Consolidation of cloud computing in ATLAS

Ryan P Taylor,[1] Cristovao Jose Domingues Cordeiro,[2] Domenico Giordano,[2] John Hover,[3] Tomas Kouba,[4] Peter Love,[5] Andrew McNab,[6] Jaroslava Schovancova[2] and Randall Sobie,[1] on behalf of the ATLAS Collaboration

[1] University of Victoria, Victoria, Canada
[2] CERN, Geneva, Switzerland
[3] Brookhaven National Laboratory, Upton, USA
[4] Czech Academy of Sciences, Prague, Czech Republic
[5] Lancaster University, Lancaster, UK
[6] University of Manchester, Manchester, UK

Abstract. Throughout the first half of LHC Run 2, ATLAS cloud computing has undergone a period of consolidation, characterized by building upon previously established systems, with the aim of reducing operational effort, improving robustness, and reaching higher scale. This paper describes the current state of ATLAS cloud computing. Cloud activities are converging on a common contextualization approach for virtual machines, and cloud resources are sharing monitoring and service discovery components. We describe the integration of Vacuum resources, streamlined usage of the Simulation at Point 1 cloud for offline processing, extreme scaling on Amazon compute resources, and procurement of commercial cloud capacity in Europe. Finally, building on the previously established monitoring infrastructure, we have deployed a real-time monitoring and alerting platform which coalesces data from multiple sources, provides flexible visualization via customizable dashboards, and issues alerts and carries out corrective actions in response to problems.

1. Introduction

The ATLAS experiment [1] at the LHC now has a long history of activity in cloud computing. Whereas previous efforts progressed from initial exploration and evaluation of cloud technologies [2] to continuing research and development [3] and an eventual transition towards stable operations and integration of cloud resources [4], the activity described in this paper focuses on ongoing improvement and consolidation of the systems and workflows that are now well-established.

2. The Vacuum platform

The Vacuum platform is a model of Virtual Machine (VM) lifecycle management wherein resource providers spontaneously create and contextualize experiment-specific VMs, which obtain jobs to run from a central task queue [5]. In 2016 a new VM definition was developed and deployed on five sites using the Vacuum platform in the UK. As with the previous VM definition for Vacuum sites [6], CernVM [7] is used as a base image. However, the new definition has been rewritten from scratch to use the industry-standard cloud-init[1] format for contextualization, and

[1] `http://cloud-init.io/`

to run standard ATLAS pilot jobs via HTCondor [8] instead of acquiring job payloads directly. This aligns the workflow with that of other ATLAS cloud platforms and the conventional grid methodology, while retaining the feedback-based philosophy of the Vacuum model. Future work is planned to continue this process of convergence between the differing ATLAS VM definitions.

Vacuum resources are integrated into ATLAS distributed computing operations using a dedicated pilot factory [9] to submit pilot jobs. This cleanly isolates Vacuum resources while using the same services that are familiar to the operations team. The management of ATLAS robot proxies is done via the CERN MyProxy [10] service, allowing ATLAS to control which sites may retrieve payloads. The peak capacity for ATLAS on Vacuum computing sites in the UK is approximately 1,500 cores.

3. Simulation at Point 1

The Simulation at Point 1 (Sim@P1) project provides additional computing resources to the ATLAS experiment by opportunistically exploiting the Trigger and Data Acquisition (TDAQ) farm [11]. Using Openstack,[2] up to 70,000 cores in 2,300 compute nodes can be used for offline computing when not needed for TDAQ purposes. Figure 1 shows the usage over the last year. Since the start of Run 2 in March 2015, Sim@P1 produced more than 500 million Monte Carlo events over 46 million CPU-hours, a significant portion of the events simulated for the whole experiment.

Recent developments in the Sim@P1 infrastructure have focused on exploiting the whole farm effectively during short periods of availability. Shifters can now quickly and easily switch resources between the TDAQ and Sim@P1 modes of operation using a simple web interface created by the TDAQ team. In order to consume available resources in a more agile manner, Event Service [12] jobs were used. Rather than taking several hours to complete, Event Service jobs process events and store output continuously. Since this preemptible workload can be terminated at any time, processing time is not wasted when nodes need to be returned to TDAQ use on short notice. Due to the ease of use of the switcher, the effectiveness of Event Service jobs, and the fast ramp-up to convert the farm to running jobs, ATLAS can simulate events on Sim@P1 even during periods of availability as short as a few hours.

