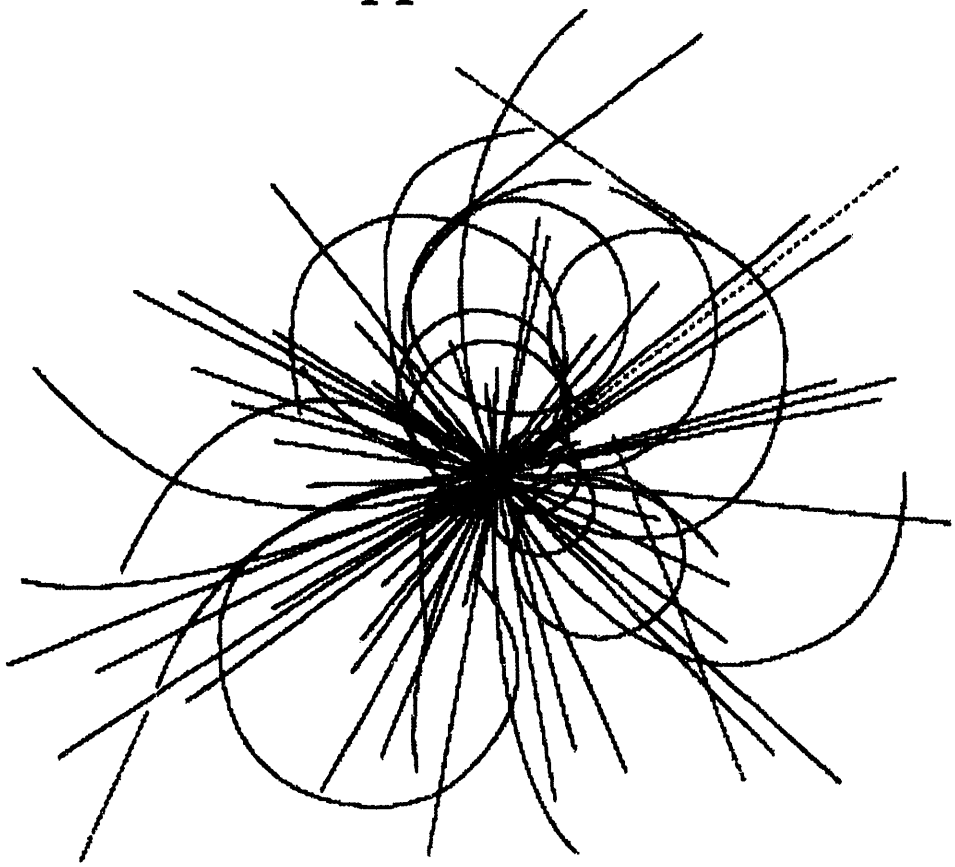
SSCL-Preprint-321

SSCL-Preprint-321
May 1993
Distribution Category: 400

B. Scipioni
D. Liu
T. Song

SISSY: An Example of a Multi-Threaded, Networked, Object-Oriented Databased Application



Superconducting Super Collider Laboratory

To be published in *Supercollider 5*

SSCL-Preprint-321

SISSY: An Example of a Multi-Threaded, Networked, Object-Oriented Databased Application*

B. Scipioni, D. Liu, and T. Song

Superconducting Super Collider Laboratory†
2550 Beckleymeade Ave.
Dallas, TX 75237

May 1993

*Presented at the Fifth Annual International Symposium on the Super Collider, May 6–8, 1993 San Francisco, CA.

†Operated by the Universities Research Association, Inc., for the U.S. Department of Energy under Contract No. DE-AC35-89ER40486.

