Computing and Software for Big Science (2019) 3:8
https://doi.org/10.1007/s41781-019-0023-6

ORIGINAL ARTICLE

Using ATLAS@Home to Exploit Extra CPU from Busy Grid Sites

Wenjing Wu[1] · David Cameron[2] · Di Qing[3]

Received: 26 November 2018 / Accepted: 10 April 2019

Abstract
Grid computing typically provides most of the data processing resources for large high-energy physics experiments. However, typical grid sites are not fully utilized by regular workloads. To increase the CPU utilization of these grid sites, the ATLAS@Home volunteer computing framework can be used as a backfilling mechanism. Results show an extra 15–42% of CPU cycles can be exploited by backfilling grid sites running regular workloads, while the overall CPU utilization can remain over 0.9. Backfilling has no impact on the failure rate of the grid jobs, and the impact on the CPU efficiency of grid jobs varies from 0.02 to 0.11 depending on the configuration of the site. In addition, the throughput of backfill jobs in terms of CPU time per simulated event is the same as for resources dedicated to ATLAS@Home. This approach is sufficiently generic that it can easily be extended to other clusters.

Keywords BOINC · ATLAS@Home · CPU utilization · Grid site · Backfilling

Introduction

Large high-energy physics (HEP) experiments require a huge amount of computing resources for their data processing [1, 2]. The ATLAS experiment is the largest of the LHC experiments in terms of both the pledged and acquired computing resources and its computing infrastructure [3, 4] is built on grid computing. ATLAS jobs are a mixture of single-core and multi-core [5] workflows which typically use between 4 and 12 cores on a single node (depending on site configuration). The real-time computing resources acquired from grid sites to ATLAS in 2018 are around 2.5 million HEPSPEC06[1] [6]. ATLAS also uses an increasing level of opportunistic computing resources such as clouds, High-Performance Computing [7], and volunteer computing.

Even though grid sites provide 75% of the total computing resources to ATLAS, opportunistic computing resources play an important role. One such resource is the volunteer computing project ATLAS@Home [8, 9] which uses the BOINC [10, 11] middleware to harness worldwide heterogeneous volunteer computers. The ATLAS@Home project is integrated into the ATLAS workload management system PanDA [12, 13], and processes ATLAS simulation tasks [14, 15]. Simulation is a CPU intensive task which, on average, consumes over half the wall time of ATLAS CPUs on grid sites.

Most grid sites are clusters managed by batch systems such as HTCondor [16], SLURM [17] and PBS [18], and the scale of the sites ranges from a few hundred to tens of thousands of cores. However, when the CPU time utilization of several dedicated ATLAS grid sites was measured, results showed that none of these clusters were being fully used. In other words, both the wall time utilization and CPU time utilization rates were not as high as expected even there were enough jobs in the queues to saturate the sites. This means that a significant percentage of cluster resources were being wasted due to not fully utilizing the idle CPU cycles and, hence, the need to seek solutions to improve the CPU time utilization.

The rest of this paper is organized as follows: Sect. 2 analyzes the CPU time utilization of the ATLAS grid sites, Sect. 3 introduces a new method of backfilling the grid sites, Sect. 4 presents results of backfilling two ATLAS grid

✉ Wenjing Wu
wuwj@ihep.ac.cn

1 Institute of High Energy Physics, CAS, 19B Yuquan Road, Beijing 100049, China

2 Department of Physics, University of Oslo, P.b. 1048 Blindern, 0316 Oslo, Norway

3 TRIUMF, Vancouver, BC V6T2A3, Canada

[1] HEPSPEC06 is the HEP-wide benchmark for measuring CPU performance and the official CPU performance metric used by the Worldwide LHC Computing Grid. The average performance of one CPU core is around 10 HEPSPEC06 for the ATLAS grid sites.

Table 1 The average utilization of typical ATLAS grid sites over a period of 100 days

Site	Amount of cores	Avg. ϵ_{CPU}	Avg. u_{cpu}	Avg. u_{wall}
BEIJING	634	0.81	0.55	0.68
SiGNET	5288	0.77	0.68	0.88
TOKYO	6144	0.85	0.72	0.85
AGLT2	10,224	0.84	0.61	0.72
MWT2	16,250	0.84	0.70	0.83

sites, Sect. 5 measures the impact of backfilling, and Sect. 6 concludes.

