EPJ Web of Conferences **295**, 07035 (2024) https://doi.org/10.1051/epjconf/202429507035
CHEP 2023

Optimization of opportunistic utilization of the ATLAS high-level trigger farm for LHC Run 3

Ivan Glushkov[1*], *Chris* Lee[2], *Alessandro* di Girolamo[3], *Rodney* Walker[4] and *Carlo Alberto* Gottardo[3] on behalf of the ATLAS Computing Activity

[1]University of Texas at Arlington, Arlington, USA
[2]Stoney Brook University, New York, USA
[3]CERN, Geneva, Switzerland
[4]LMU, Munich, Germany

Abstract. The ATLAS Trigger and Data Acquisition (TDAQ) High Level Trigger (HLT) computing farm contains 120 000 CPU cores. These resources are critical for the online selection and collection of collision data in the ATLAS experiment during LHC operation. Since 2013, during a longer period of LHC inactivity, these resources are being used for offline event simulation via the "Simulation at Point One" project (Sim@P1). With the recent start of LHC Run 3 and the flat computing budget expected in the near future, finding ways to maximize resource utilization efficiency is of paramount importance. Recent improvements in the ATLAS software stack can potentially allow the utilization of the Sim@P1 even during LHC operation for the duration of the LHC inter-fill gaps. While previous papers on the Sim@P1 project emphasized the technical implementation details, the current contribution is presenting results of a variety of tests that led to the optimal configuration of the job submission infrastructure which would allow the use of Sim@P1 during LHC Run 3.

* Corresponding author: Ivan.Glushkov@cern.ch

Introduction

ATLAS [1] is one of the four experiments at the Large Hadron Collider (LHC) [2] at CERN. It is located at Point 1 on the LHC ring where the High-Level Trigger (HLT) computing farm [3] is installed amongst all other dedicated hardware. It consists of 119 744 x86 computing cores (Table 1) tailored to the needs of the ATLAS trigger system. This system has been used since 2013 for offline ATLAS data processing in periods in which the resources are not being used by HLT within the Simulation at Point 1 (Sim@P1) project, the details of which are described elsewhere [4, 5]. With the approaching of Run 2 and the resource restrictions to be followed by the ATLAS computing, we looked at optimization of the utilization of this resource which would potentially allow us to use it as effectively as possible.

Table 1. The hardware currently available at the HLT farm.

Name	CPU	HT Cores	RAM, GB	VM cores	Nodes
AS-2124BT	2 x AMD EPYC 7302 16	64	125	60	1216
S2600TPR	Xeon E5-2660 v4 @ 2.00GHz	56	62	52	440
S2600KPR	Xeon E5-2650 v4 @ 2.20GHz	48	62	44	360

Sim@P1

Sim@P1 is the largest non-HPC opportunistic compute resource available to ATLAS Distributed Computing (ADC). It has provided 6.7 billion simulated events or 15% of all simulated events in ATLAS in 2022 as shown in Figure 1. It is limited only to Geant4 simulation [6] production due to its hardware configuration limitation, specifically the RAM available per slot, which is dictated by the needs of the HLT system. With the flat budget that ATLAS computing is being faced with in foreseeable future, in this work we are looking into the optimizations needed to run this resource not only – as we do now - during longer LHC stops (long shutdowns, technical stops, machine development breaks) but also during the operation of LHC in the time between fills (Inter Fill Time, IFT) when the HLT farm is not needed. This requires looking at every step of starting the resources, feeding them with the proper workload and stopping them as a potential source of delay or bottleneck for getting maximum output in each IFT.

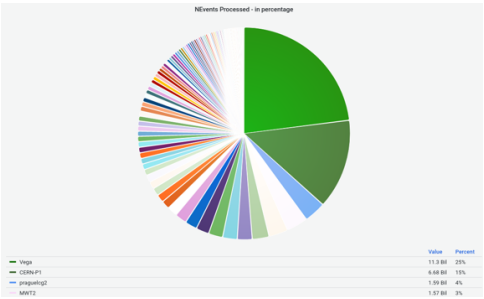


Figure 1. Simulated events in 2022 at ATLAS. Sim@P1 is the second most significant contributor with 15% or 6.7 billion events.

