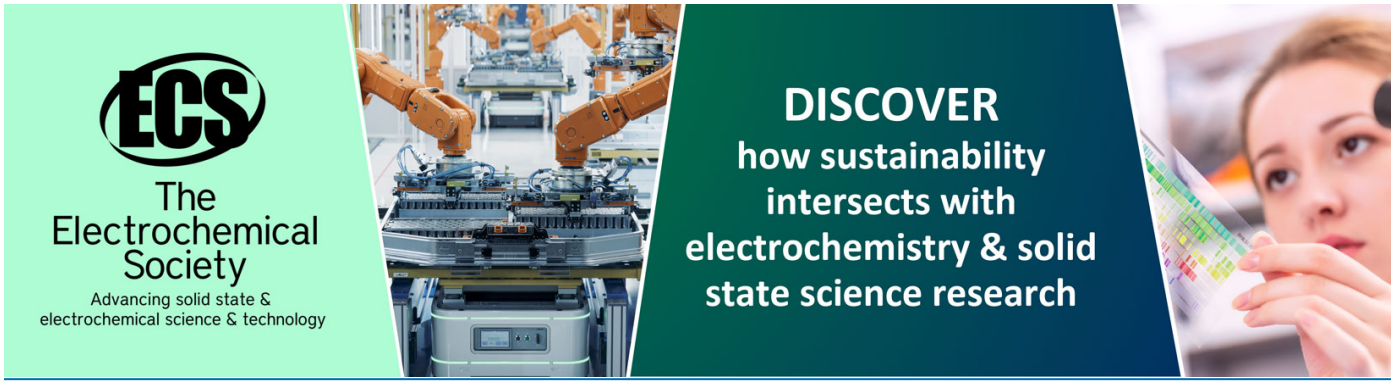
Exploiting the ALICE HLT for PROOF by scheduling of Virtual Machines



Journal of Physics: Conference Series **331** (2011) 072054 doi:10.1088/1742-6596/331/7/072054

Exploiting the ALICE HLT for PROOF by scheduling of Virtual Machines

Marco Meoni[1], Stefan Boettger[2], Pierre Zelnicek[2], Volker Lindenstruth[3] and Udo Kebschull[2]

[1]Ecole Polytechnique Fédérale de Lausanne, Station 1, 1015 Lausanne, Switzerland
[2]Kirchhoff Institute for Physics, Im Neuenheimer Feld 227, 69120 Heidelberg
[3]Frankfurt Institute for Advanced Studies, Ruth-Moufang-Str. 1, 60438 Frankfurt am Main, Germany

E-mail: marco.meoni@epfl.ch, boettger@kip.uni-heidelberg.de

Abstract. The HLT (High-Level Trigger) group of the ALICE experiment at the LHC has prepared a virtual Parallel ROOT Facility (PROOF) enabled cluster (HAF - HLT Analysis Facility) for fast physics analysis, detector calibration and reconstruction of data samples. The HLT-Cluster currently consists of 2860 CPU cores and 175TB of storage. Its purpose is the online filtering of the relevant part of data produced by the particle detector. However, data taking is not running continuously and exploiting unused cluster resources for other applications is highly desirable and improves the usage-cost ratio of the HLT cluster. As such, unused computing resources are dedicated to a PROOF-enabled virtual cluster available to the entire collaboration. This setup is especially aimed at the prototyping phase of analyses that need a high number of development iterations and a short response time, e.g. tuning of analysis cuts, calibration and alignment. HAF machines are enabled and disabled upon user request to start or complete analysis tasks. This is achieved by a virtual machine scheduling framework which dynamically assigns and migrates virtual machines running PROOF workers to unused physical resources. Using this approach we extend the HLT usage scheme to running both online and offline computing, thereby optimizing the resource usage.

1. Motivation

Data taking in the ALICE[1] HLT online system is not running continuously and exploiting unused cluster resources for other applications is highly desirable and improves the usage-cost ratio. For this purpose unused computing resources are dedicated to a PROOF[2]-enabled virtual cluster, thereafter called HLT Analysis Facility (HAF).

PROOF is a toolkit vastly used by physicists at the CERN LHC for prompt analysis and reconstruction of data. It is especially aimed at the prototyping phase of event-based analyses that need a high number of development iterations and a short response time, thus providing an interactive alternative to batch systems. This project aims at providing dynamic allocation of virtual resources for running PROOF in addition to the HLT online system for cluster finding and tracking.