Utilization of Grid Sites

Analysis from the ATLAS Job Archive

To understand the utilization rate of grid sites, a few example sites from ATLAS are studied. The selected sites are of different scale and have a large geographic separation, and they are dedicated Tier 2 sites to ATLAS, so the CPU time and wall time of ATLAS jobs is representative of the overall usage of the clusters. CPU efficiency (ϵ_{CPU}) is used to measure the efficiency of the jobs, and wall time utilization (u_{wall}) and CPU time utilization (u_{cpu}) measure how fully these clusters are being utilized. Assuming that, in a given period M days(including downtime), the total wall time (in seconds) of all jobs is T_{wall}, the total CPU time (in seconds) of all jobs is T_{CPU}, and the total number of available cores of the site is N_{core}, then:

$$u_{\mathrm{wall}} = \frac{T_{\mathrm{wall}}}{3600 \times 24 \times M \times N_{\mathrm{core}}}, \tag{1}$$

$$u_{\mathrm{cpu}} = \frac{T_{\mathrm{cpu}}}{3600 \times 24 \times M \times N_{\mathrm{core}}}, \tag{2}$$

$$S_{\mathrm{CPU}} = \frac{T_{\mathrm{cpu}}}{T_{\mathrm{wall}}}. \tag{3}$$

As shown in Table 1, five ATLAS sites were chosen from Asia, North America, and Europe. They have different scales in terms of the number of cores, and they use different local batch systems. From the selected sites, the average u_{wall} is around 0.85, and the corresponding u_{cpu} is around 0.70. Ideally, u_{wall} should be close to 1, but there are several reasons why grid sites cannot achieve this, as follows.

1. Sites often have downtime for scheduled maintenance or unexpected problems.
2. The inefficiency of both the grid scheduling system and local batch systems. In the ATLAS case, the central PanDA scheduling system is rather conservative, and sites are assigned fewer jobs during the periods before and after downtimes.
3. Over 50% of the ATLAS worker nodes run multi-core jobs which have lower CPU efficiency compared to the single-core jobs (the same job running in multi-core can drop 0.05–0.15 in CPU efficiency value compared to single-core). This is due to the fact that certain stages of the multi-core job can only use a single core and, hence, leave the other allocated cores idle.
4. Sites with fixed partitioning of worker nodes between single-core and multi-core ATLAS jobs can have idle worker nodes when the mix of workloads assigned to the site does not well match the partition well.
5. For sites configured to mix single- and multi-core jobs (multi-core number could range from 4 to 128 depending on the sites' configuration) on the same worker nodes, the multi-core jobs may need to wait for a number of single-core jobs to finish to obtain the number of cores which they require.

In the best case, even if the site has u_{wall} of 1, u_{cpu} would still be less than 1, because the CPU efficiency of the jobs is always less than 1, so the CPU time utilization is always lower than wall time utilization. Different types of job demonstrate different CPU efficiency, in the case of ATLAS computing, the CPU efficiency of jobs can range from lower than 0.5 to as high as 0.98 depending on the ratio of IO operations in the job.

Observations from a Site's Local Monitoring

Using local monitoring tools to look at the CPU time utilization of single-worker nodes in different periods, it was observed that, in the long run, the CPU time utilization of the worker nodes was not as high as expected.

On a worker node for the ATLAS BEIJING site, the CPU time utilization of grid jobs can reach 91% over a 24 hour period, because this worker node is running highly CPU efficient simulation jobs. However, on the same worker node, looking over a period of 2 weeks, the CPU time utilization is only 0.69. This is because the site had two scheduled downtime in those 2 weeks, and also because of the inefficiency of the job scheduling and the jobs.

Using ATLAS@Home to Backfill the Sites

The Basic Idea

From Sect. 2, it can be seen that, with the traditional batch system assignment of one job slot per core, the CPU cycles can never be 100% utilized due to the job CPU efficiency. The key is to have more than one job slot on each core, but jobs must have different priorities; otherwise, more wall time and CPU time would be wasted on the scheduling of CPU cycles between different jobs at the operating system level. In addition, sites use different batch systems, so it is not easy to implement a universal configuration for all batch systems, and some batch systems may not support the feature of defining more than one job slot per core and assigning different priorities to different jobs.

