The PROOF benchmark suite measuring PROOF performance

S Ryu[1] **and G Ganis**[2]

[1] KISTI, 245 Daehak-ro Yuseong-gu, Daejeon 305-806, Korea
[2] CERN, Geneva, Switzerland

E-mail: sangsuryu@kisti.re.kr

Abstract. The PROOF benchmark suite is a new utility suite of PROOF to measure performance and scalability. The primary goal of the benchmark suite is to determine optimal configuration parameters for a set of machines to be used as PROOF cluster. The suite measures the performance of the cluster for a set of standard tasks as a function of the number of effective processes. Cluster administrators can use the suite to measure the performance of the cluster and find optimal configuration parameters. PROOF developers can also utilize the suite to help them measure, identify problems and improve their software. In this paper, the new tool is explained in detail and use cases are presented to illustrate the new tool.

1. Introduction

The Parallel ROOT Facility, PROOF, is an extension of the ROOT system [1] aimed at speeding-up analysis using multiple ROOT processes in parallel on a cluster of computers or a multi-core computer [2, 3].

PROOF implements multi-process parallelism to address specifically typical analysis problems encountered in HEP, which in most cases are either embarrassingly parallel or can be formulated as such. For systems like PROOF, scalability with respect to available resources is a natural metric of the efficiency of architecture design and implementation. The inherent scalability of a well-designed and efficient parallel system should be close-to-linear, with minimum overhead. When relevant, external factors, like the available I/O bandwidth, may change this picture [4]. Spotting, identifying and understanding departures of system scalability from linearity are important steps in system optimization process, and essential for any further improvement.

The new PROOF benchmark suite is a new module built into PROOF, providing a standard way to measure the scalability of PROOF system as a function of active processes in the cluster. It is a simple, easy to use, yet flexible enough tool with full support for user-specific customization.

2. PROOF benchmark suite

The new benchmark suite is a framework to perform scalability measurements on a PROOF cluster in a standard way. The primary goal of the new benchmark suite is to determine optimal configuration parameters for a set of machines to be used as PROOF cluster. The suite measures the performance of the cluster for a set of standard tasks as a function of the number of effective processes. From these results, indications about the optimal working point of the cluster – e.g. the maximum or optimal number of concurrent processes - should be derived. For large facilities, where a multi-tier architecture with a super-master supervising multiple sub-masters may help improve scalability and merging

performance, the suite should also give indications about the optimal number of sub-masters into which the cluster should be partitioned.

The new suite should be beneficial to PROOF site administrators who wish to check their installation, find bottlenecks, and optimize configuration parameters. PROOF developers will find the suite useful in understanding and improving PROOF.

2.1. Design requirements

The new benchmark has been designed to be easy to use and flexible. This means that defaults are meaningful for most typical cases and straightforward to run, with only few or no settings from the user. At the same time the suite allows users to benchmark user-specific or experiment-specific cases, requiring dedicated datasets, selectors and software packages.

2.2. Analysis modes with PROOF

Typical analysis tasks in HEP application can be categorized into two types – cycle-driven analysis and data-driven analysis. Cycle-driven analysis work is typically CPU-intensive but it could also be I/O, network, or RAM intensive; generation of Monte Carlo events is a good example. For data-driven analysis, the unit of process is entries of a TTree fetched from distributed files. The analysis task reads in events from real data files from experiment or data files generated from simulation. This type of analysis task is typically disk I/O intensive but it could also be network, RAM, or CPU intensive. The new suite addresses both types of analysis tasks.

2.3. Key components

The suite consists of a set of client-side classes, of which essential classes are a steering class for user interface and a set of selectors for default tasks.

2.3.1. Steering class. TProofBench is a user interface class for the benchmark of the system. With the interface, the user makes a connection to PROOF cluster, prepares an output file for results of the test, uploads relevant selectors as PAR files [2], runs benchmark, and displays results. For typical use, this is the only class relevant to the user of the suite.

