PDS: A Performance Database Server

MICHAEL W. BERRY[1], JACK J. DONGARRA[1,2], BRIAN H. LaROSE[1], AND TODD A. LETSCHE[1]

[1]*Department of Computer Science, University of Tennessee, Knoxville, TN 37996*
[2]*Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN 37831*

ABSTRACT

The process of gathering, archiving, and distributing computer benchmark data is a cumbersome task usually performed by computer users and vendors with little coordination. Most important, there is no publicly available central depository of performance data for all ranges of machines from personal computers to supercomputers. We present an Internet-accessible performance database server (PDS) that can be used to extract current benchmark data and literature. As an extension to the X-Windows-based user interface (Xnetlib) to the Netlib archival system, PDS provides an on-line catalog of public domain computer benchmarks such as the LINPACK benchmark, Perfect benchmarks, and the NAS parallel benchmarks. PDS does not reformat or present the benchmark data in any way that conflicts with the original methodology of any particular benchmark; it is thereby devoid of any subjective interpretations of machine performance. We believe that all branches (research laboratories, academia, and industry) of the general computing community can use this facility to archive performance metrics and make them readily available to the public. PDS can provide a more manageable approach to the development and support of a large dynamic database of published performance metrics. © 1994 John Wiley & Sons, Inc.

1 INTRODUCTION

Given the current evolution of computer hardware technology, computer vendors are consistently producing more advanced versions of current machines as well as introducing new architectures that can cause significant increases in system performance. This seemingly exponential growth in machine performance is accompanied by new varieties of computer benchmarks to track this growth. Until recently, serial benchmarks [1] have been the primary available measures of processor performance. With the advent of parallel benchmarks [2, 3] the complexity of benchmark acquisition and presentation will certainly increase. For example, classifications of parallel benchmarks may be based on communication characteristics, processor utilization and load balancing, data layout or mappings, and even parallel I/O constructs. Hence, the number of variations of a single parallel benchmark program can be large. The ability to store, organize, and disseminate credible computer benchmark data is of paramount importance if we are to categorize the performance of computers ranging from laptop computers (e.g., Apple Powerbook) to massively parallel machines (e.g., Thinking Machines, CM-5).

The performance database server (PDS) developed at the University of Tennessee and Oak Ridge National Laboratory is an initial attempt at performance data management. This on-line database of computer benchmarks is specifically

Received July 1993
Revised November 1993

designed to provide easy maintenance, data security, and data integrity of the benchmark information contained in a dynamic performance database. In Section 2, we briefly discuss some of the presentation formats currently used for benchmarks; we also give a short description of Xnetlib [4], which supports PDS. In Section 3, we discuss four major categories of benchmarks that we consider appropriate for PDS. The design and implementation of PDS are presented in Sections 4 and 5, respectively; illustrations of sample PDS queries are also provided in Section 5. Section 6 comprises a brief summary, a preview of future work, and instructions for acquiring PDS.

2 BENCHMARK PRESENTATIONS

Currently there are as many presentation formats as there are benchmarks. Any PDS must be able to present a variety of benchmark formats, and yet standardize the actual storage of them. As an example, consider the differences in the LINPACK and Perfect benchmarks.

2.1 Extremes

The LINPACK benchmark [5] uses three numbers: $n = 100$ Mflops,* n = 1,000 Mflops, and theoretical peak performance. The first two numbers reflect the solution of linear systems of order 100 and 1,000, respectively. Theoretical peak performance is defined as the absolute upper bound on performance that is enforced by architectural limits.

The Perfect benchmarks [6] are very different from LINPACK. Perfect is a set of 13 scientific application programs (over 50,000 lines of Fortran-77) that are to be singly executed and measured for elapsed CPU time and Mflops. The performance of each of the 13 programs can be reported from an unoptimized (baseline) or optimized run. Hence, there can be as many as 52 numbers produced for each machine configuration considered.

LINPACK and Perfect are just two examples of the high variability in benchmark presentations. Nonetheless, these presentation formats are derived from the very nature of the benchmarks themselves and should be preserved at least in part. Indeed, from a database user's perspective, it is desirable for any user interface to a benchmark collection to retrieve and present data in a way consistent with the benchmark's original design. That is, performance interpretation by the interface should be minimized and left to the user.

To this end, we have designed PDS with a simple tabular format that involves displaying the data in rows (machine configuration) and columns (data, results, statistics, etc.). Graphical representations of tabular data, such as the representation by SPEC [7] with the SPECmarks, are straightforward.

2.2 Xnetlib

The Netlib software distribution system maintained at the University of Tennessee and Oak Ridge National Laboratory has been in development for several years and has established a large database of scientific (numerical) software and literature. Netlib has over 120 different software libraries, as well as hundreds of articles and bibliographic information.

Originally, Netlib software library access involved the use of electronic mail to form and process queries. However, in 1991, an X-windows interface, Xnetlib, was developed by Dongarra et al. [4] in order to provide a more immediate and efficient access to the Netlib software libraries. To date, there have been over 3,200 requests for the Xnetlib tool. In turn, the number of Netlib acquisitions has also escalated. In fact, there were over 86,000 Xnetlib-based transactions and over 150,000 electronic mail acquisitions from Netlib in 1992 alone.

Before discussing the design of PDS within the Xnetlib tool, we briefly classify many of the most popular benchmarks used today.

3 A TAXONOMY OF BENCHMARKS

Following the classification originally discussed by Berry et al. [8], we examine benchmarks from two perspectives: type and use.

3.1 Classification According to Type

Although there are numerous performance metrics, we classify them into four major categories, or types: synthetic, kernel, algorithm, and application.

* Millions of floating-point operations per second.