Due to the nature of the TDAQ workload for which the farm was designed, the nodes in the farm are suited for CPU-intensive, low-I/O tasks, so Sim@P1 was originally limited to running only simulation jobs. However, there were sometimes not enough jobs of this type to occupy the available resources. Therefore, to increase overall usage efficiency of the farm, reconstruction and reprocessing jobs were enabled despite their somewhat higher I/O and memory requirements. Moreover, the system was reconfigured to use the dynamic partitioning feature of HTCondor in order to run single-core as well as multi-core jobs in a flexible way.

4. CERN cloud benchmark suite

The CERN cloud benchmark suite is a configurable framework for running a selection of benchmarks, and transporting, storing and analyzing the measured results [13]. It enables users to profile the performance of computing resources in a unified way, for the purpose of quantifying variability and inhomogeneity, comparing actual and expected performance, and identifying performance issues. Use cases include running acceptance tests and quantifying procured resources, conducting stress tests, continuous benchmarking to monitor performance on an ongoing basis, and predicting job run times. The suite includes the following benchmarks: DB12 (Dirac Benchmark 2012), a fast benchmark based on Gaussian random number generation with the Python multiprocessing library; Whetstone [14]; and simulating 100 single muon events with the ATLAS kit validation application [15]. The results of the DB12 benchmark

[2] `https://www.openstack.org/`

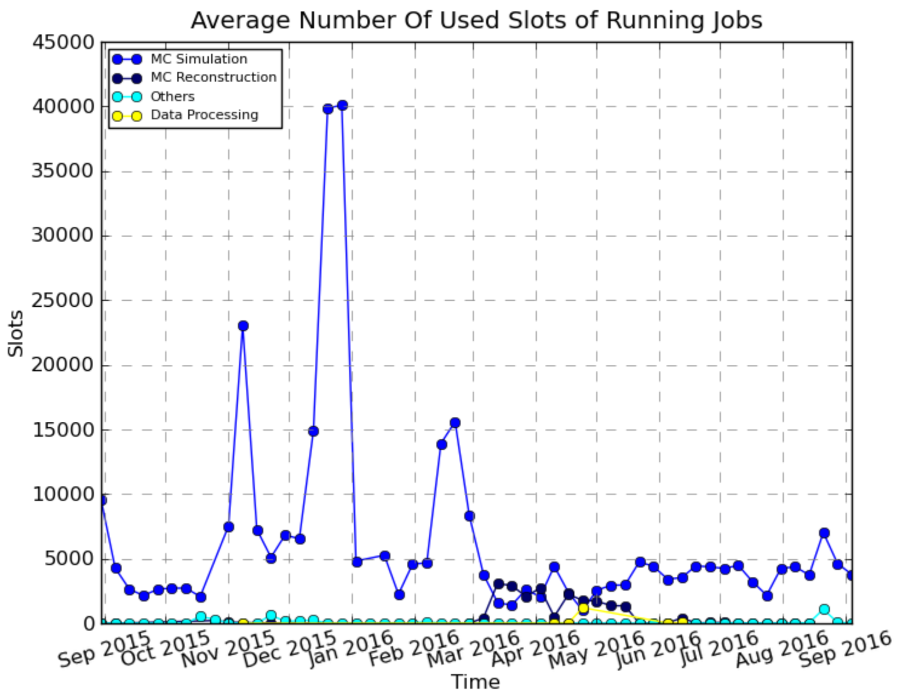


Figure 1. Cores used by Sim@P1 over the last year. Throughout Run 2 approximately 5,000 cores were dedicated to Sim@P1, with higher opportunistic usage of the TDAQ farm during technical stops and machine development periods.

are approximately equivalent to the HEP-SPEC06 benchmark [16], a standard metric for comparison. The suite provides the capability to transport benchmark results via ActiveMQ[3] to an Elasticsearch[4] instance at CERN and then visualize them with Kibana.[5]