For many reasons not all resources in the HLT are always fully exploited and their usage can unpredictably vary over space and time. First, the HLT online system can be reconfigured at runtime by reallocating processing components to other physical resources. Second, specific ALICE sub-detectors can be excluded from a run and will not deliver data to the HLT. Third, specific phases in the

ALICE experiment life-cycle like calibration and maintenance do not require HLT to process data. Fourth, the data-rate arriving at HLT may vary due to general LHC experiment layout[3].

The HLT cluster runs a dedicated application with specific requirements regarding CPU-load, network and memory usage. These requirements are formalized as policies. A virtual machine (VM) scheduler is the heart of the HAF. The VM-scheduler knows these policies, monitors the cluster environment and thereby infers where free resources exist and where policies are violated. Based on this information it decides whether queued PROOF-VMs can be run on free resources or whether running VMs need to be live migrated. The main objective of the scheduler is to resolve violations as soon as possible and to keep as many VMs running as possible. The decision is calculated using an optimization algorithm for properly choosing among migration, suspending or stopping a given VM.

This document is structured as follows. Section 2 describes architecture and current components of the ALICE HLT computing farm. In Section 3 we sketch the design model of the VM-scheduler and its prototype. Section 4 depicts our experimental environment across 64 nodes and introduces the challenges we face in running a toolkit for I/O intensive interactive data processing in the free slots left by the HLT services. Performance plots are discussed along with functioning principles that can be derived at this early stage. Section 5 outlines the conclusions.

2. The ALICE HLT Computing Farm

The ALICE HLT computing farm is based upon the Linux Operating System and is the basement for a software framework for online analysis of events seen by the ALICE detectors. It consists of two major sections: the infrastructure section and the compute section. The infrastructure section provides the base services needed to operate a cluster, for example gateways, portal server, management & monitoring systems and the shared global storage. The compute section is further divided into three layers. The first layer consists of the front-end processor nodes. These nodes are equipped with the Read-Out Receiver Card (RORC), which receives the data from the detectors via 400 optical Detector Data Links (DDL). The second layer is made of the general compute nodes. Some of these general compute nodes are equipped with GPGPU cards. The data output nodes build the last layer.

All nodes are connected to two Gigabit Ethernet networks, one for remote access, monitoring and administration, the second for operation and control. The operation and control network is used for the shared filesystem and the data analysis framework control channels. The data transport for the analysis framework is done on a dedicated data transport network exclusive to the compute section. Table 1 reflects the installed equipment within the compute section as for 1st November 2010.

Table 1. Compute section of the HLT Farm and installed equipment.

Nodes	224
Compute	2860 cores and 16320 stream cpus (34 gpgpu)
Storage	64 TB of global shared storage and 111,52 TB of local disk storage
Network	• 1 Infiniband Network with 40 GBit/s as data transport network • 1 Gigabit Ethernet with 1 GBit/s as operation and control network • 1 Gigabit Ethernet with 1 GBit/s as monitoring and administration network

3. A VM-Scheduler

Several questions arise aiming to exploit free resources for third party applications like PROOF.

- What are usable resources and how can we identify them?
- How can we avoid interference with the HLT online system?
- How are third party applications assigned to these free resources?
- How can this assignment be adapted on the fly when resource availability changes?

First of all, the requirements of the HLT online system and its constituting components have to be defined. We set up service-level objectives[4] for each particular component and derive policies, which specify minimal, maximal or average resource consumption along several dimensions like number of cores, amount of memory, network bandwidth and CPU-load. Available resources can be determined by combining the knowledge about the allocation of components to physical cluster nodes and their respective policies. Policy compliance and policy violation are recognized by monitoring the cluster[5].

Third party applications may interfere with the HLT online system. We minimize the probability by using virtual machines that encapsulate functionality and avoid harmful interaction by running malicious code. Policies are defined to precisely state what are available resources and to which extent applications can use them. Despite of having defined resource availability and provisioning third party applications to such resources only, policy violations may occur. This can be due to component reallocation or any other unexpected change in resource consumption by the HLT or third party applications. Therefore, policies are equipped with a time-limit property that determines the maximum amount of time a policy violation can be sustained by a component. The scheduler for third party applications makes sure that the policy violation is resolved within the specified time frame.

They choice of virtual machines for running third party applications is obvious: Arbitrary applications can be hosted regardless of the operating environment they need. They possess features like suspend/resume and live-migration. If a policy violation occurs, a VM can be stopped and restarted at a later stage, or suspended and resumed preserving already computed results, which is a better option for achieving an improved result/time ratio in the cluster. Live-migration adds a further possibility for efficient reacting to policy violations.

