PDS: A Performance Database Server

Michael W. Berry
(Corresponding Author)
Department of Computer Science
University of Tennessee
Knoxville, TN 37996
berry@cs.utk.edu
Office: (615) 974-3838, FAX: (615) 974-4404

Jack J. Dongarra[1]
Department of Computer Science
University of Tennessee
Knoxville, TN 37996
Mathematical Sciences Section
Oak Ridge National Laboratory
Oak Ridge, TN 37831
dongarra@cs.utk.edu

Brian H. LaRose[1]
Department of Computer Science
University of Tennessee
Knoxville, TN 37996
larose@cs.utk.edu

Technical Paper Submission

January 27, 1995

[1]Research supported by NASA under grant number NAG-5-2083, and by ARPA under grant number DAAL03-92-G-0284

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) which 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 feel 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.

Keywords: benchmarks, database, DBMS, performance, repository, Xnetlib

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 which can cause leaps 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 ([14]) have been the primary available measures of processor performance. With the advent of parallel benchmarks ([2], [9]), 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., 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 designed to provide easy maintenance, data security, and data integrity in 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 [7], which supports PDS. In Section 3, we discuss four major categories of benchmark 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 performance database server 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 [8] uses three numbers: $n = 100$ Mflops[1], $n = 1000$ Mflops, and theoretical peak performance numbers. The first two numbers reflect the solution of linear systems of order 100 and 1000, respectively. Theoretical peak performance is defined as the absolute upper bound on performance which is enforced by architectural limits.

The Perfect Benchmarks [5] 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 nonconformity 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 (numbers). Graphical representations of tabular data, such as the representation by SPEC [12] 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)

[1]Millions of floating-point operations per second.

software and literature. The Netlib software library 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. [7] in order to provide a more immediate and efficient access to the Netlib software libraries. To date, there have been over 2500 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 in [4], 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.

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 timing some "stress testing" of reading and writing of very large data files. Dhrystones and Whetstones [14] are examples of once-popular synthetic benchmarks which are rarely used today. Dhrystones were designed to stress integer performance and use many string operations. 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 [11] 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 [8], Slalom [9], and the NAS Fortran kernels [1].

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 [5] 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 Figure 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, while others are 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 the growing interest in distributed programming environments such as PVM [3] and Linda [6], we anticipate benchmarks for homogeneous and heterogeneous networks of machines in the near future.

Benchmark Name	Target Machines: Workstations/ Personal Computers	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	

Figure 1: Classification by benchmark use

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. Parsing 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

$$\text{EXTRACT ROW ACCT\_NO} = \text{R103049},$$

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 (*i*) what kind of data is available, (*ii*) how to request/extract the data, and (*iii*) what form to expect the returned data to be in. 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

(1) the ability to extract specific machine and benchmark combinations that are of interest,

(2) the ability to search on multiple keywords across the entire dataset, and

(3) the ability to 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 [10]. This database query language offers several advantages. It is easy to manipulate. It is also efficient, being based on a *perl* model [13] that uses Unix[2] 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 the performance data.

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 pre-existing Xnetlib tool and was extended to provide support for the performance data. Figure 2 illustrates this client-server interface.

The Xnetlib client is a sufficient X-Windows interface that retrieves data from the server. This client is a *view-only* tool which 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, and

(4) a few additional features that aid the user in acquiring benchmark documentation and references.

[2] Unix is a trademark of AT&T Bell Laboratories.