Using ATLAS@Home meets the above requirements in terms of being independent from the site's local batch system and having the ability to use different job priorities. Using the ATLAS@Home platform to run ATLAS@Home jobs in the background of the regular grid job workload effectively exploits CPU cycles which cannot be fully utilized by the grid jobs.

The Advantages of ATLAS@Home jobs

When ATLAS@Home started, it was aimed towards the general public; most of whom were running hosts with the Microsoft Windows operating system. Therefore, it was developed to use virtualization to provide the required Linux-based computing environment (operating system and dependent software installation). Later, as more and more Linux hosts joined the project, containerization and native running were developed to replace virtualization on Linux hosts. This improved the average CPU efficiency of the ATLAS@Home jobs by up to 0.1 and is also more lightweight to deploy as it does not require the pre-installation of virtualization software.

Like many volunteer computing projects, the ATLAS@Home project uses the BOINC middleware to manage job distribution to volunteer hosts. A BOINC project defines jobs in a central server, and volunteers install the BOINC client software and configure it to pull jobs from the servers of the projects to which they would like to contribute. A grid site volunteers to run ATLAS@Home installs the BOINC client on its worker nodes and configures it to take jobs from the ATLAS BOINC server. So far, five ATLAS Tier 2 sites have started to use ATLAS@Home to backfill their entire clusters. In this paper, "BOINC jobs" are defined as the jobs which BOINC controls on a worker node (as opposed to grid jobs controlled by a batch system), whereas ATLAS@Home is the general framework for volunteer computing in ATLAS.

One key feature of BOINC is that the processes are set to the lowest priority in the operating system, so they only use CPU cycles when they are not being used by any other higher priority processes. In particular, for Linux systems, it uses the non-preempt scheduling mechanism for CPU cycles, which means that the higher priority processes will always occupy the CPU unless they voluntarily release it. This feature guarantees that starting low priority processes, such as all the processes spawned by the BOINC jobs, will not increase the wall time of the higher priority processes due to switching CPU cycles between processes. Hence, BOINC should not impact the CPU efficiency of the higher priority grid jobs. Of course, the CPU efficiency might be lower due to the memory contention of both jobs (overflowing of memory into swap space can prolong the wall time of the jobs).

Another advantage of using BOINC to add the extra job slots is that these jobs are from two different batch systems: the higher priority jobs from the local batch system of the cluster and the lower priority jobs from BOINC. They are invisible to each other, and the local batch system does not know that the BOINC jobs exist, so it will still send as many jobs as it is configured to. In other words, this does not affect the wall time utilization of the higher priority grid jobs.

BOINC provides a convenient way to schedule payloads to the worker node, because it is already fully integrated into ATLAS distributed computing systems. Alternative methods of over-committing resources would require either requesting sites to re-configure batch systems to allow over-commit, or developing a way to schedule jobs behind the batch system—essentially duplicating BOINC's functionality.

The multi-core simulation jobs of ATLAS@Home use very little memory (less than 300 MB per core for 12-core jobs), and the majority of ATLAS grid jobs (except for special jobs requiring higher memory) use less than 1.5 GB memory per core. This means that grid jobs and BOINC jobs usually have enough memory to co-exist on the same worker node, and the BOINC jobs can also be kept in memory, while they are suspended (if, for example, no CPU cycles are available). Therefore, the BOINC jobs do not get preempted even if the grid jobs are using 100% of the CPU, and hence, no CPU cycles are wasted. The ATLAS@Home jobs can also run in the backfilling model on sites with jobs from other applications or VOs, as we also successfully tested this model in a cluster with a mixture of jobs from over ten different applications and VOs.

There is on-going work to integrate ATLAS@Home with the ATLAS Event Service [19], a framework which reduces the granularity of processing from the job-level to the event-level. Events are uploaded to grid storage as they are produced which make it ideal for opportunistic resources where jobs may be terminated at any point. For ATLAS@Home, it will be useful in cases where memory requirements are tighter and BOINC jobs cannot be held in memory, so that, when a BOINC job is preempted, only the current event being processed is lost.