One of the applications of the benchmark suite within ATLAS is to quantify the performance of various clouds, and in particular to provide accounting information. This requires a measurement of the HEP-SPEC06 performance of each VM, as well as a measurement of the walltime and CPU time used by the VM. Benchmarking clouds accurately has been challenging because running the HEP-SPEC06 benchmark itself is prohibitively time-consuming, but the use of the DB12 fast benchmark addresses this by estimating the HEP-SPEC06 score in just a few minutes. The benchmark suite is run at boot time on each VM, and the measured performance is stored in a local file. The CPU time spent on running jobs is extracted from the system information in `/proc/stat`. Every 15 minutes each VM transmits the time and benchmark data to a remote server.[6] If a VM has stopped sending data in the last hour, it is assumed to have terminated, so the time and benchmark data are then combined to generate HEP-SPEC06-normalized accounting data. These data are used to generate accounting reports showing the CPU time used by jobs and the total walltime delivered by the cloud provider, thereby giving an estimate of the usage efficiency of the cloud resources, as shown in Table 1.

[3] `http://activemq.apache.org/`

[4] `https://www.elastic.co/products/elasticsearch`

[5] `https://www.elastic.co/products/kibana`

[6] Ideally the time measurements would be taken during the shutdown phase of the VM, but this would not work for opportunistic cloud resources where a VM may be terminated without warning.

Cloud	HEP-SPEC06	CPU time [h]	Wall time [h]	Efficiency
A	15.6	376.1	440.9	85.3%
B	16.7	33742.1	44347.8	76.1%
C	14.2	2721.1	3195.5	85.2%
D	10.8	27.4	30.9	88.7%
E	12.8	47.1	55.1	85.5%
F	21.1	2168.4	2805.3	77.3%
G	9.1	51.1	255.1	20.0%

Table 1. Accounting data for all job types combined, for the 30-day period ending Dec. 6, 2016. The benchmark score is measured by DB12 as an approximation of HEP-SPEC06.

5. Amazon scale tests

Several technical barriers have impeded ATLAS's use of Amazon Web Services (AWS) at large scales. In order to acquire resources on Amazon's Elastic Compute Cloud (EC2) at an economical price, it is essential to use the Spot market, which costs about 4 to 10 times less than using on-demand instances, resulting in core-hour costs comparable to or even less than the cost of purchasing and operating dedicated resources. However, nodes acquired on the Spot market may be terminated at any time, so workloads must be interruptible in order to exploit the Spot market effectively. Additionally, in order to ensure that data storage, access, and transfers are affordable and fast, it is important to establish network peerings and use cloud storage effectively.

The RHIC/ATLAS Computing Facility at Brookhaven National Laboratory has been collaborating with Amazon on a pilot project to address these issues. The event-based processing model of the Event Service was an ideal way to provide the interruptible workload needed to deal with Spot instance termination. During the pilot project, preliminary testing of Event Service jobs uncovered some issues that were addressed by the developers, thereby contributing to the commissioning and improvement of the Event Service system overall.

To attain scalable, high-availability data storage on AWS at low cost, the object-based Amazon Simple Storage Service (S3) must be used. However, work on integrating native use of object stores into the ATLAS distributed data management system is still underway [17], and is complicated by S3's lack of support for third-party transfers and GridFTP. In the meantime, S3-based storage elements were set up to bridge the object storage with the grid, as shown in Figure 2. To enable data access for jobs, the pilot job scripts were adapted to interact with S3 directly.

To prevent data transfers between BNL and Amazon from traveling over the public internet, ESnet has established high-bandwidth (20-100 Gbps) direct network peering between its network and AWS. Because usage of these peering points does not incur costs for Amazon, the usual data egress charges are waived as long as substantial compute resources are purchased.

In September 2015, a 40,000-core scaling run was conducted over one week using 8-core VMs, completing 437,000 ATLAS event generation and simulation jobs on the EC2 US East region (see Figure 3). Input data was automatically pre-subscribed to the region and copied to S3. Output was copied back to ATLAS storage systems asynchronously after the run. A larger-scale run using 80,000 cores was begun in March 2016, but was hampered by an API safeguard put in place by Amazon to protect from denial of service. HTCondor was temporarily blocked from accessing the EC2 API due to the high connection rate resulting from the large number of VMs in use. Once the problem was understood, the HTCondor team updated the software to respond to the error by backing off instead of continually retrying. Unfortunately, by the time the software was ready to perform another scaling test, the Amazon grant had been exhausted. Continuation of the scaling tests is contingent on another procurement.