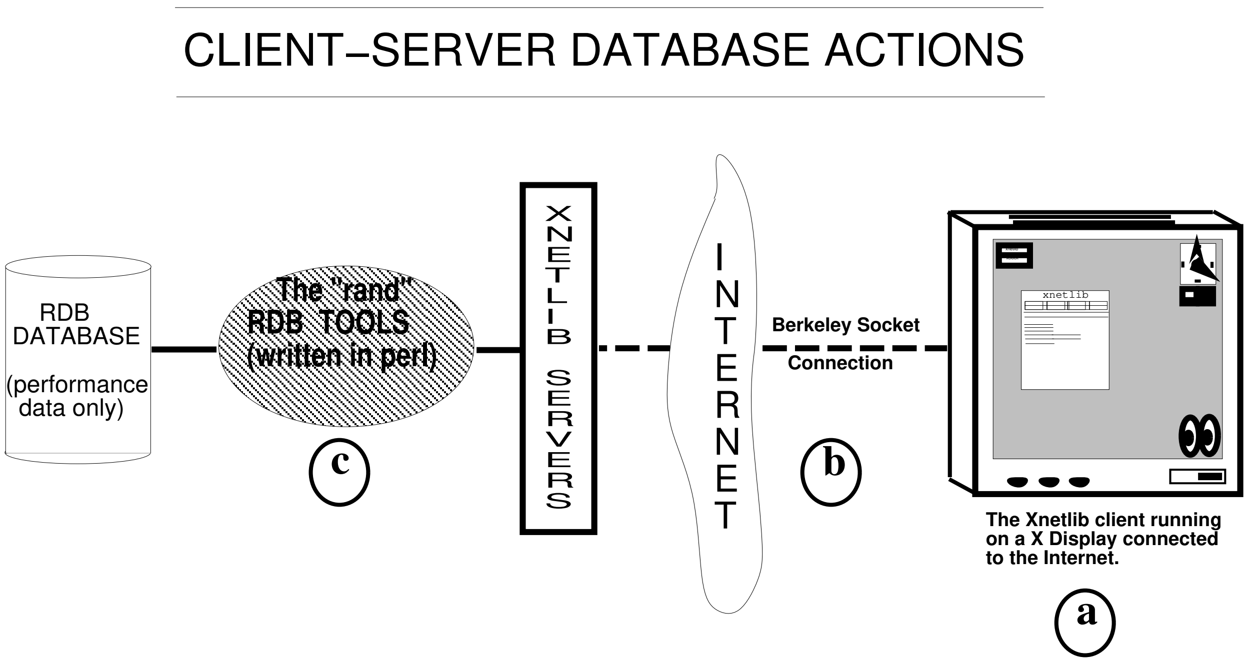

Figure 2: The PDS client-server interface: the X workstation (a) communicates over the Internet via Berkeley socket connection (b) to the Xnetlib server, which queries the database using rdb tools (c) and returns benchmark data via the socket connection.

Figure 3 illustrates the PDS start-up window which 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, Bibliography`).

As denoted in Figure 4, 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` options are menu-driven data access paths within PDS.

With the `Browse` facility in PDS (see Figure 5), 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

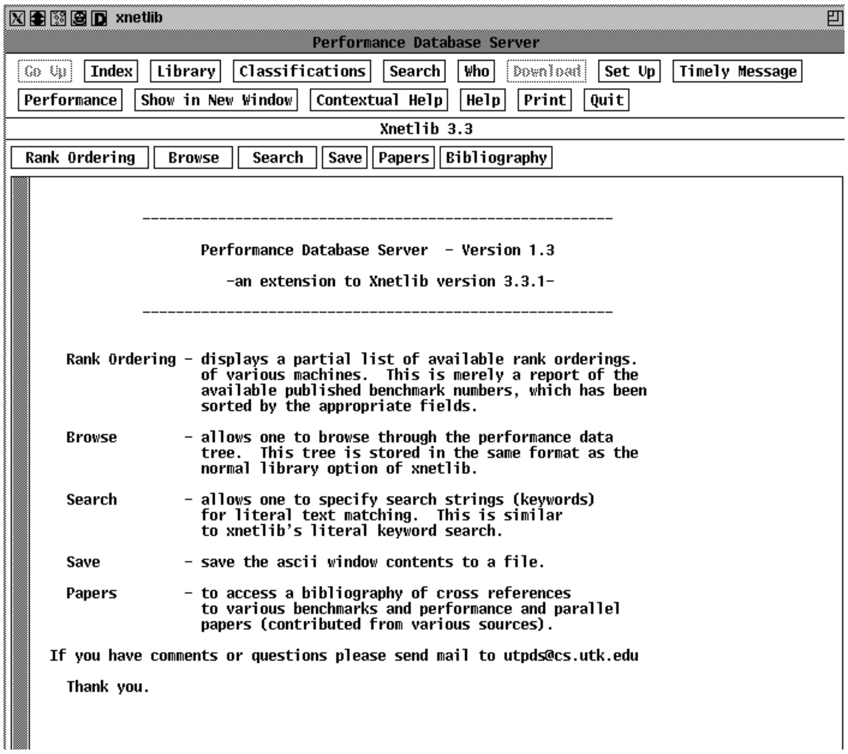

Figure 3: PDS start-up window describing available services

Figure 6. Notice the column headings which will vary with each benchmark. The returned data is displayed into an ASCII widget with scrollbars when needed.

