

# Simulating the Farm Production System Using the MONARC Simulation Tool

Y. Wu<sup>1</sup>, I. C. Legrand<sup>2</sup>, I. Gaines<sup>1</sup> and V. O'Dell<sup>1</sup> for the CMS collaboration

1. MS 234, Fermilab, PO Box 500, Batavia, IL 60510, USA

2. C.C. Lauritsen Laboratory of High Energy Physics, Caltech, Pasadena, CA 91125, USA

## Abstract

The simulation program developed by the "Models Of Networked Analysis at Regional Centers" (MONARC) project is a powerful and flexible tool for simulating the behavior of large scale distributed computing systems. In this study, we further validate this simulation tool in a large-scale distributed farm computing system. We also report the usage of this simulation tool to identify the bottlenecks and limitations of our farm system.

Keywords: MONARC, simulation, distributed computing, CMS production

## 1. Introduction

The "Models Of Networked Analysis at Regional Centers" (MONARC) project [1] developed a powerful and flexible tool for simulating the behavior of large scale distributed computing systems. The simulation tool, which is based on Java technology, is well written to describe the concurrent running processes. The CMS (Compact Muon Solenoid) experiment, which will run at the Large Hadron Collider (LHC), requires very large scale distributed computing systems to analyze the expected petabytes of data. It is important to model these systems to optimize the proposed architecture.

The validation of the MONARC simulation tool has been continuing in the past several years. It was first verified by several institutes in the Testbeds Working Group of the MONARC project. However, these tests were mainly done at several specially designed small-scale test beds [2, 3]. The only known large-scale test was done at CERN, Switzerland [4]. They simulated a PC farm system with 70 dual-CPU processing nodes and 35 database servers. They reproduced the measured results quite well using the simulation tool. Further validation is needed in a variety of large-scale distributed computing systems. We believe this is an essential step in the evaluation and development of the computer architectures for future CMS production runs, as part of a continuous cycle of modeling the system, test and measurement, simulating the system, and validation of the simulation.

This paper reports further validation of the simulation tool in a large-scale farm system. We also report our usage of the MONARC simulation system to recognize bottlenecks in our farm system and to provide information on how to improve the performance of our farm system.

## 2. Design considerations

The details of the MONARC tool and its design considerations have been described by Legrand and Newman [5]. Here we briefly describe some of the main features of the tool. The simulation tool was designed for the study of very large distributed computing systems. It utilizes object-oriented design to map the logical components into the simulation program and offers a flexible and extensible

solution for modeling large-scale systems. A process-oriented approach for discrete event simulation using threaded objects offers the flexibility in simulating the complex behavior of large scale computing systems.

The MONARC simulation program adopts the multi-thread feature of the Java™ technology and uses it to provide a dedicated scheduling mechanism for the simulation program. Shared resources, such as CPU and I/O links, are implemented as normal objects in the simulation program. The threaded object, also called “Active Object” (having an execution thread, program counter, stack, mutual exclusion mechanism...), is the basic class that must be inherited by all the entities in the simulation that require a time dependent behavior. Objects that extend this basic class may implement any specific time-dependent behavior to simulate the real processes in the distributed computing system. The interrupt mechanism for the “Active Objects” offers an effective way to simulate discrete event processes assuming a “continuous” flow in time between events that modify parts of the system. Together with an interactive GUI, this program provides a powerful and flexible tool for simulating the behavior of large scale distributed computing systems and is well suited to be used to evaluate and design our farm system.

### 3. Simulating the farm production system

Simulation of CMS events is a complex process involving both signal and background (pileup) events with both input and output using an Objectivity database. The farm system we studied consists of one I/O node to submit jobs, one 4-CPU 500MHz Pentium machine as the signal database server, and nine dual-CPU 750 MHz Pentium machines used as pileup database servers. We also have 29 dual-CPU 750MHz machines as working nodes that run the reconstruction software to process the events. All the machines are linked by 100Mb/s Ethernet. Redhat Linux 6.1 is the standard OS on these computers and Objy/DB v5.2.1 is used as the ODBMS. The activities of each farm node (e.g., CPU, memory, disk, network traffic, etc.) are recorded in a log file at 90 second intervals.