Synthetic Benchmarks

Synthetic benchmarks are not representative of any real computation: rather, they exercise various basic machine functions. IOZone, a package written by Bill Norcott (norcott_bill@tandem-.com), primarily tests disk throughput by "stress testing" the reading and writing of very large data files. Dhrystones and Whetstones [1] are examples of once-popular synthetic benchmarks that are rarely used today. Dhrystones were designed to stress integer performance and use many string operations. Whetstones consist of ten modules that perform a variety of numerical computations such that a significant portion of run-time is spent in mathematical library routines and trigonometric functions. The use of Whetstones is on the decline because they prevent vectorization and various compiler-based optimizations.

Kernel-based Benchmarks

Kernel-based benchmarks contain sections, or kernels, of a sample application code. A large library of routines with many different functions may be characterized by a small code sample. An example might be a loop that is processed millions of times in the application. The Livermore Loops [9] are representative of this type of benchmark. These benchmark programs contain intensive floating-point operations: and, as is typical of a kernel benchmark, they stress a single functional unit of the hardware.

Algorithm

Algorithm-based benchmarks are implementations of well-defined algorithms that vary slightly over different platforms. Algorithms that have been optimized are implemented in that optimized format. Because of the large number of operations typically processed, however, small variations in specific implementations or machine-specific calls can be masked in the long run. Thus, barring new optimizations or radically new approaches, these benchmarks give consistent measures of performance over various platforms and implementations. Examples of algorithm benchmarks are LINPACK [5], Slalom [3], and the NAS Fortran kernels [10].

Application

Application benchmarks may be complete samples of engineering, scientific, or business applications. These applications typically stress several functional groups of the hardware. The Perfect benchmarks [6] are examples of application benchmarks. This benchmark suite comprises 13 different scientific and engineering applications that can be run in predefined configurations. Such benchmarks are especially interesting to scientists whose research may closely resemble that modeled by the benchmarks. Application benchmarks are the closest performance estimation to actually running a real engineering application on candidate hardware.

3.2 Classification According to Use

Benchmarks can also be classified by their use (see Table 1) in an attempt to describe operating environments and to address performance concerns within such environments. Some benchmarks are indicative of workstation or personal computer (PC) environments, whereas others are

Table 1. Classification by Benchmark Use

	Target Machines		
Benchmark Name	Workstations/PCs	Supercomputers/ Parallel Computers	Peripheral Devices
Linpack	X	X	
Perfect		X	
SPEC	X		
IOZone	X		X
Dhrystones	X		
Livermore Loops	X	X	
Slalom	X	X	
Flops	X		
Whetstones	X	X	
NAS parallel		X	

intended for mainframes or supercomputers. The Dhrystones, for example, were designed to test integer performance, spend significant time in string operations, and are therefore considered more representative of a workstation environment. The NAS parallel benchmarks [2], on the other hand, are intensively parallel and considered representative of a multiprocessor system environment.

In general, however, peripheral devices and local area networks (LANs) generally lack adequate benchmarks. One notable exception is the IOZone benchmark, which could be classified as a peripheral benchmark because it tests the disk performance and I/O bandwidth. Given the growing interest in distributed programming environments such as PVM [11] and Linda [12], we anticipate benchmarks for homogeneous and heterogeneous networks of machines in the near future.

4 DESIGN OF A PERFORMANCE DATABASE

Because of the complexity and volume of the data involved in a performance database, it is natural to exploit a database management system (DBMS) to archive and retrieve benchmark data. A DBMS will help not only in managing the data, but also in assuring that the various benchmarks are presented in some reasonable format for users: table or spreadsheet where machines are rows and benchmarks are columns.

Of major concern is the organization of the data. It seems logical to organize data in the DBMS according to the benchmarks themselves: a LINPACK table, a Perfect table, etc. It would be nearly impossible to force these very different presentation formats to conform to a single presentation standard just for the sake of reporting. Individual tables preserve the display characteristics of each benchmark, but the DBMS should allow users to query all tables for various machines. Loading benchmark data into these tables is straightforward provided a customized parser is available for each benchmark set. In the parsing process, constructing a raw data file and building a standard format ASCII file eases the incorporation of the data into the database.

The functionality required by PDS is not very different from that of a standard database application. The difference lies in the user interface. Financial databases, for example, typically involve specific queries like `EXTRACT ROW ACCT_NO = R103049`, in which data points are usually discrete and the user is very familiar with the data. The user, in this case, knows exactly what account number to extract, and the format of retrieved data in response to queries. With our performance database, however, we would expect the contrary: The user does not really know (1) what kind of data is available, (2) how to request/extract the data, and (3) what form to expect the returned data. These assumptions are based on the current lack of coordination in (public domain) benchmark management. The number of benchmarks in use continues to rise with no standard format for presenting them. The number of performance-literate users is increasing, but not at a rate sufficient to expect proper queries from the performance database. Quite often, users just wish to see the best-performing machines for a particular benchmark. Hence, a simple rank-ordering of the rows of machines according to a specific benchmark column may be sufficient for a general user.

Finally, the features of the PDS user interface should include the ability to

1. Extract specific machine and benchmark combinations that are of interest
2. Search on multiple keywords across the entire dataset
3. View cross-referenced papers and bibliographic information about the benchmark itself.

We include (3) in the list above to address the concern of proliferating numbers without any benchmark methodology information. PDS would provide abstracts and complete papers related to benchmarks and thereby provide a needed educational resource without risking improper interpretation of retrieved benchmark data.

5 PDS IMPLEMENTATION

In this section, we described the PDS tool developed and maintained at the University of Tennessee and the Oak Ridge National Laboratory. Specific topics include the choice of DBMS, the client-server design, and the interface. Specific features of PDS such as `Browse`, `Search`, and `Rank-Ordering` are also illustrated.