We use the lease abstraction[6] for describing a number of VMs hosting a distributed application (such as PROOF) and its specific properties. A user requests a lease and the life cycle of a lease and its associated VMs is controlled by a scheduler. Once the scheduler accepts a lease-request, a priority is calculated and the lease is put in a priority queue. The scheduler knows the allocation of HLT components in the cluster, their policies and the monitoring data about the actual cluster environment. The queue is processed using a first-fill priority scheduling approach based on the run-to-competion paradigm.

If policy violations occur, a more aggressive approach has to be taken. The processing of the lease queue is stopped and all VMs/leases contributing to policy violations are collected. Possible reactions are the live-migration of a single VM or the suspending or even stopping of their leases. As an initial solution, stopping/killing of all concerned leases is chosen. However this is not desirable and therefore a search upon the solution space using a hill climbing approach is done. First, the solution space is minimized by only taking into account a) operations which resolve a number of violations rather than only a single violation and b) operations which do not exceed the time-limit of the violated policies. The solution space is traversed trying to keep higher priority leases running on expense of suspending/stopping lower prioritized leases. The search runs until this formula holds: *time(searchSolutions) + time(applyBestSolution) = min(timelimit_violatedPolicies)* . Then the best-found solution is applied. Further details on the VM-Scheduler can be found here[8].

For the PROOF use-case some additional remarks are helpful. Since PROOF is an interactive application, the suspending of VMs or whole leases is not an option. Futhermore, PROOF can deal with changing of worker availability. Therefore we choose a lease-size of one VM, mark the lease as interactive (thereby telling the scheduler not to suspend it) and submit a number (e.g. 64) of lease requests to the scheduling framework. If a VM violates a policy and is eligible for migration, it will be stopped and put back on the priority queue if the migration is not feasible. A PROOF session will lose results from that VM, whereas any subsequent session will work with the remaining running VMs/leases.

The VM-scheduling framework has two benefits in the HAF. PROOF processing is enabled in concurrency with the HLT online system and PROOF sessions are kept running using live-migrations even if resource availability in HLT for third party applications changes. A prototypical scenario is

represented in figure 1. Three physical nodes are shown and a policy concerning the maximum allowed CPU-load for each node is indicated. The scheduler knows five PROOF leases consisting of one VM each, two of them are running and three are waiting in the queue. A policy violation on node2 occurs and the scheduler checks whether live-migration is feasible or VM2 has to be stopped. The evaluation shows that a migration to node3 will not violate policies on node3 and will resolve the policy violation within the time-limit of the violated policy on node2. After that, the processing of the queue continues and available resources on node1 are detected. VM3 is found to be suitable and run on node1. The current PROOF sessions will remain processing with VM1 and VM2 and a new one will now work with VM1, VM2 and VM3.

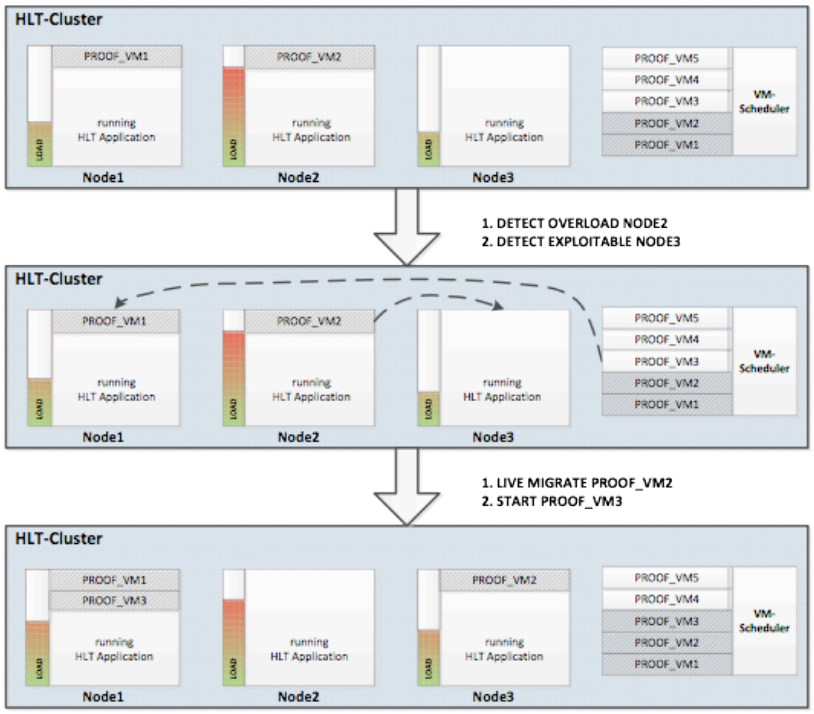


Figure 1. In this example a policy violation is recognized for node2 and node1 is under-utilized. The VM-scheduler decides to start the queued PROOF_VM3 on node1 and to live-migrate PROOF_VM2 from node2 to node3.