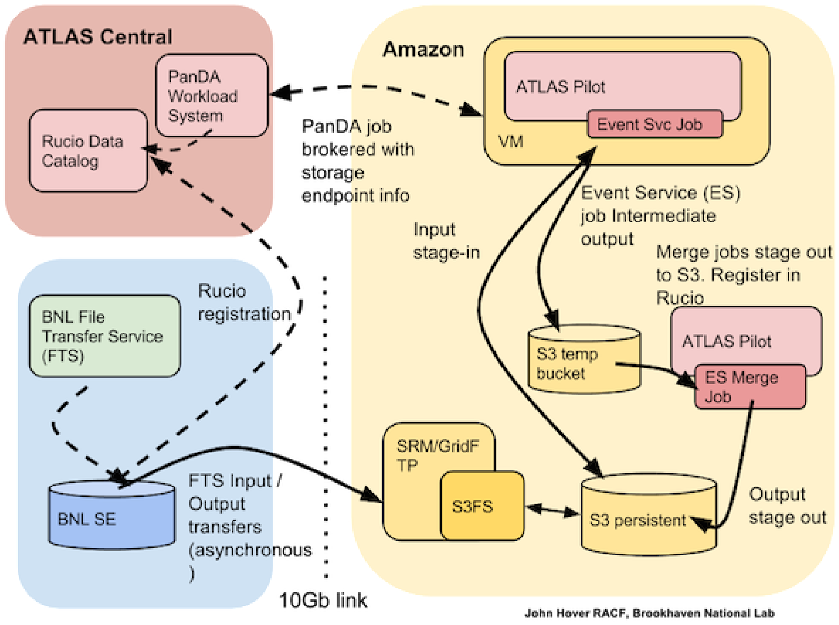

ATLAS Central
PanDA Workload System
Rucio Data Catalog
Amazon
ATLAS Pilot
Event Svc Job
VM
PanDA job brokered with storage endpoint info
Event Service (ES) job Intermediate output
Input stage-in
Merge jobs stage out to S3. Register in Rucio
Rucio registration
BNL File Transfer Service (FTS)
S3 temp bucket
ATLAS Pilot
ES Merge Job
SRM/GridFTP
S3FS
S3 persistent
BNL SE
FTS Input / Output transfers (asynchronous)
Output stage out
10Gb link
John Hover RACF, Brookhaven National Lab

Figure 2. Interfacing Amazon S3 storage with the grid. In each Amazon region, a SRM and GridFTP service were set up to mount the contents of the S3 storage via s3fs, allowing Rucio integration and data movement with FTS.

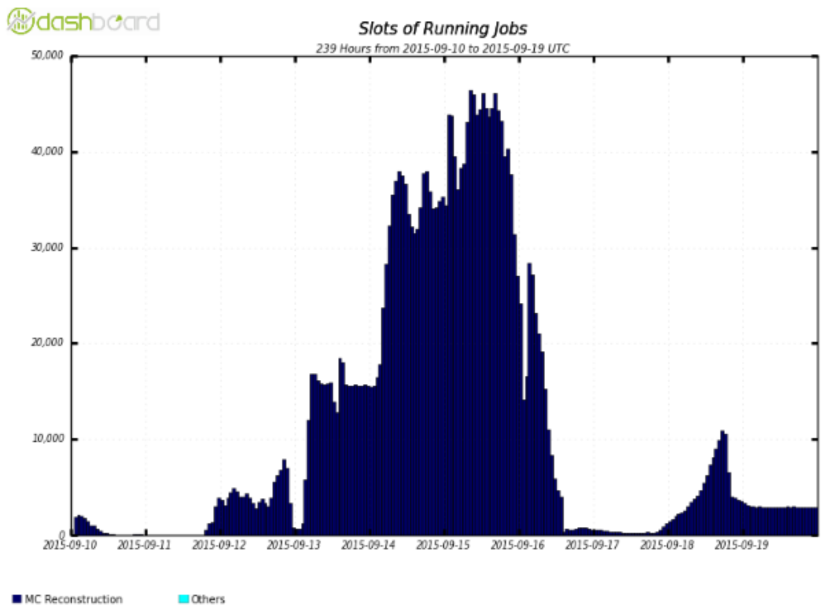


Figure 3. Approximately 40,000 cores used by ATLAS jobs on Amazon EC2.

