

# Report on the HPSS Database Benchmark

*D. Fisher*

**June 19, 2001**

*U.S. Department of Energy*

Lawrence  
Livermore  
National  
Laboratory

## DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

This report has been reproduced  
directly from the best available copy.

Available to DOE and DOE contractors from the  
Office of Scientific and Technical Information  
P.O. Box 62, Oak Ridge, TN 37831  
Prices available from (423) 576-8401  
<http://apollo.osti.gov/bridge/>

Available to the public from the  
National Technical Information Service  
U.S. Department of Commerce  
5285 Port Royal Rd.,  
Springfield, VA 22161  
<http://www.ntis.gov/>

OR

Lawrence Livermore National Laboratory  
Technical Information Department's Digital Library  
<http://www.llnl.gov/tid/Library.html>

# **Report on the HPSS Database Benchmark**

**June 19, 2001**

**David Fisher**

## **1 Introduction**

Concerns about the long-term viability of SFS as the metadata store for HPSS have been increasing. A concern that Transarc may discontinue support for SFS motivates us to consider alternative means to store HPSS metadata. The obvious alternative is a commercial database.

Commercial databases have the necessary characteristics for storage of HPSS metadata records. They are robust and scalable and can easily accommodate the volume of data that must be stored. They provide programming interfaces, transactional semantics and a full set of maintenance and performance enhancement tools.

A team was organized within the HPSS project to study and recommend an approach for the replacement of SFS. Members of the team are David Fisher, Jim Minton, Donna Mecozzi, Danny Cook, Bart Parlman and Lynn Jones.

We examined several possible solutions to the problem of replacing SFS, and recommended on May 22, 2000, in a report to the HPSS Technical and Executive Committees, to change HPSS into a database application over either Oracle or DB2.

We recommended either Oracle or DB2 on the basis of market share and technical suitability. Oracle and DB2 are dominant offerings in the market, and it is in the best interest of HPSS to use a major player's product. Both databases provide a suitable programming interface. Transaction management functions, support for multi-threaded clients and data manipulation languages (DML) are available. These findings were supported in meetings held with technical experts from both companies. In both cases, the evidence indicated that either database would provide the features needed to host HPSS.

## **2 Database Benchmark Tool**

The next step in the process was to conduct a performance benchmark of DB2 and Oracle. For comparison, SFS was benchmarked as well. The performance criterion we were most interested in was the rate at which HPSS user files could be created. For short files, the time to put a file into HPSS is dominated by the time to create the metadata. We created a program that emulated the metadata changes made by HPSS when a user file is created, and used it to measure the sustainable rate of user file creation. Then this program was "ported" to DB2 and Oracle, to obtain sustained rate performance numbers from those platforms.

It was understood that the SFS benchmark program would not directly "port" to either database; considerable re-writing of the program would be necessary. SFS uses a record oriented interface, while databases use SQL, which is oriented towards database tables

and rows, and operations on fields in those rows. We followed a set of porting rules which required the database benchmark programs to use a C language callable library interface (rather than say, embedded SQL), and to follow the basic algorithm of the SFS benchmark program. We permitted the database benchmarks to be written using best practice methods for the database in question, as long as those methods preserved the basic algorithm of the SFS benchmark, and could be applied to the eventual HPSS production interface. Members of the team inspected both programs to ensure a “level playing field”.

## **2.1 SFS Benchmark**

The SFS benchmark program emulates the steps taken by HPSS when a user file is created. A non-transactional read of a Name Server table is followed by a transaction in which the Name Server table is read again, and then a record is inserted in the table. Most fields in this record contain fixed values, but values of fields used as primary keys were given random values. The transaction continues by inserting a corresponding Bitfile Server record, after which the transaction is committed. This sequence of operations is enclosed in a loop which is repeated a configurable number of times.

The program emulates the multi-threaded nature of HPSS servers by creating a configurable number of threads, and running a separate instance of the benchmark loop in each. OFD management and other details of the use of the SFS interface are similar to the use by HPSS servers.

By December 2000, the SFS benchmark program was debugged and working as desired. The program was “frozen” at that time.

## **2.2 DB2 Benchmark**

A copy of the “frozen” SFS benchmark program was sent to Shirley Woods at ORNL who was assigned the task of porting it to DB2. Shirley created a database schema based on the SFS definitions of the subject HPSS metadata tables, using the appropriate DB2 data types. Members of our team reviewed this schema, some revisions were made, and the schema was approved. Shirley re-wrote the test loop using DB2 CLI library functions as the database interface. She optimized the performance of the program by extracting from the loop the statement preparation steps that were invariant and could be performed once prior to entry to the loop. Members of the team inspected the program and determined that it conformed to our porting guidelines.

