Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

THE MONARC TOOLSET FOR SIMULATING LARGE NETWORK-DISTRIBUTED PROCESSING SYSTEMS

Iosif C. Legrand
Harvey B. Newman

Charles C. Lauristsen Laboratory of High Energy Physics
California Institute of Technology
Pasadena, CA 91125, U.S.A.

ABSTRACT

The next generation of High Energy Physics experiments have envisaged the use of network-distributed Petabyte-scale data handling and computing systems of unprecedented complexity. The general concept is that of a "Data Grid Hierarchy" in which the central facility at the European Laboratory for Particle Physics (CERN) in Geneva will interact and coherently manage tasks shared by and distributed amongst national "Tier1 (National) Regional Centres" situated in the US, Europe, and Asia. CERN and the Tier1 Centers will further communicate and task-share with the Tier2 Regional Centers, Tier3 centers serving individual universities or research groups, and thousands of "Tier4" desktops and small servers.

The design and optimization of systems with this level of complexity requires a realistic description and modeling of the data access patterns, the data flow across the local and wide area networks, and the scheduling and workload presented by hundreds of jobs running concurrently on large scale distributed systems exchanging very large amounts of data.

The simulation toolset developed within the "Models Of Networked Analysis at Regional Centers" - MONARC project provides a code and execution time-efficient design and optimisation framework for large scale distributed systems. A process-oriented approach for discrete event simulation has been adopted because it is well suited to describe various activities running concurrently, as well the stochastic arrival patterns typical of this class of simulations. Threaded objects or "Active Objects" provide a natural way to map the specific behaviour of distributed data processing (and the required flows of data across the networks) into the simulation program.

This simulation program is based on Java2$^{(TM)}$ technology because of the support for the necessary methods and techniques needed to develop an efficient and flexible distributed process oriented simulation. This includes a convenient set of interactive graphical presentation and analysis tools, which are essential for the development and effective use of the simulation system.

The design elements, status and features of the MONARC simulation tool are presented. The program allows realistic modelling of complex data access patterns by multiple concurrent users in large scale computing systems in a wide range of possible architectures. Comparison between queuing theory and realistic client-server measurements is also presented.

1 INTRODUCTION

The future Large Hadron Collider (LHC) experiments have envisaged Computing Models involving many hundreds of physicists doing analysis at institutions around the world (Newman 1999). These Models encompass a complex set of wide-area, regional and local-area networks, a heterogeneous set of compute- and data-servers, and a yet-to-be determined set of priorities for group-oriented and individuals' demands for remote data and compute resources. Each of the experiments foresees storing and partially distributing data volumes of Petabytes per year, and providing rapid access to the data over regional, continental and transoceanic networks. Computational Grid technology (Foster and Kesselman 1999) extended to data intensive tasks and worldwide scale could be used to effectively manage such systems (Newman 2000). Distributed systems of this size and complexity do not exist yet, although systems of a similar size to those foreseen for the LHC experiments are predicted to come into operation by around 2005.

The aim of this paper is to describe the simulation program, being developed by the MONARC project, as a design and optimization tool for such large scale distributed computing system. The goals are to provide a realistic simulation of distributed computing systems, customized for specific physics data processing and to offer a flexible and dynamic environment to evaluate the performance of a range of possible data processing

architectures. The simulation program was designed as a tool to study very large distributed systems. It is not intended to be a detailed simulator for basic components such as operating systems, data base servers or routers. Instead, based on realistic mathematical models and measured parameters on test bed systems for all the basic components, it aims to correctly describe the performance and limitations of large distributed systems with complex interactions. At the same time it provides a flexible framework for evaluating different strategies for middleware design, providing dynamic load balancing and optimising resource utilisation as well as turnaround time for high priority tasks.

2 DESIGN CONSIDERATIONS

The simulation and modelling task for MONARC requires the description of both simple and complex data processing programs, running on large scale distributed systems, interacting and exchanging very large (and small) amounts of data.

An Object Oriented design, which allows an easy and direct mapping of the logical components into the simulation program and provides the interaction mechanism, offers the most flexible, extensible solution for modelling such large-scale systems. This design approach also copes with systems which may scale and change dynamically.

A process-oriented approach for discrete event simulation is well suited to describing concurrent running programs, as well as all the stochastic arrival patterns that characterise how such systems are used. Threaded objects, or "Active Objects" (having an execution thread, program counter, stack, mutual exclusion mechanism...), offer great flexibility in simulating the complex behaviour of distributed data processing programs. This approach offers a natural way of describing complex running programs that are data dependent and which concurrently compete for shared resources (Legrand 2000).