6. CERN commercial cloud procurement

Over the past two years, CERN procurement initiatives have led to several partnerships between the CERN IT department and commercial cloud providers such as ATOS, Microsoft Azure, T-Systems, Deutsche Börse Cloud Exchange, and IBM SoftLayer [18]. In the latest procurement, CERN awarded a four-month contract to T-Systems to use the Open Telekom Cloud IaaS for physics data processing. The resources consisted of 1,000 VMs, each with 4 vCPUs, 8 GB of RAM, 100 GB of local disk storage, and a public IPv4 address. In addition, 500 TB of disk space was provided on a central cloud storage system and interfaced to the grid using a DPM[7] storage element. The site was connected to CERN over the GÉANT network with a dedicated bandwidth of 10 Gbps.

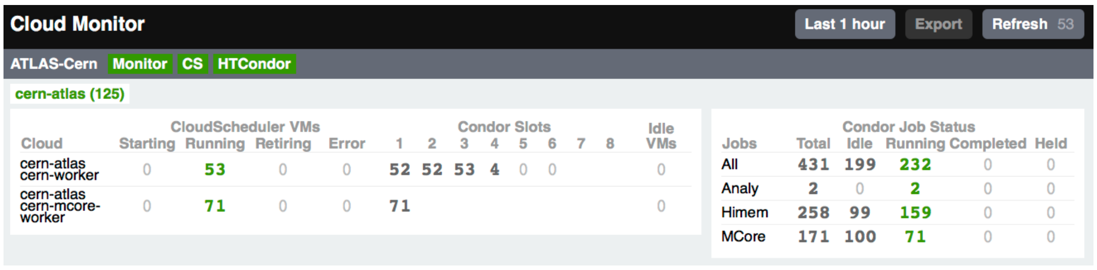
The resources were exposed to the four WLCG virtual organizations via CERN's HTCondor batch system. ATLAS submitted several kinds of workloads, ranging from Monte Carlo simulation to reconstruction of simulated data and real physics data. Some of the jobs were configured to remotely access data from the EOS[8] storage system at CERN over the WAN, while others were configured to access data from the central cloud storage.

7. Cloud monitoring

When operating several HTCondor/Cloud Scheduler [19] instances, each with several clouds, job types, and VM types, it is essential to have a unified monitoring interface to present a coherent and real-time view of all VMs and jobs [20]. The Cloud Monitor stack [21] accomplishes this using an array of industry-standard tools, and some specially developed daemons and web apps. The data collection and transport layer consists of Logstash[9] for collecting and forwarding log files, Ganglia[10] and Sensu[11] for collecting system metrics and other data, and RabbitMQ for message transport. The data processing and storage layer uses Graphite[12] for time-series data storage, the Elasticsearch datastore and search engine for log messages, and MongoDB[13] for storing structured but schema-less data. The visualization layer uses Grafana[14] for flexible dashboarding and graphing of time-series data, Kibana for the display and analytics of log data, and a web front-end built with Flask,[15] PyMongo,[16] and Plotly.js.[17] Ansible[18] is used to automate the deployment and configuration of all the components in the Cloud Monitor stack, which ensures consistency of the services, reduces overall complexity of managing the stack, and provides a self-documenting approach to configuration.

Each HTCondor/Cloud Scheduler server runs the Sensu client, which schedules health checks of the VM management and batch job services, and triggers email notifications or corrective event handlers in case a check fails. The Sensu client also schedules the periodic execution of a data collection script which gathers information on the status of running jobs and VMs from three sources: HTCondor, Cloud Scheduler, and the clouds hosting the VMs.[19] These data are then timestamped and transmitted using the RabbitMQ message bus.