Operational Sequence

The sequence of switching from HLT operation to Sim@P1 processing and producing simulated events is defined by the following steps:

1. HLT shifter decides when the resources can be handed to Sim@P1. This is a decision based on the LHC and ATLAS experiment's immediate operational plans. In the case of planned intervention or technical stop - this is a trivial step. In the case of nominal LHC operation, one cannot predict after the end of an LHC fill, when the next LHC fill will start and the HLT farm resources would need to be switched back. One of the questions that this work is trying to answer is what is the minimal IFT for which useful events might be produced if the resources are being switched to Sim@P1 mode. While the switch is a complicated process it is simplified down to a simple web interface described in details in Ref. [7]. It still requires a manual intervention.
2. After the resources are switched to Sim@P1 mode Puppet is being run on all nodes in order to configure the worker nodes (WN) and add them to an HTCondor pool. While details on the initialization were discussed in a previous CHEP contribution [5], the important feature for this study is that Puppet runs on different machines are run at random times in order not to saturate the network. Additionally, in every one hour, Puppet runs are paused for five minutes to avoid network congestion for other, higher priority services on the Point 1 network.
3. Once the first WNs start appearing in the HTCondor pool, a dedicated instance of the ATLAS pilot submission system Harvester [8] can start submitting jobs. How fast Harvester can submit jobs in the burst operational mode when Sim@P1 comes online depends on the rate at which it gets jobs from the ATLAS workflow management system (WFMS), the rate at which HTCondor can consume them, the hardware resources of the node hosting the service as well as on the configuration values of the service.
4. The workload distribution at ATLAS is handled by PanDA [9]. PanDA should tell Harvester that there is a job to be run at that particular resource. The main delay that can come from PanDA, provided there are matching workloads available in the system would come from the brokerage process.

The brokerage process that serves processing loads to up to 1×10^6 computing cores and optimizes the process based on the changes in the stakeholders of the system. Sudden changes like the appearance of 100k compute cores in the system takes some time to be fully utilized. To overcome this feature, a specific parameter per compute resource has been introduced which allows to define a fixed number of jobs that will be queued no matter if the resource is currently available or not. This feature is useful for fast filling of burst resources like Sim@P1, but it is a problem for workflows which can be stuck queued there for hours in case the resource is not available for a long time.

5. There should be matching workloads available in the system. Due to the limitations of the Sim@P1 resource, not only we can run only simulation, but also not every simulation would be brokered there. The best solution to ensure fast filling of the resource would be to have a dedicated, low priority simulation submitted and pinned to Sim@P1.

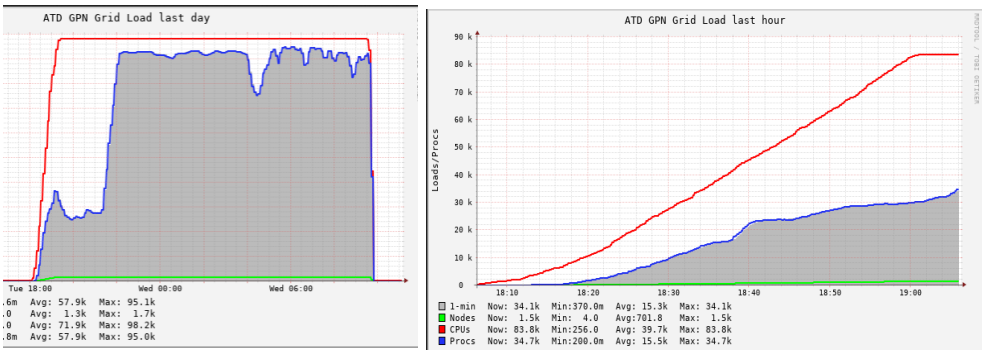
Optimizations

1.1 Conditions

Since the computing resources used in this work are usually needed for the HLT during LHC operation and in the rest of the time they are vital for ATLAS simulation data production, there is rarely time to perform tests and optimization of the system. We have used such an occasion in the period of 6th to 20th in September 2022, during an LHC technical stop and a temporary low of simulation requests from ATLAS when we have managed to perform all tests and optimizations described below. Only 98.1k cores were available for these tests, but the results are easily extendable to the full size of the resource.

1.2 Step 1: WFMS

There are only a limited number of jobs that can be run on Sim@P1 and as it is visible from Figure 2(a), if left to the WFMS alone, the resource could be filled fully – in some cases, depending on the current jobs mix and input data location in the system - only after a couple of hours. To ensure consistency and comparability of all results we needed a large sample of similar standard simulation jobs. Thus, we have used one large standard *pp* interaction FastSim [9] simulated sample. To achieve the same effect in production, we need to ensure a constant flow of low priority FastSim workload available in the system and assigned to Sim@P1.