The MONARC simulation program is built with Java(TM) technology. Java has built-in multi-thread support for concurrent processing, which can be used for simulation purposes by providing a dedicated scheduling mechanism. Java also offers good support for distributed objects (RMI and CORBA) architectures and for graphics. The flexible graphics tools, and facilities to analyse data interactively, are essential in any simulation project.

The tool's "simulation engine" provides a dedicated scheduling mechanism that is based on semaphores for the "Active Objects". It also provides a mechanism to dynamically add or remove objects from the system. Handling dynamically loadable modules is essential to describe complex configurations which may change or evolve in time. The "Active Object" is the basic class that must be inherited by all the entities in the simulation, which require a time dependent behaviour. It provides the methods for synchronous and asynchronous communications with other objects, and the mechanism to communicate with the simulation engine so that it can be interrupted, suspended and resumed during execution. Objects which extend this basic class may implement any specific time dependent behaviour, which can be a function of messages or data received, its previous state(s), and its access to certain shared resources. In this way it is possible to implement highly non-linear processes such as caching and swapping. It also offers a means of describing the stochastic input pattern for jobs and activities in the system.

As the number of jobs necessary to be simulated in such applications may be huge, a dedicated structure that allows "Active Objects" recycling was implemented to improve the simulation efficiency. The interrupt mechanism, implemented as an atomic (synchronised) self addressed event, for the "Active Objects" offers an effective way to simulate discrete event processes assuming a "continuous" flow in time between events which modify parts of the system.

Shared resources, like CPU or I/O links, are represented in the simulation as normal objects, but access to their different update methods needs to be made, synchronised with the external "running" entities. There is a mutual exclusion mechanism when accessing unique atomic parts that avoids interruption: this guarantees the correct representation of the execution of concurrent processes.

A detailed but still intuitive Graphical User Interface (GUI) to the simulation program allows the user to change parameters dynamically, to load user-defined modules with specific time response functions, and to monitor and analyse the simulation results. It provides a powerful development tool for evaluating and designing large scale distributed systems.

3 COMPONENTS MODEL

The simulation program requires the abstraction of all components from the real system and their time dependent interactions. This abstracted model has to be equivalent to the original system in the key respects that concern us. The simulation engine is designed to be generic for any distributed system. However, there are certain HEP-specific system components that are specially modelled to make the tool useful to the physics community. The major components are described below.

3.1 Data Model

The current data model follows the Objectivity/DB architecture and the basic object data design used in HEP. This model allows an efficient way to describe very large database systems with a huge number of stored objects.

The atomic unit object is the "Data Container", which emulates a database file containing a set of objects of a certain type. In the simulation, data objects are assumed to be stored in such "data container" files in a sequential order. In this way the number of objects used in the simulation to model large number of real objects is dramatically reduced, and the searching algorithms are simple and fast. Random access patterns, necessary for realistic modelling of data access, are simulated by creating pseudo-random sequences of indices. Clustering factors for certain types of objects, when accessed from different programs, are simulated using practically the same scheme to generate a vector of intervals. A "Database Unit" is a collection of containers and performs an efficient search for type and object index range.

The database server component simulates the client-server mechanism used to access objects from a database. It implements response time functions based on data parameters (page size, object size, access is from a new container, etc.), and hardware load (how many other requests are active at the same time). In this model it is also assumed that the database servers control the data transfers from/to the mass storage system.

Different policies for storage management may be used in the simulation. The model is designed to handle a very large number of objects while at the same time providing an automatic storage management scheme. It allows the emulation of different clustering schemes in the data for different types of data access patterns, and the simulation of ordered data access when following the associations between the data objects, even if the objects reside in databases located in different database servers.

3.2 Multitasking Data Processing Model

This is based on sharing resources such as CPU, memory and I/O between concurrently running tasks by scheduling their use for very short time intervals. The model is based on an "interrupt" driven mechanism implemented in the simulation engine. It calls the interrupt method implemented in the "Active Objects", which is the base class for all "running entities". The way it works is shown schematically in Figure 1.

