TPC Benchmark™ B Full Disclosure Report Using Sun Microsystems SPARCserver 630MP, SPARCserver 670MP, and SYBASE SQLServer 4.8

Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94043

Revision 1, of September, 1991
Effective September 30, 1991

The Sun logo, Sun Microsystems, and Sun Workstation are registered trademarks of Sun Microsystems, Inc.
SPARC is a registered trademark of SPARC International, Inc. SPARCstation, SPARCware, SPARCcompilers, and SPARCserver are trademarks of SPARC International, Inc. licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based on an architecture developed by Sun Microsystems, Inc. Sun, Sun-2, Sun-3, Sun-4, Sun386*i*, SunInstall, SunView, NeWS, SunOS, DeskSet, SunPHIGS, XGL, DLX, SunNet Manager, and SunCD are trademarks and NFS and PC-NFS are registered trademarks of Sun Microsystems, Inc.
TCP-B Benchmark™ is a trademark of the Transaction Processing Performance Council.
SYBASE SQL Server 4.8 is a trademark of Sybase, Inc.
OPEN LOOK and UNIX are registered trademarks of UNIX System Laboratories, Inc.
All other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations.

Copyright © 1991 Sun Microsystems, Inc. - Printed in U.S.A.

Restricted rights legend: use, duplication, or disclosure by the U.S. government is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 52.227-7013 and in similar clauses in the FAR and NASA FAR Supplement.

The Sun Graphical User Interface was developed by Sun Microsystems Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface which license also covers Sun's licensees.

Abstract

This report documents the compliance of the Sun TPC Benchmark B tests on the SPARC Server 630MP and SPARC Server 670MP systems, running SYBASE SQLServer 4.8, with the TPC Benchmark B Standard Specification.

Two standard metrics, transactions per second (TPS) and price per TPS, are reported. Throughout this report, TPS refers to the tps B performance metric, in accordance with the TPC Benchmark B Standard.

Company Name	System Name	Database Software	Operating System
Sun Microsystems	SPARCserver 630MP	SYBASE SQLServer 4.8	SunOS 4.1.2 SunDBE 1.2

Total System Cost	TPC-B Throughput*	Price Performance
-Hardware -Software -5 year Maintenance	Sustained maximum throughput of system running TPC Benchmark B expressed in transactions per second	Total system cost / TPC-B throughput ($252812 / 121.49 tps-B)
$252812	121.49 tps-B	$2081 per tps-B

Company Name	System Name	Database Software	Operating System
Sun Microsystems	SPARCserver 670MP	SYBASE SQLServer 4.8	SunOS 4.1.2 SunDBE 1.2

Total System Cost	TPC-B Throughput*	Price Performance
-Hardware -Software -5 year Maintenance	Sustained maximum throughput of system running TPC Benchmark B expressed in transactions per second	Total system cost / TPC-B throughput ($269949 / 121.49 tps-B)
$269949	121.49 tps-B	$2222 per tps-B

First Printing September, 1991

Sun Microsystems, Inc. believes that the information in this document is accurate as of its publication date. The information in this document is subject to change without notice. Sun Microsystems, Inc. assumes no responsibility for any errors that may appear in this document.

The pricing information in this document is believed to accurately reflect prices in effect on the indicated dates. However, Sun Microsystems, Inc. provides no warranty on the pricing information in this document.

These benchmark runs and results were internally audited by the Database Engineering Group of Sun Microsystems, Inc.

The performance information in this document is for guidance only. System performance is highly dependent on many factors including system hardware, system and user software, and user application characteristics. Customer applications must be carefully evaluated before estimating performance. Sun Microsystems, Inc. does not warrant or represent that a user can or will achieve a similar performance expressed in transactions per second (TPS) or normalized price/performance ($/TPS). No warranty on system performance or price/performance is expressed or implied in this document.

Copyright © 1991 Sun Microsystems, Inc.

All Rights Reserved.

Contents

Preface

This report documents the compliance of the Sun Microsystems TPC Benchmark B testing on the SPARCserver 630MP and 670MP running SYBASE SQLServer 4.8 with the *TPC Benchmark B Standard Specification*[1]. The TPC Benchmark B Standard represents an effort by Sun Microsystems Inc., Sybase Inc., and other members of the Transaction Processing Performance Council (TPC) to create an industry-wide benchmark for evaluating the performance and price/performance of transaction processing systems.

These tests were run using the SYBASE SQLServer 4.8 running with SunOS 4.1.2 and SunDBE 1.2 (the Sun Database Excelerator) on the SPARCserver 630MP and 670MP.

SunDBE 1.2 will be generally available in calendar Q1/1992; Sybase 4.8 on the Sun platform will be generally available at the same time. The SPARCserver 630MP and SPARCserver 670MP products are available for customer shipment in Q4, calendar 1991. The performance results reported here were with pre-release versions of each product; FCS results will be at least as good.

Document Structure

The *TPC Benchmark B Full Disclosure Report is* organized as follows:

- The main body of the document lists each item in Clause 10 of the TPC Benchmark B Standard and explains how each specification is satisfied.
- Appendix A contains the Open Client and C source code of the application program used to implement the TPC Benchmark B transaction.
- Appendix B contains the definition of the stored procedures used in the tests.
- Appendix C contains the database definition files used in the tests.
- Appendix D contains code for the program used to populate the database as well as sample rows from the respective tables.
- Appendix E contains code to show how the driver program generates and records the transactions.

---

1. TPC Benchmark B Standard Specification, Transaction Processing Performance Council, August, 1990.

- Appendix F lists the parameter settings used to set up the operating system, application, and database for the tests.

Additional Copies

To request additional copies of this report, write to the following address:

TPC Benchmark Reports

Sun Microsystems, Inc.

2550 Garcia Ave.

Mountain View, CA 94043

Attn: Karen Galatis

Mailstop: PAL1-324

Or, copies may be ordered from the administrator of the TPC:

Shanley P.R.

777 N First Street, Suite 600

San Jose, CA 95112-6311

(408) 295-8894

FAX (408) 295-2613

TPC Benchmark™ B Full Disclosure

The *TPC Benchmark B Standard Specification* requires test sponsors to publish, and make available to the public, a full disclosure report for the results to be considered compliant with the Standard. The required contents of the full disclosure report are specified in Clause 10.

This report is intended to satisfy the Standard's requirement for full disclosure: it documents the compliance of the benchmark tests reported in the *TPC Benchmark B Results for the Sun SPARCserver 630MP and 670MP, running the SYBASE 4.8 SQLServer.*

In the *Standard Specification*, the main headings in Clause 10 are keyed to the other clauses. The headings in this report use the same sequence, so that they correspond to the titles or subjects referred to in Clause 10.

Each section in this report begins with the text of the corresponding item from Clause 10 of the *Standard Specification*, printed in italic type. The plain type text that follows explains how the tests comply with the TPC Benchmark B requirement. In sections where Clause 10 requires extensive listings, the section refers to the appropriate appendix at the end of this report.

1 - General Items

1.1 Sponsor

A statement identifying the sponsor of the benchmark and any other companies who have participated.

This benchmark test was sponsored by Sun Microsystems, Inc. and Sybase, Inc.

1.2 Application Code and Definition Statements

- *Program listing of application code and definition language statements for files / tables.*

Appendix A contains the Open Client and C source code of the application program used to implement the TPC Benchmark B transaction; Appendix B contains the definition of the stored procedures used in the tests; Appendix C contains the database definition files used in the tests; Appendix D contains the program used to populate the database as well as sample rows from the respective tables; Appendix E contains code to show how the driver program generates and records the transactions; Appendix F lists the parameter settings used to set up the database for the tests.

1.3 Parameter Settings

Settings for all customer-tunable parameters and options that have been changed from the defaults found in actual products; including but not limited to:

- Database options
- Recovery /commit options
- Consistency/locking options
- System parameters, application parameters, and configuration parameters

Test sponsors may optionally provide a full list of all parameters and options.

Sun has elected to provide a listing of all parameters and options.

Appendix F contains the database configuration parameters used in the TPC Benchmark B tests.

1.4 Configuration Diagrams

Configuration diagrams of both benchmark configuration and the priced system, and a description of the differences.

Configuration Items for the SPARCserver 630MP

In both the priced and tested systems, a single SPARCstation ELC was used to run the driver software and the database client sessions. It had 20MB of memory. A single SPARCserver 630MP acted as the DBMS server node as well as the boot node for the ELC (known in Sun parlance as a diskless client). The SPARCserver 630MP was connected to the client node via ethernet. It had 128MB of memory, used 2 SCSI Host Adaptors (plus the internal Host Adaptor) and a total of 8 1.3 GB SCSI disks, with 1 669MB disk, used to store 1/6 of the history tables. We configured the SUT configuration this way because 30 days of history table storage require use of the extra disk. The priced system was identical except that we priced the more expensive, and better performing, 1.3 GB disk, as Sun no longer offers the 669MB disks on our price list. Sybase SQLServer version 4.8 was used as the dataserver with OpenClient on the SPARCstation ELC.

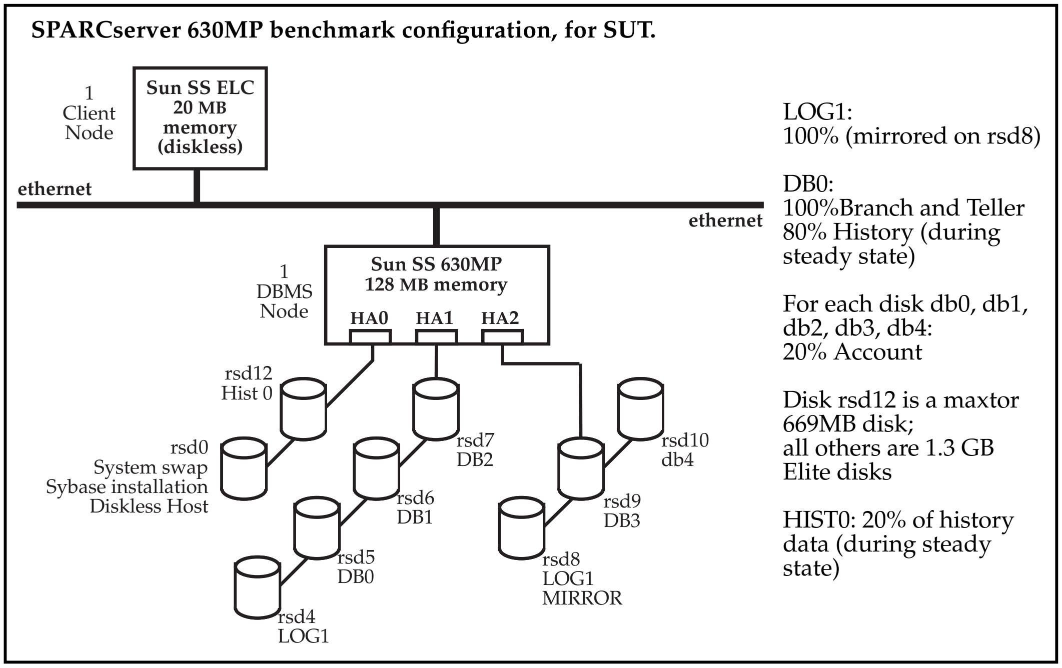

Figure 1-1 The Sun SPARCserver 630MP benchmark configuration, for SUT.