4. The HAF: HLT Analysis Facility

We begin by describing our experimental environment because the target of this work lies in evaluating the utilization of the VM-scheduler at the HLT computing farm with the additional load generated by PROOF. We were granted access to HLT production machines and created lightweight PROOF VMs. The PROOF workers are deployed to 64 VMs with 2 cores and 2GB of memory each, whereas the PROOF master runs on a 4-core machine equipped with 8GB of memory.

What makes PROOF a system very well suited for testing the VM-scheduler is a) it leverages a master/slaves architecture with many nodes collaborating to achieve a common task b) it generates bursts of CPU consumption along with intensive network and I/O traffic c) it allocates potentially large memory structures, these depending on the type of user analysis and the objects that are filled. As such, PROOF acts as a perfect payload for the VM-scheduler. In fact, it produces heavy load and allows for stressing of every parameter adopted by the policies. Moreover, should a policy violation occur, prompt reactions and fast migrations are needed to avoid reduction of the event processing rate.

Since the execution of the HLT software components must not be affected by concurrent PROOF sessions, PROOF VMs are the first entity to be moved around the cluster if possible. This live migration should be as fast and transparent as possible to preserve PROOF load balancing of the workload according to the computing node performances and to optimize the efficient use of the resources.

In order to reduce the impact of the VM migration, PROOF machines are kept very small in size, with just 1GB images containing the O.S., the services handling the communication protocol and adequate space for local sandboxes. The HLT Gigabit Ethernet network infrastructure provides suitable bandwidth to move VMs around the cluster and enable workers dynamically.

The majority of the software involved in a PROOF-based Analysis Facility, such as the ROOT framework on top of which PROOF is built and the AliEn Grid middleware for package synchronization, is shared among the nodes and installed on an AFS volume. Similarly, event-data is staged from the Grid to a high-speed dedicated storage mounted as NFS volume. The latter choice prevents us from exploiting data locality. We have lightweight VMs where each node does not process local data but rather accesses a single dedicated storage concurrently. Thus we trade local I/O rate for network bandwidth and reduce the average processing rate by a factor of ten. Nevertheless, lightweight VMs, remote software distribution and shared data storage embody a scenario that is completely new for a PROOF Analysis Facility and allows for easy cloning and allocation of working nodes.

Figure 2 displays the first target of the paper, namely the proof of the principle of using PROOF within the HLT virtual environment. Tests are performed running physics analysis tasks processing 160 Mio events distributed in 16k files and plotting momentum distributions in several shapes and dimensions. Input files are stored in ROOT format and each one is approximately 50MB in size. Examples of the PROOF I/O rate are shown in two different long-running scenarios.