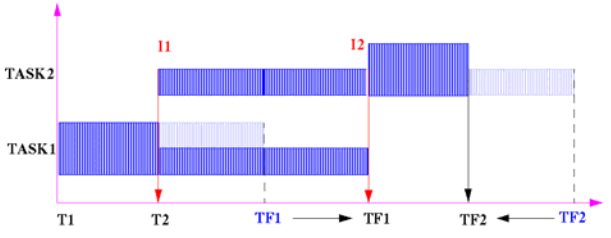


Figure 1: Modelling Multitasking Processing Based on an "Interrupt" Scheme

Referring to this figure, when a first job (Task1) starts, the time it takes is evaluated (original TF1), and this "Active" object enters into a wait state for this amount of time unless it is interrupted. If a new job (Task2) starts on the same hardware, it will cause an interrupt to the first task. Both tasks will share the same CPU power and the time to complete for each of them is re-computed assuming that they share the CPU equally or based on a running priority scheme (new TF1 and original TF2). Then both jobs will enter into a wait state and listen for other interrupts. When the first job (Task1) is finished, it creates another interrupt to re-distribute the resources for the remaining jobs. This model assumes that resource sharing is maintained between any discrete events (e.g. new job submission, job completion) that occur during the simulated time interval.

3.3 Network Model

Accurate and efficient simulation of networking is also a major requirement for the MONARC simulation project. The simulation program should offer the possibility to simulate data traffic for different protocols on both LAN and WAN. This has to be done without precise knowledge of the network topology. We note that it is practically impossible to simulate the network on a packet-by-packet basis for large amounts of data.

User-defined time dependent functions are used to evaluate the effective bandwidth. The approach used to simulate the data traffic is based on an "interrupt" scheme similar to the multitasking model described above. When a message transfer starts between two end points in the network, the time to completion is calculated. This transfer time is calculated using the minimum speed of all the links between the end points, and it may be a function of the network protocol being used. The time to complete is used to generate a wait statement that can subsequently be interrupted in the simulation. If a new message is initiated, an interrupt is generated for the LAN/WAN object. The speed for each transfer affected by the new one is re-calculated, assuming that the transfers are running in parallel and share the bandwidth (with weights depending on the protocol). With this new speed the time to completion for all the messages affected is re-evaluated and inserted into the priority queue for future events. This approach requires an estimate of the data transfer speed for each component and the round trip time for each network. For a long distance connection an "effective speed" between two points has to be used. This value can be fully time dependent to emulate "outside" traffic sharing the same lines, as well as the averaged effects of overheads and performance limitations associated with certain network protocols and conditions such as packet loss.

This approach for data transfer provides an effective and accurate way of describing many large and small data

transfers occurring in parallel on the same network. This model cannot describe speed variation in the traffic during one transfer if no other transfer starts or finishes. This is a consequence of the fact that we have only discrete events in time. However, by using smaller packages for data transfer or artificially generating additional interrupts for LAN/WAN objects, the time interval for which the network speed is considered constant can be reduced. As before, this model assumes that the data transfer between time events is done in a continuous way utilising a certain part of the available bandwidth.

3.4 Arrival Patterns

A flexible mechanism of defining the stochastic process of submitting jobs is necessary. This is done using the "dynamic loadable modules" feature in Java, which supports the ability to include (threaded) objects into running code. These objects are used to describe the behaviour of a "User" or a "Group of Users". It should be able to describe both batch and interactive sessions, and also to use any time dependent distribution describing how jobs are submitted. An "Activity" object is the base class for all activity processes to estimate the time dependent job arrival patterns and correlation. These Activity objects are in fact the job injectors into the simulation framework.

The user can provide very simple sections of Java code, to override the "RUN" method of the "Activity" class, and provide the time dependent profile of different job submission activities, as shown schematically in Figure 2. Any number of "Activity" objects may be dynamically loaded via the GUI allowing them to be studied independently or all together.

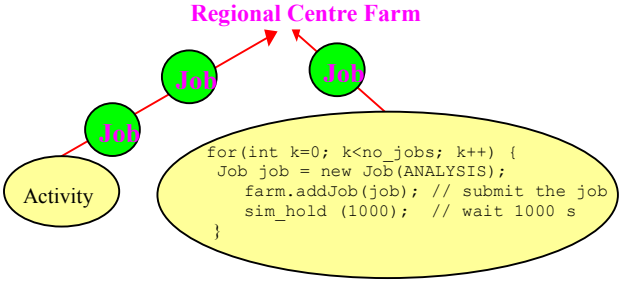


Figure 2: Modelling Jobs Submission into the System

4 REGIONAL CENTER MODEL

The "Regional Centre" (Figure 3) is a complex, composite object containing a number of data servers and processing nodes, all connected to a LAN. Optionally, it may contain a Mass Storage unit and can be connected to other Regional Centres. Any regional centre can dynamically instantiate a set of "Users" or "Activity" objects, which are used to generate data processing jobs based on different scenarios. Inside a Regional Centre different job scheduling policies may be used to distribute jobs to processing nodes.