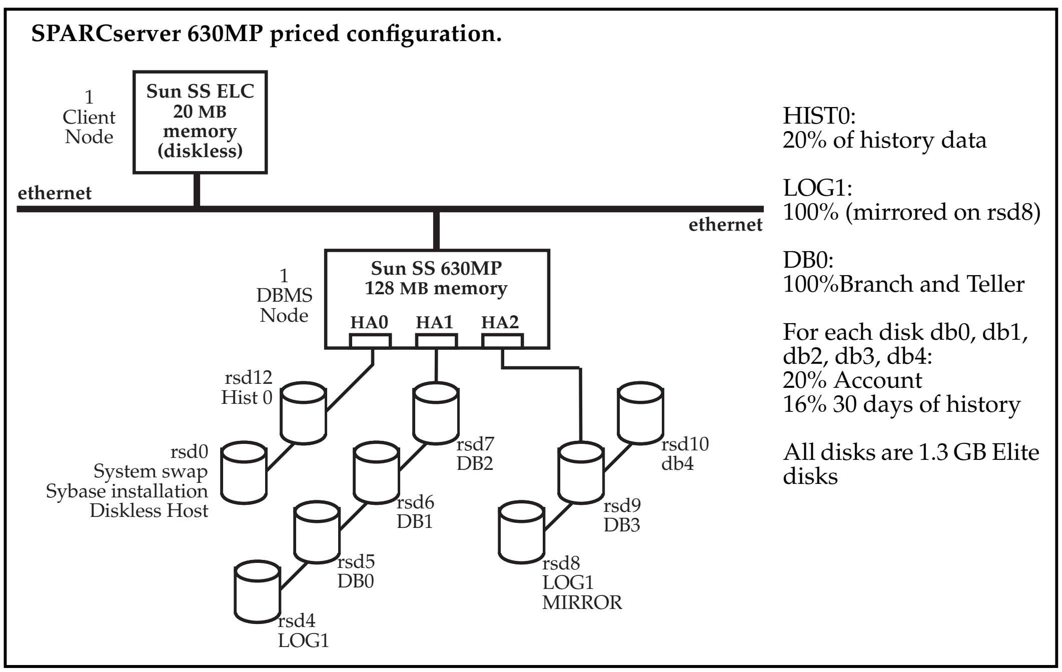

Figure 1-2 The Sun SPARCserver 630MP priced configuration.

The tested configuration for the SPARCserver 670MP was identical in every respect to that used for the SPARCserver 630MP runs. For this reason, we made use of the same TPS-B figure for the SPARCserver 670MP. Note that the SPARCserver 670MP is the same as the SPARCserver 630MP (same CPU, memory, disks, busses, etc.) except that it has a larger chassis to hold more VME boards. We provide separate SPARCserver 670MP pricing, and a separate cost/TPS figure, as shown in Section 9 of this Full Disclosure Report.

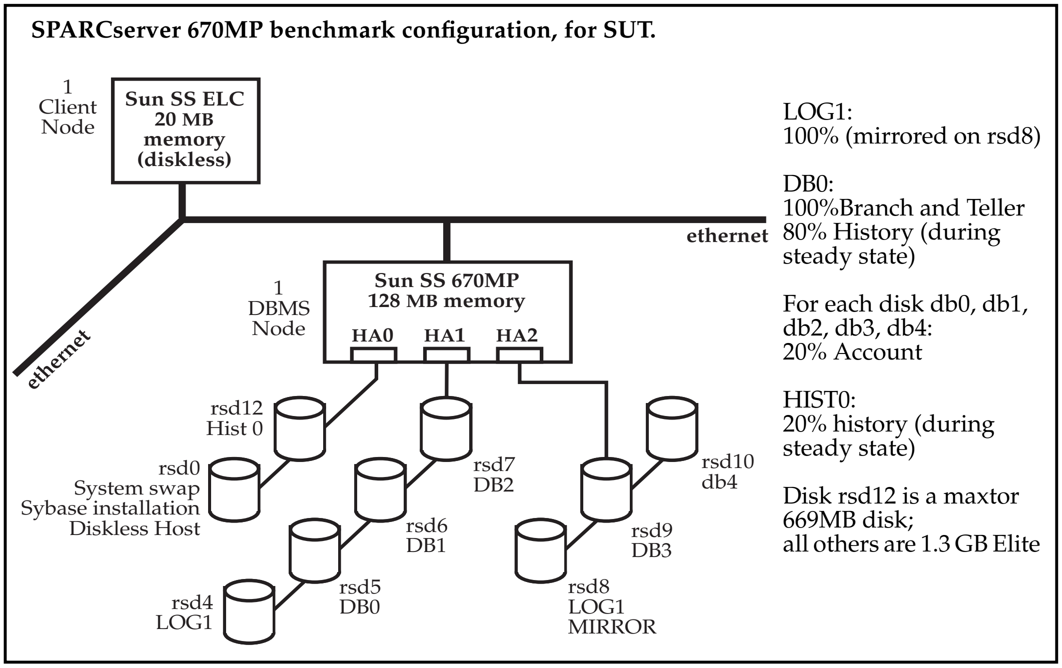

Figure 1-3 The Sun SPARCserver 670MP benchmark configuration, for SUT.

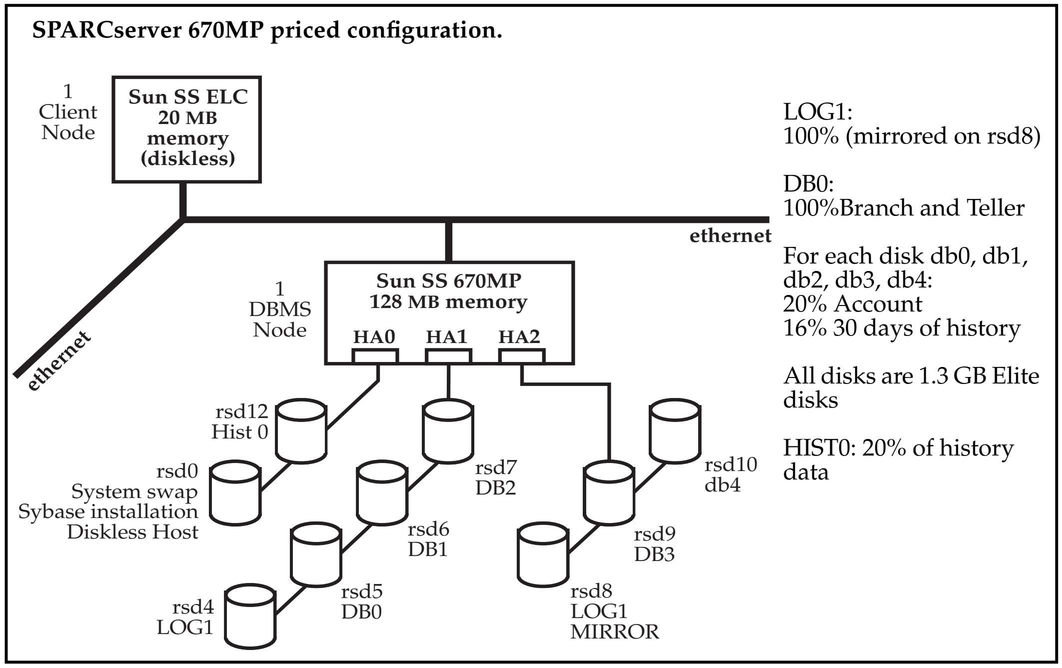

Figure 1-4 The Sun SPARCserver 670MP priced configuration.

2 - Clause 2 Related Items

ACID Properties

Results of the ACIDity test (specified in Clause 2) must describe how the requirements were met.

Clause 2 of the *TPC Benchmark B Standard Specification* lists specific tests to ensure the atomicity, consistency, isolation, and durability (ACID) properties of the SUT. The following subsections show how the tests required in Clause 2 were performed. All mechanisms needed to insure full ACID properties were enabled during both the measurement and test periods.

Note – The fully-scaled database was used to conduct all atomicity consistency, and isolation tests, as required by the TPC-B standard (clause 2).

2.1 Atomicity Tests

Atomicity of Completed Transaction

Perform the standard TPC Benchmark B transaction for a randomly selected account and verify that the appropriate records have been changed in the Account, Branch, Teller, and History files / tables.

The following test was performed, and verified the atomicity of completed transactions:

1. Select a random Branch record.
2. Select a random Teller record.
3. Select a random Account record.
4. For the selected Account record, count the History records and sum their delta values.
5. Using the randomly selected records, perform the following steps:
   a. Update the Branch record.
   b. Update the Teller record.
   c. Update the Account record.
   d. Insert the History record.
   e. Commit the transaction.
   f. Select the Branch record.
   g. Select the Teller record.
   h. Select the Account record.
6. For the selected Account record, count the History records and sum their delta values. Verify that the History record count and delta sum reflect the committed transaction.

Atomicity of Aborted Transaction

Perform the standard TPC Benchmark B transaction for a randomly selected account, substituting an ABORT of the transaction account for the COMMIT of the transaction. Verify that the appropriate records have not been changed in the Account, Branch, Teller, and History files /tables.

The following test was performed, and verified the atomicity of aborted transactions:

1. Select a random Branch record.
2. Select a random Teller record.
3. Select a random Account record.
4. For the selected Account record, count the History records and sum their delta values.
5. Using the randomly selected Branch, Teller and Account records from above, do the following:
   a. Update the Branch record.
   b. Update the Teller record.
   c. Update the Account record.
   d. Insert the History record.

   e. Abort the transaction and perform a rollback recovery.
   f. Select the Branch record.
   g. Select the Teller record.
   h. Select the Account record.
6. For the selected Account record, count the History records and sum their delta values. Verify that the History record count and delta sum have not changed.

2.2 Consistency Tests

Consistency is the property of the application that requires any execution of a transaction to take the database from one consistent state to another.

The following tests were performed, and verified the consistency property of transactions:

1. Consistency of Branch and Teller records before transaction
   a. Select Branch balances for each Branch record.
   b. Select the sum of Teller balances for each Branch record.
   c. Verify that the balance for each Branch record is equal to the balance of its Teller records.
2. Consistency of Branch and Teller records after the required number of transactions on the SPARCserver 630MP and 670MP.
   a. For the entire History file, count the History records and sum their delta values.
   b. Perform the standard TPC Benchmark B test and record the number of committed transactions.
   c. Repeat step 1.
3. Consistency of History files
   a. For the entire History file, count the History records and sum their delta values.
   b. Verify that this History record count equals the sum of History record count taken in step 2A plus the number of committed transactions.
   c. Verify that the difference between the final History delta sum and the initial History delta sum equals the difference between the final and initial Branch record balances.

2.3 Isolation Tests

Operations of concurrent transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed to completion in some order.