(a) (b)

Figure 2. Speed-up of filling with jobs from WFMS at resource switching from HLT to Sim@P1. The top solid line represents the number of cores available for the Sim@P1 workload. The filled area represents the number of processes actually running on the resources. a) Filling of resources with time without any optimization. The delay between the two is due to the lack of workload in the WFMS with resource requirements matching the resources available at Sim@P1. b) Filling of resources with time using dedicated workflow and 16 core jobs. The rate at which the resources are being filled with jobs is lower than the rate with which new resources are being added to the system. Note the difference in timescales between a) and b)

1.3 Step 2: Number of cores per job

In order to try to fill the resources as fast as we are getting them and minimize the effect observed on Figure 2(a) without creating additional bottlenecks in the submission infrastructure, a test was performed where each job was run at 44 cores (Figure 3). The maximum number of cores that can be used on every VM at Sim@P1 was set to be 44. As visible from Figure 3, the bottleneck for fast filling of the resource becomes the switching of the resources to Sim@P1 mode which is handled by Puppet.

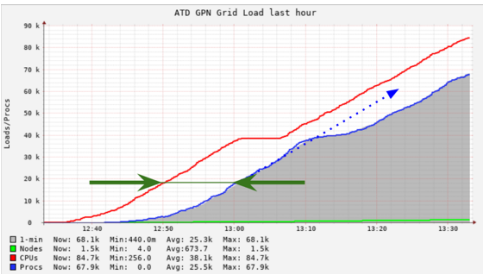
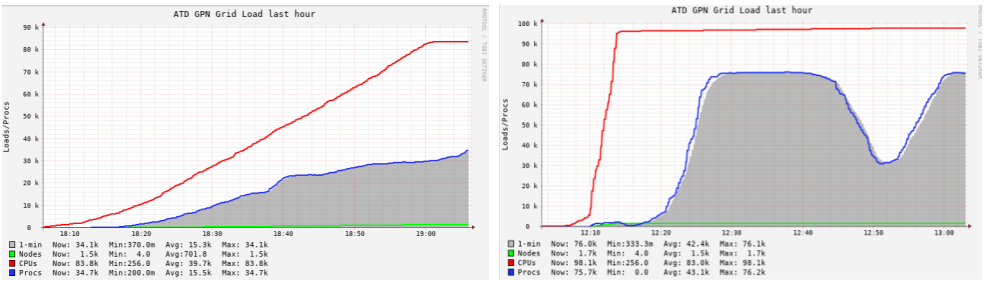


Figure 3. Filling Sim@P1 with 44-core jobs. The top solid line represents the number of cores available for the Sim@P1 workload. The filled area represents the number of processes actually running on the resources. The plateau of the top line comes from the current Puppet configuration discussed in the Operational Sequence section. The dotted arrow indicates the direction of the increase in the number of jobs if the inclination was limited by the job submission infrastructure. The full arrows denote irreducible job initialization time which is approximately 10 minutes.

1.4 Step 3: Puppet

We have tested the influence of the Puppet run on the start-up time of the system and by forcing Puppet to run on VM creation time. While we have managed to get a speed-up of up to 8 times on start-up times, Puppet managed to saturate the network which resulted in disruption of operations of the TDAQ monitoring system and other vital systems at Point 1. Thus, this mode of operation is highly unlikely before achieving a detailed understanding of the effect of other systems.
A side effect that was observed throughout the tests and which is visible on Figure 4(b) is the dip of the filled area. This is because the jobs with which the test was performed were very similar in length and starting a new job in the place of the old one requires the usual initialization time of 10 minutes in which only one core of the multi-core job is being used. This "feature" does not exist in production since there we run a mixture of jobs with random lengths ending and starting at random times.



(a) (b)

Figure 4. Speed-up of the resource switching from HLT to Sim@P1 mode. The top solid line represents the number of cores available for the Sim@P1 workload. The filled area represents the number of processes actually running at Sim@P1. a) Current configuration: A puppet run is triggered at random nodes at random times. The resource switching rate is approximately 1.5k slots/min. b) Proposed configuration: Puppet run is triggered at the moment at which the resources are switched from HLT to Sim@P1 operation. The resource switching rate is approximately 12k slots/min.

1.5 Step 5: Core number variation effects

From pure efficiency of resource utilization, the best results we get with the 8-core jobs – we get 90% of the resources being used (Fig. 5a) and with CPU / wall clock efficiency of 94% (Fig. 5b). Running with such configuration results in jobs with an average duration of 175 min (Fig. 5c) with tails up to 272 min (Fig. 5d). On the other extreme, if we go for jobs with 52 cores, we get very short jobs (34 mins, Fig. 5c) but with filling efficiency of 75% (Fig. 5a) and CPU / wall clock efficiency of 76%. Since in operational conditions, we cannot know in advance what would be the length of the current IFT, the optimal setting would be to start with high core count jobs and as time progresses gradually switch to lower count but higher efficiency jobs.

