A Mobile-Agent Based Performance-Monitoring System at RHIC

R.W. Ibbotson, B. Gibbard, D. Stampf, T. Throwe

Brookhaven National Laboratory, Upton, NY, USA

Abstract

The computing environment for event reconstruction and analysis at RHIC currently includes a farm of 200 Pentium-based computers, several NFS and AFS servers, an HPSS storage system and 10TB of RAID disk. The configuration of this environment is continuously evolving to meet the expanding needs of the experiments as the RHIC facility nears operation.

A system is being developed at BNL to monitor the performance of these various subsystems, as well as to understand changes in the performance due to configuration changes. An array of metrics is being developed to provide measures of latencies, transfer rates and network connectivity. This system will provide a continuous accessible log of these metrics and will allow correlations between metrics to be studied.

In order to determine the performance of specific tasks on these distributed systems and provide platform independence, a Java-based mobile-agent system is being used as the framework for task distribution. Both the measurements and the code used to perform the measurements are stored in a central database for scalability and reproducibility. The design of the system and initial experiences in the RHIC Computing Facility will be presented.

Keywords: monitoring, mobile agents, Java, RHIC

1 Introduction

The primary goals of monitoring a distributed computing environment are (1) performance evaluation and performance tuning, (2) fault recovery/diagnostics, (3) early detection of errors or lapses in proper performance of the system, and (4) identification of under-utilized resources. The first of these is critical to the second and third, since if the nominal performance of a system is not known, it cannot be evaluated as functioning properly or improperly. The present system attempts to address the first three of these goals in order to provide both an evaluative and diagnostic tool for the administrators of the systems at the RHIC Computing Facility (RCF) and a monitoring tool for the users.

The task of monitoring a distributed computing environment such as that at the RCF is clearly more involved than the task of monitoring a single system. The computing environment at the RCF includes a farm of $\approx$200 Pentium-based computers used for event reconstruction and analysis, several Solaris and AIX file servers, an HPSS storage system, and several data-mining computers. The performance of a particular task in such a distributed system depends on several facets of the entire computing environment, almost always including the network throughput/topology. As an example, the speed with which data can be transferred to or from a network-mounted disk depends on the load on the server and client computers, the architecture of the client and server computers, and the status of the network connection between the two. Performance of complex systems such as HPSS will involve still more variables.

Rather than attempting to measure the performance of each facet of the systems at the RCF, the present system attempts to provide "end-to-end" tests which are sensitive to several components of the RCF. This tool is expected to be useful in the initial stages of the debugging process only, but is also expected to provide performance measurements useful to the users of the systems. The measurements themselves can be placed in three categories: repeatable tasks which provide information on some aspects of the RCF systems, information from in-house software systems, and information taken directly from vendor-supplied monitoring systems.

2 Design of the System

In order to be useful for evaluation, diagnostics and monitoring, it was decided that the system must have the following features: Measurements of the present performance of the system must be viewable and comparable to past measurements of the same type, and the correlations between different metrics must be available and viewable. The technique used for performing measurements should be independent of operating system as much as possible, so that changes in the performance due to large configuration changes can be tracked. The measurements should be automated and minimally impact the systems being measured. A further desired feature is that the rate at which sampling is performed be variable. This would allow tests to be performed at a high rate, possibly impacting the systems measurably, in order to assist in diagnosing and resolving existing problems.

In order to provide these capabilities, the RHIC Instrumentation Facility has been designed following a task-driven, centralized model. In this model, tasks are initiated at a central server computer and the measurements performed by these tasks are time-stamped and stored in a database at the server location. Since all results are time-stamped and stored in the same database, different metrics can be easily correlated in time. In the present case, the tasks themselves (actually the Java byte-code defining the tasks) are stored in the same database. This ability to store the code for the measurements in the central database is provided by the nature of the Mobile Agent system used to distribute the tasks to the "target" hosts. This allows a simple "re-measurement" of historic measurements and assures that the same code is used to perform the measurements in all cases regardless of target host. The centralization of the system also allows the simple addition of new metrics at the server without modifications to the target computers. The variability in sampling rate can also be achieved by adjustments at the central computer only. The tasks are designed to be distributed and executed using a mobile agent system. For the present monitoring system, we have used the IBM Aglets [1] system. Before describing the manner in which mobile agents have been utilized, a brief overview of the technology will be presented.

Mobile Agent systems employ the "remote programming" model, as compared to the more familiar "remote procedure call" (RPC) or "remote method invocation" model (see, for example, [2] for a comparison of the two models). Whereas RPC allows a procedure to be executed on a remote computer, the form of the procedure calls must be agreed upon in advance and implemented on both client and server. As an alternative to either model, event-driven monitoring systems have also been developed which do not require remote execution of any kind, but require installation of the event-producing software on all target hosts. In the mobile-agent paradigm, however, the program itself may halt execution, transfer to a remote host and resume execution. The procedures to be executed on the remote host do not need to be pre-determined since the agent (object with code) moves itself to the remote host. In order to maximize platform-independence, many mobile-agent architectures, including the IBM Aglets system, are written in Java. The remote host on which the agent executes must therefore be running a program which can accept agents and allow them access to the Java virtual machine (JVM). This program (the "Aglets server") also imposes

security restrictions on the agents which it accepts, and allows agents to communicate via message-passing.