SISSY: AN EXAMPLE OF A MULTI-THREADED, NETWORKED, OBJECT-ORIENTED DATABASED APPLICATION

B. Scipioni, D. Liu, and T. Song

Physics Computing Department
Superconducting Super Collider Laboratory*
2550 Beckleymeade Avenue
Dallas, TX 75237

ABSTRACT

The Systems Integration Support SYstem (SISSY) is presented and its capabilities and techniques are discussed. It is a fully automated data collection and analysis system supporting the SSCL's systems analysis activities as they relate to the Physics Detector and Simulation Facility (PDSF).[1, 2] SISSY itself is a paradigm of effective computing on the PDSF. It uses home–grown code (C++), network programming (RPC, SNMP), relational (SYBASE) and object–oriented (ObjectStore) DBMSs, UNIX operating system services (IRIX threads, cron, system utilities, shell scripts, etc.), and third party software applications (NetCentral Station, Wingz, DataLink) all of which act together as a single application to monitor and analyze the PDSF.

INTRODUCTION

Using networks of computers as a single system, that is, to perform tasks in a parallel or distributed fashion, has created a need for support services similar to those found on single computer systems. Systems administrators, systems integrators, developers and users all have need for certain information concerning the functioning and performance of a system of computers. SISSY is an automated tool which provides this information in a timely way and which relies upon UNIX standards in a heterogeneous environment.

MOTIVATION

There are two reasons for the choice of architecture for SISSY. First, it was desired to easily develop software which was maintainable, high performance, and makes maximum use of commercial applications. This was expected to result in fast development time and very low

*Operated by the Universities Research Association, Inc., for the U.S. Department of Energy under Contract No. DE-AC35-89ER40486.

manpower for development and maintenance. Second, one of the goals of the computing group is to promulgate and infuse modern software technology and engineering techniques into the HEP computing community at the SSCL. SISSY is a vehicle for demonstrating these techniques.

REQUIREMENTS

The PDSF,[1] Figure 1, is in its second phase of operations with Phase III in procurement and Phase I having been a prototype and testing ground for hardware and software integration techniques. It was mandatory, then, that all aspects of subsystem performance be monitored so systems analysis activities could provide directions for future computer acquisitions. This is an ongoing process with overall and detailed system performance and utilization reviewed on a weekly basis. The data required for this analysis effort includes average and peak utilization of all CPU, network, disk and tape systems throughout the PDSF. These hardware systems represent the bulk of the monetary resource invested and, therefore, a measure of effectiveness and efficiency of the facility is required. This analysis also points out hot spots and bottlenecks to guide future architectural changes and additions. In addition, both aggregate and individual user statistics need to be provided to the main groups of users, in particular the detector collaborations, so they may plan their current and future computing activities. The following list represents current requirements on weekly reporting, but it is only a part of the data collected and filtered:[2]

- % CPU utilization daily on a 24 hour basis, and weekly average for each cluster of compute servers, each data server, and each support computer (database and console multiplexers).
- Maximum and average data rates (KB/s) for each FDDI network every hour throughout the week. Maximum data rate (Kb/s) and packet rate (packets/s) separately for input and output for each computer and router network interface throughout the PDSF.
- Weekly average disk and tape storage in use, available and change from previous week.
- Hourly traces of the total number of distinct users with processes on the PDSF and within individual computing clusters.
- Total weekly CPU utilization (in minutes) consumed by each user, on each architecture and grouped by user organization. A list of all processes with more than one CPU minute for each user.
- Total CPU utilization for the week (in minutes) on each architecture broken down by user organization and location (in/out of the lab).