[7] http://lcgdm.web.cern.ch/dpm
[8] http://eos.web.cern.ch/
[9] https://www.elastic.co/products/logstash
[10] http://ganglia.info/
[11] Sensu also provides health checks and alerting. https://sensuapp.org/
[12] https://graphiteapp.org/
[13] https://www.mongodb.com/
[14] http://grafana.org/
[15] http://flask.pocoo.org/
[16] https://api.mongodb.com/python/current/
[17] https://plot.ly/javascript/
[18] https://www.ansible.com/
[19] If there is any discrepancy between the list of VMs as reported by Cloud Scheduler and by the clouds themselves, it is flagged for an administrator to reconcile.



cern-atlas (125)

	CloudScheduler VMs				Condor Slots								Idle
Cloud	Starting	Running	Retiring	Error	1	2	3	4	5	6	7	8	VMs
cern-atlas cern-worker	0	53	0	0	52	52	53	4	0	0			0
cern-atlas cern-mcore-worker	0	71	0	0	71								0

	Condor Job Status				
Jobs	Total	Idle	Running	Completed	Held
All	431	199	232	0	0
Analy	2	0	2	0	0
Himem	258	99	159	0	0
MCore	171	100	71	0	0

Figure 4. A portion of the Cloud Monitor overview page, showing just one Cloud Scheduler instance named "ATLAS-Cern", which runs three types of jobs (analysis, high-memory, and multi-core) on just one cloud.

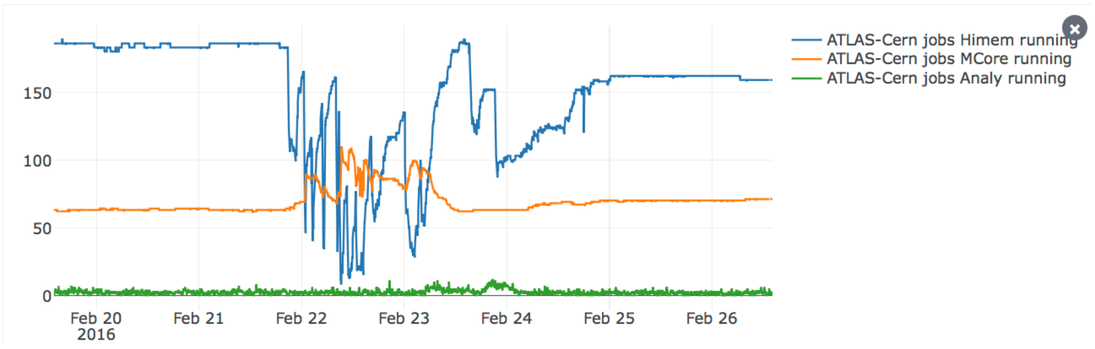


Figure 5. A graph of the number of jobs of different types running over a one-week period on the ATLAS-Cern Cloud Scheduler instance.

On the Cloud Monitor server, a daemon called `cmon` consumes the data from RabbitMQ, processes it, and stores it in Graphite and MongoDB. The Cloud Monitor server also hosts a Ganglia `gmetad` service which is simply configured to ingest data from the Ganglia monitoring system that was established in 2014 and described in Ref. [4]. In this way all the previously available monitoring information is incorporated into the Cloud Monitor stack with minimal additional overhead. Moreover, the ingested Ganglia metrics are dumped into Graphite to enable additional analytics, further leveraging the capabilities of the Cloud Monitor.

The Cloud Monitor web front-end was developed in order to concisely display a unified overview of all clouds, VMs and jobs in one window (see Figure 4), with hierarchical views showing more in-depth information and links to the web interfaces of each subsystem. Any combination of monitored metrics can be selected for display in a time-series graph over a configurable interval (see Figure 5). Any cloud can be clicked on to show the list of VMs and jobs on that cloud and their statuses. Any VM or job can be selected to show its event history, and each VM is cross-referenced with the jobs which ran on it. Moreover, the detailed view of a job shows the log output from that job.

8. Summary

During the first half of LHC Run 2, the ATLAS cloud computing group has taken the opportunity to make improvements in several areas. The new monitoring system reduces the operational effort needed to keep cloud resources running smoothly, and has a positive effect on system reliability since problems can be identified and corrected more easily. By simplifying the benchmarking process, the benchmarking suite allows us to gain deeper insight into performance characteristics and potential problems of clouds, and facilitates accounting. The Vacuum platform has converged towards using contextualization and job workflow techniques similar to those used elsewhere in ATLAS, which allows shared operational support and reduces the effort needed to maintain different approaches. With regard to commercial clouds, projects at CERN and BNL have resulted in valuable operational experience and economical resource acquisition at unprecedented scales. Improvements to the Sim@P1 infrastructure allow the significant resources of the TDAQ farm to be used for offline processing even during short LHC stops. These enhancements put ATLAS in a favourable position to continue successfully leveraging cloud technologies for the remainder of Run 2.