The `Search` option in PDS is illustrated in Figures 7 and 8. 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 7. 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 6000 linpack Perfect

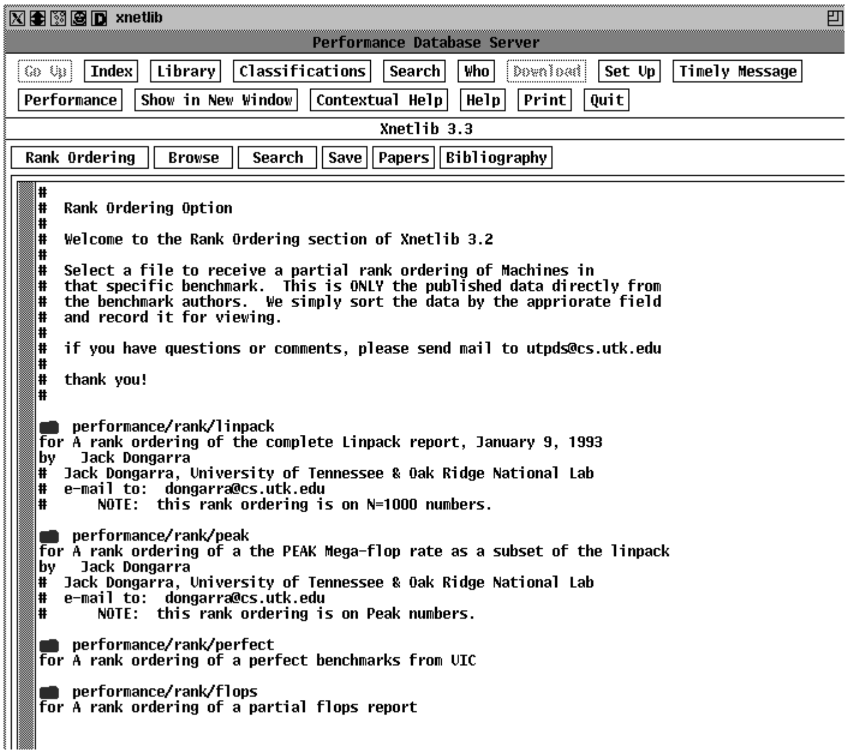

Figure 4: The PDS rank-ordering feature menu

are shown in Figure 8. The alias term *rios* is associated with the IBM RS/6000 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 to 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 Performance Database Server (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 simpli-