5.1 Choice of DBMS

Benchmark data is represented by Hobbe's RDB format [13]. This database query language offers several advantages. It is easy to manipulate. It is

Table 2. RDB Format*

01	Computer	35
02	OS/Compiler	45
03	N=100	7N
04	N=1,000	7N
05	Peak	7N

* The first column is the field number, the second is the field label, and the third is the field type, in (type-size) format. The default form is ASCII, and *N* denotes numeric data so that the Peak entry, for example, is a size 7 numeric field.

also efficient, being based on a perl model [14] that uses Unix† pipes to run entirely in memory. We have easily converted raw performance data into RDB format using only a few perl commands. Additionally, RDB provides several report features that help standardize the presentation of performance data. The RDB format specifies that database tables are defined using a schema of the form shown in Table 2.

The contents of the data files are expected to be in some regular grammar, usually a space-separated columnar format. A perl script takes a description of the columns and builds a tab-separated file. The schema is converted to a tab-separated file using the RDB command *headchg*. The data files are appended to the end of the schema file, and the resulting flat tab-separated file becomes the rdb format table. RDB uses the schema in the header to process the file. An example from the linpack.rdb is provided below.

```
   Computer          OS/Compiler      N=100   N=1000  Peak
35      45      7N      7N      7N
CRAY Y-MP C90 (16 proc. 4.2 ns) CF77 5.0 -Zp -Wd-e68    479     9715     15238
CRAY Y-MP C90 (8 proc. 4.2 ns)  CF77 5.0 -Zp -Wd-e68    468     5994     7619
CRAY Y-MP C90 (4 proc. 4.2 ns)  CF77 5.0 -Zp -Wd-e68    388     3272     3810
CRAY Y-MP C90 (2 proc. 4.2 ns)  CF77 5.0 -Zp -Wd-e68    387     1709     1905
```

This rdb table may then be searched for query matching. An example query for a Linpack benchmark with N=100 number equal to 388 is

cat linpack.rdb | row N=100 eq 388 | ptbl

and the benchmark returned is

```
Computer                          OS/Compiler         N=100   N=1000   Peak
--------------------------------  ------------------  ------- -------  ----
CRAY Y-MP C90 (4 proc. 4.2 ns)    CF77 5.0 -Zp -Wd-e68  388     3272     3810
```

† Unix is a trademark of AT&T Bell Laboratories.

5.2 Client-Server Design

Within PDS, the database manager runs only on the server, and the clients communicate via Berkeley sockets to attach to the server and access the database. This functionality was provided by the preexisting Xnetlib tool and was extended to provide support for the performance data. Figure 1 illustrates this client-server interface.

The Xnetlib client is an X-Windows interface that retrieves data from the server. This client is a view-only tool that provides the user a window into the database yet prohibits data modifications. The `Performance` button under the main xnetlib menu will provide users access to the PDS client tool. Users familiar with Xnetlib 3.0 will have an easy transition to using the PDS performance client.

5.3 PDS Features

PDS provides the following retrieval-based functions for the user:

1. A *browse* feature to allow casual viewing and point-and-click navigation through the database
2. A *search* feature to permit multiple keyword searches with Boolean conditions
3. A *rank-ordering* feature to sort and display the results for the user
4. A few additional features that aid the user in acquiring benchmark documentation and references.

Figure 2 illustrates the PDS start-up window that appears after the user selects the `Performance` button from the Xnetlib 3.3 main menu. The user then selects from the six available services (`Rank Ordering`, `Browse`, `Search`, `Save`, `Papers & Notes`, `Bibliography`).