Table 2 Utilization of BEIJING site in a busy week

	f_s	ϵ_{CPU}	u_{cpu}	u_{wall}
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

Table 3 Utilization of BEIJING site in an idle week

	f_s	ϵ_{CPU}	u_{cpu}	u_{wall}
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

The Harvest from the Grid Sites

The ATLAS@Home backfill method was tested on two ATLAS grid sites. The first is a small site in China (BEIJING) which has 464 cores and PBS as its batch system, and the second is a large site in Canada (TRIUMF) which has 4816 cores and HTCondor as its batch system. Both sites are dedicated to ATLAS, so the ATLAS job measurements can serve as an overall measure of the sites' efficiency. The BOINC software was deployed on both clusters, and the worker nodes received jobs from ATLAS@Home to run in the background, while the grid jobs were also running. To compare the difference, the CPU time utilization and wall time utilization defined in Sect. 2.1 are used.

Results from the BEIJING Site

Backfilling was started on the BEIJING site in September 2017. Results from both ATLAS job monitoring and local monitoring during this period suggest that the CPU time exploited by BOINC is dependent on the wall time and CPU time utilization of the grid jobs, having both a higher u_{cpu} from grid jobs and the supplementary BOINC backfilling jobs can lead to a higher u_{cpu} of the cluster. In addition to the u_{cpu}, u_{wall}, and S_{CPU} metrics defined in Sect. 2.1, an additional metric f_s was used to measure the effect of BOINC jobs on the success rate of grid jobs. f_s is defined as the ratio between successful jobs and total jobs.

Tables 2, 3 show the utilization of BOINC, Grid, and All jobs over two different periods of 7 days. In a busy week, the average u_{wall} of the grid jobs reaches 0.93, and the corresponding u_{cpu} is 0.80. Under these circumstances, BOINC backfilling jobs can exploit an extra 15% CPU time from the cluster, which makes the average overall u_{cpu} of the cluster reach 0.95. With backfilling jobs, the average overall u_{wall} is 1.81, which means that there are, on average, 1.81 ATLAS processes (from both the grid and BOINC backfilling jobs) running or waiting on each core.

In an idle week, the u_{wall} of the grid jobs is only 0.62, and the corresponding u_{cpu} of grid jobs is 0.48. In this case, the BOINC backfilling jobs exploit an extra 42% CPU time, which makes that the overall u_{cpu} of the cluster reach 0.90.

It can be seen that BOINC backfilling can exploit the CPU cycles which cannot be used by grid jobs, and the u_{cpu} of BOINC jobs depends on the u_{cpu} of the grid jobs. In addition, the overall u_{cpu} also depends on the u_{cpu} of the grid jobs; usually, higher u_{cpu} of grid jobs yields higher overall u_{cpu}; For 6 months in BEIJING, the average overall u_{cpu} of the site remains above 0.85.

Results from the TRIUMF Site

For the TRIUMF site, the overall u_{cpu} of the site before and after adding the BOINC backfilling jobs is compared.

Table 4 shows a 7-day period before adding the backfilling jobs, during which the average overall u_{cpu} is 0.69. Table 5 shows a 7-day period when backfilling was enabled, when the average overall u_{cpu} is 0.92 of which 0.27 is exploited by the backfilling jobs. It is also notable that the average u_{wall} of grid jobs after is 0.09 higher; in other words, the backfilling jobs do not affect the throughput of the grid jobs; after adding the backfilling jobs, the overall u_{wall} of the cluster is 1.88, corresponding to an average 1.88 ATLAS processes running or waiting on each core.

Measuring the Effects of Backfilling

To understand the impact of the backfilling jobs on the grid jobs and vice versa, several metrics are used to compare them: the ϵ_{CPU} and f_s defined, respectively, in Sects. 2.1 and 4 for grid jobs, and the CPU time per event for the BOINC jobs.

Failure of Grid Jobs

Tables 2, 3, 4, 5 show that the f_s of jobs for both sites remains very high after adding the backfilling jobs. In fact, the f_s is even 0.05 higher for TRIUMF after adding the backfilling jobs, indicating that the backfilling jobs do not have any negative effect on the grid job success rate.