The following tests were performed, and verified the isolation property of the transactions for conventional locking used by the database system:

Isolation of Completed Transactions

1. Select the Branch balance for a Branch record (Branch B).
2. Start transaction 1.
   a. Update the Branch B record with delta(1).
   b. Stop just prior to committing transaction 1.
3. Start transaction 2.
   a. Attempt to update Branch B with delta(2).
   b. Transaction 2 hangs.
4. Resume transaction 1.
   a. Update the Teller record.
   b. Update the Account record.
   c. Insert the History record.
   d. Commit transaction 1.
5. Resume transaction 2.
   a. Update the Teller record.
   b. Update the Account record.
   c. Insert the History record.
   d. Commit transaction 2.
6. Select the Branch balance for Branch B. The balance should equal the previous balance plus delta(l) and delta(2).

Isolation of Aborted Transactions

1. Start transaction 1.
   a. Update Branch B with delta(3).
   b. Stop just prior to committing transaction 1.
2. Start transaction 2.
   a. Attempt to update Branch B with delta(4).
   b. Transaction 2 hangs.
3. Resume transaction 1.
   a. Update the Teller record.
   b. Update the Account record.
   c. Insert the History record.
   d. Abort transaction 1 and perform a rollback recovery.

4. Resume transaction 2.
    a. Update the Teller record.
    b. Update the Account record.
    c. Insert the History record.
    d. Commit transaction 2.
5. Select the Branch balance for Branch B. The balance should equal the previous balance plus delta(4).

The preceding isolation tests were repeated for Teller and Account files (that is, by generating a conflict on a Teller and then an Account record).

2.4 Durability Tests

The tested system must guarantee the ability to preserve the effects of committed transactions and ensure database consistency after recovery from any one of the failures listed below:

For these tests only, a two TPS database was used, with log mirroring and concatentation enabled, as in the fully scaled configuration. Note this is explicitly permitted by the TPC-B Standard (Clause 2.5.6) in order to allow recovery times to be reasonable. To the best of the test sponsors' knowledge, a fully loaded and fully scaled database would also pass the Durability tests.

Four types of failure were induced in the SUT in order to demonstrate the requisite durability characteristics. These were:

1. Instantaneous interruption of system power/system crash. On the systems benchmarked, this also resulted in loss of system volatile memory.
2. Loss of the disk on which the SYBASE SQL 'master' database resides.
3. Loss of a disk on which the 'bench' database, or any part of it, resides.
4. Loss of the log device.

The method of recovery varied according to the type of failure:

1. Power up system, start dataserver. SYBASE automatically rolls forward all committed transactions, and rolls back uncommitted transactions, using its transaction log.
2. Replace bad disk, power up system, build default master DB from scratch, restore master from backup, start dataserver. Apply transaction logs; SYBASE automatically rolls forward any committed transactions, and rolls back uncommitted transactions.
3. Replace bad disk, power up system, start dataserver. Drop and recreate the 'bench' database as a new, empty database. Restore the database from backup, start dataserver. Apply transaction logs; SYBASE automatically rolls forward any committed transactions, and rolls back uncommitted transactions.

4. As soon as SYBASE detects a failure in writing to the log device, (or reading from it upon system startup), it automatically switches to the log mirror device. Further log writes or reads then occur only from the log mirror. (If the log mirror disk failed instead, then only the log device is used).

During each of the durability tests, we first cleared the History tables. Then three clients were started on a remote node, communicating over Ethernet with the database server. As each client completed a transaction, it wrote a record to its own local 'success' file which included the account, branch, teller, and delta for the transaction. After at least 1000 transactions had committed, we induced on the server node one of the failures listed above. The clients detected this and immediately closed their respective 'success' files.

On the server node, we then recovered the database using the appropriate method, as described above. We counted the number of committed transactions in the History tables. In each case, this number was greater than or equal to the number of transactions seen as committed by the clients in their 'success' files. We also verified the required consistency test on the Branch and Teller tables.

3 - Clause 3 Related Items

3.1 ABTH Data Storage Distribution

The distribution across storage media of ABTH (Accounts, Branch, Teller, and History) files / tables and all logs must be explicitly depicted.

See Figures 1-1, 1-2, 1-3, and 1-4.

3.2 Database Contents and Method of Population

A description of how the database was populated, along with sample contents of each ABTH file / table to meet the requirements described in Clause 3.

Database Contents

Appendix D contains the database population program and samples of the contents of the database files used in the tests.

3.3 Type of Database

A statement of the type of database utilized, e.g., relational, Codasyl, flat file, etc.

These TPC Benchmark B tests used the SYBASE SQLServer 4.8 relational database software.

4 - Clause 4 Related Items

There are no Clause 4 related items in the check-list for TPC-B.

5 - Clause 5 Related Items

5.1 Method of Verification of Random Number Generator

The method of verification of the random number generator should be described.

Random numbers used in the driver to choose accounts, branches, tellers, and deltas were generated using the random number generator described and documented in the article, "Random Number Generators: Good Ones are Hard to Find", Communications of the ACM, October, 1988, Volume 31, Number 10. The properties of this random number generator are well-known and are documented in the article as producing a uniformly distributed pseudo-random sequence.

To generate a random number, the driver programs first use a seed based on time-of-day and the process id of the respective client session. This guarantees that each run and each client, is mathematically independent of previous runs.

The history records of the entire steady-state period, for all branches, were examined and shown to satisfy the TPC-B requirement that approximately 15% of the accounts were from a different branch.

5.2 Use of Horizontally Partitioned History Tables

We used a total of 32 history tables, all identical other than name, a strategy that is permitted by the TPC-B standard (see Clause 5.2.1). On each transaction, each of the 48 drivers selects one of these history tables as a modulus function of the randomly chosen account number. Thus, each driver selects a new history table for each transaction. Note that the client system is fully priced as part of the SUT.

6 - Clause 6 Related Items

Report all the data specified in Clause 6.6, including maximum and average residence time, as well as performance curves for number of transactions vs. residence time (see clause 6.6.1) and throughout vs. level of concurrency for three data points (see clause 6.6.5).

The graphs and tables in this section show the residence time performance results.

6.1 Residence Time Versus TPS for the Sun SPARCserver 630MP and 670MP

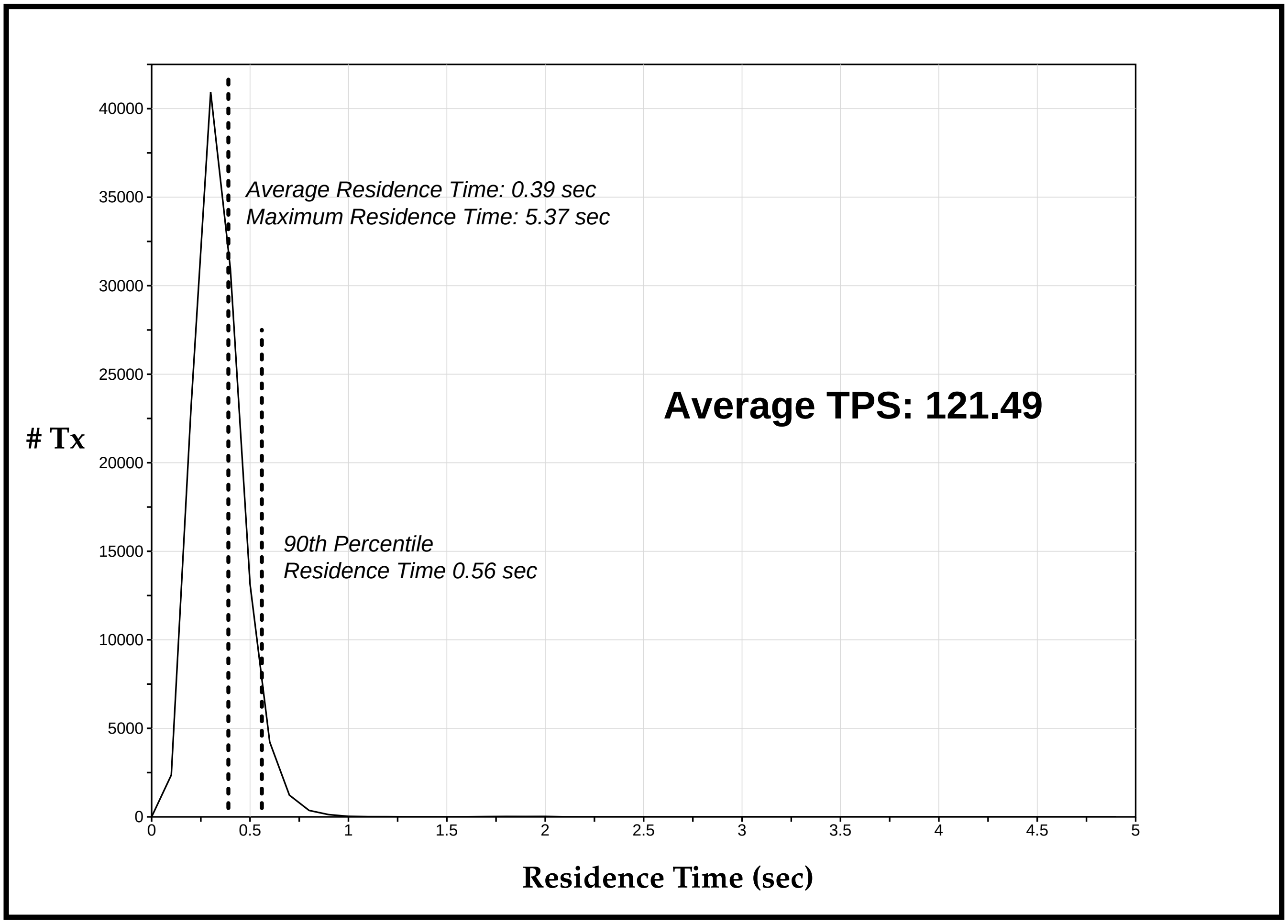

Figure 1-5 Residence Time Distribution (SYBASE SQLServer on SPARCserver 630MP and 670MP)

Remote Transactions (see Clause 6.6.2):	15%
Home Transactions:	85%
Transactions started and not completed during measurement interval (see Clause 6.6.3):	0.04%

