JLAB-PHY-97-01

Database Driven Scheduling for Batch Systems

Ian G. Bird, Rita Chambers, Mark E. Davis, Andy Kowalski, Sandy Philpott, Dave Rackley, Roy Whitney

SURA/Jefferson Lab, 12000 Jefferson Avenue, Newport News, VA 23606

Experiments at the Jefferson Laboratory will soon be generating data at the rate of 1 TB/day. In this paper we present a database driven scheme that we are currently implementing in order to ensure the safe archival and subsequent reconstruction of this data. We use a client-server architecture implemented in Java to serve data between the experiments, the mass storage, and the processor farm.

Key words: Database; Client/Server; Batch; Farm; Java;

By mid-1997, experiments running at the Thomas Jefferson National Accelerator Facility ("Jefferson Lab") will generate some 10 MB/s of raw data that will be copied to mass storage and subsequently reconstructed on a 300 SPECint95 farm of Unix processors. The design of the hardware and issues addressed in the selection of the system are discussed in another paper submitted to this conference [1]. Data is archived in a StorageTek robotics silo, interfaced to four high performance STK Redwood tape transports. Data is moved in and out of the silo via RAID staging areas. In this paper we discuss the system we have designed to ensure the safe archival and subsequent processing of the data. The control system has been implemented in Java, using the features of remote method invocation and database access via the Java Database Connectivity (JDBC) interface.

The system architecture is based on a client-server model with three main server processes: a tape scheduler, a job scheduler and a database server. User interaction with all three servers is via a single graphical interface running on any network connected machine, as well as command line interaction for certain frequently used operations. Most of the user interaction can also be achieved through Web browsers. Scheduling and submission of jobs to the CPU farm is managed by LSF [2]. The database is used as a checkpoint for error recovery and system startup as well as for general data tracking and logging. A schematic of the system is shown in Figure 1.