2.3.2. Selectors. The suite provides one default selector for each of analysis task types. TSelHist is a default selector class to be used for cycle-driven task and class TSelEvent for data-driven analysis task. These classes are uploaded to the cluster as PAR files at the beginning stage of benchmarking. The selector TSelHist intensively generates random numbers following normal distribution to fill 1-D, 2-D, or 3-D histograms, which will be merged at PROOF master and returned to client. 3-D histogram can be used to study the impact of merging large outputs on the scalability of the system. The selector TSelEvent reads in test events from files, which have been generated through TProofBench user interface class prior to benchmark, and fills in histogram with information from events read.

2.4. Modes of scan

To obtain the scalability of the system, the suite repeatedly performs measurements while enabling a certain number of additional workers at each step. In worker-scan mode, with default parameters, scan starts with one active worker, enabling another worker at each step until all available workers in the cluster are active. In core scan mode, with default parameters, measurement starts with one active worker on every node in the cluster, enabling additional worker on every node simultaneously at each step, until all available workers on nodes are active. The order workers are activated is determined by system configuration on master node. User can change how many active workers to start with, how many workers to activate each step, and the number of active workers to finish scanning at. With worker scan mode, overall behaviour of the system can be investigated, while core-scan mode is more effective for the investigation of the behaviour of the system inside a node. To minimize the statistical fluctuation of measurement and maximize measurement accuracy, measurements are repeated (four

times, by default) for each step of the scan. For data-driven mode, file caches on worker nodes are cleaned after each measurement.

2.5. Generation of data files

For default data-driven benchmark, files with events should be generated on worker nodes, which can be done with user interface class TProofBench. The events are of type Event of ROOT[3]. Default is to generate two files with 30,000 events each for every worker in the system. All relevant parameters are configurable by user.

2.6. Output of benchmark

For cycle-driven test, average event rate (number of events processed on all active workers in unit time) and its RMS are calculated at each measurement point. For data-driven test, average I/O rate (MB read in on all active workers in unit time) and its RMS are calculated for each measurement point as well as average event rate and its spread. Event rate and I/O rate are plotted on display as a function of the number of active workers in the cluster and updated as progress is made, as well as saved to output files. Normalized event rate and normalized I/O rate (event rate and I/O rate divided by the number of active workers in the cluster, respectively) are also displayed to supplement the interpretation of system behaviour. Packet information from each active worker in the cluster such as CPU time, process time, and latency is saved to an output file for possible further investigation after the benchmark has finished.

2.7. Availability and documentation

The new benchmark suite is available from ROOT v5.29 and on. As it is a client-side module, it can be imported into previous ROOT versions. Additional documentation is available online at http://root.cern.ch/drupal/content/new-benchmark-framework-tproofbench.

3. Use cases

To illustrate the tool, some example results obtained from runs on PROOF facilities for ALICE experiment [5] and on a cloud facility [6] are shown. PROOF facilities that were used are summarized in table 1. All benchmark runs were performed with default parameters, if not stated otherwise.

Table 1. PROOF facilities used for use case study.

	ALICE CAF	KIAF	PoD on Frankfurt cloud
Nodes	1 master, 58 workers	1 master, 4 worker	50
CPU	2 / node, 4 cores/CPU (Intel Xeon L5520, 2.27 GHz)	2 / node, 6 cores/CPU (Intel Xeon X5650, 2.67 GHz)	2 / node, 12 cores / CPU (AMD Opteron 6172, 2.1 GHz)
RAM	24 GB / node	24 GB / node	64 GB / node
Storage	Local SATA disks	5 TB/node (NAS) 300 GB/node (SAS)	-

3.1. Cycle-driven benchmark on a homogeneous cluster

Figure 1 shows an example result of cycle-driven benchmark with default configuration on KIAF which is a homogeneous ALICE PROOF cluster. Total event rate divided by the number of active workers in the cluster as shown in figure 2 provides closer view on the scaling of the system.