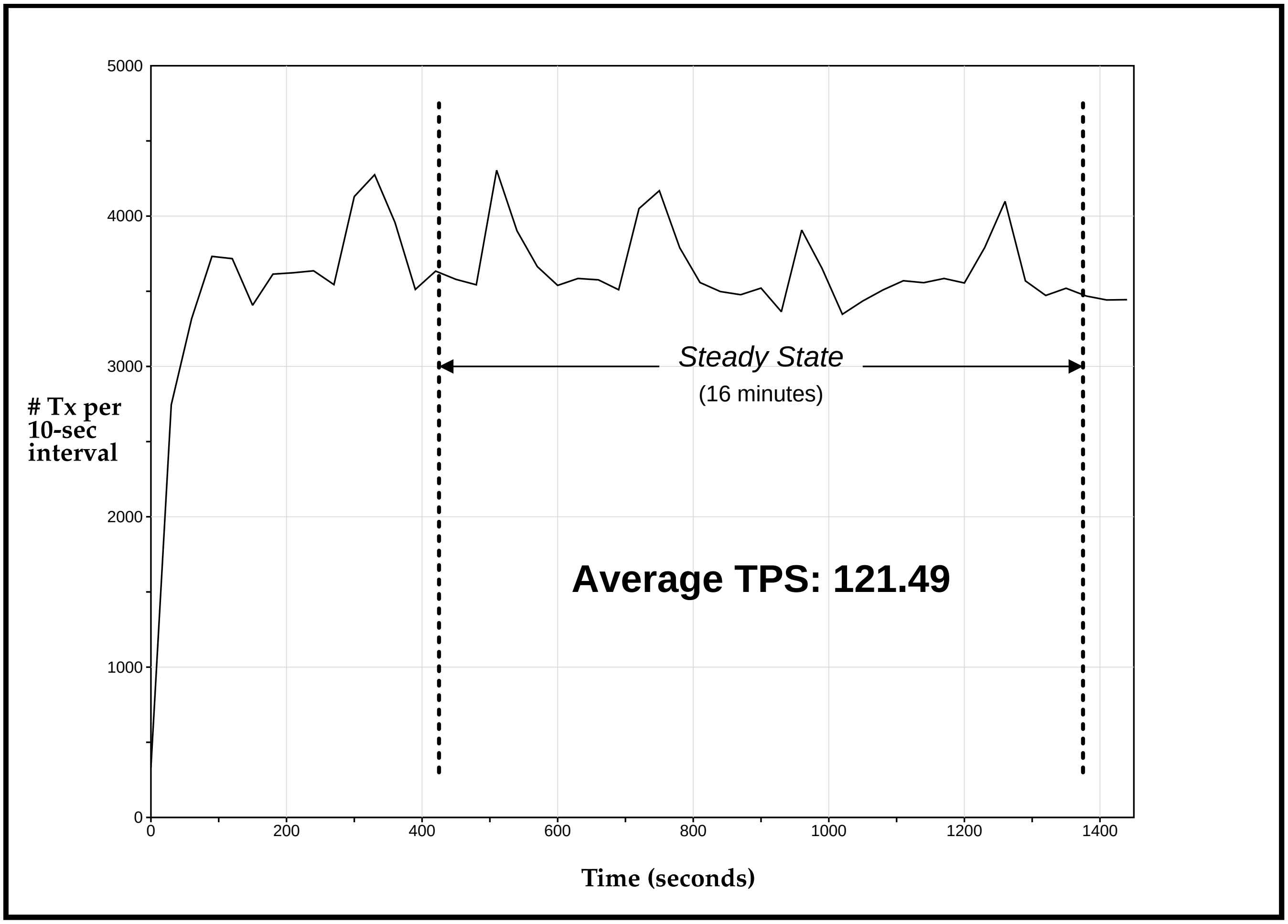

Figure 1-6 Steady State Transaction Rates (SYBASE SQLServer on SPARCserver 630MP and 670MP)

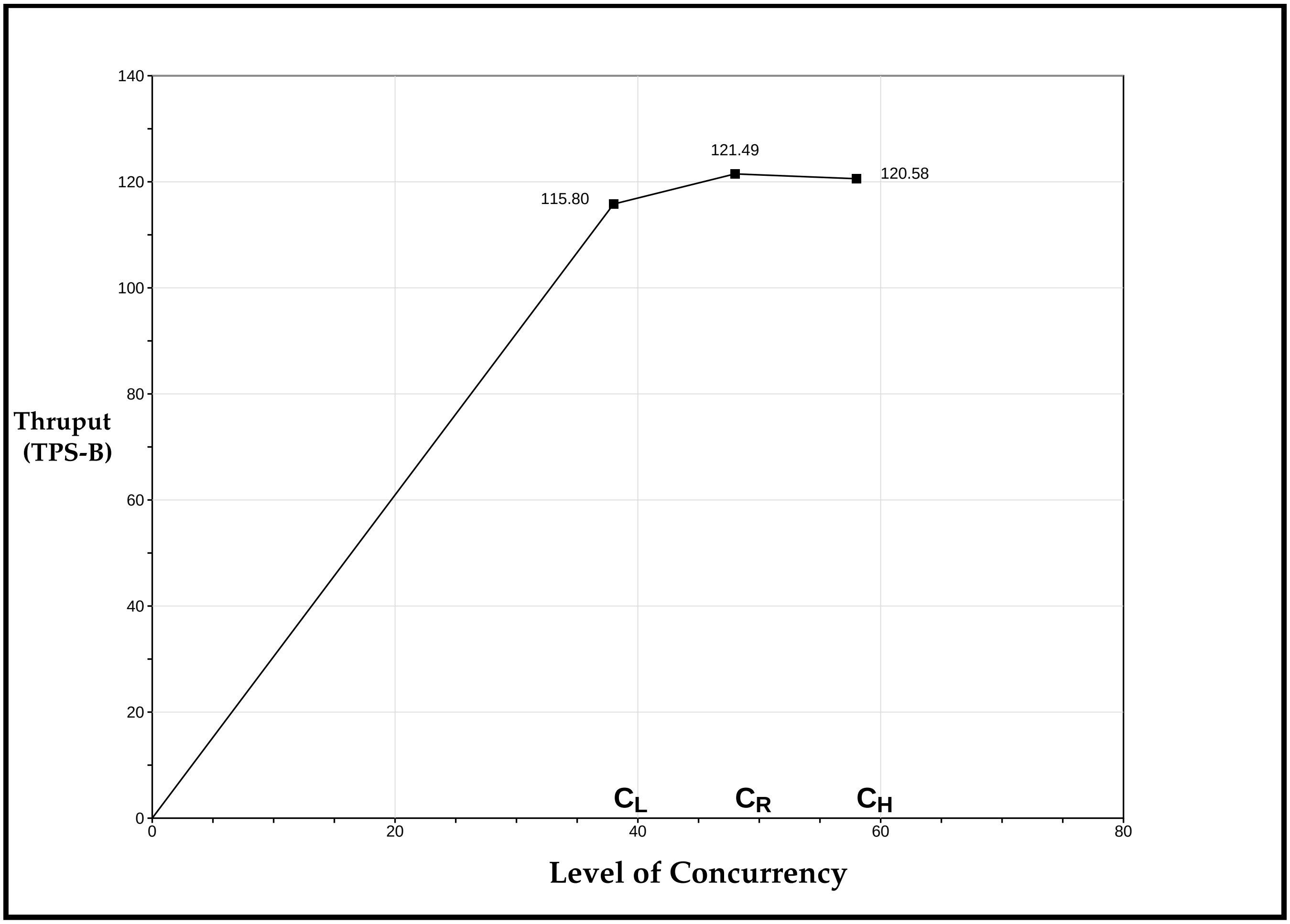

Figure 1-7 TPS vs. Concurrency (SYBASE SQLServer on SPARCserver 630MP and 670MP)

Note:

C_R is 48 clients

C_L is 38 clients

C_H is 58 clients

For all these benchmark runs, the database was scaled for 122.0 TPS.

7 - Clause 7 Related Items

7.1 Determining Steady State

The method used to determine that the SUT had reached a steady state prior to commencing the measurement interval should be described.

Confirmation that the system is in steady state during the timed measurement interval is determined by design as well as by inspection.

The design of the benchmark driver program is such that all clients wait on a master program to signal them to commence ramp-up work. It does this only when it detects that all clients are connected to the SQLserver and ready to do work. During the ramp-up, which lasts at least 420 seconds, all clients begin executing the identical TPC-B transaction that they do during the timed steady state run. At the end of the ramp-up period (which is the beginning of the steady state interval), each client independently begins keeping track of the number and characteristics of its committed transactions. The audited benchmark steady state period lasts for 16 minutes. At the end of the steady state period, each client continues executing TPC-B transactions for another 180 seconds, but does not include them in its steady-state numbers. When the run completes, the clients individually and independently report to the master their accumulated transaction and residence time results. The driver then calculates the required aggregate numbers to report.

In order to confirm by inspection that true steady-state was indeed reached, the history table was sampled before and after the indicated steady state interval, to demonstrate that the average TPS rate during ramp-up and ramp-down is statistically identical to the average rate during the steady state interval. The graph shown in Figure 1-6 indicates the number of transactions completed in each 30-second interval of the respective benchmark runs. This includes ramp-up, steady-state, and ramp-down. The steady-state period (16 minutes) is indicated on the graphs. Note the dips in transaction rate that occur during steady-state for each run; these are the database checkpoints.

SPARCserver 630MP and 670MP log size calculations

Clause 7.1 requires that the system be configured so that it is possible to run the test at the reported TPS-B rate for a continuous period of at least eight hours.

We observed that the log disk (and thus the mirror of the log too) was being written to at a rate of 21 writes per second, each of 2048 bytes in size. For sake of conservative estimates, we computed log size using 23 writes/sec. Thus, an average of 5.28 transactions committed in each log write. In order to run for 8 hours, we needed (2*2048*60*60*8)/5.28)*121.49 bytes of log and mirror space, or 1.294 GB for each. The log and mirror disks are each large enough to hold this quantity of log data on a single spindle. So we configured one 1.304 GB disk for the DB log, and one 1.304 GB disk for the mirror of the DB log. As required by the TPC-B standard, the media used to store the 8 hours of log data was entirely configured on the SUT system.

7.2 Work Performed During Steady State

A description of how the work normally performed during a sustained test (for example, checkpointing, writing redo / undo log records, etc., as required by Clause 7.2), actually occurred during the measurement interval.

During steady state, each TPC-B transaction results in there being, on average, one disk read of an account page from the database, one write of an independent, updated, account page to the database, and a log write (which is likely part of a group commit). At least once during steady state, the SQLserver performs a checkpoint, during which all updated buffers (branch, teller, account, history) are written to disk. See Appendix F for a discussion of the impact on transaction performance of the buildmaster ‘cbufwashsize’ parameter.

7.3 Determining Reproducibility

A description of the method used to determine the reproducibility of the measurement results.

Each experiment at the maximum targeted TPS level was repeated at least three times to ensure reproducibility. The computed TPS for each experiment was within 2% of the reported TPS, and the 90th percentile residence time was under two seconds.

7.4 Duration of Measurement Period

A statement of the duration of the measurement period for the reported tps (it should be at least 15 minutes and no longer than one hour).

Each experiment was run for at least 26 minutes. The duration of the steady state period was 16 minutes, preceded by a 7 minute ramp-up period and followed by a 3 minute ramp-down period.

8 - Clause 8 Related Items

8.1 Description of RTE

If the RTE is commercially available, then its inputs should be specified. Otherwise, a description should be supplied of what inputs (e.g., scripts) to the RTE were used.

The RTE used was jointly developed by Sybase, Inc. and Sun Microsystems Inc. It is proprietary. Inputs to the RTE include the number of clients to initiate, how long they are to run, sleep time (if any), TPS scaling, times for ramp up and ramp down, the type of transaction workload to run, and a number of other parameters. Appendix A contains the entirety of the transaction portion of the driver software that implements the TPC-B transaction. Appendix E contains the Open Client code for the section of the RTE used to control transaction submission and timing during ramp up, steady state, and ramp down.