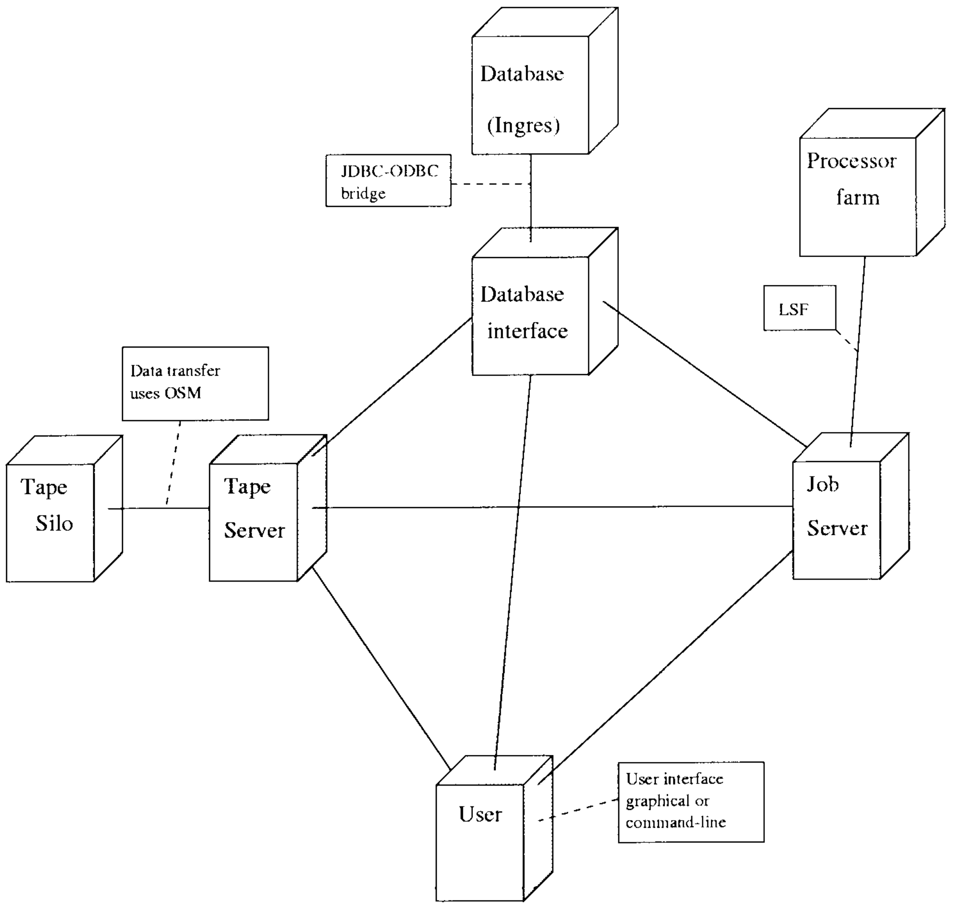

Fig. 1. Job and tape scheduling system overview

In order to guarantee archiving of the incoming raw data and to achieve optimum use of the Redwood tape drives, there is a unique access route to the tape silo – the tape server process. There are several types of tape access demand – that of ensuring that the raw data (at 10 MB/s sustained) is safely archived; data movement to and from the processing farm; and finally that of general tape archiving. This last includes, for example, copying data onto DLT for off-site export, as well as backups and user-initiated archiving of files. It is the responsibility of the tape server to schedule these various demands and to allocate them tape drives. At least one of the drives will be occupied with the copying of raw data to tape - this operation taking the highest priority. Requests to and from the farm take precedence over general requests. The database is notified of each request, source file and destination and changes in the request status, and can be used to restart the server when necessary without loss of data. The actual movement of data into and out of the silo is done by the hierarchical storage manager OSM [3] with modifications from DESY and by Jefferson Lab.

The second major component of the system is the job scheduler. It is the responsibility of this server to accept both "classic" requests like "run this job on this file" and more commonly "process all today's data from experiment X and write the output to tape - here is the command to run". As the larger experiments will generate some 500 2 GB files per day, this is the primary mode that the system is designed to deal with.

The server, when accepting such a request, will query the database to find the location of "today's data", break the resulting list of files into chunks corresponding to physical tapes and submit streams of jobs to the batch management system - LSF. In doing the submission, the system will generate on the fly the scripts necessary to control LSF. Each job cluster (corresponding to the files on a tape) consists of one control job whose sole function is to copy the data out of the silo and onto the staging disks, and as many processing jobs as there are files on the tape. Each of these jobs will run on a CPU node of the farm. As soon as a file (there may be up to 25 2-GB files on a tape) is available on the staging area, the job running on the farm will copy it to its local disk and begin processing it. Once the processing is finished, the data is copied back to the staging area. The server process will collect outputs from each initial request ("process today's data") and schedule copies back to tape when it has reasonable amounts of processed files available. While there is no attempt to keep files in sequence, files corresponding to different initial requests are not mixed. Network data transfers use the CERN rfio software.

At each stage the database is notified. It is by querying the database that the job server keeps track of the status of the jobs and requests. We also use the database to recover from system failures. All monitoring and querying of the system from the user interface is achieved by database queries of request and job status rather than direct query of the servers, although that facility is available to administrators. In addition statistics on the use of the farm and tape system will be generated from the database tables and viewable on the Web.

Given that the system is required to be accessible from Web based applications, and that a client-server solution arises naturally from the necessary distribution of the various pieces of control software across processors, the use of Java for the implementation was suggested. With the advent of JDK 1.1 there are new facilities for object communication between processes - the RMI (remote method invocation) interface coupled with the object serialization capabilities allow simple object-based communication between processors. Also in this release of Java is the JDBC interface which allows direct communication from Java code to a relational database. In our current implementation each server process communicates directly with the database, the actual implementation of that communication being hidden from the server via an interface of database objects that are natural objects inside the server and that may eventually map onto several tables inside the database. This mapping is dealt with by the interface. At the moment we do not use a separate database server -

individual database objects can communicate directly with the database, and it is the database itself that deals with locking issues etc. If in the future we introduce a server front-end to the database (which would be for reasons of efficiency yet to be investigated), the program interface will remain as it is now, only the communication paths will change. Also, the current implementation should allow the future substitution of the relational database with an object database with no or minimal changes to the applications.

At the time of writing (Feb. 1997) the main components of the system have been prototyped, and the design concepts proven. Remote method invocation in Java works well and flexibly solves the problem of object communication across a network. The access to the database via the JDBC driver also works well and enables an object view of the relational database which is essentially decoupled from the structure of the database itself. We intend to have the complete system running in test-production mode by April 1997, and in full production mode by the summer. Future improvements will involve increases in processing power, and the use of event (or groups of event) level parallelism on the farm - especially for CPU intensive Monte-Carlo simulations. For the control software, the use of database triggers combined with pairs of Observer/Observable objects to propagate database state changes in place of database queries will be investigated, as well as a more fully-featured and flexible user interface, and perhaps future extension to an interactive analysis facility.

References

[1] "Data Management for Batch Systems"; I .G .Bird, R. Chambers, M. E. Davis, A. Kowalski, S. Philpott, D. Rackley, R. Whitney.
Paper submitted to this conference.

[2] "LSF - Load Sharing Facility"; Platform Computing Corporation.
http://www.platform.com

[3] "OSM - Open Storage Manager"; Computer Associates.
http://www.cai.com