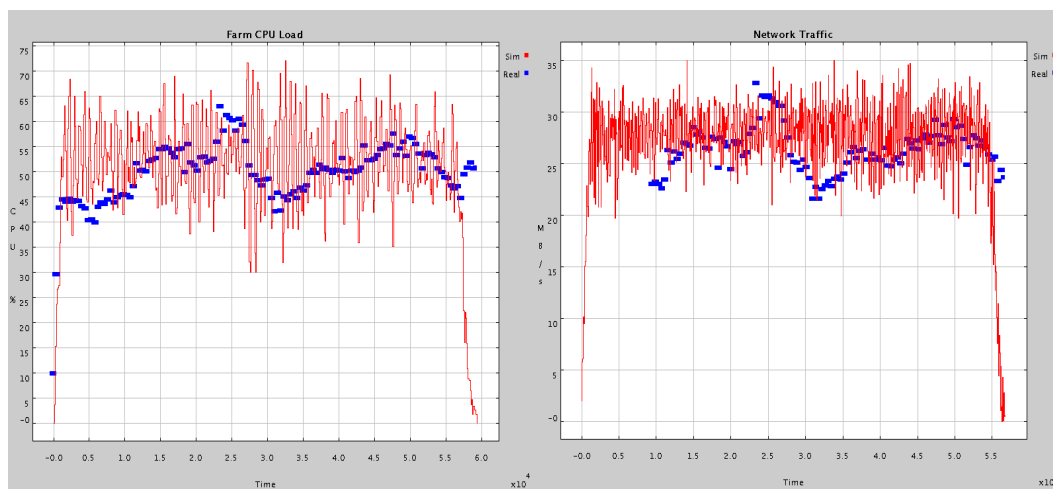


Figure 1. The measured and simulated CPU usage (left) and network traffic (right) on Fermilab farm system. Blue squared symbol: measured results; red lines: simulated results.

The running program on the Fermilab farm we studied was the digitisation with pileup events, which is the final step of the ORCA (Object Oriented Reconstruction for CMS Analysis) process. It is an I/O intensive program, reading out 1 signal event from the signal database server and randomly reading out 25 MB of minimum bias events from the 9 pileup database servers. 500 events were combined into one batch job. Each batch job was submitted with 21-second intervals using the farm batch system developed at Fermilab [6].

The real job parameters, CPU time per event and input and output event sizes, together with the hardware configuration of our farm system were used as input parameters for the simulation program. The simulation program was also modified to describe the detailed behavior of the production process reading in the signal events and pileup events.

By using these parameters, the simulation program reproduces the CPU utilization, job efficiency and network traffic quite well. Figure 1 shows the measured and simulated CPU usage and network traffic. The agreement indicates the simulation tool is sufficiently powerful to accurately model the overall performance of multiple jobs distributed over a large-scale farm system. Similar behavior between the simulated network traffic and measured results is another indication that the farm production system is well modeled and simulated.

#### 4. Discussion

From Figure 1, we can see that the CPU usage in our farm system is quite low, only around 50%. This is due to the nature of the running digitisation job and the limits of the farm system. The digitisation job is very I/O bound. It needs 25MB of minimum bias events to process one signal event. So, if we assume that it needs 30 seconds of CPU time to process one signal event and the system has a 50% CPU efficiency, it will take about 1 minute to process a signal event. To process each signal event, it reads out 1 MB of signal data and writes out 1 MB of output, and it also needs ~25 MB of pileup events. So, this means that each program needs 27 MB of I/O activity per minute. Since we have 58 processes running in parallel, the mean rate on our farm is around 26 MB/s. To get an overall 90% CPU usage on our farm system, we would need to have a much higher data rate, around 47MB/s. As our farm system cannot provide such a high data rate, the CPU usage is quite low.