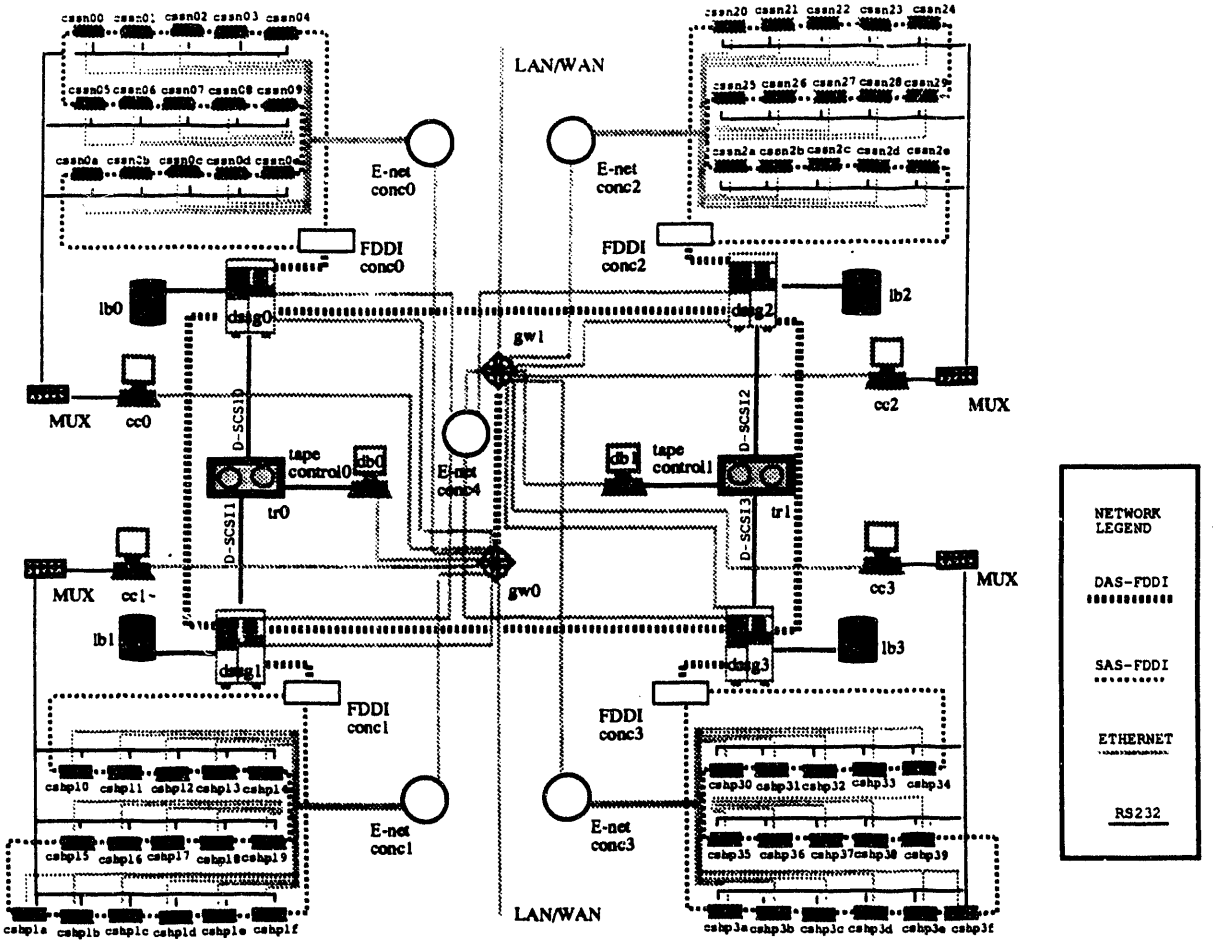


Figure 1. Physics Detector Simulation Facility (PDSF) Architecture.

DESIGN

There is a particular software design strategy implicit in the hardware architecture of the Physics and Detector Simulation Facility (PDSF) at the Superconducting Super Collider Laboratory (SSCL).[2] In particular, the many nodes which comprise the compute servers on the PDSF are intended, by design, to be able to both compute and perform I/O to network, disk and tape independently, or in parallel. In addition, the data servers act as a collection point for files common to applications running in parallel on the clusters of nodes, and as launch points for jobs executing on the compute servers. For this reason Symmetric Multi–Processing (SMP) is required for the data servers. Any application which takes maximum advantage of this architecture will be both multi–threaded and networked. In the case of SISSY there is a collector process on each data server, which is a multi–threaded object. Each thread executes a (non–blocking) Remote Procedure Call (RPC) to collect data from one of the compute servers of the associated cluster, as shown in Figure 2. In this way the RPCs can be used as a parallel network program, and all nodes in the system can be polled simultaneously rather than visited in series. This results in both higher data consistency and an increase in performance by more than an order of magnitude. The polling schedule and type of information requested is maintained in a database and queried by the collectors in order to determine when to execute data collection. This schedule can be changed asynchronously, and take immediate effect by a database update. Each compute server throughout the PDSF collects the requested information in parallel, usually by executing a UNIX command, then filters and packages the result in SQL and ships it back to appropriate collector. Here at the collector, along with information from the other nodes in the cluster, it is inserted over the network into the database. Scripts can be executed at any time to query the latest information in the database, but normally this is done