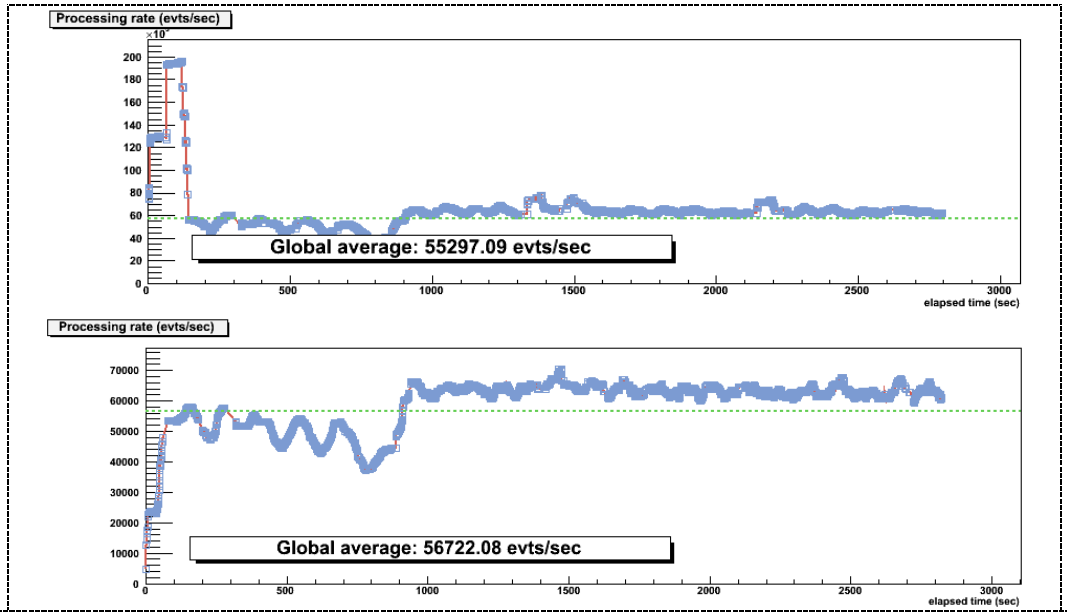


Figure 2. Examples of PROOF analysis tasks at the HAF. The first plot shows a burst of processed events when the cluster is only used by PROOF. Then the performance drops because the network is loaded by concurrent processes that swap files from a common data storage. The second plot depicts the simulation of the processing rate that gradually increases when the number of external processes is reduced.

The two simulations aim to understand the main source of bottlenecks that can affect our early HAF setup. Since the HLT framework does not produce significant hard-disk I/O and data is stored on

a shared volume, network represents a potential reason for PROOF performance drop off. In case of overload, the network policy can mark a migration as a bad choice for the VM-scheduler. This also means, except for network criterion, there is no bonus if external processes do vanish or stop.

The HAF is currently monitored using a MonALISA[7] Web repository. It displays real-time plots and running histories about active sessions, number of events processed, CPU-load at the different nodes, used disk space and network traffic. These statistics are shown at http://proofmaster:8080, which is accessible from inside the HLT network.

5. Conclusions and Outlook

We have extended the ALICE HLT farm usage scheme for running both online and offline computing. For this purpose, the PROOF toolkit has been enabled in the cluster. Soon it will run along with the VM-based framework offering the opportunity to exploit a large number of CPUs for interactive distributed parallel data processing and increase their usage-cost ratio.

Utilization constraints enforced us to design and test a novel schema of PROOF Analysis Facility. To name a few, we must not interfere at all with the running HLT components and we do need to limit the size of the PROOF VMs for an efficient live migration whenever it occurs. The results achieved so far look promising and show room for further developments. The measure of performance loss when running PROOF in lightweight VMs rather than natively will lead to explore the future feasibility of using cloud-based Analysis Facility setups.

The exercise has enabled a fully operational virtual PROOF cluster that can be opened to the entire ALICE collaboration providing exploitable resources at any time HLT data taking is not running at full speed. The additional load introduced by PROOF gives the opportunity to refine the scheduler policies that are applied for limiting the competition between the two applications. The watermark for best lightweight machines should be further investigated in order to measure whether the trade of all-shared software for a minimal VM size is the way to go. On the other side, single data storage represents a major limitation as it prevents from exploiting data locality and cache effects, although shared storage is the most reasonable solution with a VM setup. Other filesystems can be tested in the future and multiple (disjunct) servers are possible. This could help increasing performance. Due to the chosen setup, test PROOF tasks show that network bandwidth can be a major reason for performance bottlenecks. Hence, VM-remapping must be carefully evaluated. Nevertheless, real analysis tasks can produce large memory structure and also trigger live-migration upon memory criterion at a frequency that might not be sustainable.

Finally, upcoming new PROOF features can increase the flexibility of the VM-scheduler too. For example, dynamic run-time node allocation may introduce the possibility to perform on-the-fly VM suspending/resuming, which are now limited when the HAF nodes are not running PROOF sessions.

References

[1] ALICE Technical Proposal for a Large Ion Collider Experiment at the CERN LHC. CERN/LHCC/95-71, 15 December 1995
[2] Brun, R., Rademakers, F.: ROOT - An Object Oriented Data Analysis Framework. Nucl. Instr. And Meth. A 502 339-346 (2003)
[3] The Lage Hadron Collider Nature Vol 448 Issue No. 7151 / 19 July (2007)
[4] Rick Sturm, Wayne Morris, and Mary Jander. Foundations of Service Level Management. SAMS Publishing, Apr 2000.
[5] Camilo Lara. The SysMES Architecture: System Management for Networked Embedded Systems and Clusters, DATE07 conference, PhD Forum, Nice, France, 2007
[6] D. Irwin, J. Chase, L. Grit, A. Yumerefendi, D. Becker, and K. G. Yocum, "Sharing networked resources with brokered leases," Proc. USENIX Technical Conference, Boston, USA, (2006)
[7] MonALISA: http://monalisa.cacr.caltech.edu
[8] Stefan Boettger, Camilo Lara, Timo Breitner, Udo Kebschull and Volker Lindenstruth. Virtual Machine Scheduling for Special Purpose Clusters, Proc. SNA+MC2010, Tokyo, Japan, 2010