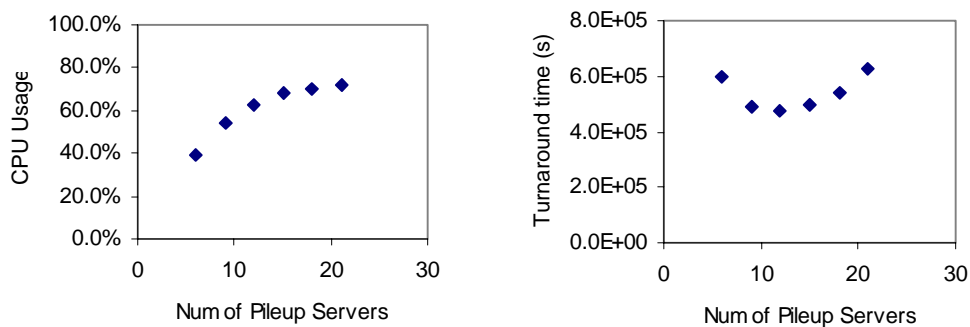


Figure 2. The variation of the CPU usage and 1000-job turnaround time as a function of the number of pileup servers.

We can use the MONARC simulation program to effectively arrange our farm configuration. That is, we can predict how to achieve the maximum farm performance under the current farm setup by

simulating different numbers of farm nodes as pileup servers. Figure 2 shows the CPU usage and 1000-job turnaround time as a function of the number of pileup servers under the current farm setup. From this figure, we can see that there is a minimum turnaround time when there are 12 pileup servers. This means we can achieve maximum job throughput with 12 farm nodes as pileup DB servers. It is probably the optimum setting for our current farm system, which will have a CPU efficiency of 62.5%. Similarly, the simulation tool can be used to determine what hardware investments will provide the largest increments in farm system performance.

## 5. Conclusions

The MONARC project has developed a powerful simulation tool to evaluate the performance of distributed computing systems. This study further verifies the use of this simulation tool in a large-scale farm system. The MONARC simulation program correctly describes the overall performance and limitations of our farm system. This study is a clear indication that the simulation tool can be used to simulate multiple jobs on a large-scale distributed computing system.

By monitoring and modeling the production farm system, we are able to identify the bottlenecks and limitations of our farm system and propose new optimized farm configurations to obtain better performance. Modification of the parameters in the MONARC simulation tool can help us identify the sensitivity to the model details. This study is quite useful for us to understand the current farm system. This study also shows that the MONARC simulation tool, if properly used, can not only be used to simulate the computing farm system, but also provide very useful information to future designs of large scale distributed farm systems.

## Acknowledgement

The authors want to thank the help of the CMS production team at Fermilab, especially G. Graham, H. Wenzel and S. Aziz. We would also like to thank D. Stickland and T. Wildish at CERN for the help in providing the production parameters.

## References

1. The MONARC Project: <http://www.cern.ch/MONARC/>.
2. Y. Morita, Validation of the MONARC Simulation Tools, CHEP2000, <http://chep2000.pd.infn.it>.
3. A. Brunengo et al., "WAN Test-bed with Objectivity 5.2 in a multi-server configuration", CHEP2000, <http://chep2000.pd.infn.it>.
4. H.B. Newman and I.C. Legrand, "Simulating Distributed Systems", ACAT2000, <http://conferences.fnal.gov/acat2000>.
5. I.C. Legrand and H. B. Newman, "The Monarc Toolset for Simulating Large Network-Distributed Processing Systems", Proc. of the 2000 Winter Simulation Conference. J.A. Joines, R.R. Barton, K.Kang, and P.A. Fishwick, eds.: <http://www.wintersim.org/prog00.htm>.
6. J. Fromm, K. Genser, T. Levshina, I. Mandrichenko, "FBSNG - Batch System for Farm Architecture", CHEP2001, <http://www.ihep.ac.cn/~chep01/>.