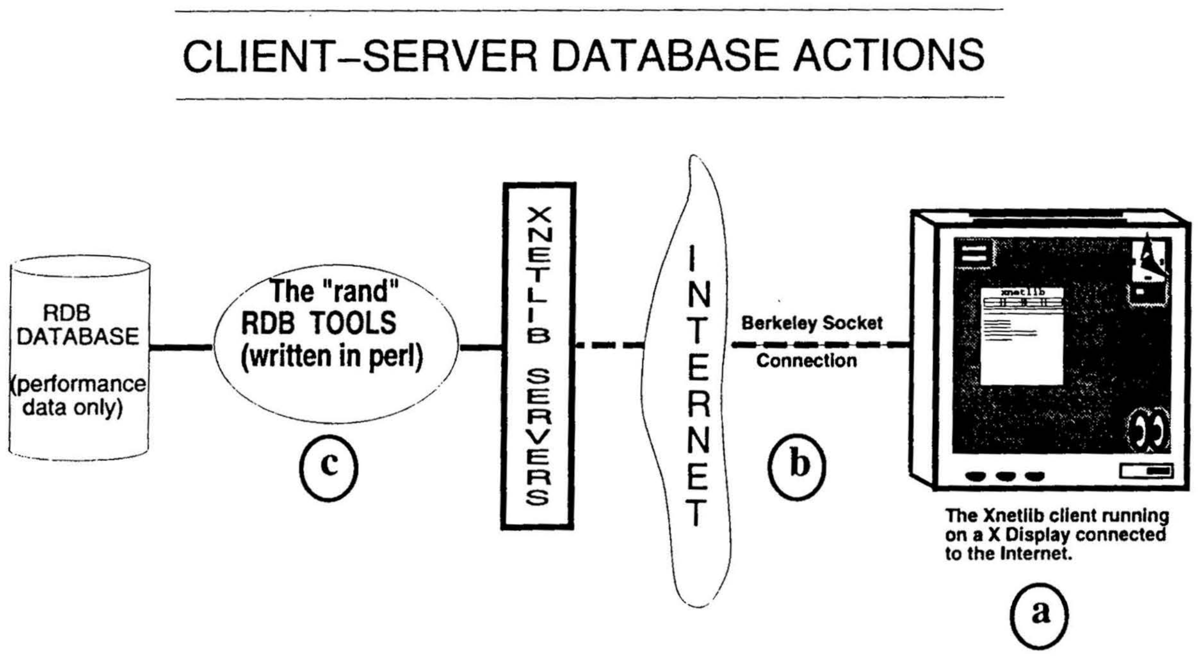
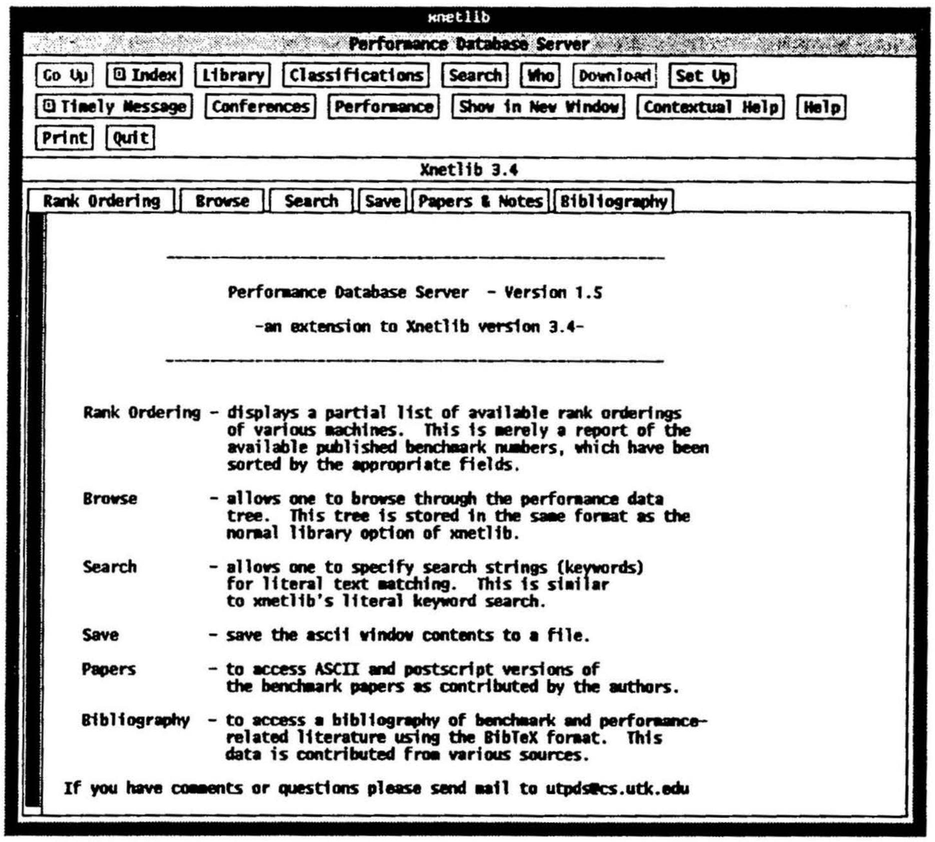
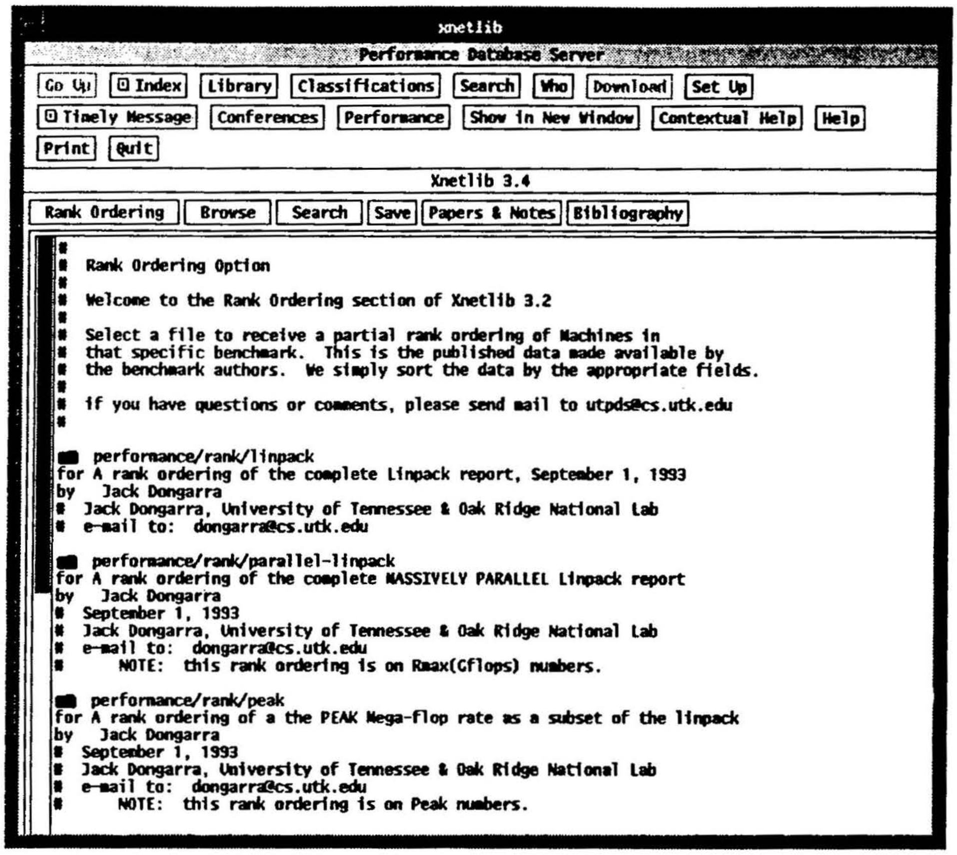

CLIENT–SERVER DATABASE ACTIONS
RDB DATABASE (performance data only)
The "rand" RDB TOOLS (written in perl)
XNETLIB SERVERS
INTERNET
Berkeley Socket Connection
The Xnetlib client running on a X Display connected to the Internet.
a
b
c

FIGURE 1 The PDS client-server interface: the X workstation (a) communicates over the Internet via Berkeley socket connection (b) to the Xnetlib server, which queries the database using rdb tools (c) and returns benchmark data via the socket connection.