## **2.3 Oracle Benchmark**

A copy of the “frozen” SFS benchmark program was sent to Lloyd Arrowood at ORNL who was assigned the task of porting it to Oracle. Lloyd created a database schema based on the SFS definitions of the subject HPSS metadata tables, using the appropriate Oracle data types. Members of our team reviewed this schema, some revisions were made, and the schema was approved. Lloyd re-wrote the test loop using Oracle OCI library functions as the database interface. As in the DB2 benchmark, the Oracle benchmark was optimized by removing invariant code from the test loop. SQL statements that were invariant in the loop were prepared once by each thread, then the prepared statements

were executed repeatedly with only the data values to be inserted varying from one iteration of the loop to the next. Members of the team inspected the program and determined that it conformed to our porting guidelines.

### **3 Testbed Description**

A fundamental requirement for the benchmark test was that all three benchmark programs must be run on the same test platform. In addition, the test platform should be a reasonably up to date system in terms of CPU and I/O performance. We found such a system in the Probe testbed at ORNL. Probe was created to test new equipment, new protocols, and software for storage applications. We were given access to a system, “Marlin”, which is an IBM RS/6000 model H70, with 4 PowerPC RS64-II CPUs, 4MB of L2 cache, 2 GBs of RAM, an IBM 2104 SCSI disk array and an IBM PCI 4 channel Ultra 3 SCSI RAID adaptor. A Texas Memory Systems disk, essentially a RAM disk, was available during the first round of runs of all three benchmark programs.

### **4 Running the Benchmark**

To avoid introducing unnecessary variables in the testing process, the following set of rules for running the benchmark programs were established:

- All tables must be empty at the start of a test run.
- Databases do not need to checkpoint during the test run.
- The test system (Marlin) runs only the designated benchmark during the test run.
- Log archiving, or its equivalent, is enabled during the test.
- The benchmark program runs 20 threads, with each thread performing 1000 pseudo file creates. An appropriate shell script runs the program in a loop for a total of 50 iterations for a total of 1,000,000 pseudo file inserts. The script redirects the standard output from the benchmark into a file.
- While the test is in progress, a tool such as “monitor” is run to collect CPU, disk and memory paging statistics.
- The test result consists of the benchmark standard output log and the monitor output log.

## **5 Benchmark Results**

### **5.1 SFS Results**

#### **5.1.1 Environment**

The benchmark was run against TX Series 4.3. The SFS data volume and log volume were on separate “disks” in the TMS disk system. Each TMS disk had its own Fibre Channel adaptor, an Emulex LP8000 with the Emulex driver. Both adaptors were mounted in 64-bit PCI slots. SFS log archiving was disabled.

#### **5.1.2 Results**

The benchmark achieved a rate of approximately 155 file insert operations per second. However, as the size of the SFS files grew, some performance degradation was observed.

In the last iteration of the test, the rate was reduced to 122 operations per second. The disk was essentially idle during the test, memory paging was minimal (< 20 faults/second), and CPU utilization was 100%.

## **5.2 DB2 Results**

### **5.2.1 Environment**

The benchmark was run against Version 7.1 of the DB2 Universal Database. DB2 was configured to allow the client to communicate with the server over the TCP/IP loopback interface. The TMS disk was used in early runs of the benchmark, but was not available for later runs. The absence of the TMS disk did not change the ranking or relative performance of the databases and SFS materially.

Both tables and indices were built on a Database Managed Space tablespace. The tablespace was configured on a container that consisted of a single AIX raw logical volume. The raw logical volume was configured on an 8-disk LUN (RAID 5-E) of the IBM 2104 SCSI disk array.

Logging was configured to use three 100MB log files. The log file directory was on an AIX JFS logical volume configured on a LUN of the IBM 2104. Log archiving was disabled for the benchmark. Because of the size of the log files, no database checkpoint occurred during the benchmark.

Since the benchmark required no long running queries, the degree of database parallelism was set to one. A single 200MB database buffer pool was configured for DB2 for its cache table and index pages. The size of the agent pool was set to 23 (20 + 3) to match the number of benchmark connections, plus a few spares.

The number of page cleaners and I/O servers was set to 4, one per processor, in order to optimize loading and flushing of the buffer pool.

