0.53	0.80	0.93
All	0.99	0.53	0.95	1.81

Busy Week

BEIJING Site CPU Utilization(464 CPU cores in 7 days)

Fri Apr 6 00:00:00 2018 to Fri Apr 13 00:00:00 2018

	suc_rate	cpu_eff	cpu_util	wall_util
BOINC	1.00	0.47	0.42	0.88
Grid	0.96	0.61	0.48	0.62
All	0.98	0.61	0.90	1.50

Idle Week

- The CPU utilization of TRIUMF site is **improved by 23%** (from 69% to 92%, note the grid workload is even higher after backfilling)
- BEIJING site, the Avg. CPU utilization improvement (in 6 months period) is **26%**
- The Peak CPU utilization reaches **95%** in a week basis, remains **90%** in long term

Improvements for sites (AGLT2)

AGLT2 Site CPU Utilization(10888 CPU cores in 10 days)

Mon Mar 11 00:00:00 2019 to Thu Mar 21 00:00:00 2019

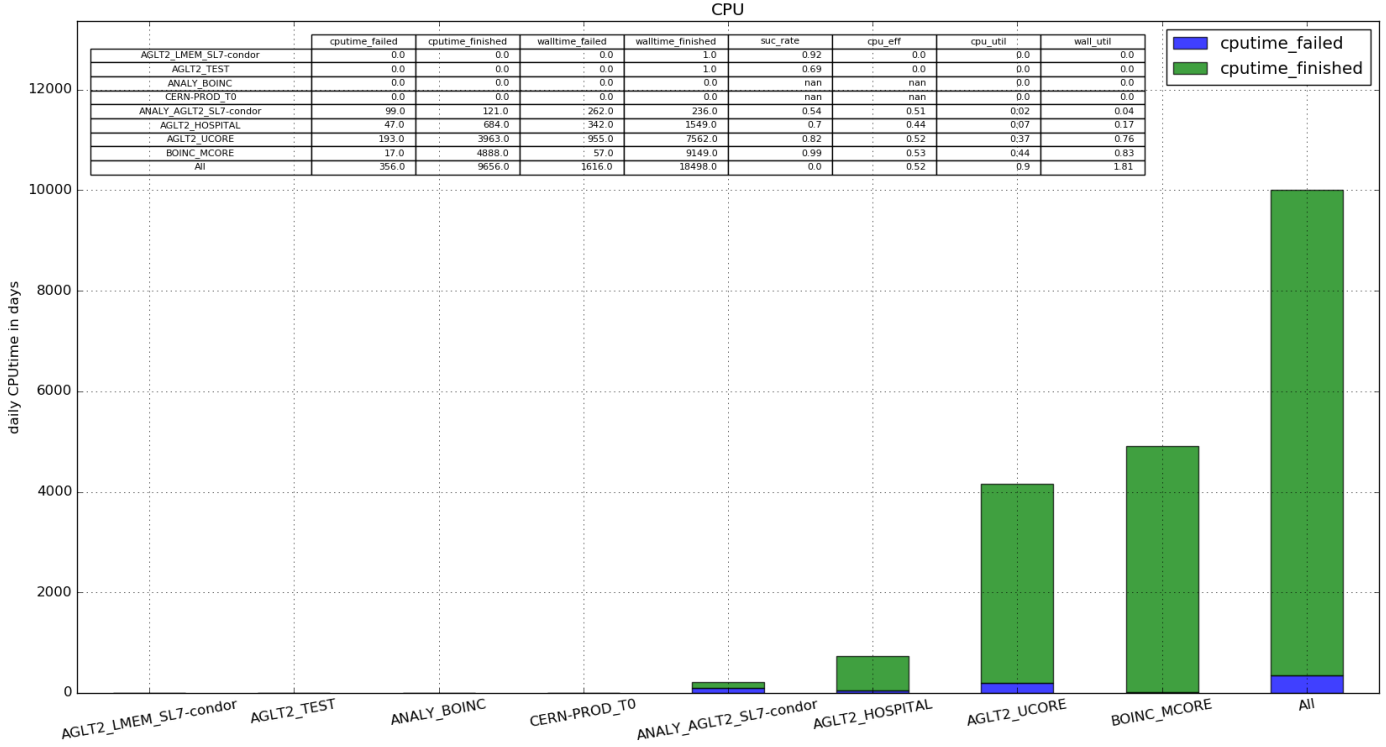
	suc_rate	cpu_eff	cpu_util	wall_util	cputime
BOINC	1.00	0.50	0.40	0.80	4348.0
Condor	0.91	0.53	0.53	0.96	19958.0
All	0.95	0.53	0.93	1.76	10113.0

- At peak, 93% of CPU utilization
- In long turn (past 100 days), average 90% CPU utilization

AGLT2 Site CPU Utilization(11136 CPU cores in 100 days)
between Sun Jun 2 00:00:00 2019 and Tue Sep 10 00:00:00 2019
generated at 2019/9/10

CPU

	cputime_failed	cputime_finished	walltime_failed	walltime_finished	suc_rate	cpu_eff	cpu_util	wall_util
AGLT2_LMEM_SL7-condor	0.0	0.0	0.0	1.0	0.92	0.0	0.0	0.0
AGLT2_TEST	0.0	0.0	0.0	1.0	0.69	0.0	0.0	0.0
ANALY_BOINC	0.0	0.0	0.0	0.0	nan	nan	0.0	0.0
CERN-PROD_T0	0.0	0.0	0.0	0.0	nan	nan	0.0	0.0
ANALY_AGLT2_SL7-condor	99.0	121.0	262.0	236.0	0.54	0.51	0.02	0.04
AGLT2_HOSPITAL	47.0	684.0	342.0	1549.0	0.7	0.44	0.07	0.17
AGLT2_UCORE	193.0	3963.0	955.0	7562.0	0.82	0.52	0.37	0.76
BOINC_MCORE	17.0	4888.0	57.0	9149.0	0.99	0.53	0.44	0.83
All	356.0	9656.0	1616.0	18498.0	0.0	0.52	0.9	1.81



Exception at AGLT2

- BOINC jobs are not really backfilling, they use the same amount of CPU and have the same priority as the HTCondor jobs, which led the CPU efficiency of the HTCondor jobs to be lower than they should be.
- This might be due to the CPU schedule policy. We started to use cgroup to control the CPU usage by BOINC jobs, and the CPU efficiency of HTCondor jobs get back to normal.
- With cgroup, the average CPU Efficiency for AGLT2 HTCondor jobs increases from 50% to 70%, the overall CPU utilization for AGLT2 is 88%, with 66% being used by the HTCondor jobs and 22% by BOINC jobs. This is only measured for 3 days . We need to observe the long term usage.

Manpower for managing

- We experienced a lot of problems and keep improving our monitoring and management scripts, to make the backfilling jobs more autonomous
- Implemented a few scripts, to
  - Dynamically adjust/kill/start backfilling jobs (number of cores per jobs, CPU usage etc.) according to the available resources on the system
  - Measure resource efficiency, part of check_MK checks
  - Remove zombie processes or files left by killed jobs
- After it was stabilized, it takes ~5% of time, to monitor the status. We haven't had issues relating to backfilling jobs for a couple of months.
- For +20% extra CPU, it is worth it!

References

- Wu, Wenjing, David Cameron, and Di Qing. "Using ATLAS@ Home to exploit extra CPU from busy grid sites." *Computing and Software for Big Science* 3.1 (2019): 8.