weekly by an automated procedure. Every Thursday morning a process awakens and executes all the database scripts, feeds the resulting tables to a spreadsheet application running in batch mode and automatically pipes the output viewgraphs and tables to a color printer for presentation.

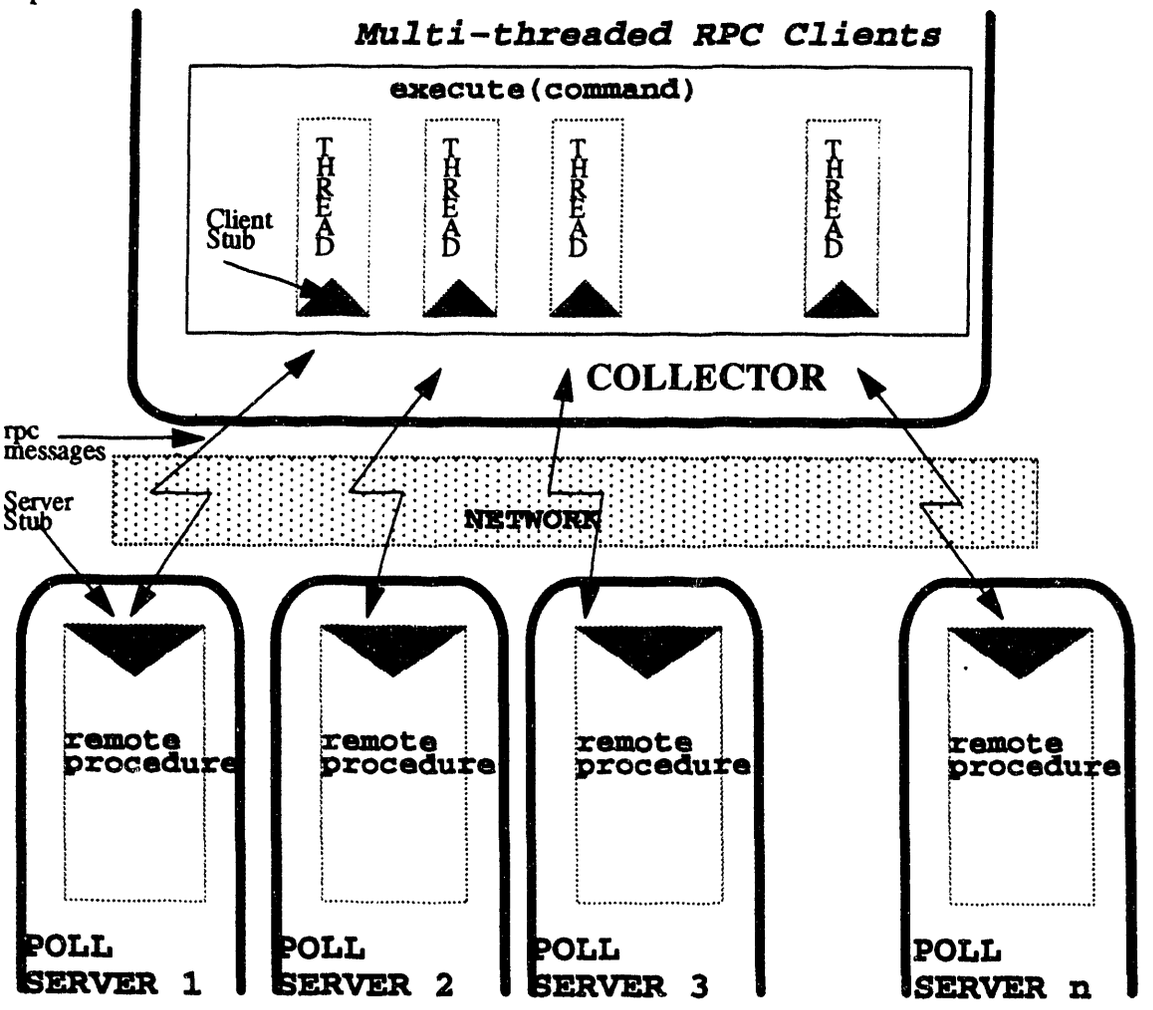


Figure 2. Threaded RPCs.

IMPLEMENTATION

All software written for SISSY is written in C++. The polling data are kept in Object Store object oriented database management system (OO–DBMS). An OO–DBMS is used because the polling process is a continuous loop through the collector objects. So the procedural logic of the queries and the repetitive nature of the polling benefit greatly from the features of the OO–DBMS. This ensures that the appropriate data persists in the program's data structures, reducing traffic and simplifying code. The actual systems data, however, is returned as SQL statements and inserted into two R–DBMS (Sybase) on two database workstations. The data entities were designed using both Software Through Pictures (IDE) and ERdraw (LBL), and the SQL create table statements (and design) were then generated automatically. The collector programs are instances of the same C++ class with all the functional details of the RPCs hidden away in the object methods. Each collector object invokes a public threaded method which then calls the RPC by invoking a private method. The threads mechanisms is provided through the SGI m_* library routines. These require the shared data to be global to the threaded function, which is natural for data which are static object members. There are only two tricks to get this to work correctly. First the RPC clients must be created in series because of the way they use

operating system resources. This goes fast and is put into the class constructor. Second, the threaded function must be of static storage class and visibility with its shared data global to the module. This allows the same copy of the method to be shared by all objects of the class and inhibits the compiler from putting in an extra pointer to the method and thus