### **5.2.2 Results**

In fifty iterations of the benchmark program, DB2 averaged 1351 pseudo file insert operations per second across all threads, with the high and low being 1399 and 1311 operations per second. The benchmark drove all CPUs at near 100% utilization (only around 2% idle across all four processors). A small amount of paging was noticed, with about 750 pageouts per 'monitor' sample period (every 10 seconds).

Flushing pages from the database buffer pool seemed to dominate the I/O activity, with the bytes written per second in the 4MB range. The rate of pages read from disk barely exceeded 300KB per second. Due to the small level of paging observed, it appears that additional memory might have increased performance.

## **5.3 Oracle Results**

### **5.3.1 Environment**

The test was run against Oracle Enterprise Edition, version 8.1.6. The TMS disk was used in early runs of the benchmark, but was not available for later runs. The absence of the TMS disk did not change the ranking or relative performance of the databases and SFS materially.

A number of changes were made to the configuration of the Oracle instance on Marlin to maximize the benchmark speed. The test used locally managed tablespaces with larger extent sizes to eliminate frequent and unnecessary space allocation. Larger redo log files were used to reduce the number of database checkpoints, which had been occurring every two minutes, and the redo log buffer was tuned for best performance. Each table and table index had more than one freelist assigned to eliminate bottlenecks associated with heavy row insert activity. The System Global Area was reduced in size to eliminate unused shared memory buffers. Data storage files were positioned on the devices to minimize I/O bottlenecks.

### **5.3.2 Results**

In fifty iterations of the benchmark program, Oracle yielded an average of 1157 pseudo file insert operations per second across all threads, with the high and low being 1245 and 633 operations per second. This test included two database checkpoints (which should have been avoided) that lowered performance to about 600 per second in one case and 950 per second in the other.

## **5.4 Performance evaluation conclusion**

Both DB2 and Oracle exhibited at least an 8X performance improvement over SFS. The results of the benchmark indicate that Oracle and DB2 performance is similar, in very carefully tuned environments, with DB2 holding a slight edge. Based on these results, either database represents a major step forward over SFS and both are acceptable in terms of performance.

The benchmark program has illustrated that both DB2 and Oracle can be integrated with HPSS and used to store and retrieve HPSS metadata. The exercise of creating a program to drive a database, using HPSS data structures, was part of the intended product of the benchmark effort. While the benchmark programs are “quick and dirty”, they are sufficiently representative of how HPSS works to show that either database could be used in place of SFS.

## **6 Technical Evaluation**

In this section the technical strengths and weaknesses of Oracle and DB2 that are germane to their use by HPSS are described. This section concludes with a database recommendation based on technical merit.

## **6.1 DCE compatibility**

The DB2/CLI library seems to be “DCE aware”, i.e., programs that invoke DCE header files compile and link without conflicts between the DCE environment and the database environment. The Oracle OCI library has direct conflicts with DCE. Both DCE and Oracle invoke C language “typedef” commands to define types such as “boolean”. To resolve this conflict, the portion of the Oracle benchmark program that includes the Oracle header definitions was compiled separately from the rest of the program, using a set of modified HPSS header files. Linking both modules into a single executable was successful.

To use Oracle in HPSS, we would have to segregate all of the OCI interface functions from the rest of the HPSS source and compile them with a special set of HPSS header files modified to avoid the DCE/Oracle conflicts. We have dealt with a similar problem in the common services portion of the HPSS source tree. We have to create an XDR interface between two HPSS servers using data structure definitions meant for DCE. We solve this problem with a Perl script that generates XDR compatible header files from the HPSS headers. We believe we can solve the DCE/OCI incompatibility with a similar work-around.

## **6.2 Standards**

DB2/CLI is based on the Microsoft Open Database Connectivity specification and the ISO Call Level Interface International Standard (ISO/IEC 9075-3:1995 SQL/CLI) for SQL/CLI. All or most of the symbols, function names and definitions follow these standards. A complete discussion of this topic can be found in chapter one of the IBM *DB2 Universal Database Call Level Interface Guide and Reference*. The Oracle OCI library provides functions with semantics similar to the standard DB2 functions, but the function names and calling sequences and other details do not conform to the standard. Our conclusion is that Oracle OCI is not based on a standard interface model.

## **6.3 Locking**

Oracle provides row-level locking, with no lock escalation. This means that locks are taken on individual rows as transactions progress, and that these locks do not escalate to pages or tables. This provides a greater level of concurrency in a transaction intensive environment. The possibility of transaction lock induced deadlocks is greatly decreased.