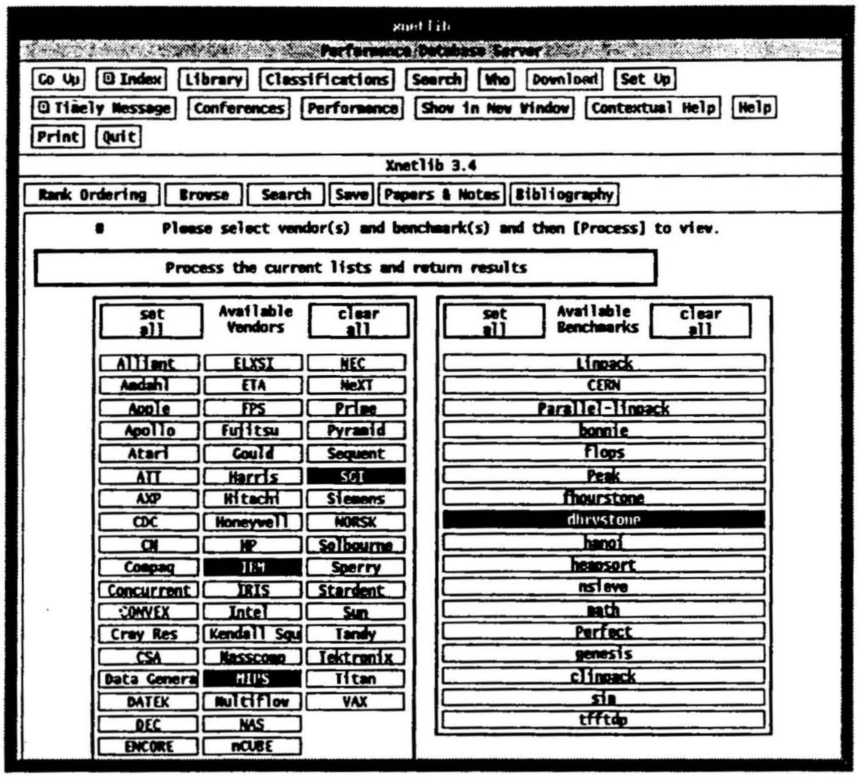
```
Performance Database Server  - Version 1.5

     -an extension to Xnetlib version 3.4-


Rank Ordering - displays a partial list of available rank orderings
                of various machines.  This is merely a report of the
                available published benchmark numbers, which have been
                sorted by the appropriate fields.

Browse        - allows one to browse through the performance data
                tree.  This tree is stored in the same format as the
                normal library option of xnetlib.

Search        - allows one to specify search strings (keywords)
                for literal text matching.  This is similar
                to xnetlib's literal keyword search.

Save          - save the ascii window contents to a file.

Papers        - to access ASCII and postscript versions of
                the benchmark papers as contributed by the authors.

Bibliography  - to access a bibliography of benchmark and performance-
                related literature using the BibTeX format.  This
                data is contributed from various sources.

If you have comments or questions please send mail to utpds@cs.utk.edu
```

FIGURE 2 PDS start-up window describing available services.



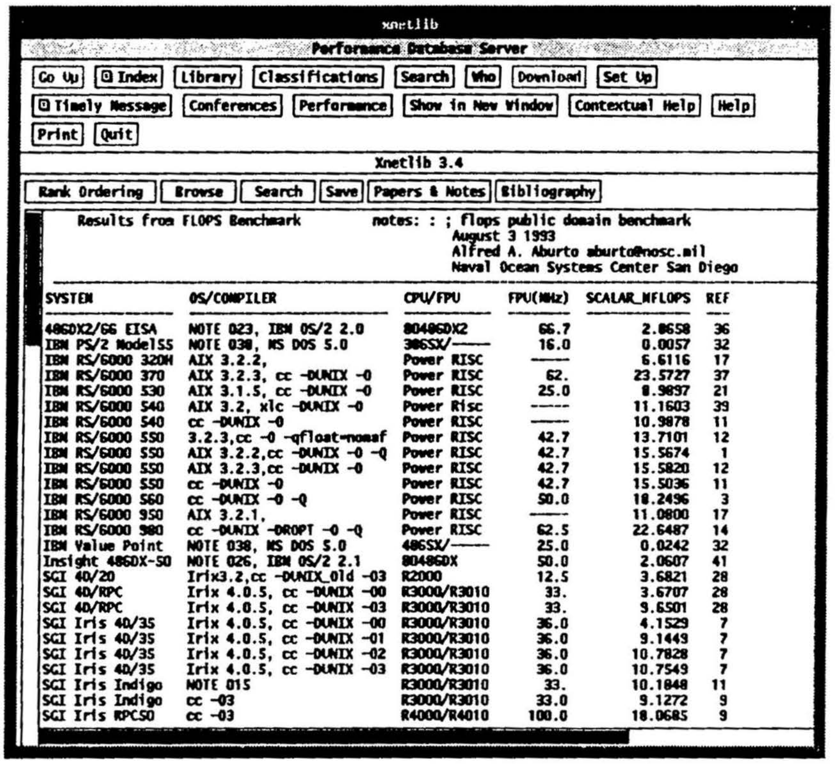
```
#
#  Rank Ordering Option
#
#  Welcome to the Rank Ordering section of Xnetlib 3.2
#
#  Select a file to receive a partial rank ordering of Machines in
#  that specific benchmark.  This is the published data made available by
#  the benchmark authors.  We simply sort the data by the appropriate fields.
#
#  If you have questions or comments, please send mail to utpds@cs.utk.edu
#

 performance/rank/linpack
for A rank ordering of the complete Linpack report, September 1, 1993
by   Jack Dongarra
#  Jack Dongarra, University of Tennessee & Oak Ridge National Lab
#  e-mail to:  dongarra@cs.utk.edu

 performance/rank/parallel-linpack
for A rank ordering of the complete MASSIVELY PARALLEL Linpack report
by   Jack Dongarra
#  September 1, 1993
#  Jack Dongarra, University of Tennessee & Oak Ridge National Lab
#  e-mail to:  dongarra@cs.utk.edu
#      NOTE:  this rank ordering is on Rmax(Gflops) numbers.

 performance/rank/peak
for A rank ordering of a the PEAK Mega-flop rate as a subset of the linpack
by   Jack Dongarra
#  September 1, 1993
#  Jack Dongarra, University of Tennessee & Oak Ridge National Lab
#  e-mail to:  dongarra@cs.utk.edu
#      NOTE:  this rank ordering is on Peak numbers.
```

