PROCEEDINGS OF SCIENCE

Explore multi core virtualization on the ATLAS@home project

Wenjing Wu[1]
IHEP
19B Yuquan Road, Beijing, 100049 China
E-mail:wuwj@ihep.ac.cn

David Cameron[2]
Department of Physics, University of Oslo
P.b. 1048 Blindern, N-0316 Oslo, Norway
E-mail: david.cameron@cern.ch

Andrej Filipcic[3]
Department of Experimental High Energy Physics, Jozef Stefan Institute
Jamova 39, P.o.Box 3000, SI-1001 Ljubljana, Slovenia
E-mail:andrej.filipcic@ijs.si

Abstract. The exploitation of volunteer computing resources has become a popular practice in the HEP (High Energy Physics) computing community because of the huge amount of potential computing power it provides. ATLAS@home, as a pioneer project in this practice, uses the BOINC middleware to harness thousands of worldwide volunteer computers, and it has been harvesting a very considerable number of computing resources since its official launch in 2013. With more and more resource diverse volunteer computers participating in this project, it is very necessary to explore ways to optimize the usage of volunteer computing resources in terms of memory, storage and CPU to qualify more computers of running ATLAS@home jobs. The ATLAS software Athena has already supported multi-core processing, and been running stably and efficiently on its Grid sites for over 1 year. Statistics from Grid sites show that using multi-core processing can significantly reduce the total memory usage upon utilizing the same amount of CPU cores. Based on this practice, we explore the usage of multi-core virtualization on the ATLAS@home project, which is to spawn one virtual machine with all the available CPU cores from the volunteer computer, and then a multi-core Athena job is launched inside this virtual machine. This Athena job uses all the available CPU cores from the virtual machine. The ATLAS@home multi-core application was officially launched in July 2016 and the statistics from a few months' full load running confirmed the reduction of total memory usage, however the performance of this practice is different from the Grid sites due to the virtualization. Through the historical statistics and testing, we find out that factors including the allocation of CPU cores, different versions of hypervisors and the scheduling policies of the BOINC middleware can significantly impact the performance of the multi-core application.

In this paper, we will cover the following aspects of this practice: 1) the implementation of the multi-core virtualization; 2) experiences gained through a few months' full load running; 3) tuning and optimization of its performance.

[1]Wenjing Wu

PoS(ISGC2017)014

International Symposium on Grids and Clouds 2017 -ISGC 2017-
5-10 March 2017
Academia Sinica, Taipei, Taiwan

1. Introduction

The concept and practice of volunteer computing in science became popular in the late 1990s after the big public visibility and success of the SETI@home project [1] which tries to search for Extra Terrestrial Intelligence from the OutSpace. Afterwards, the software used by SETI@home was rewritten and named as BOINC (Berkeley Open Infrastructure for Network Computing) [2][3], so it became a generic middleware which can harness volunteer computers to run computing tasks in a high throughput flavor. Traditionally, the application for the computing had to be ported on different platforms that the volunteer computers provide, together with linkage to the BOINC client APIs which would allow a better control of the running process by the BOINC client. As of writing, the scale of the volunteer community has reached around 500 thousands volunteers providing over 800 thousands volunteer computers which could achieve a real time computing power of 21.2 PetaFLOPS [4].

In 2004, CERN started the LHC@home[5]project at its 50th anniversary. The project features an application named SixTrack which simulates particles traveling through the LHC ring. As the SixTrack application is a small FORTRAN program, it was possible to port it to various platforms such as Windows, Linux and Mac OS. Immediately the LHC@home project attracted intensive attention from the volunteer computing community to the field of High Energy Physics and successfully gained a large amount of CPU to meet its computing requirement. With this successful use case, people started to explore the possibility of running the platform dependent HEP experiment software on volunteer computers. This required virtualization technologies which would convert the heterogeneous volunteer computers into the "volunteer clouds" [6] and provide the suitable operating system and software environment that all the HEP experiment computing require.