CPU Efficiency of Grid Jobs

To study the effect of backfilling on CPU efficiency of grid jobs, a reliable and stable set of jobs needed to be found.

Table 4 Utilization of TRIUMF site before backfilling

	f_s	ϵ_{CPU}	u_{cpu}	u_{wall}
BOINC	0	0	0	0
Grid	0.90	0.80	0.69	0.88
All	0.90	0.80	0.69	0.88

Table 5 Utilization of TRIUMF site after enabling backfilling

	fs	ϵ_{CPU}	u_{cpu}	u_{wall}
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

Rather than using all the ATLAS jobs over a certain period of time, only simulation jobs whose wall time was longer than 0.3 CPU days were selected. There were several reasons for this: simulation jobs, on average, use over 50% of a sites CPU time, there is usually a constant flow of them over time, and these jobs have much higher and more stable ϵ_{CPU} compared to the other types of ATLAS jobs. In addition, restricting to jobs longer than 0.3 CPU days leads to average ϵ_{CPU} above 0.95 and increases the sensitivity of the measurement of the effect of backfilling.

Table 6 shows the average ϵ_{CPU} for sample sets of simulation tasks running on the BEIJING and TRIUMF sites, before and after backfilling. For BEIJING, the ϵ_{CPU} of grid simulation jobs drops by 0.0148 after adding the backfilling jobs. This is expected, as a little bit of extra wall time can be added to the grid jobs if there is memory contention between the grid and BOINC jobs when the work nodes are running the large memory ATLAS jobs (around 2% ATLAS jobs require more than 1.5 GB memory per core to run).

When comparing the ϵ_{CPU} in TRIUMF, the difference is larger. The ϵ_{CPU} of grid simulation jobs drops by 0.1113 after adding the backfilling jobs. The drop can mainly be ascribed to two reasons. First, the memory usage of grid jobs in TRIUMF is higher, since it runs 6-core multi-core jobs compared to 12-core in BEIJING. TRIUMF also runs a larger variety of ATLAS jobs, some of which have higher memory requirements. Second, TRIUMF uses c groups [20] to control the resource allocation between grid and BOINC jobs. With cgroups, BOINC jobs could "steal" the CPU cycles from the grid jobs; in other words, with cgroups, BOINC is allocated more CPU cycles than it should have been.

However, this is tunable from both the BOINC and site's resource allocation, depending on whether the goal of the site is to maximize the overall CPU time utilization of the cluster or to minimize the ϵ_{CPU} drop of the grid jobs. For sites dedicated to ATLAS, since both grid and

Table 6 CPU efficiency of grid jobs before and after backfilling

Site	Sample size	Avg. ϵ_{CPU}
BEIJING before	930	0.97
BEIJING after	539	0.96
TRIUMF before	2872	0.97
TRIUMF after	3568	0.86

backfilling jobs are ATLAS jobs, a simple policy could be to maximize the overall CPU time utilization. However, other considerations could have an effect on such a policy; for example, grid jobs generally have a higher priority than BOINC jobs and a site could decide the grid jobs should not be penalized in any way.

Impact of Backfilling on ATLAS@Home

The effects on running BOINC jobs in backfill mode can be measured by comparing similar jobs running on dedicated (BOINC-only) nodes and backfill nodes which have the same hardware configuration. The following results came from one set of 48 cores dedicated for BOINC jobs and another set of 400 cores which ran both grid jobs and backfilling jobs. Both the dedicated and non-dedicated cores are from the BEIJING ATLAS Tier 2 site, and the work nodes have the same hardware, so the results are comparable. The metric used for comparison is the consumed CPU time per simulation event processed (a BOINC job consists of processing 200 events).