8.2 Network Bandwidth

If the SUT contains a WAN or a LAN network, its bandwidth should be specified.

The Ethernet used in the LAN complies with the IEEE 802.3 standard and has a bandwidth of 10 mbps.

9 - Clause 9 Related Items

9.1 Hardware and Software Components

A detailed list of hardware and software used in the priced system. Each item must have vendor part number, description, and release / revision level, and either general availability status or committed delivery date. If package pricing is used, contents of the package must be disclosed.

9.1.1 Priced System Configuration Tables for the Sun SPARCserver 630MP and 670MP

Component	Product	Quantity
Server Processor	SPARCserver 630MP	1
Client Processor	SPARCstation ELC	1
Server Memory		128 MB
Client Memory		20 MB
SCSI Host Adapter	SCSI	2
Server Disks	SCSI	9
Client Disks	SCSI	0
Tape drive		1
Operating System	SunOS 4.1.2, SunDBE 1.2	1
Database	SYBASE SQLServer 4.8 + OpenClient	1

Component	Product	Quantity
Server Processor	SPARCserver 670MP	1
Client Processor	SPARCstation ELC	1
Server Memory		128 MB
Client Memory		20 MB
SCSI Host Adapter	SCSI	2
Server Disks	SCSI	9
Client Disks	SCSI	0
Tape drive		1
Operating System	SunOS 4.1.2, SunDBE 1.2	1
Database	SYBASE SQLServer 4.8 + OpenClient	1

9.1.2 Package Pricing For the Sun SPARCserver 630MP and 670MP

I. SPARCserver 630MP as the server and SPARCstation ELC as the client

Cost summary:	SPARCserver 630MP hardware/software	$122,694
	SPARCserver 630MP hardware/software maintenance	44,824
	SPARCstation ELC hardware/software	10,195
	SPARCstation ELC maintenance	3,634
		$181,347
	Sybase SQLServer 4.8	21,460
	Sybase SMP Access Fees	2,700
	Sybase ELC OpenClient	2,360
		$26,520
	Sybase maintenance	44,945

Price per TPS ($/tpsB) @121.49 TPS = $2081

Server System:

Description	Order Number	List Price	Monthly Maint	Total* Maint
SPARCsystem 630MP, Model 140 with 4 CPUs, 128 MB ECC memory, 1.3GB SCSI disk, 644 MB SunCD	S630-140-128-P51	67,500	420	23,940
US Country Kit	X311L	0	0	0
150 MB 1/4" internal SCSI tape	660L	1,400	10	570
1.3 GB internal SCSI drive	572A	5,300	50	2,850
Sbus SCSI Card	1054A	1,600	20	1,140
Sbus SCSI Card	1054A	1,600	20	1,140
SCSI Expansion pedestal with 2 1.3GB SCSI drives	X750A	15,995	100	5,700
2 1.3 GB SCSI drives with tray	X756A	10,000	100	5,700
1.3 GB SCSI drive	X572A	5,300	50	2,850
1.3 GB SCSI drive	X572A	5,300	50	2,850
1.3 GB SCSI drive	X572A	5,300	50	2,850
Wyse terminal	WY-50	404	50/yr	200
SunOS 4.1.2 on CD-ROM	SX-21	200	155	8,835
SunOS 4.1.2 documentation of CD	SX-23	495	N/A	N/A
SunOS unlimited users right-to-use	SYSLUN	2,000	N/A	N/A
SunOS Database Excelerator on CD-ROM	DBE-1.2-4-4-21	300	20	1,140
		122,694		59,765
Less 25% for prepaid Sun maintenance				14,941
				44,824

Client Driver System:

Description	Order Number	List Price	Monthly Maint	Total* Maint
SPARCstation ELC, 17" monochrome monitor, 8 MB memory	4/25FM-8	4,995	50	2,850
4 MB memory expansion	X104Q	1,000	0	0
4 MB memory expansion	X104Q	1,000	0	0
4 MB memory expansion	X104Q	1,000	0	0
US Country Kit	X300Q	0	0	0
SunOS 4.1.2 on CD-ROM	SX-21	200	35	1,995
SunOS unlimited users right-to-use	SYSLUN	2,000	N/A	N/A
		10,195		4,845
Less 25% for prepaid Sun maintenance				1,211
				3,634

II. SPARCserver 670MP as a the server and SPARCstation ELC as the client

Cost summary: SPARCserver 670MP hardware/software $137,694

II. SPARCserver 670MP as the server and SPARCstation ELC as the client

Cost summary:	SPARCserver 670MP hardware/software	$137,694
	SPARCserver 670MP hardware/software maintenance	46,961
	SPARCstation ELC hardware/software	10,195
	SPARCstation ELC maintenance	3,634
		________
		$198,484
	Sybase SQLServer 4.8	21,460
	Sybase SMP Access Fees	2,700
	Sybase ELC OpenClient	2,360
		________
		$26,520
	Sybase maintenance	44,945

Price per TPS ($/tpsB) @121.49 TPS = $2222

Server System:

Description	Order Number	List Price	Monthly Maint	Total* Maint
SPARCsystem 670MP, Model 140 with 4 CPUs, 128 MB ECC memory, 1.3 GB SCSI disk, 644 MB SunCD	S670-140-128-P51	82,500	470	26,790
US Country Kit	X311L	0	0	0
150 MB 1/4" internal SCSI tape	660L	1,400	10	570
1.3 GB internal SCSI drive	572A	5,300	50	2,850
Sbus SCSI Card	1054A	1,600	20	1,140
Sbus SCSI Card	1054A	1,600	20	1,140
SCSI Expansion pedestal with 2 1.3GB SCSI drives	X750A	15,995	100	5,700
2 1.3 GB SCSI drives with tray	X756A	10,000	100	5,700
1.3 GB SCSI drive	X572A	5,300	50	2,850
1.3 GB SCSI drive	X572A	5,300	50	2,850
1.3 GB SCSI drive	X572A	5,300	50	2,850
Wyse terminal	WY-50	404	50/yr	200
SunOS 4.1.2 on CD-ROM	SX-21	200	155	8,835
SunOS 4.1.2 documentation of CD	SX-23	495	N/A	N/A
SunOS unlimited users right-to-use	SYSLUN	2,000	N/A	N/A
SunOS Database Excelerator on CD-ROM	DBE-1.2-4-4-21	300	20	1,140
		137,694		62,615
Less 25% for prepaid Sun maintenance				15,654
				46,961

Client Driver System:

Description	Order Number	List Price	Monthly Maint	Total* Maint
SPARCstation ELC, 17" monochrome monitor, 8 MB memory	4/25FM-8	4,995	50	2,850
4 MB memory expansion	X104Q	1,000	0	0
4 MB memory expansion	X104Q	1,000	0	0
4 MB memory expansion	X104Q	1,000	0	0
US Country Kit	X300Q	0	0	0
SunOS 4.1.2 on CD-ROM	SX-21	200	35	1,995
SunOS unlimited users right-to-use	SYSLUN	2,000	N/A	N/A
		10,195		4,845
Less 25% for prepaid Sun maintenance				1,211
				3,634

*Sun hardware has a 90 day warranty. Thus to cost 5 years of hardware maintenance, a total of 57 months is calculated. There are several types of hardware maintenance. The one included in the above calculations is for same day on-site service which provides a four-hour on-site response to service requests received Monday through Friday, 8:00am to 5:00pm local time.

Sun also has several types of software maintenance. The one included in the above calculations is called AnswerLine Service which provides toll-free telephone support between 6:00 a.m. to 5:00 p.m. PST, Monday through Friday, with a software engineer response within two business hours. AnswerLine Service also includes software and documentation updates, monthly Software Technical Bulletins, and access to Sun's on-line support services. SYBASE SQL maintenance costs 15%/year of the base license fee. The Wyse terminal can be bought and serviced from many distributors (not Sun). The Wyse terminal has a one year warranty.

9.2 Performance and Price/Performance

The total price of the entire configuration is required including hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used shall be disclosed. The following table shows the measured tpsB and price/tpsB results for the tested systems.

Company Name	System Name	Database Software	Operating System
Sun Microsystems	SPARCserver 630MP	SYBASE SQLServer 4.8	SunOS 4.1.2 Sun-DBE 1.2

Total System Cost	TPC-B Throughput*	Price Performance
-Hardware -Software -5 year Maintenance	Sustained maximum throughput of system running TPC Benchmark B expressed in transactions per second	Total system cost / TPC-B throughput ($252812 / 121.49 tps-B)
$252812	121.49 tps-B	$2081 per tps-B

Company Name	System Name	Database Software	Operating System
Sun Microsystems	SPARCserver 670MP	SYBASE SQLServer 4.8	SunOS 4.1.2 Sun-DBE 1.2

Appendix A: Application Code

This appendix displays the entirety of the transaction portion of the proprietary driver software that implements the TPC-B application. The code is written in the C language, and uses the Sybase Open Client (dblib).

```
/*  Sybase/Sun Code  */
#include <stdio.h>
#include "sybfront.h"
#include "sybdb.h"
#include "tp.h"

extern CONTROL control;
extern XACT21  data21;
extern int teller, branch, account;

void
stpl_1()
{
    /* Send the transaction information to the dataserver */
     deadlock:

     sprintf (control.command, "stpl_%d", (data21.account % HISTORY)
+1);


     dbrpcinit(DB, control.command, 0);
     dbrpcparam(DB, NULL, 0, SYBINT4, -1, -1, &(data21.account));
     dbrpcparam(DB, NULL, 0, SYBINT4, -1, -1, &(data21.teller));
     dbrpcparam(DB, NULL, 0, SYBINT4, -1, -1, &(data21.branch));
     dbrpcparam(DB, NULL, 0, SYBFLT8, -1, -1, &(data21.delta));
       dbrpcparam(DB, NULL, 0, SYBINT4, -1, -1, &(data21.location));
       dbrpcparam(DB, NULL, (BYTE)DBRPCRETURN, SYBFLT8, -1, -1,
     &(balance));
```

```
dbrpcsend(DB);
        dbsqlok(DB);

        /* Did we deadlock? */
        if (*((DBBOOL *) dbgetuserdata(DB)) == TRUE)
           goto deadlock;

     while(dbresults(DB) != NO_MORE_RESULTS) {
     while (dbnextrow(DB) != NO_MORE_ROWS)
     ;
     }
}


void
stpl_1_args()
{
    int abranch;
    DBINT my_delta;

    /*  Select delta value.  Range is +/-999999  */
  my_delta = irand() % MAX_DELTA2;
  if (my_delta > MAX_DELTA )
     my_delta = - (my_delta - MAX_DELTA);
    /* Now convert to Sybase server type */
  dbconvert((DBPROCESS *) NULL, SYBINT4, &my_delta, -1,
       SYBFLT8, &data21.delta, -1);

    /* Select teller and branch values */
    data21.teller = (irand() % ( control.scale * teller )) + 1;
    data21.branch = ((data21.teller-1)/(teller/branch)) + 1;

  /* Now figure out if customer's branch is local or remote */
  if ( control.scale == 1 || drand() > .15 ) {
           abranch = data21.branch;
        } else {
        do {
               abranch = (irand() % ( control.scale * branch )) + 1;
           } while (abranch == data21.branch);
        }


    /* Now compute account number at the branch */
```

```
    data21.account =
      (abranch - 1)*(account/branch) + (irand() % (account/branch))
+ 1;

    /* Indicate the location of the account */
    if (abranch != data21.branch)
        data21.location = 999;
    else
        data21.location = 1;

}
```