The development and advance of the key technologies such as CernVM[7], CVMFS[8], VirtualBox[9] and the vboxwrapper[10] from BOINC made this goal possible, and also yielded very reasonable performance. With the advancements of the related technologies, more and more HEP experiments[11][12]started to explore the possibility of harnessing volunteer computing resources and integrating them into their grid computing platforms, as the grid computing platforms are usually the major computing resources for HEP computing and upon which the computing workflows are designed. ATLAS was the pioneer in this field with the success in both gaining a wide range of publicity and the amount of free resource from its volunteer computing project ATLAS@home [13][14].

2. ATLAS@home project and its harvest

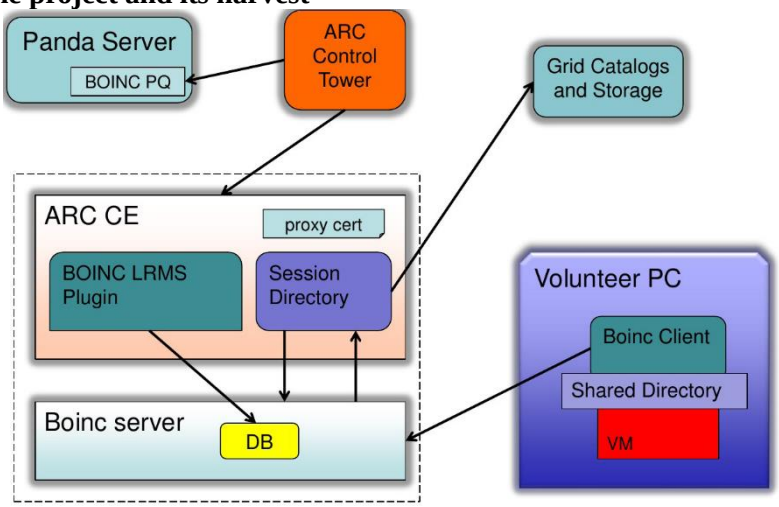

Figure 1. The architecture of ATLAS@home

There are a few reasons to start the ATLAS@home project: to add computing resource to the current ATLAS distributed computing platform; to increase the publicity of the ATLAS experiment and find solutions to manage the Tier 3 resources which are usually owned by collaborating users without IT expertise. With all these goals, the ATLAS@home project started

its development work in 2013, with its first prototype server located at IHEP, Beijing.

In the development of ATLAS@home, the main challenge was to integrate the volunteer computing resource into the ATLAS PanDA [15] system in a secure way as there was the contradict between the GSI authentication requirement of the grid services and the untrusted feature of volunteer computers. As shown in Figure 1, thanks to the flexibility in the implementation of the ATLAS pilot[16] and the feature of ARC CE[17] which could separate the job payload running from its data operations, we adopted the ARC CE to bridge the volunteer computing resource and the PanDA system, so the dynamical resource pool of volunteer computing appears as a grid site to the PanDA system in a transparent way, and yet no user credentials which are usually required to be placed on the work nodes of grid sites need to be placed on any of the volunteer computers. The ATLAS@home "site" processes simulation jobs which handles a small amount of input and output files. A typical ATLAS@home job processes less than 100 events simuation which normally takes a few hours CPU time, and the corresponding input and output files are respectively around 100MB.This design is to suit the feature of volunteer host.More details of the ATLAS@home system design can be found in paper[13].

With a few months of development work and beta testing, the project was officially launched to the public in the early 2014. Within a few months of production running, ATLAS@home attracted over 20 thousands volunteers registering with over 40 thousands computers, and the scale of resource kept growing until it reached a stable level of around 80 thousands registered computers. Soon, ATLAS@home became the second largest simulation site for the ATLAS computing, and accounted for 3% of the whole ATLAS computing power which includes around 150 thousands of dedicated CPU cores from its over 100 grid sites and also opportunistic resources such as HPC centers and cloud computing resource. The ATLAS@home project completed 3.3 Million simulation jobs which consumed 43 Million CPU hours and processed 91 Million events in the physical year of 2016. The number of concurrent running jobs remains around 6000, and the average job failure rate in a year is 11.84% which is nearly 5% lower than that of the overall ATLAS simulation jobs. This proves the resource from ATLAS@home is both sustainable, reliable and steady. As regards to performance measured by the average CPU time spent on a single event, the ATLAS@home site is among the slowest grid sites, but this is expected due to the non-dedication feature of the volunteer computers.