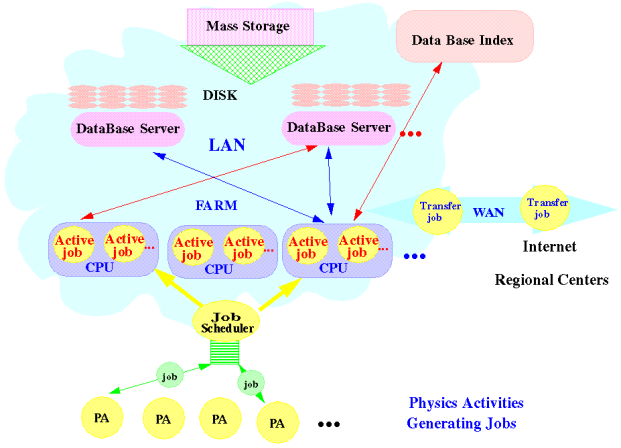


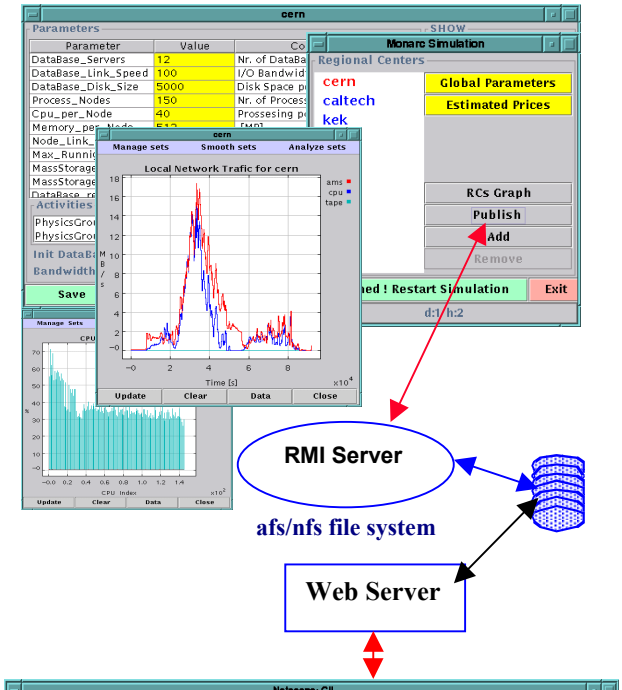
Figure 3: An Example Regional Center (Sub)-Model

5 THE GRAPHICAL USER INTERFACE AND AUXILIARY TOOLS

An adequate set of GUIs to define different input scenarios, and to analyse the results, is essential for the simulation tools. The aim in designing these GUIs was to provide a simple but flexible way of defining the parameters for simulations and the presentation of results.

The number of regional centres considered can be changed through the main window of the simulation program. The "Global Parameters" frame allows the (mean) values and their statistical distributions for quantities, which are common in all Regional Centres to be changed. The hardware cost estimates for the components of the system may also be obtained. For each Regional Centre in the simulation, the user may interactively select the parameters which are graphically presented (CPU usage, memory load, load on the network, efficiency, Database servers' load etc). Basic mathematical tools are available to examine all simulation results: computation of integrated values, mean values and integrated mean values.

To publish or store the simulation results and all the relevant files used in the simulation, an automatic procedure has been developed. This allows publishing locally, or to a MONARC Web server. The Web Page thus offers a repository for different simulation studies within the MONARC Collaboration (The MONARC repository Web page). There can be found the configuration files, the Java source code used to certain modules and the results (tables and graphic output) for any given simulation runs. The aim of this facility is to provide an easy way to share ideas and results. The publishing procedure is implemented in Java using the Remote Method Invocation mechanism. The schematic view of how this works is shown in Figure 4.