FIGURE 3 The PDS rank-ordering feature menu.



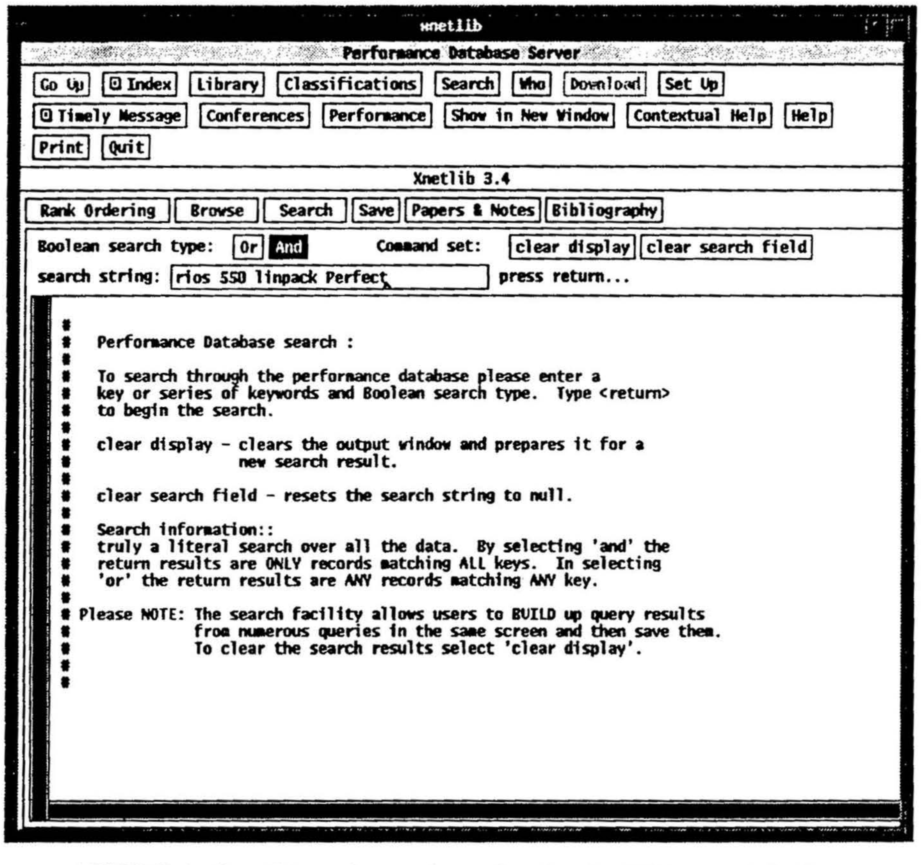
```
#        Please select vendor(s) and benchmark(s) and then [Process] to view.
```

FIGURE 4 The browse facility provided by PDS.



```
    Results from FLOPS Benchmark        notes: : ; flops public domain benchmark
                                                   August 3 1993
                                                   Alfred A. Aburto aburto@nosc.mil
                                                   Naval Ocean Systems Center San Diego

SYSTEM            OS/COMPILER                CPU/FPU        FPU(MHz)  SCALAR_MFLOPS  REF
-------------     ------------------------   ------------   --------  -------------  ---
486DX2/66 EISA    NOTE 023, IBM OS/2 2.0     80486DX2           66.7         2.0658   36
IBM PS/2 Model55  NOTE 038, MS DOS 5.0       386SX/-----        16.0         0.0057   32
IBM RS/6000 320H  AIX 3.2.2,                 Power RISC      -------         6.6116   17
IBM RS/6000 370   AIX 3.2.3, cc -DUNIX -O    Power RISC          62.        23.5727   37
IBM RS/6000 530   AIX 3.1.5, cc -DUNIX -O    Power RISC         25.0         8.9897   21
IBM RS/6000 540   AIX 3.2, xlc -DUNIX -O     Power Risc      -------        11.1603   39
IBM RS/6000 540   cc -DUNIX -O               Power RISC      -------        10.9878   11
IBM RS/6000 550   3.2.3,cc -O -qfloat=nomaf  Power RISC         42.7        13.7101   12
IBM RS/6000 550   AIX 3.2.2,cc -DUNIX -O -Q  Power RISC         42.7        15.5674    1
IBM RS/6000 550   AIX 3.2.3,cc -DUNIX -O     Power RISC         42.7        15.5820   12
IBM RS/6000 550   cc -DUNIX -O               Power RISC         42.7        15.5036   11
IBM RS/6000 560   cc -DUNIX -O -Q            Power RISC         50.0        18.2496    3
IBM RS/6000 950   AIX 3.2.1,                 Power RISC      -------        11.0800   17
IBM RS/6000 980   cc -DUNIX -DROPT -O -Q     Power RISC         62.5        22.6487   14
IBM Value Point   NOTE 038, MS DOS 5.0       486SX/-----        25.0         0.0242   32
Insight 486DX-50  NOTE 026, IBM OS/2 2.1     80486DX            50.0         2.0607   41
SGI 40/20         Irix3.2,cc -DUNIX_Old -O3  R2000              12.5         3.6821   28
SGI 4D/RPC        Irix 4.0.5, cc -DUNIX -O0  R3000/R3010         33.         3.6707   28
SGI 4D/RPC        Irix 4.0.5, cc -DUNIX -O3  R3000/R3010         33.         9.6501   28
SGI Iris 4D/35    Irix 4.0.5, cc -DUNIX -O0  R3000/R3010        36.0         4.1529    7
SGI Iris 4D/35    Irix 4.0.5, cc -DUNIX -O1  R3000/R3010        36.0         9.1449    7
SGI Iris 4D/35    Irix 4.0.5, cc -DUNIX -O2  R3000/R3010        36.0        10.7828    7
SGI Iris 4D/35    Irix 4.0.5, cc -DUNIX -O3  R3000/R3010        36.0        10.7549    7
SGI Iris Indigo   NOTE 015                   R3000/R3010         33.        10.1848   11
SGI Iris Indigo   cc -O3                     R3000/R3010        33.0         9.1272    9
SGI Iris RPC50    cc -O3                     R4000/R4010       100.0        18.0685    9
```