3. ATLAS@home multi core application

3.1 Multi core support in the ATLAS software

Athena [18] is the software framework for the ATLAS production and analysis computing, initially it supported only the single core usage, namely one job utilized one core. In order to save the overall memory usage on a multi core work node, Athena evolved into the AthenaMP [19] which is a mutiple processor application in the software framework and soon it will involve into the multiple threading application. AthenaMP allows one job to utilize a flexible number of cores on a work node. In AthenaMP, the job reads the input event data during the initialization once, and then the event data can be shared among all the cores occupied by this job, hence it uses much less memory compared to running several single core jobs on all these cores separately. Depending on the number of cores allocated to the job, the multi core job can save up to 50% of memory usage.

With this advantage, more and more grid sites start to enable the support of multi core in their local resource manager, and as of February 2017, the ATLAS multi core has been deployed on 62% of its computing nodes.

3.2 Motive to deploy multi core in ATLAS@home

For the ATLAS@home project, the transition from single core to multi core is extremely important as the majority of the volunteer computers have more than one available cores, and in the single core usage,mutiple virtual machines need to be spawned in order to utilize all the available cores. This is very inefficient in terms of memory, hard disk and bandwidth usage.In these days, the performance loss including CPU, memory, hard disk and network has been reduced to below 5% with the layer of virtualization, however the total performance loss of running several instances of virtual machine is still very considerable. Also due to the design of running ATLAS

jobs inside the virtual machine, an extra amount of resource will be wasted if it is to run several virtual machines on the same physical machine. This includes memory usage, hard disk and network traffic, as each single virtual machine requires 2.3GB memory to run a single core simulation job, 10GB of hard disk from the physical host to store both the virtual machine image file and accommodate the extra hard disk space inside the virtual machine, and redundant network traffic as each virtual machine needs to download the software which are not pre-cached in the CVMFS in the image file, and the software cache can't be shared among different instances of virtual machine running on the same physical machine. So in a nutshell, enabling multi core in ATLAS@home will significantly reduce the usage of memory, hard disk and network traffic, which are usually desired improvements by the volunteers and could potentially attract more volunteers to participate in the project, and will definitely lead to a full exploitation of the existing resources attached to the project.