Appendix B: Stored Procedure Definitions

This appendix contains the definition of the Sybase Stored Procedures used to implement the TPC-B transaction.

```
CREATE PROCEDURE nstpl_1
        @acct   int,
        @teller int,
        @branch int,
        @delta  float,
        @location int,
        @balance float output
AS

BEGIN TRANSACTION

UPDATE    account
    SET   balance = balance + @delta
    WHERE number  = @acct

SELECT @balance = balance from account
        where number = @acct

UPDATE    teller
    SET   balance = balance + @delta
    WHERE number  = @teller

UPDATE    branch
    SET   balance = balance + @delta
    WHERE number  = @branch
```

```
INSERT nhistory_1
    (
        account,
        teller,
        branch,
        amount,
        amounti,
        timestmp,
        location,
        other
    )
VALUES
    (
        @acct,
        @teller,
        @branch,
        @delta,
        0,
        getdate(),
        @location,
        "hi"
    )

COMMIT TRANSACTION
```

Note – Thirty one additional stored procedures were used, identical to the above except in name and the history table used in the transaction. For example, the stored procedure stpl_2 inserts into table history_2 rather than history_1. The driver software randomly chooses one of the 32 stored procedures to execute: this distributes the insert load over 32 history tables rather than a single one. The driver system is fully costed, per TPC-B clause 8.4.3.

Appendix C: Database Definitions

This appendix contains the database schema files used in the tests. Note that there are occassional extra attributes contained in the records.These have no relevance to the conduct or definition of the TPC-B benchmark, and were included in the database entirely for analytic reasons. The lengths of the respective records are correct as specified in the TPC-B standard.

```
/* SQL script to create database and tables for TPC-B Sybase/Sun
benchmark */

drop   table branch
go
create table branch
(
       number        int,
       name          char(16),
       branch        int,
       balance       float,
       altnumber     char(12),
       joinkey       int,
       field7        char(12),
       field8        char(10),
       field9        char(10),
       field10       char(10),
       field11       char( 6),
       balancei      int
)
create unique clustered index ibranch on branch(number)with
fillfactor=1
go

drop   table account
go
```

```
create table account
(
       number       int,
       name         char(16),
       branch       int,
       balance      float,
       altnumber    char(12),
       joinkey      int,
       field7       char(12),
       field8       char(10),
       field9       char(10),
       field10      char(10),
       field11      char( 6),
       balancei     int
)
create unique clustered index iaccount on account(number) with
fillfactor=100
go

drop   table teller
go
create table teller
(
       number       int,
       name         char(16),
       branch       int,
       balance      float,
       altnumber    char(12),
       joinkey      int,
       field7       char(12),
       field8       char(10),
       field9       char(10),
       field10      char(10),
       field11      char( 6),
       balancei     int
)
create  unique  clustered  index  iteller  on  teller (number)
with  fillfactor=25
go
```

Branch Table Schema Information

```
Name                           Owner                          Type
------------------------------ ------------------------------ ----------------
branch                         dbo                            user table

Data_located_on_segment        When_created
------------------------------ --------------------
default                         Sep 19 1991  4:39AM

Column_name     Type            Length Nulls Default_name    Rule_name
--------------- --------------- ------ ----- --------------- ---------------
number          int                  4     0 NULL            NULL
name            char                16     0 NULL            NULL
branch          int                  4     0 NULL            NULL
balance         float                8     0 NULL            NULL
altnumber       varchar             12     0 NULL            NULL
joinkey         int                  4     0 NULL            NULL
field7          char                12     0 NULL            NULL
field8          char                10     0 NULL            NULL
field9          char                10     0 NULL            NULL
field10         char                10     0 NULL            NULL
field11         char                 6     0 NULL            NULL
balancei        int                  4     0 NULL            NULL

index_name           index_description
   index_keys
-------------------- --------------------------------------------------------
  ----------------------------------------------------------------------
ibranch              clustered, unique located on default
    number
No defined keys for this object.
```

Teller Table Schema Information

```
Name                           Owner                          Type
------------------------------ ------------------------------ ----------------
teller                         dbo                            user table

Data_located_on_segment        When_created
------------------------------ --------------------
default                         Sep 19 1991  4:39AM

Column_name     Type            Length Nulls Default_name    Rule_name
--------------- --------------- ------ ----- --------------- ---------------
number          int                  4     0 NULL            NULL
name            char                16     0 NULL            NULL
branch          int                  4     0 NULL            NULL
balance         float                8     0 NULL            NULL
altnumber       varchar             12     0 NULL            NULL
joinkey         int                  4     0 NULL            NULL
field7          char                12     0 NULL            NULL
field8          char                10     0 NULL            NULL
field9          char                10     0 NULL            NULL
field10         char                10     0 NULL            NULL
field11         char                 6     0 NULL            NULL
balancei        int                  4     0 NULL            NULL

index_name           index_description
   index_keys
-------------------- --------------------------------------------------------
  ----------------------------------------------------------------------
iteller              clustered, unique located on default
    number
No defined keys for this object.
```

Account Table Schema Information

```
Name                           Owner                          Type
------------------------------ ------------------------------ ----------------
account                        dbo                            user table

Data_located_on_segment        When_created
------------------------------ --------------------
default                         Sep 19 1991  4:39AM

Column_name     Type            Length Nulls Default_name    Rule_name
--------------- --------------- ------ ----- --------------- ---------------
number          int                  4     0 NULL            NULL
name            char                16     0 NULL            NULL
branch          int                  4     0 NULL            NULL
balance         float                8     0 NULL            NULL
altnumber       varchar             12     0 NULL            NULL
joinkey         int                  4     0 NULL            NULL
field7          char                12     0 NULL            NULL
field8          char                10     0 NULL            NULL
field9          char                10     0 NULL            NULL
field10         char                10     0 NULL            NULL
field11         char                 6     0 NULL            NULL
balancei        int                  4     0 NULL            NULL

index_name           index_description                    index_keys
-------------------- ------------------------------------ --------------------
-------------------- ------------------------------------ ----------
iaccount             clustered, unique located on          default
     number
No defined keys for this object.
```

History Table Schema Information

Note – The remaining 31 history tables are similarly defined.

```
 Name                           Owner                          Type
 ------------------------------ ------------------------------ ----------------
 nhistory_1                     dbo                            user table

 Data_located_on_segment        When_created
 ------------------------------ --------------------
 histseg                         Sep 22 1991 12:06PM

 Column_name     Type            Length Nulls Default_name    Rule_name
 --------------- --------------- ------ ----- --------------- ---------------
 account         int                  4     0 NULL            NULL
 teller          int                  4     0 NULL            NULL
 branch          int                  4     0 NULL            NULL
 amount          float                8     0 NULL            NULL
 amounti         int                  4     0 NULL            NULL
 timestmp        datetime             8     0 NULL            NULL
 location        int                  4     0 NULL            NULL
 other           char                14     0 NULL            NULL
Object does not have any indexes.
No defined keys for this object.
```

Method of creating Database for SPARCServer 630MP and 670MP

```
disk init name="logs",physname="/dev/rsd4h",vdevno=1,size=117000
go
disk init name="db1",physname="/dev/rsd6g",vdevno=2,size=139500
go
disk init name="db2",physname="/dev/rsd7g",vdevno=3,size=139500
go
disk init name="db3",physname="/dev/rsd9a",vdevno=4,size=139500
go
disk init name="db4",physname="/dev/rsd10a",vdevno=5,size=139500
go
create database bench on
                         default=45,
                         db1=45,
                         db2=45,
                         db3=45,
                         db4=45,
                         default=45,
                         db1=45,
                         db2=45,
                         db3=45,
                         db4=45,
                         default=45,
                         db1=45,
                         db2=45,
                         db3=45,
                         db4=45,
                         default=45,
                         db1=45,
                         db2=45,
                         db3=45,
                         db4=45,
                         default=45,
                         db1=45,
                         db2=45,
                         db3=45,
                         db4=45,
                         default=45,
                         db1=45,
                         db2=45,
                         db3=45,
                         db4=45
                    log on logs=227
go
checkpoint
go

use master
go
```

```
DISK MIRROR
NAME   = “logs”,
MIRROR = “/dev/rsd8f”,
writes = noserial
go

checkpoint
go

checkpoint
go

disk init name=”db6”, physname=”/dev/rsd3h”,vdevno=7, size=12000
go

alter database bench on db6=20
go

use bench
go

sp_addsegment histseg, db6
go

checkpoint
go
```

Logical Devices

```
device_name                    physical_name
  description
   status cntrltype device_number low         high
------------------------------ ---------------------------------------------
  ---------------------------------------------------------------------------------
----------------------------------------------------------------------------------
--------------------------------------------------------------------------------
  ------ --------- ------------ ---------- ----------
db1                            /dev/rsd6g
  special, physical disk, 272 MB
        2         0             2    33554432    33693931
db2                            /dev/rsd7g
  special, physical disk, 272 MB
        2         0             3    50331648    50471147
db3                            /dev/rsd9a
  special, physical disk, 272 MB
        2         0             4    67108864    67248363
db4                            /dev/rsd10a
  special, physical disk, 272 MB
        2         0             5    83886080    84025579
db5                            /dev/rsd12f
  special, physical disk, 58 MB
        2         0             6   100663296   100693295
db6                            /dev/rsd3h
  special, physical disk, 23 MB
        2         0             7   117440512   117452511
diskdump                       /dev/null
  disk, dump device
       16         2             0           0       20000
logs                           /dev/rsd4h
   special, MIRROR ENABLED, default disk, mirrored on '/dev/rsd8f', reads mirrored,
physical disk, 228 MB
      706         0             1    16777216    16894215
master                         d_master
  special, default disk, physical disk, 272 MB
        3         0             0           0      139499
tapedump1                      /dev/rmt4
  tape, 625 MB, dump device
       16         3             0           0       20000
tapedump2                      /dev/rst0
  disk, dump device
       16         2             0           0       20000
```