[3] See the files test/Event.h and test/Event.C in any ROOT installation.

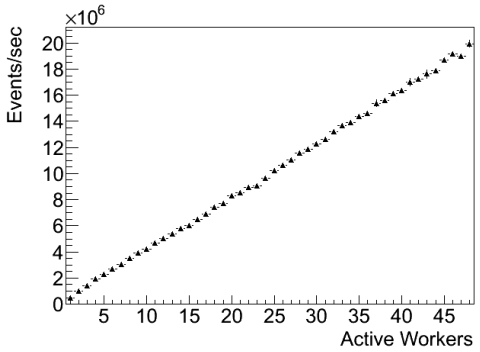

Figure 1. Event rate from a cycle-driven benchmark on KIAF.

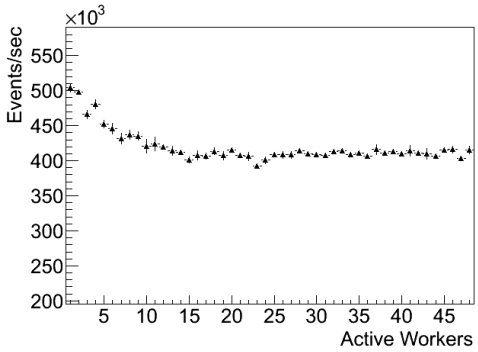

Figure 2. Events rate normalized by number of active workers from a cycle-driven benchmark on KIAF.

3.2. Cycle-driven benchmark on ALICE PROOF facility

Cycle-driven benchmark results on ALICE PROOF facility CAF [5] are shown in figure 3 and figure 4. The facility is currently actively utilized for analysis tasks by ALICE collaboration. The facility consists of 3 groups of computers with different performance, formed over time by adding new computers to existing cluster. The plots clearly show transition between the groups.

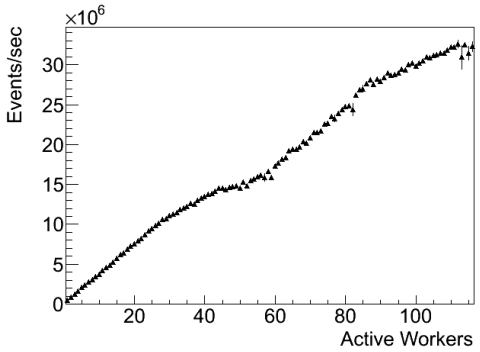

Figure 3. Event rate from a cycle-driven benchmark on ALICE CAF. Transition from one type of computers to another is clearly shown.

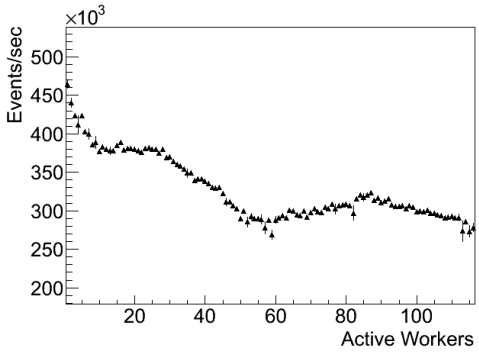

Figure 4. Normalized event rate from a cycle-driven benchmark on ALICE CAF.

3.3. Cycle-driven benchmark on PoD PROOF cluster

Figure 5 shows a cycle-driven benchmark result on PoD [7] cluster where PROOF cluster is dynamically set up on Frankfurt cloud facility [6]. We can see here that the system scales almost linearly up to around 300 workers, where the impact of the serial implementation of packet distribution by master becomes visible.

3.4. Data-driven benchmark of two storage systems on ALICE KIAF

Figure 6 shows results from a data-driven benchmark with test events on ALICE KIAF cluster [5]. KIAF had 2 storage systems available on the system – local SAS disks on every node and a NAS storage system mounted to all worker nodes via network. At 4~5 active workers per node, rate starts to saturate to the value representing the total amount of I/O that the device can provide [4]. The new benchmark suite is very effective in benchmarking and comparing performance of different hardware with PROOF.