3.3 Multi core support from BOINC

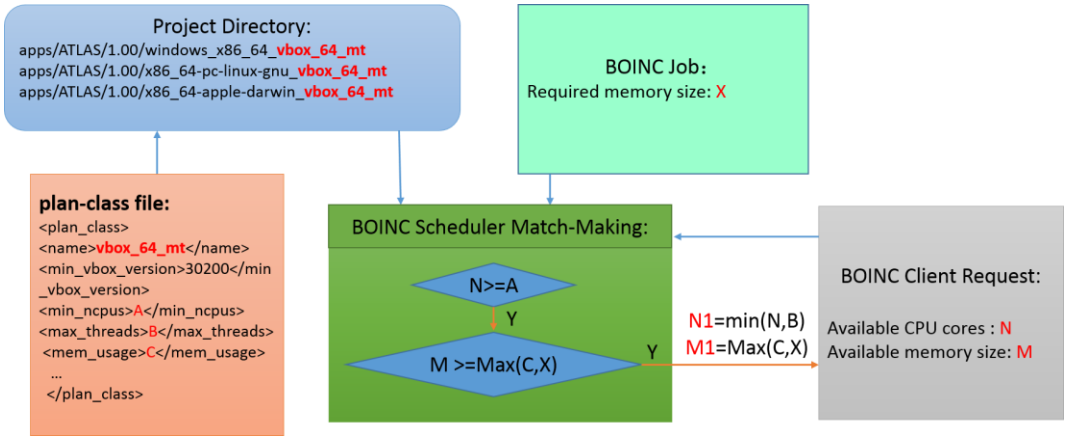

Figure 2. Multi core scheduling in BOINC

In BOINC, there is a piece of software named vboxwrapper which runs on the BOINC client side to control the creation and lifecycle of virtual machines, and it supports allocating multiple cores to a single virtual machine. As shown in Figure 2, on the BOINC server side, there is already a mechanism called plan-class, in which the project can define different plan-class tags and attributes, then associate the attributes with different application directories via the tags. Attributes include the range of CPU cores and memory size for the virtual machine, the minimum and maximum Virtualbox versions required to use on the volunteer hosts. All the attributes are passed to the BOINC scheduler which makes a match-making between the required resources from the job associated to the application and the available resources reported by the client request. However, in the original BOINC scheduler, it only supports a fixed amount of memory from plan-class regardless of the number of CPU cores, and that means the project has to define a memory size based on the maximum number of CPU cores. This isn't efficient given the memory size is proportional to the number of CPU cores in the ATLAS multi core job, and setting up a big memory size will exclude hosts with smaller memory size from running ATLAS jobs.

3.4 Customization for ATLAS@home multi core

One demanded feature for ATLAS@home multi core application is to have dynamical memory allocation according to the actual number of CPU cores used on the volunteer host, hence we expanded the plan-class definition and modified the BOINC scheduler. As shown in Figure 3, in plan-class, there are 2 tags used to specify the memory (mem_usage_base which defines the basic memory usage and mem_usage_per_cpu which defines the extra amount of memory usage for each core), and the overall memory size is calculated by the equation M=C+N*D, whereas M is the total memory size, N is the number of CPU cores allocated to the virtual machine, C is the mem_usage_base and D is the mem_usage_per_cpu. And in the

BOINC scheduler, the actual CPU core number to be allocated to the virtual machine is also calculated based on the available memory size from the client.

Empirically, we set the basic memory usage to 1.3GB, and the extra memory usage for each core to be 1GB, and the core number range from 1 to 12. As shown in Figure 4, the total memory usage is significantly reduced by enabling multi core on the volunteer computer.

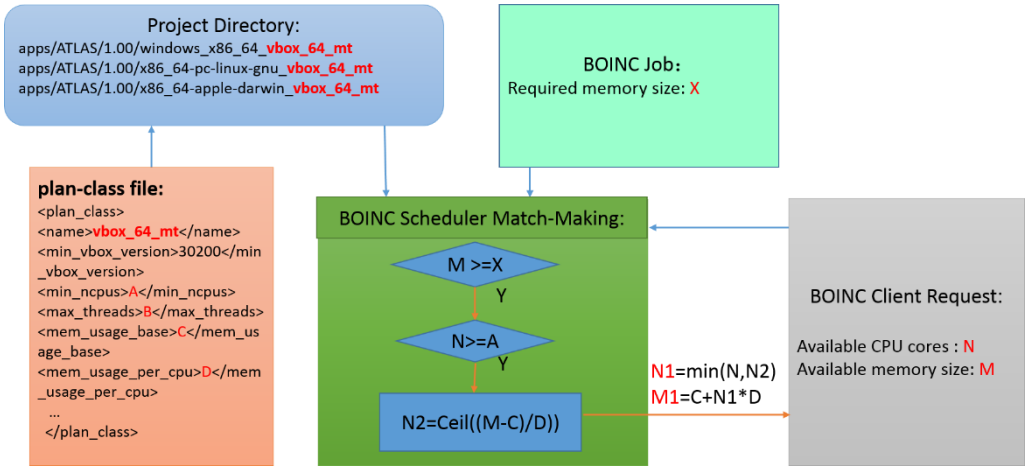

Figure 3. Multi core scheduling in ATLAS@home

Number of Cores	Sing Core Memory Usage(GB)	Multi Core Memory Usage(GB)	Improvement percentage
1	2.3	2.3	0%
2	4.6	3.3	28%
3	6.9	4.3	38%
4	9.2	5.3	42%
5	11.5	6.3	45%
6	13.8	7.3	47%
7	16.1	8.3	48%
8	18.4	9.3	49%
9	20.7	10.3	50%
10	23	11.3	51%
11	25.3	12.3	51%
12	27.6	13.3	52%