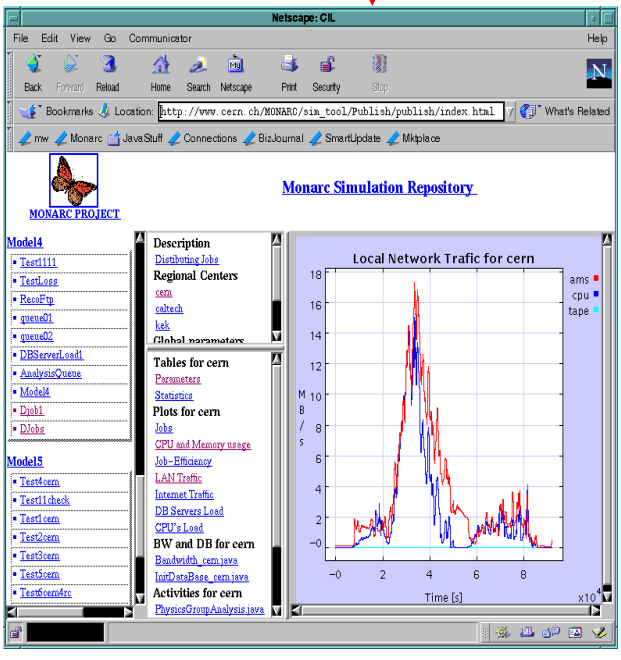


Figure 4: Simulation GUI and the "Publishing" Mechanism

6 COMPARISON WITH QUEUING THEORY

A few basic comparisons of the simulation program with Queuing Theory (Haverkort 1998) have been made.

6.1 M|M|1 Model

This model consists of a queuing station where jobs arrive with a falling exponential inter-arrival time distribution with rate λ. Furthermore, the job time service requirements are also negative exponentially distributed with mean E[S] =1/μ.

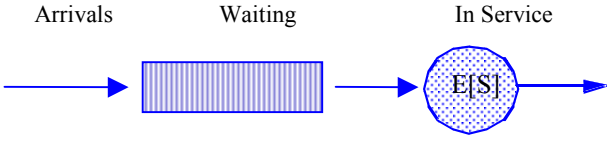


Figure 5: M|M|1 Queuing Model

The simplest queueing model M|M|1 theory gives the formula for mean number of jobs in the system and the mean response time:

$$E[N] = \frac{\rho}{1-\rho} \quad \text{and} \quad E[R] = \frac{E[S]}{(1-\rho)} \qquad (1)$$

where E[N] is the mean number of jobs in the system, E[R] mean response time of the system, E[S] - mean serve time of the system, and the utilisation ρ=λ/μ.

This can be described in the simulation program as a data base server acting as a queuing station for data request from clients with the same time distribution. The results for different arrival rate shown in Figure 6 reproduce the mean number of jobs in the queue.

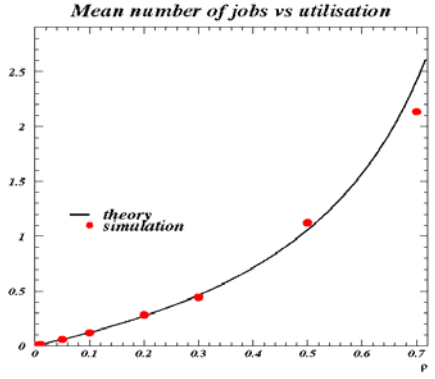


Figure 6: Comparison with Queuing Theory for the M|M|1 Model

6.2 M|M|1 Network Queue Model

This type of queuing model consists of a chain of M|M|1 queues. In this case the mean total number of jobs in the system and the mean total response time of the network are defined by:

$$E[N] = \sum_{i=1}^{r} E[N_i] = \sum_{i=1}^{r} \frac{\rho_i}{1-\rho_i} \qquad E[R] = \sum_{i=1}^{r} E[R_i] = \sum_{i=1}^{r} \frac{E[Si]}{(1-\rho i)} \qquad (2)$$

where the utilisation for each stage is $\rho_i = \frac{\lambda}{\mu_i}$

In the simulation program, it can be modelled by creating a sequence of jobs processing correlated data and having different sizes and processing times. Figure 7 shows the mean number of jobs and the mean response time as a function of system utilisation for the M|M|1 network queue model.