confusing the threads library routines. Some of the network data is collected by NetCentral Station network management software through the SNMP daemons executing on all computers and routers in the systems. NetCentral is implemented on top of Sybase which makes for a seamless integration of commercial and homegrown tools residing on a single database. The report generation is provided by WINGZ using its hyperscript batch capability and then printed on a Tektronix Phaser II color printer.

FAULT TOLERANCE AND UPGRADE

Currently higher fault tolerance and a natural language interface are being incorporated into SISSY. Since there are two database computers, either can pick up the data collection of the other if it is down.This up/down information on each node in the system will be maintained in the database by a program which attempts a remote execution. Since each computer has a minimum of two network interfaces (Ethernet and FDDI) if a data server is down severing the FFDI connectivity another data server can pick up the task over Ethernet. Design is complete and implementation underway. Also the natural language interface development has just begun using Natural Language from Natural Language, Inc. as a conversational English database interface.

CONCLUSION

SISSY is a portable, automated, high performance, systems analysis and maintenance tool based upon homegrown and commercial software, which is low in manpower requirements and provides the necessary information to successfully operate a parallel, distributed, computing environment. It is also a successful example of simple software development techniques which may be useful in production physics code running on the PDSF and other HEP computing facilities.

REFERENCES

1. G. Chartrand, L. Cormell, R. Hahn, D. Jacobson, H. Johnstad, P. Leibold, M. Marquez, B. Ramsey, L. Roberts, B. Scipioni, N. Shivapuja, G. Yost, "Physics and detector simulation facility specifications," SSCL-275, Attachment A, July (1990).
2. B. Scipioni, "Physics and detector simulation (PDSF) architecture/utilization", IISSC5(these proceedings).

END

DATE FILMED

11/01/93