Since jobs from the same simulation task take a similar time to simulate each event, 8012 sample jobs from 8 different simulation tasks were selected to compare the dedicated and backfill nodes. The sample jobs are the entire ATLAS@Home jobs finished on both the dedicated and non-dedicated cores of BEIJING Tier 2 site mentioned above. As shown in Table 7, for each task, the CPU time per event for the BOINC jobs differs by only 1–4% between the dedicated and backfill cores. In some cases, the CPU time per event on the dedicated cores could be higher than the non-dedicated cores, this is because the CPU time per event is not an absolutely fixed value for jobs within the same task even running on the exact same hardware, and they could vary by within 5%. This indicates that the CPU time exploited by the BOINC backfilling jobs (when they are actually using CPU) is similar to the CPU time from dedicated nodes. The ϵ_{CPU} is a clear indicator of whether the job is run on dedicated or backfilling cores—ϵ_{CPU} for backfilling jobs is much lower, because they have to wait for CPU cycles to be released by higher priority processes.

Table 7 CPU time per event comparison for BOINC jobs

Task	Dedicated sample jobs	Dedicated cpu (s) per event	Dedicated ϵ_{CPU}	Backfilling sample jobs	Backfilling cpu (s) per event	Backfilling ϵ_{CPU}	Offset (%) CPU time per event
1	673	172.02	0.91	3235	165.59	0.35	4
2	15	225.21	0.93	241	219.68	0.32	2
3	59	255.41	0.94	320	246.82	0.49	3
4	255	200.99	0.92	1220	198.55	0.34	1
5	74	211.48	0.93	334	204.66	0.38	3
6	60	289.73	0.94	320	291.78	0.43	1
7	78	481.49	0.95	284	471.89	0.49	2
8	248	218.78	0.93	596	220.32	0.51	1

Conclusion

There are many factors causing low overall CPU efficiency of grid sites, and this study shows that, for ATLAS grid sites, it is very difficult to achieve CPU time utilization above 0.70 of the CPU time available from the site. The ATLAS@Home framework provided a convenient solution to experiment with backfilling grid sites thanks to a few unique and convenient features of the ATLAS@Home jobs. Running BOINC backfilling jobs on two ATLAS grid sites (one small site and one medium size site) has demonstrated that using backfilling can exploit a considerable amount of extra CPU time which could not otherwise be used by grid jobs. With backfilling jobs, the overall CPU time utilization reaches over 0.90 for both sites. This improves the overall CPU time utilization of the cluster by 0.15–0.42 depending on the workload of the grid jobs. The impact of the backfilling jobs was also measured. From the grid jobs' point of view, there is no impact on the failure rate. The impact on the CPU efficiency of grid jobs is between 0.01 and 0.11 depending on the configuration of the site, the memory usage of grid jobs and the resource allocation configuration. From the BOINC jobs' point of view, the CPU time exploited in the backfilling model generates the same amount of events as the CPU time from resources dedicated to BOINC.

Based on both the improvement of the overall CPU time utilization of the site and the impact on the CPU efficiency on the grid jobs, for the sites dedicated to ATLAS, it is recommended to prioritize the improvement of the overall CPU time utilization even if it means sacrificing a small amount of CPU efficiency of grid jobs. For non-dedicated sites, the BOINC resource allocation can be tuned to balance the overall CPU time utilization improvement and the sacrificing of the CPU efficiency of higher priority jobs. This method has so far been deployed on ATLAS grid sites, but, since deploying BOINC for backfilling does not use anything ATLAS-specific, the approach and results could also be extended to general purpose (non-ATLAS) clusters, as long as the u_{cpu} of the cluster does not reach 1. Due to the lack of supporting features from most batch systems and the S_{CPU} of the jobs, it is impossible to have the u_{cpu} of the cluster reach 1 even with fully saturated jobs to the batch system; neither can BOINC serving in a solo model. However, BOINC has the advantage of serving as the background jobs to exploit the idle CPU cycles which are not being fully utilized by the higher priority jobs and yet not impact the higher priority jobs.