Figure 4. Memory usage improvement from multi core

4. Performance issues and tuning

4.1 Performance related to CPU cores

ATLAS@home multi core application initially set the CPU core range between 1 and 12, so it could efficiently use all the available CPU cores that volunteer hosts allocate to the ATLAS@home project. This is especially useful for some of the powerful volunteer hosts. However it is identified from the ATLAS job dashboard [20] that with the same simulation job—100 events per job—the average CPU time of each event on some hosts is 10 times more than the other hosts. As shown in Figure 5, the CPU performance is almost 2 times worse on 12 cores compared to the other number of cores. This plot is based on a statistics from one month period of jobs of the BOINC_MCORE site (the multi core application for the ATLAS@home project). Also as shown in Figure 6, the older version of Virtualbox led to very bad performance with multi cores compared to the new version. With further tests and analysis, two causes are found: 1) the older virtualbox version—older than virtualbox 5.0.0—performs very bad for multi core virtual machines; 2) Using multi core across physical CPUs leads to bad performance too, namely if a physical CPU has only 4 cores, then creating a virtual machine with more than 4 cores results in using cores across different physical CPUs, then the performance drops badly. As the majority of

volunteer hosts has no more than 8 cores on each physical CPU, ATLAS@home multi core app reduces the maximum CPU core to 8 instead of using 12.

4.2 Impacts to the ATLAS@home project

The ATLAS@home multi core application was officially launched in June 2016, and the single core and multi core applications coexisted until December 2016. The volunteers became very excited about the multi core application for its improvements on the resource usages, and gradually migrated from the single core to the multi core application. In order to keep the consistency, the multi core application also supports single core usage, namely the minimum number of cores can be set up to 1. As shown in Figure 7, after switching to multi core, the total CPU wall time from ATLAS@home increased by 20% in a few months, but the total number of simulated events decreased by 10%. There are 3 reasons for it:1)for small simulation jobs which last a few CPU hours, multi core jobs is less CPU efficient(Job CPU efficiency is defined by CPU time dividing Wall time) compared to single core. But for big simulation jobs which last over tens of hours, the difference in CPU efficiency is ignorable between single and multi core jobs. This is because in multi core jobs, during the initialization time and events merging time of the job, more cores would be idle;2)The CPU time per event varies by a few times from simulation task to simulation task, so the total number of events simulated across different tasks is not accurately translated into the amount of CPU power used in simulating these events;3) The availbale resources for ATLAS@home can be dynamical due to feature of volunteer hosts coming and leaving dynamically.

As shown in Figure 8, as regards to the CPU performance measured by the average CPU time spent on each event(assume for each simulation task, jobs of different number of cores all participated in processing jobs from this task), the single core has the best performance while in some multi core cases, the performance can be up to 40% worse compared to the best case. This is rather a software performance and job setup restriction issue.

Based on the above analysis, it is not clear whether ATLAS@home gained more CPU power after switching to multi core, but it is clear that the multi core jobs are generally less CPU efficient compared to single core, so even if the overall computing power from ATLAS@home remains the same, we could lose an overall of 10% of productivity of events due to the CPU efficiency loss. On the volunteers' side, using multi core can save a big amount of memory, hard disk and bandwidth, so this is a feature to please our volunteers, and we also hope that volunteers can allocate more CPU cores to the project because of this optimization gradually. As of revising the paper in October 2017, we find more accurate ways to measure the computing power and CPU efficiency of ATLAS@home from the ATLAS analytic platform, however the analytic platform only stores job arhive information back to August 2016 which is after we switching to multi core in June 2016, so we can't draw conclusions from the ATLAS analytic platform either.