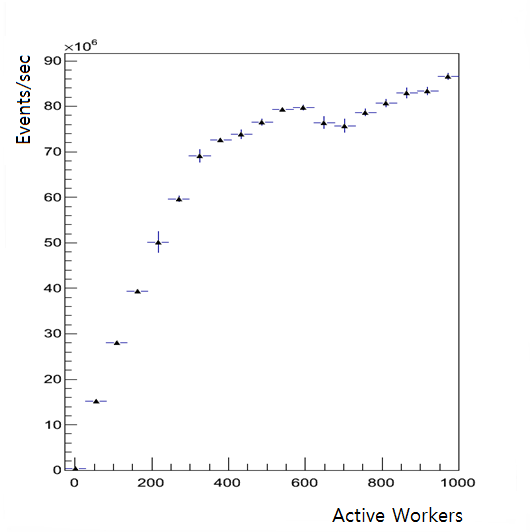

Figure 5. Event rate from a cycle-driven test on POD cluster on cloud system. Measurements were made for every 50 workers. Courtesy of A Manafov at GSI, Darmstadt.

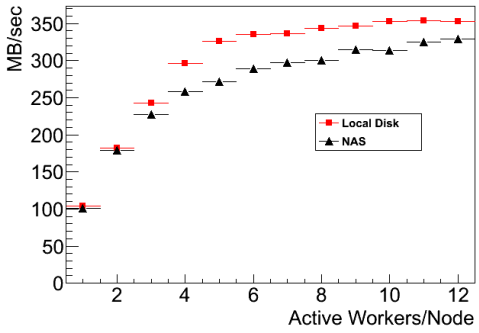

Figure 6. I/O rate from a data-driven benchmark on KIAF cluster. This is core scan result. Square points are with local SAS disks on every worker node, triangular points with a NAS system mounted to every worker node via network.

4. Conclusions

PROOF benchmark suite is a new module in PROOF system, which is easy to use, yet flexible enough to support user-specific or experiment-specific requirements. The new suite will help site administrator check their installation and optimize its configuration parameters. PROOF software developers can use the suite to spot, identify problems, and improve their software. The suite has been available from ROOT version 5.29/02 and on.

Acknowledgements

The authors gratefully acknowledge KIAF administrators at KISTI, Daejeon for PROOF facility and their support, A. Manafov at GSI, Darmstadt for the benchmark plots for PoD cluster on cloud system, and ALICE collaboration for ALICE PROOF facility.

References

[1] Antcheva I *et al*, ROOT – A C++ framework for petabyte data storage, statistical analysis and visualization *Comp. Phys. Comm.* **180**/12 (2009) 2499-2512. See also http://root.cern.ch

[2] Ballintijn M, Brun R, Rademakers F and Roland G 2003 The PROOF distributed parallel analysis framework based on ROOT *Proceedings of CHEP03 Intl. Conf. (La Jolla, California, US, March 2003*

[3] Ballintijn M, Brun R, Canal P, Gulbrandsen K, Roland G and Rademakers F 2005 Super scaling PROOF to very large clusters *Proceedings of CHEP04 Intl. Conf. (Interlaken, Switzerland , 27 September – 01 October 2004)*

[4] Aguado-Sanchez C, Blomer J, Buncic P, Charalampidis I, Ganis G, Nabozny M and Fademarkers F 2010, Studying ROOT I/O performance with PROOF-Lite *Proceedings of CHEP 2010 Intl. Conf. (Taipei, Taiwan , 11-22 October 2010)*

[5] ALICE analysis facilities http://aaf.cern.ch

[6] http://www.frankfurt-cloud.com/

[7] Malzacher P and Manafov A 2010 PROOF on Demand *J. Phys.: Conf. Ser.* **219** 072009