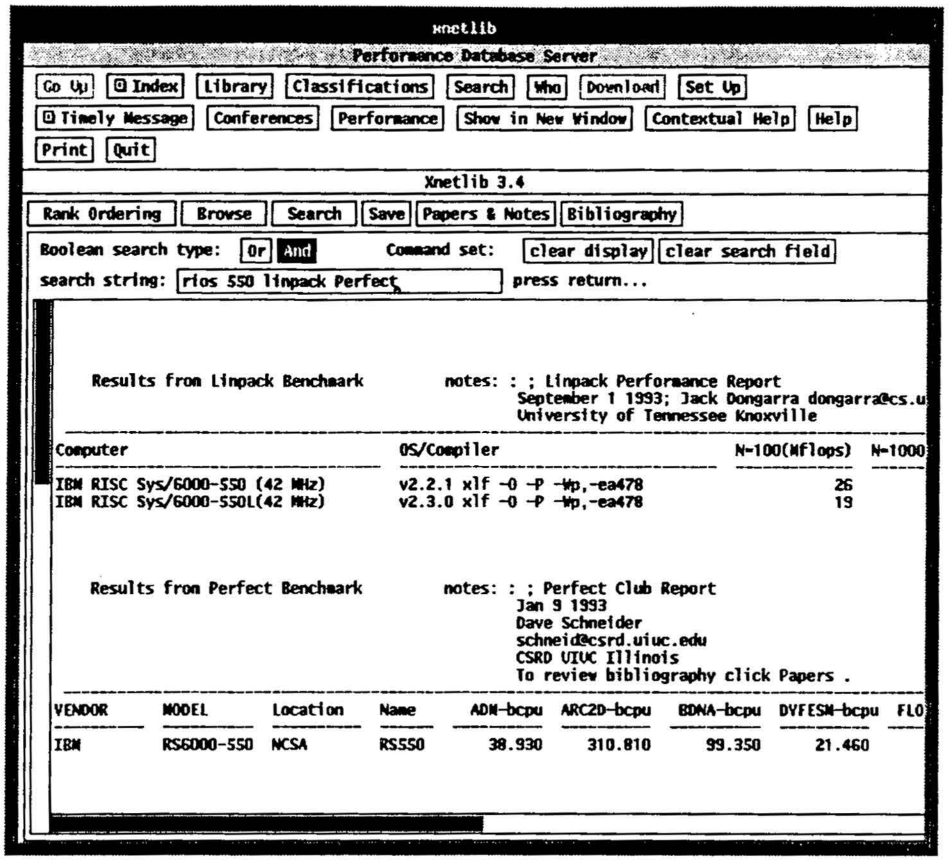
FIGURE 5 Sample data returned by the PDS Browse facility.



```
search string: rios 550 linpack Perfect
```

```
#
#   Performance Database search :
#
#   To search through the performance database please enter a
#   key or series of keywords and Boolean search type.  Type <return>
#   to begin the search.
#
#   clear display - clears the output window and prepares it for a
#                   new search result.
#
#   clear search field - resets the search string to null.
#
#   Search information::
#   truly a literal search over all the data.  By selecting 'and' the
#   return results are ONLY records matching ALL keys.  In selecting
#   'or' the return results are ANY records matching ANY key.
#
# Please NOTE: The search facility allows users to BUILD up query results
#              from numerous queries in the same screen and then save them.
#              To clear the search results select 'clear display'.
#
#
#
```

FIGURE 6 Specifying a keyword search using the PDS Search facility.



```
Boolean search type: Or [And]
search string: rios 550 linpack Perfect

   Results from Linpack Benchmark        notes: : ; Linpack Performance Report
                                                 September 1 1993; Jack Dongarra dongarra@cs.u
                                                 University of Tennessee Knoxville
------------------------------------------------------------------------------------------
Computer                          OS/Compiler                          N=100(Mflops)  N=1000
-------------------------------   ------------------------------       -------------  ------
IBM RISC Sys/6000-550 (42 MHz)    v2.2.1 xlf -O -P -Wp,-ea478                     26
IBM RISC Sys/6000-550L(42 MHz)    v2.3.0 xlf -O -P -Wp,-ea478                     19


   Results from Perfect Benchmark        notes: : ; Perfect Club Report
                                                 Jan 9 1993
                                                 Dave Schneider
                                                 schneid@csrd.uiuc.edu
                                                 CSRD UIUC Illinois
                                                 To review bibliography click Papers .
------------------------------------------------------------------------------------------
VENDOR    MODEL       Location  Name      ADM-bcpu  ARC2D-bcpu  BDNA-bcpu  DYFESM-bcpu  FLO
------    -------     --------  ----      --------  ----------  ---------  -----------  ---
IBM       RS6000-550  NCSA      RS550       38.930     310.810     99.350       21.460
```

FIGURE 7 Results of a keyword search using the PDS Search facility.