References

[1] The ATLAS Collaboration 2008 The ATLAS Experiment at the CERN Large Hadron Collider *JINST* **3** S08003 (doi:10.1088/1748-0221/3/08/S08003)

[2] Barreiro Megino F H *et al* on behalf of the ATLAS Collaboration 2012 Exploiting virtualization and cloud computing in ATLAS *J. Phys.: Conf. Ser.* **396** 032011 (doi:10.1088/1742-6596/396/3/032011)

[3] Panitkin S *et al* on behalf of the ATLAS Collaboration 2014 ATLAS cloud R&D *J. Phys.: Conf. Ser.* **513** 062037 (doi:10.1088/1742-6596/513/6/062037)

[4] Taylor R P *et al* on behalf of the ATLAS Collaboration 2015 The evolution of cloud computing in ATLAS *J. Phys.: Conf. Ser.* **664** 022038 (doi:10.1088/1742-6596/664/2/022038)

[5] McNab A 2016 The Vacuum platform. Proceedings of the 22nd CHEP conference *J. Phys.: Conf. Ser.*

[6] McNab A, Love P, MacMahon E 2015 Managing virtual machines with Vac and Vcycle *J. Phys.: Conf. Ser.* **664** 022031 (doi:10.1088/1742-6596/664/2/022031)

[7] Blomer J *et al* 2014 Micro-CernVM: slashing the cost of building and deploying virtual machines *J. Phys.: Conf. Ser.* **513** 032009 (doi:10.1088/1742-6596/513/3/032009)

[8] Thain D, Tannenbaum T, Livny M 2005 Distributed computing in practice: the Condor experience *Concurrency Computat.: Pract. Exper.* **17** pp 323-356 (doi:10.1002/cpe.938)

[9] Caballero J *et al* on behalf of the ATLAS Collaboration 2012 AutoPyFactory: A scalable flexible pilot factory implementation *J. Phys.: Conf. Ser.* **396** 032016 (doi:10.1088/1742-6596/396/3/032016)

[10] Basney J, Humphrey M, Welch V 2005 The MyProxy online credential repository *Softw: Pract. Exper.* **35** pp 801-816 (doi:10.1002/spe.688)

[11] Kouba T 2016 Evolution and experience with the ATLAS Simulation at Point1 project. Proceedings of the 22nd CHEP conference *J. Phys.: Conf. Ser.*

[12] Calafiura P *et al* 2015 The ATLAS Event Service: A new approach to event processing *J. Phys.: Conf. Ser.* **664** 062065 (doi:10.1088/1742-6596/664/6/062065)

[13] Giordano D *et al* 2016 Benchmarking cloud resources. Proceedings of the 22nd CHEP conference *J. Phys.: Conf. Ser.*

[14] Curnow H J, Wichman B A 1976 A Synthetic Benchmark *Computer Journal* **19** (1) pp 43-49 (doi:10.1093/comjnl/19.1.43)

[15] De Salvo A, Brasolin F 2010 Benchmarking the ATLAS software through the Kit Validation engine *J. Phys.: Conf. Ser.* **219** 042037 (doi:10.1088/1742-6596/219/4/042037)

[16] Michelotto M *et al* 2010 A comparison of HEP code with SPEC benchmarks on multi-core worker nodes *J. Phys.: Conf. Ser.* **219** 052009 (doi:10.1088/1742-6596/219/5/052009)

[17] Garonne V, Guan W 2016 Object-based storage integration within the ATLAS DDM system. Proceedings of the 22nd CHEP conference *J. Phys.: Conf. Ser.*

[18] Cordeiro C 2016 CERN computing in commercial clouds. Proceedings of the 22nd CHEP conference *J. Phys.: Conf. Ser.*

[19] Sobie R *et al* 2016 Context-aware distributed cloud computing using Cloud Scheduler. Proceedings of the 22nd CHEP conference *J. Phys.: Conf. Ser.*

[20] Sobie R 2016 Context-aware cloud computing for HEP *Proceedings of Science* PoS(ISGC 2016)033

[21] Ring D 2016 Dynamic monitoring and alerting for high throughput computing clouds, University of Victoria Faculty of Engineering Work Term Report (`http://cern.ch/go/66zv`)