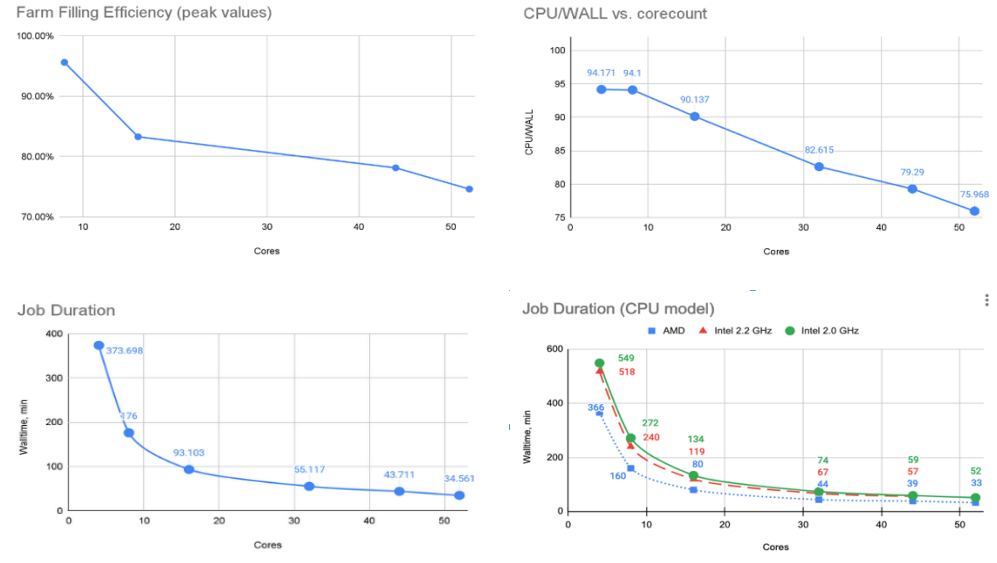


Figure 5. Effect of variation of number of cores per job on a set of jobs. a) Percentage of cores available for Sim@P1 used for workflow processing b) Job efficiency calculated as the ratio between computing time times the number of cores over wall clock time. c) Average wall clock time duration of jobs. d) Wall clock time job duration per CPU model.

Conclusions

In this work, a first set of operational parameters' optimization tests were performed at Sim@P1. A set of parameters is being proposed that would allow the production of simulated events in IFT as short as 1 hour. Higher order optimizations are possible, but they would require further testing. The settings proposed are:

- Dedicated to Sim@P1 low-priority FastSim data samples
- A variety of samples ensures different job lengths
- Constantly queued jobs from the WFMS
- The resource should be filled simultaneously with 8-core and 44-core jobs
- Puppet run should be forced at switching time from HLT to Sim@P1 operation

References

1. ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, JINST **3**, S08003 (2008) https://doi.org/10.1088/1748-0221/3/08/S08003
2. L. Evans, P. Bryant, *LHC Machine*, JINST **3** S08001 (2008) https://doi.org/10.1088/1748-0221/3/08/S08001

3. ATLAS Collaboration, *Technical Design Report for the Phase-I Upgrade of the ATLAS TDAQ System*, (2013) 120-122, https://cds.cern.ch/record/1602235
4. S. Ballestrero et al., *Design and Performance of the Virtualization Platform for Offline computing on the ATLAS TDAQ Farm*, J. Phys.: Conf. Ser. **513** (2014) 032011, http://doi.org/10.1088/1742-6596/513/3/032011
5. F. Berghaus, *ATLAS Sim@P1 upgrades during long shutdown two*, EPJ Web Conf. **245** (2020) 07044, https://doi.org/10.1051/epjconf/202024507044
6. S. Agosyinelli, *GEANT4—a simulation toolkit*, NIM A, **506** (2003) 250-303, https://doi.org/10.1016/S0168-9002(03)01368-8
7. S. Ballestrero et al., *Evolution and experience with the ATLAS Simulation at Point1 Project*, J. Phys.: Conf. Ser. **898** (2017) 082012, https://doi.org/10.1088/1742-6596/898/8/082012
8. T. Maeno et al. on behalf of the ATLAS Collaboration, *Harvester: an edge service harvesting heterogeneous resources at ATLAS*, EPJ, **214**, (2019) 03030 https://doi.org/10.1051/epjconf/201921403030
9. F H Barreiro Megano et al. On behalf of the ATLAS Collaboration, *PanDA for ATLAS distributed computing in the next decade*, J. Phys.: Conf. Ser. **898** (2017) 052002, http://doi.org/10.1088/1742-6596/898/5/052002
10. ATLAS Collaboration, *The ATLAS Simulation Infrastructure*, Eur. Phys. J. C **70** (2010) 823-874, https://doi.org/10.1140/epjc/s10052-010-1429-9
11. ATLAS Collaboration, *AtlFast3: The Next Generation of Fast Simulation in ATLAS*, Comput. Softw. Big Sci. 6 (2022) 1, https://doi.org/10.1007/s41781-021-00079-7