It is possible to get into serious problems with a poorly configured DB2 system. Situations can arise where lock escalation can cause deadlocks, but this should not happen in a properly configured database. It is possible that the DB2 locking mechanism is faster than Oracle’s because all of the locks are kept in memory. We believe that Oracle’s locks are kept in a column in the table, but we wonder if this approach may involve some extra I/O.

## **6.4 Multi-level versioning**

Oracle provides a multi-level versioning feature that allows data that is being modified in a transaction to be referenced outside of the transaction. The outside reference gets the

previous version of the data. Once the transaction commits, subsequent references get the updated version of the data. In addition, a time can be specified in the SQL SELECT statement and the version of the data in effect at that time is returned.

This feature eliminates the problem of “dirty reads”. A reference to an item of data that is being modified either gets the unmodified value, or the new committed value. There are several algorithms in HPSS dealing with dirty reads that would be simplified by this feature.

## **6.5 DB agent memory size**

Two models for representing the threads in the Oracle benchmark program were explored during the development of the program. The agent per thread model creates a process for each thread in the benchmark, while the Multi-Threaded Server (MTS) model funnels requests from threads through a dispatcher to a fixed number of agents. MTS is designed to limit the impact on system resources by a large number of users. Experimentation with these two models showed that the agent per thread model achieves significantly higher performance than MTS. DB2 also uses the agent per thread model. Both databases provide a means to configure the number of agent processes available to serve client requests.

A significant difference in the size of the agent processes was observed. DB2 agents use about 700KB of memory, while Oracle agents use about 24MB. While the amount of real memory used by these processes may be manageable, we are concerned that a large number of 24MB processes may be needed to support HPSS, and they may impose substantial requirements on the amount of memory needed by host nodes.

## **7 Technical recommendation**

Either database performs at a rate almost an order of magnitude greater than SFS, even when SFS was given an edge with a RAM disk system. The benchmarks demonstrate that HPSS metadata can be mapped into database data types, that the transaction management tools are suitable, and that using a database for HPSS metadata storage and retrieval is feasible.

For our purposes, neither database holds a significant technical advantage over the other. Oracle has locking strategies and multi-level versioning not found in DB2, but DB2 is a better choice for the DCE environment and uses a standardized interface that may reduce the work required to add a second database platform. We believe that HPSS could run over either database effectively and efficiently.

We conclude that HPSS should be re-hosted over Oracle or DB2 rather than remaining on SFS, and recommend DB2 as our initial target platform. The benchmarks demonstrated that either database will provide a higher metadata transaction rate than SFS, and will support growing metadata stores with less performance loss. DB2’s better fit in the DCE environment, and its conformance to interface standards make it our first choice.

## 8 Interface library

We plan to develop a database interface library for HPSS that will provide an abstraction layer between the database API and HPSS. This library will hide the details of the database API, and will facilitate the use of databases other than our recommended initial target. The library will be designed to minimize the amount of database dependent code outside of the library. The principal design objective of this library is to make porting of HPSS to a different database a practical matter.

## 9 Appendix A

The following information represents a summarized account of the results obtained from the benchmark testing for each database tested. To obtain the detailed information that these results are based on, please contact:

David Fisher  
Lawrence Livermore National Laboratory  
dsf@llnl.gov  
(925) 422-4215

## 10 Appendix B

The SFS benchmark was run with and without the Texas Memory Systems disk. The performance numbers shown in this report were obtained using the disk.

In addition to the Probe SFS benchmark, a number of tests were run with the SFS benchmark program at LANL. The purpose of these tests was to learn how SFS performance varies with disk configuration, and to experimentally determine the best disk configuration for large production HPSS systems. Several tests were run on a 44P-270 with a Sun T300 disk array configured as a RAID 1 SFS log volume and a RAID 5 SFS data volume. These disk systems were configured to minimize disk waiting in the benchmark. A number of conclusions can be drawn from these tests:

1. SFS will run at 100% CPU utilization in the first few iterations of the test.
2. As the size of the SFS data file increases, HPSS file insert performance decreases. Performance does not scale up with CPU speed unless the data volume is stored on a striped RAID 5 system, or a system such as the TMS.
3. The SFS log volume must be placed on a high performance disk such as a cached RAID, or it will constrain SFS from fully utilizing the CPU with resultant performance degradation.
4. The primary SFS limitation appears to be CPU resource utilization as the benchmark always used 100% of the CPU when the disk was configured to eliminate I/O bottlenecks. Results better than those seen with the TMS disk do not seem to be obtainable.

University of California  
Lawrence Livermore National Laboratory  
Technical Information Department  
Livermore, CA 94551