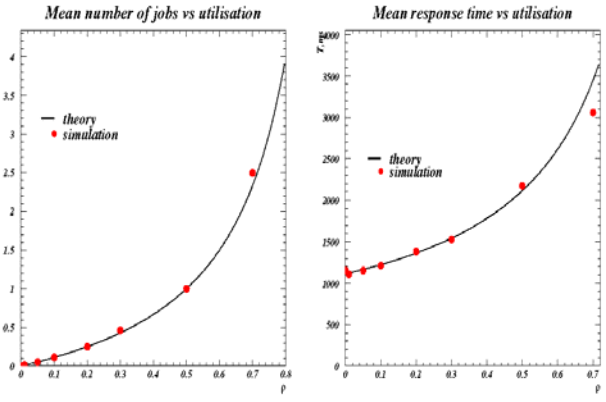


Figure 7: Comparison with Queuing Theory for the M|M|1 Network Model

7 TESTBED MEASUREMENTS AND VALIDATION OF THE SIMULATIONS

Distributed applications on wide area networks give rise to stringent performance demands that are not satisfied by any existing data, CPU and network management/monitoring infrastructure. The aim of the testbeds has been to study the efficiency and behaviour as a function of both the network characteristics, and the parameters of an ODBMS based application (Objectivity/DB) for distributed analysis of experimental data. The dynamic usage of system and network resources at host and application level has been measured in different client/server configurations, on several LAN and WAN layouts. Measurement evaluation has identified system bottlenecks and resource limitations. In addition, efficient working conditions in the different scenarios have been defined, and some critical behaviour has been observed when moving away from the optimal working conditions.

The future improvement of the monitoring tools, providing online visualisation of resource utilisation, has been identified as important, not only for troubleshooting, but also for the development of authorisation policy and workload management in general.

The evaluation of a computer and network system involves the iteration of measurement, modelling of the system behaviour, development of the simulation tools and then validation (Haverkort, 1998). With sufficient iterations of the above cycle, one can predict the behaviour of the system for various types of loads with sufficient accuracy. Therefore the validation of the MONARC Simulation tools should be closely related to the required "level of detail" as the project aims for improved accuracy with greater detail in the system modelling.

In particular, sharing of the common resources such as CPU, storage I/O bandwidth, local and wide-area network bandwidth, queuing mechanisms, and the performance of the distributed ODBMS systems are shown to be the key parameters to estimate the overall performance of the regional centres models.

7.1 Comparison with Simulation

Several testbed environments have been set up by the Testbeds Working Group at CERN, KEK, INFN, Caltech, and SLAC. These sites are connected with various types of wide area networks, such as dedicated satellite links, ATM permanent virtual circuits and QoS services (Morita et al. 2000).

Examples of physics analysis applications using Objectivity/DB have been developed and tested in these environments. Monte Carlo simulated raw data was converted into Objectivity/DB database format. A simple C++ program was written to read every object in the event using a database iterator. Multiple jobs were run on the system with three configurations: (1) Local file database access on one machine (machine A), (2) Local file database access on another machine (machine C) and (3) a pair of machines acting as client and server of Objectivity/DB AMS (Advanced Multi-thread Server). The job execution time and the CPU utilisation were measured as a function of the number of multiple concurrent jobs. The profile of the jobs, such as CPU cycles per event, were deduced from two machines with different CPU power and disk I/O speed.

The simulation results reproduce the testbed measurements very well for the concurrent running of jobs as shown in Figure 8.

The same set of job profile parameters also qualitatively reproduces the distribution of job execution time for concurrently competing for the same resources as shown in Figure 9.

Another set of measurements was performed on a QoS network using various link speeds between the AMS server and clients. The data model used in the measurements was a set of event data of various sizes. The job profile parameters were extracted from the single job configuration and the behaviour of the concurrent job execution was reproduced with the simulation program. Results for multiple concurrent clients are presented in Figure 10, for a 100Mbps Local Area Network (a) and a 2Mbps Wide Area Network (b). The simulation tool reproduces the measured job execution time of concurrent Object database access using local and wide area networks. A set of tools and methods has been developed to make