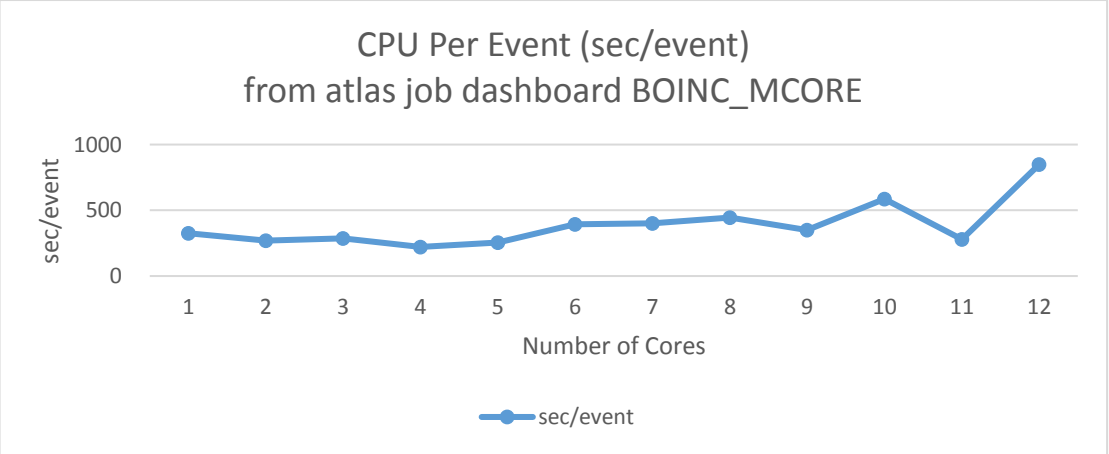

Figure 5. CPU performance differences among different number of cores for the BOINC multi core application

	HOST 1(HT)	HOST 1(HT)	HOST 2 (Non HT)	
VritulBox version/Core No.	v4.2.16(sec/event)	v5.1.2(sec/event)	v5.1.2(sec/event)	Memory (GB)
4	400	330	405	5.7
6	600	355	400	7.3
8	1000	390	490	8.9
10	1500	550	None	10.5
11	None	650	None	11.3
12	2000	735	6328	12.1
		HOST 1: 32 logical Cores, HT enabled	HOST2: 16 Cores, Non HT	

Figure 6. CPU performance difference between different virtualbox versions (The comparison is between Virtualbox v4.2.16 and v5.1.2, HT means enabling Hyper Threading, and Non HT means disabling Hyper Threading)