Appendix D: Database Contents Samples

This appendix contains the code for the program used to populate the database, as well as sample rows from the respective tables.

```
/*
**   header file for runload.c
*/

# define     BIGBRANCH 58   /* the number of branches */
# define     NAMELEN 16
# define     FIELD7LEN 12
# define     FIELD8LEN 10
# define     FIELD9LEN 10
# define     FIELD10LEN 10
# define     FIELD11LEN  6
# define     ALTNUMBERLEN12

struct bench
{
      long  number;
      char  name[NAMELEN];
      long  branch;
      long  balance;
      char  altnumber[ALTNUMBERLEN];
      long  joinkey;
      char  field7[FIELD7LEN];
      char  field8[FIELD8LEN];
      char  field9[FIELD9LEN];
      char  field10[FIELD10LEN];
      char  field11[FIELD11LEN];
      long  balancei;
      short altnumberlen;
      short namelen;
      short field7len;
      short field8len;
      short field9len;
      short field10len;
```

```
        short  field11len;
};


# include <stdio.h>
# include "runload.h"

/*
**  Load data base
**
**    Arguments:
**        1.  rows: Number of rows (required)
**        2.  file_name:Name of file to write binary data to
**           (optional: if not specified,
**           comma-delimited strings are written
**           to stdout)
**
*/


main (argc, argv)

    int      argc;
    char     *argv[];

    {

      /*
      **  File to output results to,
      **    if the second parameter is "file"
      */

      FILE  *fp;

      int   rows,
            max_branch,
            i,
            scale
            ;

      char  *file_name
            ;

      struct bench data;

      file_name = 0;

      /* There will always be BIGBRANCH branches */
      max_branch = BIGBRANCH;
```

```
rows  =  atoi (argv[1]);

if (argc > 3)
    {
      file_name = argv[2];
      scale = argv[3};
    }

srandom (getpid());

/*
**  Open file if file_name specified
*/

if (file_name)
    {
      if ((fp = fopen(file_name, "w")) == NULL)
          {
      printf ("Unable to open %s\n", file_name);
      exit (1);
          }
    }

/* invariant data */
formdata(data.name,    NAMELEN);
formdata(data.field7,  FIELD7LEN);
formdata(data.field8,  FIELD8LEN);
formdata(data.field9,  FIELD9LEN);
formdata(data.field10, FIELD10LEN);
formdata(data.field11, FIELD11LEN);
data.balance= 1000;
data.balancei   = 1000;

/* create data */
for (i = 0; i < rows; i++)
{
      /* number */
      data.number = i + 1;

      /* branch */
      data.branch = i/scale + 1;

      /* key3 */
      data.altnumberlen = makealt(data.altnumber, i);

      /* data4 */
      data.joinkey = (i + 100003) % max_branch;

      /* output row */
      if (file_name)
```

```
                {
            fwrite((char *)&data, sizeof(data) - 14, 1, fp);
                }
            else
                {
            printf("%d,%.*s,%d,%d,%.*s,%d,%.*s,%.*s,%.*s,%.*s,%.*s\n",
            data.number, NAMELEN, data.name, data.branch,
            data.balance,data.altnumberlen,data.altnumber,
            data.joinkey,FIELD7LEN, data.field7,
            FIELD8LEN, data.field8,
            FIELD9LEN, data.field9,
            FIELD10LEN, data.field10,
            FIELD11LEN, data.field11);

                }

      }

      if (file_name)
          fclose(fp);

      exit (0);

}

formdata(cp, len)
register char*cp;
registerlen;
{
      register char*pattern, *start;

      pattern = start = "abcdefghijklmnopqrstuvwxyz";

      while (--len >= 0)
      {
      /*
      ** If we have reached the end of the string, start again
      */
      if (*pattern == '\0')
            pattern = start;

      *cp++ = *pattern++;
      }
      *cp = '\0';
}

# define   BUFCNVT12/* large enough to hold max of 12 digits */
makealt(cp, value)
char   *cp;
register longvalue;
```

```
{
	char	buf[BUFCNVT], *bp;
	register digits;

	bp = &buf[BUFCNVT];
	digits = 0;
	*--bp = '\0';

	/* convert integer to ascii */
	do
	{
		*--bp = (value % 10) + '0';
		value /= 10;
		digits++;
	} while (value);

	/* copy value back (including null byte) */
	*cp++ = 'A';

	padcpy(cp, bp, BUFCNVT, ' ');

	return (digits + 1);
}

/*
** MAKENAME - form a name for bulk load
**
**	Makename builds up a string of the form:
**		database..table
**
**	Parameters:
**		dbname - name of the database
**		tablename - name of an existing table
**
**	Returns:
**		ptr to the concatenated names
*/
char *
makename(dbname, tabname)
char *dbname;
char *tabname;
{
	static charretname[100];

	register char*cp;

	cp = retname;

	while (*dbname)
		*cp++ = *dbname++;
```

```
	*cp++ = '.';
	*cp++ = '.';

	while (*tabname)
		*cp++ = *tabname++;

	*cp = '\0';

	return (retname);
}

/* Copy a maximum of cnt characters from from_str to to_str.  If space left
** in to_str, pad with pad_char.  To_str must be exactly cnt characters long.
** To_str is *NOT* null terminated, unless pad_char is a null.
*/
padcpy(to_str, from_str, cnt, pad_char)
char    *from_str;
char    *to_str;
int             cnt;
char    pad_char;
{
        /* copy from_str into to_str.  Stop after cnt bytes or null, whichever
        ** comes first.
        */
        while ((cnt-- > 0) && (*to_str++ = *from_str++)) {
                ; /* do nothing */
        }

        /* back up over the null, if it's there */
        /* if we copied n-1 chars, we put a null in the to_str */
        if (*--to_str == '\0') {
                *to_str++ = pad_char;
        }

        /* if we stopped because of a null, pad to_str out with pad_char */
        while (cnt-- > 0) {
                *to_str++ = pad_char;
        }
}
```

Example of rows from branch table (after a period of transaction activity):

```
 number      name            branch      balance              altnumber
       joinkey     field7       field8      field9      field10     field11
       balancei
 ----------- ---------------- ----------- -------------------- ------------
      ----------- ------------ ---------- ---------- ---------- -------
      -----------
```

```
 1 abcdefghijklmnop          1     -19803541.000000 A0
      3 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
 2 abcdefghijklmnop          1     -32653621.000000 A1
      4 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
 3 abcdefghijklmnop          1     -13773291.000000 A2
      5 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
 4 abcdefghijklmnop          1       9412448.000000 A3
      6 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
 5 abcdefghijklmnop          1      29708682.000000 A4
      7 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
 6 abcdefghijklmnop          1     -58947382.000000 A5
      8 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
 7 abcdefghijklmnop          1      13511709.000000 A6
      9 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
 8 abcdefghijklmnop          1     -55062678.000000 A7
     10 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
 9 abcdefghijklmnop          1     -27336917.000000 A8
     11 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
10 abcdefghijklmnop          1      19169744.000000 A9
     12 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
11 abcdefghijklmnop          2        409064.000000 A10
     13 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
12 abcdefghijklmnop          2     -31919179.000000 A11
     14 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
13 abcdefghijklmnop          2     -17086951.000000 A12
     15 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
14 abcdefghijklmnop          2     -12013057.000000 A13
     16 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
15 abcdefghijklmnop          2       1866319.000000 A14
     17 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
16 abcdefghijklmnop          2     -49110301.000000 A15
     18 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
   1000
17 abcdefghijklmnop          2      41238162.000000 A16
     19 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
```

```
     1000
   18 abcdefghijklmnop            2      -13179308.000000 A17
         20 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
     1000
   19 abcdefghijklmnop            2       40566020.000000 A18
         21 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
     1000
```

Example of rows from teller table (after a period of transaction activity):

```
number      name             branch      balance              altnumber
       joinkey     field7       field8     field9     field10    field11
       balancei
----------- ---------------- ----------- -------------------- ------------
      ----------- ------------ ---------- ---------- ---------- -------
      -----------
    1 abcdefghijklmnop            1         -80515.000000 A0
          3 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
     1000
    2 abcdefghijklmnop            1       -3837466.000000 A1
          4 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
     1000
    3 abcdefghijklmnop            1        2563923.000000 A2
          5 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
     1000
    4 abcdefghijklmnop            1         161994.000000 A3
          6 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
     1000
    5 abcdefghijklmnop            1      -14419748.000000 A4
          7 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
     1000
    6 abcdefghijklmnop            1       -8689174.000000 A5
          8 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
     1000
    7 abcdefghijklmnop            1        3309953.000000 A6
          9 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
     1000
    8 abcdefghijklmnop            1        -280858.000000 A7
         10 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
     1000
    9 abcdefghijklmnop            1       -1817224.000000 A8
         11 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
     1000
   10 abcdefghijklmnop            1        3285574.000000 A9
         12 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
     1000
   11 abcdefghijklmnop            2       -5937216.000000 A10
```

```
      13 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
  12 abcdefghijklmnop           2       1280130.000000 A11
      14 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
  13 abcdefghijklmnop           2       5397403.000000 A12
      15 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
  14 abcdefghijklmnop           2     -10727836.000000 A13
      16 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
  15 abcdefghijklmnop           2      -8192049.000000 A14
      17 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
  16 abcdefghijklmnop           2      -2155719.000000 A15
      18 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
  17 abcdefghijklmnop           2      -8227075.000000 A16
      19 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
  18 abcdefghijklmnop           2      -2069822.000000 A17
      20 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
  19 abcdefghijklmnop           2      -5582550.000000 A18
      21 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
```

Example of rows from account table (after a period of transaction activity):

```
number      name            branch      balance              altnumber
      joinkey     field7       field8     field9     field10    field11
      balancei
----------- ---------------- ----------- -------------------- ------------
     ----------- ------------ ---------- ---------- ---------- -------
     -----------
          1 abcdefghijklmnop           1       -574851.000000 A1
             4 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
          1000
          2 abcdefghijklmnop           1         13596.000000 A2
             5 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
          1000
          3 abcdefghijklmnop           1          1000.000000 A3
             6 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
          1000
          4 abcdefghijklmnop           1          1000.000000 A4
             7 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
          1000
```

```
 5 abcdefghijklmnop           1          1000.000000 A5
       8 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
 6 abcdefghijklmnop           1       -507496.000000 A6
       9 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
 7 abcdefghijklmnop           1        444906.000000 A7
      10 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
 8 abcdefghijklmnop           1       1219209.000000 A8
      11 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
 9 abcdefghijklmnop           1         67815.000000 A9
      12 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
10 abcdefghijklmnop           1          1000.000000 A10
      13 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
11 abcdefghijklmnop           1          1000.000000 A11
      14 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
12 abcdefghijklmnop           1          1000.000000 A12
      15 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
13 abcdefghijklmnop           1       -669606.000000 A13
      16 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
14 abcdefghijklmnop           1       -805773.000000 A14
      17 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
15 abcdefghijklmnop           1          1000.000000 A15
      18 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
16 abcdefghijklmnop           1      -1064405.000000 A16
      19 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
17 abcdefghijklmnop           1        205315.000000 A17
      20 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
18 abcdefghijklmnop           1          1000.000000 A18
      21 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
19 abcdefghijklmnop           1          1000.000000 A19
      22 abcdefghijkl abcdefghij abcdefghij abcdefghij abcdef
    1000
```