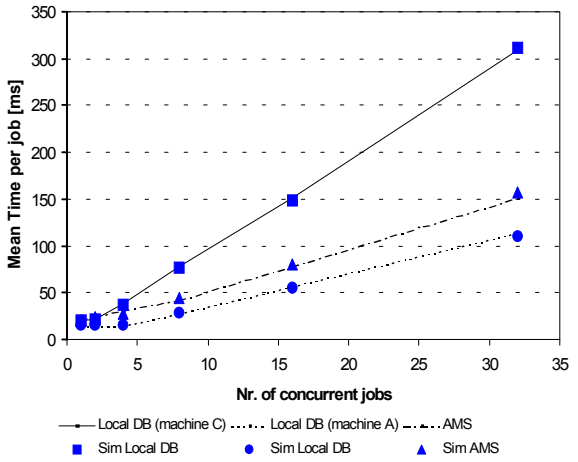


Figure 8: Comparison of Measured Values with Simulation

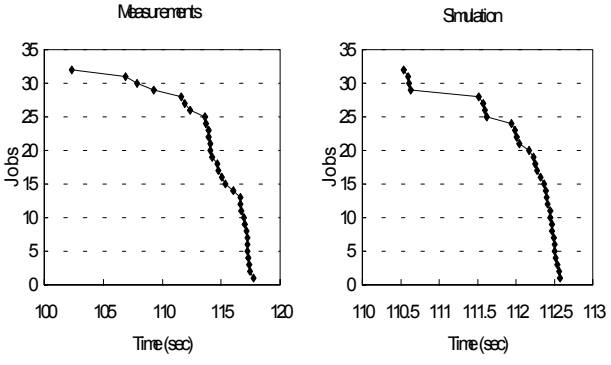


Figure 9: Profile of Job Execution Times: Measurements versus Simulated Results

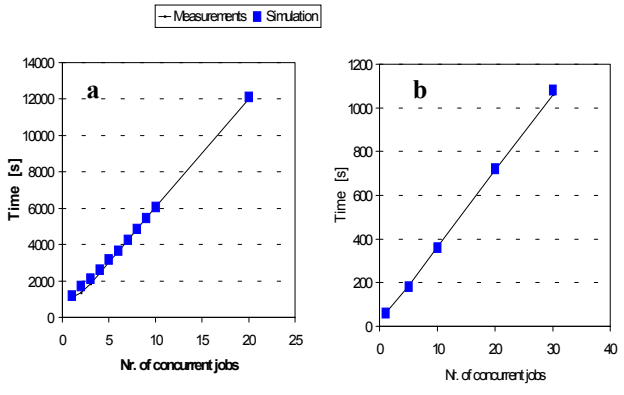


Figure 10: Comparison of the Measurements with Simulation for Concurrent Multiple Client Access: (a) WAN at 2Mbps, (b) LAN at 100Mbps

a profile of a given analysis job, and to monitor the performance of the distributed data analysis environment. The behaviour of the distributed ODBMS has been modelled and validated.

However, it is important to understand that the evaluation of the system performance is a continuous cycle of refining modelling, testing and validation. To make a reliable prediction of the system performance, more detail modelling of the analysis job and the validation of key system components such as hierarchical mass storage system are necessary.

8 A SIMPLE EXAMPLE

As an example we consider a typical activity to analyze physics events by several groups in one regional center. Assuming that 1000 jobs are submitted for execution with a specific time distribution, we compare the resource utilization in the system with the Job's response time distribution which may reflect the "user satisfaction" and the efficiency in analyzing data. In Figure 11 three cases are presented. A high resource utilization (Figure 11a) provides a quite slow response time, practically twice the "theoretical" time expected by the user. A fast response time may be achieved, but without a high utilization factor for resources (Figure 11c). Finding the cost-effective solution and the strategies in jobs submission, priority and resource allocations is one of the challenges this project has to solve.

9 SUMMARY

A CPU and code-efficient simulation approach to the problem of simulation of distributed computing systems has been developed and tested within the MONARC Collaboration. It provides a transparent way to map the distributed data processing, data transport and analysis tasks onto the simulation frame, and can describe dynamically even very complex computing models.

The Java(TM) programming environment, used extensively to build the MONARC simulation tool, is very well suited for developing a flexible and distributed process oriented simulation, equipped with adequate graphical and statistical tools.

This simulation program is still under development to include more sophisticated methods to evaluate different strategies to optimise the utilisation of resources in very large scale distributed computing systems.

ACKNOWLEDGMENTS

This work has been performed in collaboration with the MONARC project at CERN. We would like to thank CERN IT Division for the hospitality and support extended to I. Legrand during the course of this work. K. Sliwa, A. Nazarenko, A. Dorokhov, Y. Morita, L. Perini, P. Capiluppi and I. Gaines made important contributions to the simulation system features or to the definition of the physics models discussed in this paper. This work has been partially supported by DoE Grant DE-FG03-92-ER40701.