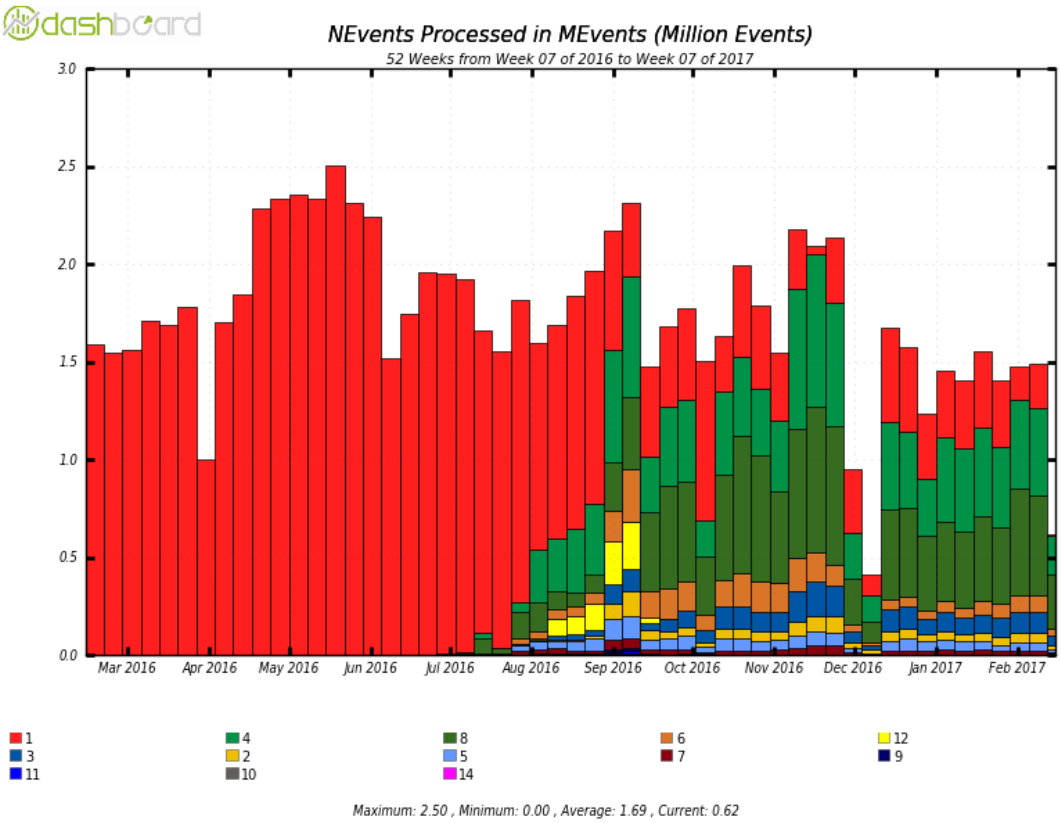

Figure 7. Events processed in a year by different number of cores on ATLAS@home

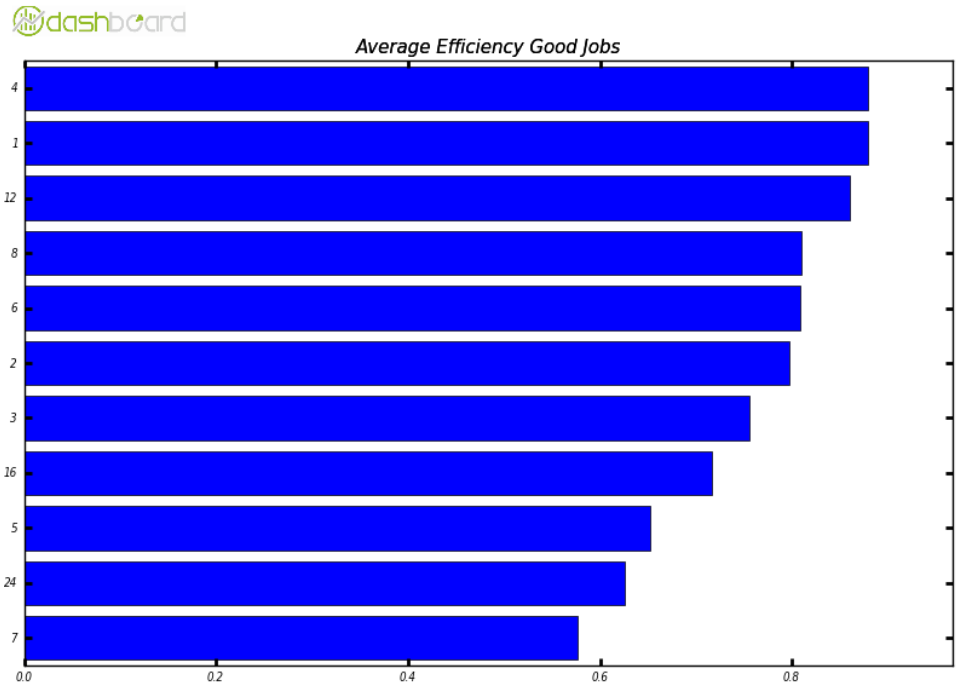

Figure 8. CPU performance difference among different number of cores (The statistics is based on a year period of all ATLAS jobs)

5. Conclusion

ATLAS@home is a pioneer project in applying volunteer computing in the computing of HEP experiments. And it has been providing 3% of the whole ATLAS computing power since 2013. The project keeps scaling up, and in order to efficiently use the available volunteer computing resource, we switched from the single core application to multi core with all the support from the Athena software, BOINC middleware and Virtualbox. Through a few months of production running, it is proved that the multi core application can significantly reduce the resource usage (memory, hard disk, and network) on the volunteer hosts by utilizing the same amount of CPU cores. At the same time, due to the AthenaMP software design, the CPU performance drops for some multi core cases. However, when taking all factors into account, multi core is still a very desirable feature for the ATLAS@home project.