Example of rows from history table (after a period of transaction activity):

```
account     teller      branch      amount               amounti
      timestmp             location    other
----------- ----------- ----------- -------------------- -----------
     -------------------- ----------- --------------
      85760           9           1         66092.000000           0
        Sep 22 1991 12:09PM          1 hi
      34080           7           1        662123.000000           0
        Sep 22 1991 12:10PM          1 hi
      48064           2           1       -173082.000000           0
        Sep 22 1991 12:10PM          1 hi
   10266976           7           1        304469.000000           0
        Sep 22 1991 12:10PM        999 hi
      14848           4           1        521124.000000           0
        Sep 22 1991 12:11PM          1 hi
      38752           2           1        536245.000000           0
        Sep 22 1991 12:12PM          1 hi
       8224           6           1        455290.000000           0
        Sep 22 1991 12:12PM          1 hi
      44352           3           1        179270.000000           0
        Sep 22 1991 12:13PM          1 hi
      91712          10           1        687081.000000           0
        Sep 22 1991 12:14PM          1 hi
      91360           2           1        742871.000000           0
        Sep 22 1991 12:14PM          1 hi
      77856           6           1        -17123.000000           0
        Sep 22 1991 12:15PM          1 hi
      13824           4           1        -59484.000000           0
        Sep 22 1991 12:16PM          1 hi
      86720           1           1        103927.000000           0
        Sep 22 1991 12:17PM          1 hi
      89600          10           1        299913.000000           0
        Sep 22 1991 12:18PM          1 hi
    8173856           1           1        465229.000000           0
        Sep 22 1991 12:18PM        999 hi
      41216          10           1       -266730.000000           0
        Sep 22 1991 12:18PM          1 hi
      35360           8           1       -915421.000000           0
        Sep 22 1991 12:19PM          1 hi
      66784           4           1        396072.000000           0
        Sep 22 1991 12:19PM          1 hi
      62528           8           1       -726013.000000           0
        Sep 22 1991 12:22PM          1 hi
      15520           6           1        439313.000000           0
        Sep 22 1991 12:22PM          1 hi
      11808           3           1       -666175.000000           0
        Sep 22 1991 12:22PM          1 hi
    8198816           3           1        563023.000000           0
```

```
     Sep 22 1991 12:22PM         999 hi
   36032          3          1      -526646.000000          0
     Sep 22 1991 12:23PM           1 hi
 7548800          9          1       685927.000000          0
     Sep 22 1991 12:23PM         999 hi
   27072          8          1       107908.000000          0
     Sep 22 1991 12:25PM           1 hi
    1568          2          1        12615.000000          0
     Sep 22 1991 12:25PM           1 hi
   62912         10          1       876688.000000          0
     Sep 22 1991 12:25PM           1 hi
   96192          1          1      -280136.000000          0
     Sep 22 1991 12:26PM           1 hi
   29792          1          1       614186.000000          0
     Sep 22 1991 12:28PM           1 hi
   46912          7          1       631572.000000          0
     Sep 22 1991 12:28PM           1 hi
 2511616          7          1       280370.000000          0
     Sep 22 1991 12:28PM         999 hi
   55008          1          1       298148.000000          0
     Sep 22 1991 12:29PM           1 hi
11918240          6          1      -632958.000000          0
     Sep 22 1991 12:29PM         999 hi
   43328          6          1      -756460.000000          0
     Sep 22 1991 12:29PM           1 hi
   30208          4          1       921794.000000          0
     Sep 22 1991 12:29PM           1 hi
   88416          3          1       972563.000000          0
     Sep 22 1991 12:30PM           1 hi
   45760          5          1      -102145.000000          0
     Sep 22 1991 12:31PM           1 hi
   63840          4          1      -187342.000000          0
     Sep 22 1991 12:31PM           1 hi
   38016          4          1      -281805.000000          0
     Sep 22 1991 12:31PM           1 hi
   86880          3          1       985098.000000          0
     Sep 22 1991 12:31PM           1 hi
    5952          1          1       429966.000000          0
     Sep 22 1991 12:31PM           1 hi
   42816          6          1       208251.000000          0
     Sep 22 1991 12:32PM           1 hi
   55776          3          1       180037.000000          0
     Sep 22 1991 12:33PM           1 hi
   92800         10          1      -813242.000000          0
     Sep 22 1991 12:33PM           1 hi
   35008          4          1        14621.000000          0
     Sep 22 1991 12:34PM           1 hi
```

Appendix E: Driver Scripts

This appendix displays a portion of the proprietary driver software that generates and records the transactions. The code is written in the C language, and uses the Sybase OpenClient (dblib).

```
/* Portion of Sybase/Sun Driver code */
  .
.
.

/*
** Run  5 transactions, not counted, so that everything is primed
and
** ready to go.
*/

for (k=0; k<5; k++)
{
   get_args();
   (*control.exec.xact)();
}

/*
** This is where all users stop and wait for the master program
** to syncronize everything.  When all the slave processes are
** 'ready', the master will let them ALL start.
*/
read_sync();
```

```
/*
** Get times from master; these indicate when rampdown,
** steady state, and rampdown end, relative to the master's
** clock.  The client's clock is already sync'ed with the master's,
** to within a few seconds..
*/

  read_times();

  difftime = gettime() - mstart_time;
if ( difftime < -(5 * MILLI) || difftime > 5 * MILLI )
{
   gethostname(hostname, sizeof(hostname));
  /* this is how much we're off from the master, in milliseconds */
   printf ("|### Client with process id %d on host %s lags the
master by %5.1f seconds\n",
          control.pid, hostname, (float) difftime / (float) MILLI);
  }

/* run for ramp up WITHOUT capturing the stats */
if ( control.ramp_up )
{
   do
   {
   /* get the next set of args for this transaction */
   get_args();
   /* run the command */
   (*control.exec.xact)();
   } while ( gettime() < mend_rampup );
}

/*
** On your mark, get set, GO.
*/

/* Record the slave & database's view of when the very first timed
   transaction starts.
*/
EXECSQL("select getdate()");
  dbbind(DB, 1, DATETIMEBIND, 0, &start_timestamp);
  while (dbnextrow(DB) != NO_MORE_ROWS)
      ;
```

```
do
{

   /*
   ** From here...
   */
   get_args();

   /*
   ** To here is OFF THE CLOCK.
   ** So, we try to do as much as we can.
   */

   start_time = gettime();/* get the starting time */
   (*control.exec.xact)();/* run the command */
   end_time = gettime();/* get the ending time */

    if (end_time <= mend_stdystate )
   {
     /* compute the how long it took to run the last transaction */
      tot_time = end_time - start_time;
      control.transcnt++; /* another one bytes the dust */
      control.transresp += (double) tot_time;
      if ( tot_time < control.fast )
      control.fastcnt++;
      if ( tot_time > control.transmax )
      control.transmax = tot_time;
      if ( tot_time < control.transmin )
      control.transmin = tot_time; /* speedy gonzalas */
      control.transsqr += (double) (tot_time * tot_time);
      /* histogram of response times */
      if ( (tot_time / BUCKET) > HIST_MAX )
      control.resp_hist[HIST_MAX - 1]++;
      else
      control.resp_hist[tot_time / BUCKET]++;
   }
   else
   {
       /* Check adherence to TPC-B Clause 6.63:  transactions that
            began during the measurement interval, but ended after
            it.  Note that for this particular client, this cannot
      exceed 1.
        */
```

```
      control.notfinish++;
    }
} while ( end_time < mend_stdystate );


/* Record the database’s view of when the very last timed
   transaction ended for this slave.
*/
EXECSQL(“select getdate()”);
  dbbind(DB, 1, DATETIMEBIND, 0, &end_timestamp);
  while (dbnextrow(DB) != NO_MORE_ROWS)
      ;

/*
** run ramp down
** the idea here is to run a little longer
** so any queries that are taking more time to
** finish will run with the same load as the fast queries.
*/

if ( control.ramp_down )
{
   do
   {
   get_args();
   (*control.exec.xact)();
   } while ( gettime() < mend_rampdown );
}





  .
.
.
/* Portion of Sybase/Sun Driver code */
```

Appendix F: System Parameter Settings

UNIX Configuration Values

The SunOS 4.1.2 DBE 1.2 generic kernel values were used on the SPARCserver 630MP and 670MP. Maxusers was set to 64 and included support for the appropriate devices, documented for the SUT. Swap space on the SPARCserver 630MP and 670MP was set to 192MB. Raw partitions were used for the log, mirror, and all database devices.

SYBASE configuration parameters on SPARCserver 630MP and 670MP

name	minimum	maximum	config_value	run_value
recovery interval	1	32767	20	20
allow updates	0	1	1	1
user connections	5	256	63	63
memory	3850	2147483647	55000	55000
open databases	5	2147483647	25	25
locks	5000	2147483647	30000	30000
open objects	100	2147483647	1000	1000
procedure cache	1	99	3	3
fill factor	0	100	0	0
time slice	50	1000	0	100
database size	2	10000	0	2
tape retention	0	365	0	0
recovery flags	0	1	0	0
serial number	1	999999	0	0
nested triggers	0	1	1	1
devices	4	256	15	15
remote access	0	1	0	0
remote logins	0	2147483647	0	0
remote sites	0	2147483647	0	0
remote connections	0	2147483647	0	0
pre-read packets	0	2147483647	0	0
upgrade version	0	2147483647	480	480
default sortorder id	0	255	50	50
default language	0	2147483647	0	0
language in cache	3	100	3	3
max online engines	1	32	4	4
min online engines	1	32	1	1

Other Configuration Parameters

The following performance parameters were specially set for these benchmark runs, and are user-settable using the "buildmaster" tool, as documented at the time the Sybase 4.8 dataserver goes to FCS.

buildmaster -d/dev/-rsd$disk -ycnblkio=500
buildmaster -d/dev/-rsd$disk -ycnblkmax=500
These parameters pertain to the maximum number of async I/O structures maintained by the SQLserver and the UNIX kernel. The default DBE 1.2 kernel sets up 500 of these also.

buildmaster -d/dev/rsd$disk -ycschedspins=0
This parameter determines whether and how long the SQLserver will sleep in its scheduler. It is pertinent only for MP systems.

buildmaster -d/dev/rsd$disk -ycindextrips=32000
This parameter determines how many times an index page may be touched before it becomes a candidate for flushing out via the SQLserver LRU algorithm. Set this high; index pages are essentially never flushed out. For a TPC-B database scaled to 122 TPS, the account table is about 6000 2K pages in size (because we use clustered indexes). There are about 43000 pages in the buffer cache, plenty for these index pages as well as the concurrent transaction activity on data pages.

buildmaster -d/dev/rsd$disk -ycbufwashsize=4000
This parameter determines the number of pages that the SQLserver will try to keep dynamically clean. It is independent of the semantics of checkpoints. However, because it maintains a cache of clean buffers, the impact of a checkpoint on performance, in terms of instantaneous transaction rates, is minimized. As can be seen from Figure 1-6, after a checkpoint, there is only a small temporary increase in throughput.

buildmaster -d/dev/rsd$disk -ycclkrate=20000
This parameter is SQLserver's estimate of the number of microseconds per OS tick.