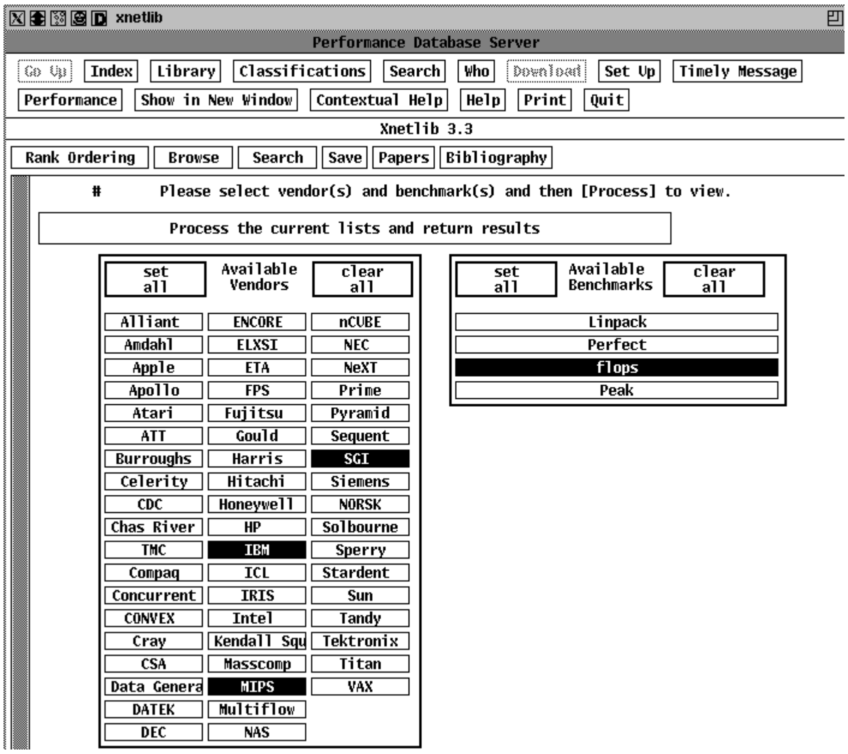

Figure 5: The browse facility provided by PDS

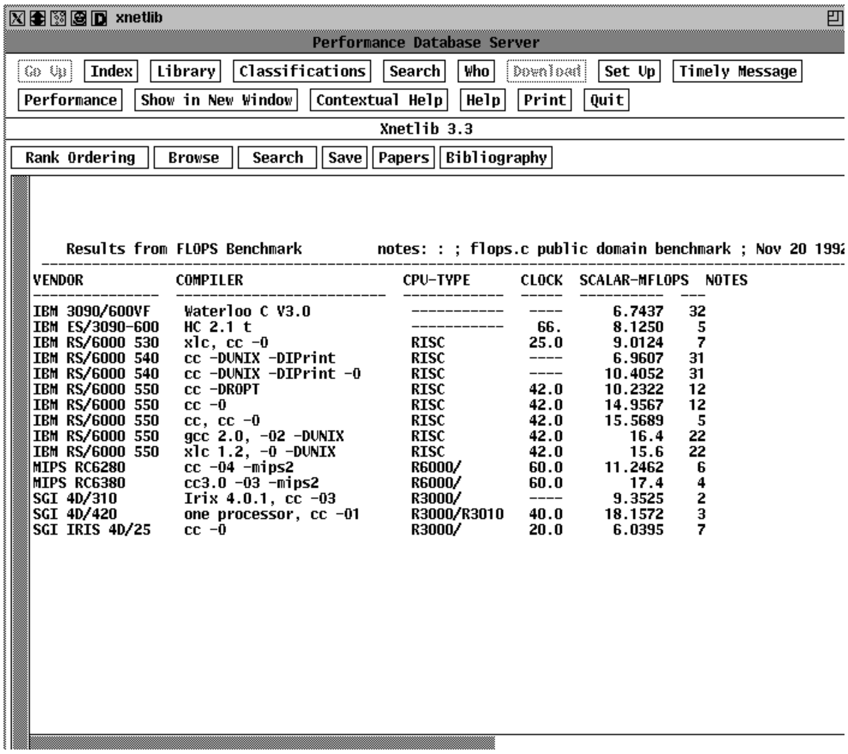

Figure 6: Sample data returned by the PDS `Browse` facility

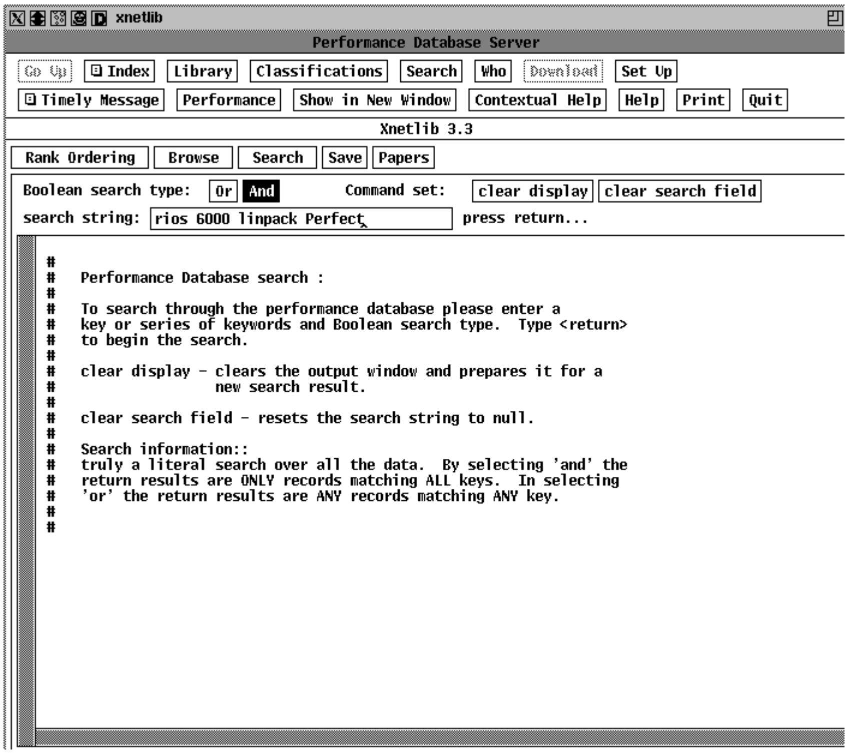

Figure 7: Specifying a keyword search using the PDS `Search` facility

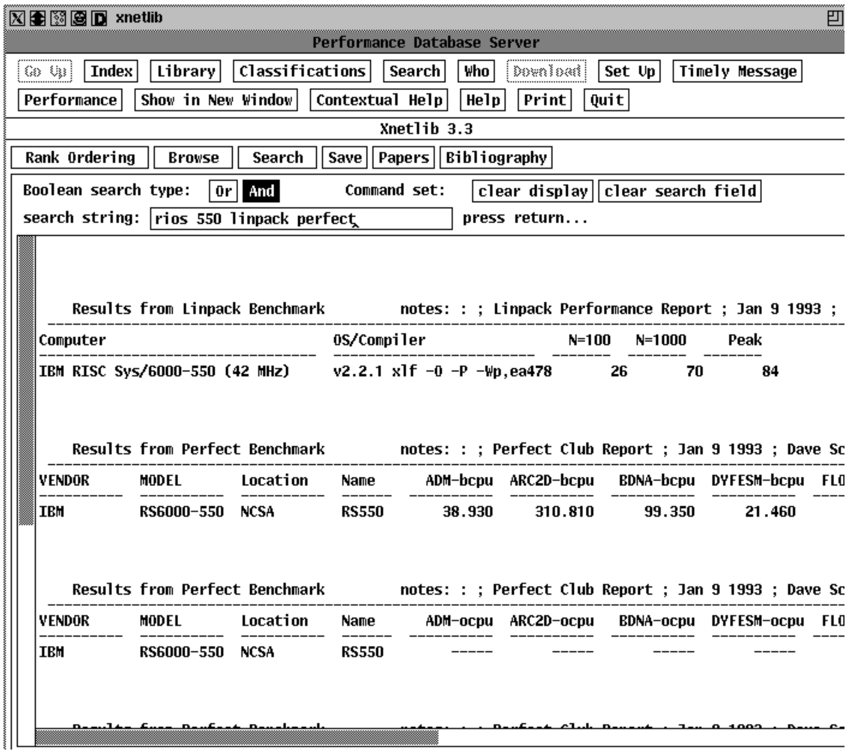

Figure 8: Results of a keyword search using the PDS `Search` facility

fied 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.

References

[1] D. Bailey and J. Barton. The NAS Kernel Benchmark Program. Technical Report 86711, NASA Ames Technical Memorandum, 1985.

[2] D. Bailey et al. The NAS Parallel Benchmarks. Technical Report RNR-91-002, NAS Systems Division, January 1991.

[3] A Beguelin, J Dongarra, G Geist, R Manchek, and V Sunderam. Solving computational Grand Challenges using a network of supercomputers. In D Sorensen, editor, *Proceedings of the Fifth SIAM Conference on Parallel Processing*. Philadelphia, PA: SIAM, 1991.

[4] M. Berry, G. Cybenko, and J. Larson. Scientific benchmark characterizations. *Parallel Computing*, 17:1173–1194, 1991.

[5] M. Berry et al. The Perfect Club Benchmarks: Effective Performance Evaluation of Supercomputers. *International Journal of Supercomputer Applications*, 3(3):5–40, Fall 1989.