Central to the Aglets system is an abstract class "Aglet" which must be extended by an object which will be a mobile agent. This class has been extended to define an abstract class "Instrument" which incorporates the basic functionality of a mobile agent which performs a task in the RHIC Instrumentation System. This functionality includes the ability to load run-time parameters from the database, store a result object after making a measurement, and insert that measurement into the database at the appropriate time. All information related to a specific measurement is stored in the database using a key which uniquely identifies a single instance of one Instrument using a specific set of parameters. This key can identify a sequence of measurements as long as they are all made by the same instance of an Instrument (which by design uses only one set of parameters).

The default mobility pattern used for these measurements involves making periodic measurements on one remote host with an arbitrary time interval between measurements. The Instrument returns to the server for this time interval (which is defined in the parameter-set stored in the database), in order to reduce the possibility of an Instrument disappearing due to a systems failure during this interval. This mobility-pattern has been created as a separate object which extends the abstract class MobilityPattern (the abstract class supplies the basic functionality needed for any such object and insures that certain functionalities will be provided by the specific implementation of a mobility pattern). An Instrument defines the pattern of mobility that it will follow by instantiating a copy of the specific MobilityPattern that it will use.

A procedure has also been developed by which all Instruments can be "tracked" during their life-cycle. At instantiation, each Instrument registers its existence with a central Tracker Aglet. If the Instrument becomes inaccessible, terminates unexpectedly or is late for an Instrument-defined deadline, the Tracker logs the information in a specific table of the database. The process of keeping the Tracker updated with the current state of the Instrument is performed by a StatusUpdater object which each Instrument possesses. This object shares information with the MobilityPattern object through pre-defined methods of the abstract superclasses. This design relies heavily on the object-oriented nature of Java, and allows considerable functionality to be built into any Instrument with minimal replication of code. The code which defines a specific task can often be written in less than 50 lines of Java.

3 Operating Experience with the System

A sample test "Instrument" was created early in the process of developing the RHIC Instrumentation Facility, so that the evolution of the system would be shaped by actual operating experience. The test consisted of a simple timed creation of a file on an arbitrary file system. The goal of this test was to monitor the access-time to an NFS-mounted file system, essentially providing mixed information about the state of the client and NFS serving computers. Tests were executed on a selection of computers in the event-reconstruction and analysis farm running NFS version 2 and version 3. A sample report of data taken with the RHIC Instrumentation system is shown in Figure 1. The factor of $\approx$3 difference in throughput between the top two curves reflects the improvement in changing from 1KB to 8KB block size for NFS-mounted volumes, although both systems exhibit similar behavior at times of high server load. The factor of $\approx$6–8 gain in throughput associated with migrating from NFS version 2 client software to NFS version 3 is clearly more than the gain associated with the further increase in block-size (to 32KB).

During the development and initial use of the system, much has been learned about how to improve the system. Although the network load due to the mobile agents (typically $\approx$15KB

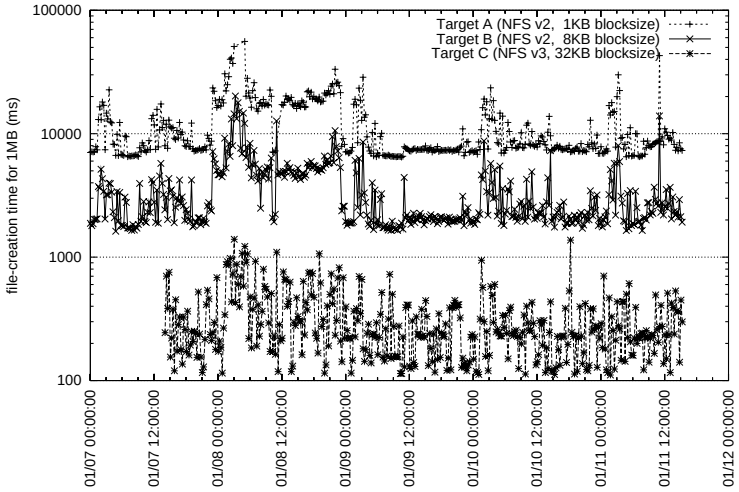

Figure 1: Report generated for tests of NFS file-access time, showing performance on three separate target computers.

for the transfer of an agent) is insignificant, the Aglets package does not appear to handle the occasional situation wherein several mobile agents arrive at one node at the same time. The agents which are transferring can detect such collisions and repeat the transfer attempt, but the frequency of collisions of agents returning to the central server grows with the number of scheduled tasks. This problem is currently being solved by developing different mobility patterns (for example, circulating among all nodes to be tested before returning to the central server). Another difficulty involves more complicated two-node tests such as a master-slave Instrument pair which measures the time taken to transfer a known quantity of data between two computers via the network. The two-node nature of this test will require a different reporting strategy than that developed for simple single-node tests. The current model of report generation will need to be replaced by a strategy in which the number of hosts which must be specified in order to retrieve a report is test-dependent.

Although there are still design issues to be solved, the initial implementation of the RHIC Instrumentation Facility is already providing useful information about the state of the RCF, and is proving useful in quantifying performance changes as the configuration of the facility evolves. The mobile-agent architecture has proven extremely useful in simplifying the evolution of the system and in expanding the role of the testing procedure. The additional volume of network traffic generated by the mobile agents has been studied and is small. We are confident that further development of the system and expansion of the tasks comprising the test suite will increase the usefulness of this system at RHIC and provide a valuable tool for the experimenters and operators alike.

References

1 D.B. Lange and M. Oshima, “Programming and Deploying Java Mobile Agents with Aglets”, *Addison-Wesley*, Reading, MA, USA, 1998.

2 M. Dönszelmann, “Mobile Agent Systems”, Cern School of Computing 1999 `http://webcast.cern.ch/Projects/CSC99/lectures`.