Acknowledgements This work was done as part of the distributed computing research and development programme within the ATLAS Collaboration, which we thank for their support. In particular, we wish to acknowledge the contribution of the ATLAS Distributed Computing team (ADC). This project is supported by the Chinese NSF Grants "Research on fine grained Event Service for the BESIII offline software and its scheduling mechanism (No. 11675201)" and "Research on BESIII offline software and scheduling mechanism on desktop Grid No. 11405195". We would also like to thank all the volunteers of ATLAS@Home who made this project possible, and also for the support of NCRC, CFI, and BCKDF (Canada) for the TRIUMF Tier 1 site. ATLAS@Home relies on many products that comprise the ATLAS distributed computing ecosystem, and so, we would like to acknowledge the help and support of PanDA, Rucio, and NorduGrid ARC. More information on running ATLAS@Home may be found on the LHC@Home website: https://lhcathome.cern.ch/lhcathome/

References

1. Shiers J (2007) The worldwide LHC computing grid (worldwide LCG). Comp Phys Commun 177:219–223
2. Bird I, Bos K, Brook N, Duellmann D, Eck C, Fisk I, Foster D, Gibbard B, Girone M, Grandi C et al (2005) LHC computing grid. Technical design report CERN-LHCC 2005-024
3. Campana S (2015) ATLAS distributed computing in LHC run 2. J Phys Conf Ser 664:032004
4. Filipcic A (2017) ATLAS distributed computing experience and performance during the LHC run-2. J Phys Conf Ser 895:052015
5. Calafiura P, Leggett C, Seuster R, Tsulaia V, Van Gemmeren P (2015) Running ATLAS workloads within massively parallel distributed applications using Athena Multi-Process framework (AthenaMP). J Phys Conf Ser 664:072050
6. Bird I (2018) Worldwide LHC computing grid: report on project status, resources and financial plan. CERN report CERN-RRB-2018-023

7. Nilsson P, Panitkin S, Oleynik D, Maeno T, De K, Wu W, Filipcic A, Wenaus T, Klimentov A (2014) Extending ATLAS computing to commercial clouds and supercomputers PoS 034
8. Adam-Bourdarios C, Cameron D, Filipcic A, Lancon E, Wu W (2015) ATLAS@Home: harnessing volunteer computing for HEP. J Phys Conf Ser 664:022009
9. Adam-Bourdarios C, Bianchi R, Cameron D, Filipcic A, Isacchini G, Lancon E, Wu W (2017) Volunteer computing experience with ATLAS@Home. J Phys Conf Ser 898:052009
10. Anderson D (2004) Boinc: a system for public-resource computing and storage. In: Proc. 5th IEEE/ACM international workshop on grid computing, pp 4–10
11. Myers D, Bazinet A, Cummings M (2007) Expanding the reach of grid computing: combining Globus-and BOINC based systems. In: Grid computing for bioinformatics and computational biology, pp 71–84
12. Maeno T (2008) PanDA: distributed production and distributed analysis system for ATLAS. J Phys Conf Ser 119:062036
13. De K, Klimentov A, Maeno T, Nilsson P, Oleynik D, Panitkin S, Petrosyan A, Schovancova J, Vaniachine A, Wenaus T (2015) The future of PanDA in ATLAS distributed computing. J Phys Conf Ser 664:062035
14. Rimoldi A, Dell'Acqua A, Gallas M, Nairz A, Boudreau J, Tsulaia V, Costanzo D (2004) The simulation for the ATLAS experiment: present status and outlook. Nucl Sci Symp Conf Rec 3:1886–1890
15. Yamamoto S, Shapiro M et al (2010) The simulation principle and performance of the ATLAS fast calorimeter simulation FastCaloSim. ATLAS report ATL-COM-PHYS 2010-838
16. Thain D, Tannenbaum T, Livny M (2005) Distributed computing in practice: the condor experience. Concurr Comput Pract Exp 17:323–356
17. Yoo AB, Jette MA, Grondona M (2003) Slurm: simple linux utility for resource management. In: Lecture notes in computer science: proceedings of job scheduling strategies for parallel processing (JSSPP). Springer, Berlin, pp 44–60
18. Feng H, Misra V, Rubenstein D (2007) PBS: a unified priority-based scheduler. ACM SIGMETRICS Perform Eval Rev 35:203–214
19. Calafiura P, De K, Guan W, Maeno T, Nilsson P, Oleynik D, Panitkin S, Tsulaia V, Van Gemmeren P, Wenaus T (2015) The ATLAS event service: a new approach to event processing. J Phys Conf Ser 664:062065
20. Heo T (2015) Control Group v2. https://www.kernel.org/doc/Documentation/cgroupv2.txt. Accessed 4 Feb 2019