[6] N Carriero and D Gelernter. Supercomputing out of recycled garbage: Preliminary experience with Piranha. In *Proceedings of the ACM International Conference on Supercomputing*. ACM, 1992.

[7] J. Dongarra, T. Rowan, and R. Wade. Software distribution using XNETLIB. *Communications of the ACM*, 1993. To appear.

[8] Jack J. Dongarra. The LINPACK benchmark: An explanation. In E.N. Houstis, T. S. Papatheodorou, and C. D. Polychronopolous, editors, *SUPERCOMPUTING, Springer Lecture Notes on Computer Science No. 297*, pages 456–474, 1987.

[9] J. Gustafson et al. Slalom Updates. *Supercomputing Review*, 4(3):56–61, March 1991.

[10] W. V. Hobbs. RDB: A relational database management system. Technical report, Rand Corporation, December 1991.

[11] Frank McMahon. The Livermore Fortran Kernels: A Test of the Numerical Performance Range. Technical report, Lawrence Livermore Lab, 1986.

[12] Joseph Uniejewski. SPEC Benchmark Suite: Designed for Today's Advanced Systems. SPEC Newsletter, Fall 1989. Volume 1, Issue 1.

[13] L. Wall and R. Schwartz. *Programming perl.* O'Reilly and Associates, Inc., Sebastopol, CA, 1990.

[14] R. P. Weicker. An Overview of Common Benchmarks. *IEEE Computer*, pages 65–76, December 1990.