As denoted in Figure 3, the Rank Ordering option in PDS allows the user to view a listing of machines that have been ranked by a particular performance metric such as megaflops or elapsed CPU time. Both Rank Ordering and Papers & Notes options are menu-driven data access paths within PDS.

With the Browse facility in PDS (see Fig. 4), the user first selects the vendor(s) and benchmark(s) of interest, then selects the large Process button to query the performance database. The PDS client then opens a socket connection to the server and, using the query language (rdb), remotely queries the database. The format of the returned result is shown in Figure 5. Notice the column headings, which will vary with each benchmark. The returned data is displayed as an ASCII widget with scrollbars when needed.

The Search option in PDS is illustrated in Figures 6 and 7. This feature permits user-specified keyword searches over the entire performance database. Search utilizes literal case-insensitive matching along with a moderate amount of aliasing. Multiple keywords are permitted, and a Boolean flag is provided for more complicated searches. Notice the selection of the Boolean And option in Figure 6. Using Search, the user has the option of entering vendor names, machine aliases, benchmark names, or specific strings, or producing a more complicated Boolean keyword search. The benchmarks returned from the Boolean And search

```
rios 550 linpack Perfect
```

are shown in Figure 7. The alias terms *rios 550* are associated with the IBM RS/6000 Model 550 series of workstations. The specification of *linpack* and *Perfect* will limit the search to the LINPACK and Perfect benchmarks only. Since any retrieved data will be displayed to the screen (by default), the Save option allows the user to store any retrieved performance data in an ASCII file.

Finally, the Bibliography option in PDS provides a list of relevant manuscripts and other information about the benchmarks.

6 SUMMARY AND FUTURE WORK

The PDS provides an on-line catalog of available public domain performance metrics. Built as an extension to Xnetlib. PDS provides multiple views into a dynamic set of data using a simplified user interface. As an objective reporting medium. PDS allows users to make machine comparisons based solely on published benchmarks, which may have a variety of presentation formats.

Future enhancements to PDS include the use of more sophisticated two-dimensional graphical displays for machine comparisons. Additional serial and parallel benchmarks will be added to the database as formal procedures for data acquisition are determined.

To receive Xnetlib with PDS support for Unix-based machines, send the electronic mail message *send xnetlib.shar from xnetlib* to `netlib@ornl.gov`. You can *unshar* the file and compile it by answering the user-prompted questions upon installation. Use of *shar* will install the full functionality of Xnetlib along with the latest PDS client tool. Questions concerning PDS should be sent to `utpds@cs.utk.edu`. The University of Tennessee and Oak Ridge National Laboratory will be responsible for gathering and archiving additional (published) benchmark data.

ACKNOWLEDGMENTS

The authors would like to thank the referees for their helpful comments and suggestions. Research supported by NASA under grant number NAG-5-2083, and by ARPA under grant number DAAL03-92-G-0284.

REFERENCES

[1] R.P. Weicker. "An overview of common benchmarks." *IEEE Comput.*, vol. 23. pp. 65–76. 1990.

[2] D. Bailey. J. Barton. T. Lasinski. and H. Simon. "The NAS parallel benchmarks." Technical Report RNR-91-002. NAS Systems Division. January 1991.

[3] J. Gustafson. D. Rover. S. Elbert. and M. Carter. "Slalom updates." *Supercomput. Rev.*, vol. 4. pp. 56–61. 1991.

[4] J. Dongarra. T. Rowan. and R. Wade. "Software distribution using XNETLIB." *ACM Transactions on Mathematical Software*, to appear.

[5] J.J. Dongarra. *SUPERCOMPUTING. Springer Lecture Notes on Computer Science No. 297.* Berlin: Springer-Verlag. 1988. pp. 456–474.

[6] M. Berry et al., "The perfect club benchmarks: Effective performance evaluation of supercomputers." *Int. J. Supercomput. Appl.*, vol. 3. pp. 5–40. 1989.

[7] J. Uniejewski. "SPEC benchmark suite: Designed for today's advanced systems." SPEC Newsletter. vol. 1. 1989.

[8] M. Berry. G. Cybenko. and J. Larson. "Scientific benchmark characterizations." *Parallel Comput.*, vol. 17. pp. 1173–1194. 1991.

[9] F. McMahon. "The Livermore Fortran kernels: A test of the numerical performance range." Technical Report. Lawrence Livermore Lab. 1986.

[10] D. Bailey and J. Barton. "The NAS kernel benchmark program." Technical Report 86711. NASA Ames Technical Memorandum. 1985.

[11] A. Beguelin. J. Dongarra. G. Geist. R. Manchek. and V. Sunderam. *Proceedings of the Fifth SIAM Conference on Parallel Processing*. Philadelphia. PA: SIAM. 1991. pp. 596–601.

[12] N. Carriero and D. Gelernter. "Linda in Context." *Communications of the ACM*, vol. 32. pp. 444–458. 1989.

[13] W.V. Hobbs. "RDB: A relational database management system." Technical Report. Rand Corporation. December 1991.

[14] L. Wall and R. Schwartz. *Programming Perl*. Sebastopol. CA: O'Reilly and Associates. Inc.. 1990.