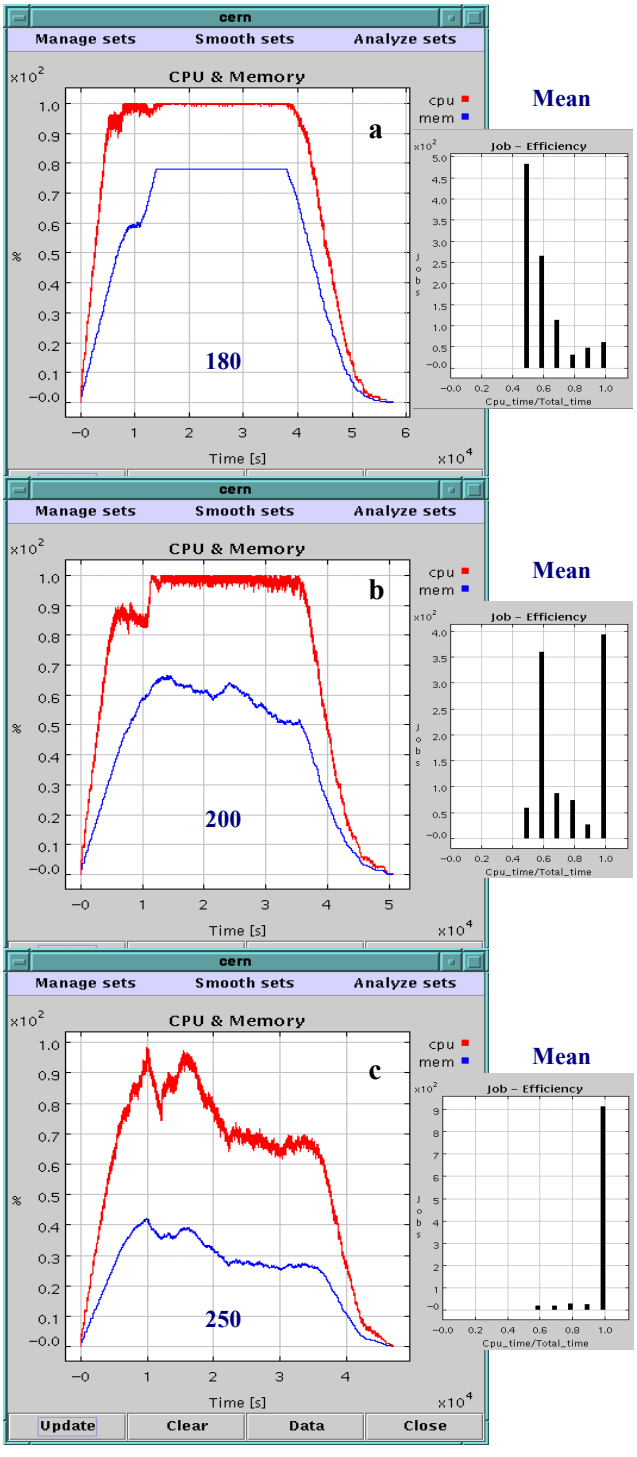


Figure 11: Resource Utilization and Job Efficiency for a Typical High Energy Physics Data Analysis Activity, as a Function of the CPU Power Installed

REFERENCES

Foster, I. and Kesselman, C., 1999. *The GRID: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann Publishers, San Francisco.

Haverkort B.R., 1998. Performance of Computer Communication Systems, John Wiley & Sons Ltd.

Legrand I., 2000. Multi-threaded, discrete event simulation of distributed computing system, CHEP2000, Padua, Italy, <http://chep2000.pd.infn.it/> paper number 148).

Objectivity: <http://www.objy.com/> and at CERN: <http://wwwinfo.cern.ch/asd/lhc++/Objectivity/>

Morita, Y. et al., 2000. Validation of the MONARC Simulation Tools, CHEP2000, Padua, Italy <http://chep2000.pd.infn.it/>, paper number 113.

Newman, H., 1999. Distributed Computing and Regional Centres. LCB Marseilles Workshop <http://lcb99.in2p3.fr/HNewman/Slide1.html>

Newman, H., 2000. *Worlwide Distributed Analysis for the Next Generations of HENP Experiments*, CHEP2000, Padua, Italy <http://chep2000.pd.infn.it/>, paper number 385.

The MONARC simulation repository <http://www.cern.ch/MONARC/sim_tool/Publish/publish/>

The MONARC Project, <http://www.cern.ch/MONARC>

AUTHOR BIOGRAPHIES

IOSIF C. LEGRAND is a Senior Software Engineer and a full time staff member of Caltech. He is the principal developer of the MONARC simulation system. His E-mail and web addresses are <Iosif.Legrand@cern.ch> and <http://home.cern.ch/cil>.

HARVEY B. NEWMAN is a Professor of Physics at Caltech and Chair of the Collaboration Board of the US high-energy groups in the CMS experiment. He was a former member of the NSFNet Technical Advisory Group in 1986 and is currently Spokesperson of MONARC and Co-Spokesperson of the Particle Physics Data Grid Project. His E-mail and web addresses are <newman@hep.caltech.edu> and <http://l3www.cern.ch/~newman>.