6. Acknowledgements

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 also thank sincerely all our volunteers for their inputs and suggestions during the testing process and their computing resource contributions!

References

[1]. Anderson D P. Boinc: A system for public-resource computing and storage[C]//Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on. IEEE, 2004: 4-10.

[2]. Anderson D P. Boinc: A system for public-resource computing and storage[C]//Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on. IEEE, 2004: 4-10.

[3]. Anderson D P, Fedak G. The computational and storage potential of volunteer computing[C]//Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on. IEEE, 2006, 1: 73-80.

[4]. BOINC usage statistics. http://boinc.berkeley.edu/

[5]. Giovannozzi M, Skands P, Zacharov I, et al. LHC@ HOME: A volunteer computing system for massive numerical simulations of beam dynamics and high energy physics events[C]//Conf. Proc. 2012, 1205201(CERN-ATS-2012-159): MOPPD061.

[6]. Carlos Aguado Sanchez, Predrag Buncic, Wenjing WU et al, Volunteer Clouds and Citizen Cyberscience for LHC Physics, Proceeding of Computing in High Energy and Nuclear Physics, Taipei, 2010, Volume 331, Part 6.
[7]. Buncic P, Sanchez C A, Blomer J, et al. CernVM–a virtual software appliance for LHC applications[C]//Journal of Physics: Conference Series. IOP Publishing, 2010, 219(4): 042003.
[8]. Aguado Sanchez C, Bloomer J, Buncic P, et al. CVMFS-a file system for the CernVM virtual appliance[C]//Proceedings of XII Advanced Computing and Analysis Techniques in Physics Research. 2008, 1: 52.
[9]. Li P. Selecting and using virtualization solutions: our experiences with VMware and VirtualBox[J]. Journal of Computing Sciences in Colleges, 2010, 25(3): 11-17.
[10]. Segal B, Aguado Sanchez C, Buncic P, et al. LHC cloud computing with CernVM[J]. PoS, 2010: 004.
[11]. Field L, Borras H, Spiga D, et al. CMS@ home: Enabling Volunteer Computing Usage for CMS[C]//Journal of Physics: Conference Series. IOP Publishing, 2015, 664(2): 022017.
[12]. Høimyr N, Marquina M, Asp T, et al. Towards a Production Volunteer Computing Infrastructure for HEP[C]//Journal of Physics: Conference Series. IOP Publishing, 2015, 664(2): 022023.
[13]. Cameron D. ATLAS@ Home: Harnessing Volunteer Computing for HEP[R]. ATL-COM-SOFT-2015-018, 2015.
[14]. ATLAS@home website: http://atlasathome.cern.ch
[15]. Maeno T, De K, Klimentov A, et al. Evolution of the ATLAS PanDA workload management system for exascale computational science[C]//Journal of Physics: Conference Series. IOP Publishing, 2014, 513(3): 032062.
[16]. Nilsson P, Caballero J, De K, et al. The ATLAS PanDA pilot in operation[C]//Journal of Physics: Conference Series. IOP Publishing, 2011, 331(6): 062040.
[17]. Filip_ci_c A 2011 arcControlTower: the System for Atlas Production and Analysis on ARC J. Phys.: Conf.Ser. vol 331 p 072013
[18]. Calafiura P, Marino M, Leggett C, et al. The athena control framework in production, new developments and lessons learned [J]. 2005.
[19]. Tsulaia V. Running ATLAS workloads within massively parallel distributed applications using Athena Multi-Process framework (AthenaMP)[C]//Proceedings of the CHEP2015 conference J. Phys.: Conf. Ser.
[20]. ATLAS job dashboard. http://dashb-atlas-job.cern.ch/dashboard/request.py/dailysummary

1.