Configuration and Tuning of Sybase System 10 for SCO UNIX Open Server 3.0 on Compaq Servers

White Paper

Prepared By

Database Engineering

Compaq Computer Corporation

August 1995

Configuration and Tuning of Sybase System 10

for SCO UNIX Open Server 3.0

on Compaq Servers

First Edition (August 1995)

Document Number 102A/0895

Compaq Computer Corporation

Table of Contents

Configuration and Tuning of Sybase System 10 for SCO UNIX Open Server 3.0 on Compaq Servers

Introduction

This document shares the knowledge acquired by Compaq Systems Engineers in the area of configuration and performance tuning of Sybase System 10 on SCO UNIX Open Server 3.0 on the Compaq ProLiant Family of Servers. It is our desire to deliver the best technical information possible on a specific topic in a timely manner and in a highly useable format. Any comments, suggestions and feedback are always appreciated.

The information presented in this document is based on Sybase System 10 for SCO UNIX Open Server 3.0. There is an abundance of information available concerning general tuning of System 10 on the UNIX platform. This document focuses on specific tuning suggestions for Compaq ProLiant servers. Wherever possible, references are made to other useful tuning documentation.

Other publications covering these and related topics listed below:

- *Configuring Compaq RAID Technology for Database Servers*, Compaq White Paper, Document Number 267A/0294
- *SYBASE SQL Server System Administration Guide*, Document ID:32500-01-1000-02
- *SYBASE SQL Server Troubleshooting Guide*, Document ID:39998-01-1000-0
- *SYBASE SQL Server Installation Guide for SCO UNIX System V/386*, Document ID:34855-01-1000-02
- Compaq Tech Communiqué, *Compaq Insight Server Management*

Tuning Goals

To achieve the best system performance possible, there are several factors which you must review. These factors include optimization of the hardware, the Sybase SQL Server, the operating system, and the application software. This paper focuses on the hardware, Sybase SQL Server, and the operating system. Although it is important to tune the application to take advantage of the system, due to the diversity of applications this is beyond the scope of this paper.

Use the tuning process and parameters in this paper as a starting point. Tuning is an iterative process that evolves as user and work loads change on your system.

An optimally tuned Sybase System 10 on SCO UNIX system should have the following characteristics:

- There is little or no waiting on I/O. You can verify this by running *sar* or *cpqmon*. This indicates that the system processor(s) always have some work to do while there are outstanding I/Os. If asynchronous I/O is enabled on your system, the wait on i/o column (%wio shown below) should always be 0, except when other non-Sybase processes are accessing the disks
- Most of the system processor utilization is in user mode. Again, you can verify this by running *sar* or *cpqmon* and looking at the percentage of processor time spent in kernel and in user time. System time can be thought of as operating system overhead such as time spent in the I/O subsystem or in system calls. The higher the percentage of user to system time that you have, the better. A reasonable range of system time, as shown below, would be 20-30%.

  The following is an example of using the *sar* command to check on the processor:

```
> sar -u 5 5

09:00:10   %usr   %sys   %wio   %idle
09:00:15   76     23     0      1
09:00:20   76     24     0      0
09:00:25   75     23     0      2
09:00:30   75     24     0      1
09:00:35   76     24     0      0

Average    75     24     0      1
```

- Users should see good response times. A system that appears to be tuned well but is experiencing poor response times could have inefficient statements in the application or excess latencies in the I/O subsystem or network. Because well-tuned database applications are processor bound, an additional system processor could improve response times, provided the application is not at fault.

I/O Tuning

In most well-tuned Sybase systems, I/O is not a limiting factor. To ensure that I/O is not a problem, verify the following factors:

- Sequential I/O's are isolated to their own controller volume.
- Random I/O's are balanced across all drives allocated to data and indexes.
- Physical disk I/O limits are not exceeded.

Separate Sequential and Random I/O's

To maximize performance on sequentially accessed data files, place these files on dedicated disks. Of primary importance are the Sybase transaction log files, which are accessed in a sequential, write-only fashion, under normal usage. Other partitions with little I/O activity can share the disk(s) with the transaction logs, such as the operating system partition and swap (unless your machine is memory limited).

In a typical, multi-user database system, file access is random. Spread out these files over as many physical disks as necessary to achieve random I/O rates that do not exceed recommendations. (See next section.) You can best achieve this by using the disk striping available with the Compaq SMART SCSI Array Controller. Spreading out the disk requests among many disks allows a high degree of parallelism to occur on accesses. Using the SMART Controller ensures that the load is balanced equally across the disks. For more information on optimizing array configurations, refer to the Compaq White Paper, *Configuring Compaq RAID Technology for Database Servers*.

Layout of Tables and Files

To improve performance where disk I/O is a problem, keep in mind the following.

- Transaction log access is 100-percent sequential I/O and should be isolated, if possible. Speed of the log is essential to the performance of the system. If possible, these drives should be fault tolerant. Hardware fault tolerance provides the maximum performance and reliability. See the Compaq White Paper, *Configuring Compaq RAID Technology for Database Servers.*
- Data file access is usually random and should be spread across as many drives as necessary. You can acheive greater I/O rates by increasing the number of physical drives. Using a striped array ensures that the I/O's are well distributed.

Checking Disk I/O Rate

Try not to overload any individual disk with random I/Os. Compaq recommends that random I/Os not exceed 50 I/Os per second per drive for 2-Gigabyte drives and not exceed 40 I/Os per second per drive for 1-Gigabyte and 500-Megabyte drives.

To determine the I/O rate per driver, first determine the number of I/O's per second to each logical volume. You can do this by using the *sar -d* or *cpqmon* commands or third-party tools. Take the number of I/O's per second to each logical volume, shown in the column below labeled *r+w/s*, and divide by the number of physical disks in that logical volume. For example, looking at the *sar* output below, you would divide the number of reads and writes per second for the first device, sd011, by the number of disks in that logical volume. If there are 7 drives in that volume, the calculation would be 156.95/7 = 22.42 I/O's per second per drive. This is an acceptable rate since it is well below the recommendations. Repeat the calculation for each device/logical volume listed in *sar*. If the calculated number exceeds recommended I/O's per second rating, adding more physical disks should improve average system performance.

The following *sar* command provides an example for determining the I/O, r+w/s, for each logical controller volume on a system.

```
> sar -d 25 1
```

```
10:51:47   device   %busy    avque    r+w/s    blks/s    avwait   avserv

10:52:12   sd011    26.13    1.0      156.95   712.22    0.00     12.12

           sd012    100.00   1.0      295.67   1178.64   0.00     13.38

           sd013    100.00   1.0      331.34   1319.40   0.00     12.92

           sd014    100.00   1.0      260.40   1029.66   0.00     14.73
```

Sybase Installation Issues

Sybase Installation Default Parameters

Sybase SQL Server boots after running *sybinit*. It starts up with one data engine and the default configuration values. You will probably want to change some or all of the following parameters. (See the section, "System 10 Server Configuration and Tuning Parameters" for more detail on these. Also, refer to the *Sybase SQL Server System Administration Guide*.)

Parameter	Description
memory	Sets the memory size, in 2-Kbyte units, that SQL Server allocates from the operating system.
devices	Sets the number of database devices the SQL Server can use. Include master and log devices in this count. The default is 10. You must set devices before executing the **disk init** command; this initialize your devices so the server restarts successfully.
user connections	Sets the maximum number of connections that can connect to the SQL Server at the same time.
max online engines	Sets the number of data server engines that start up when SQL Server boots. For a one-processor environment, this should be set to one. For SMP environments, one engine per processor is common. If the users are performing several non-server processes, then configure the number of data engines to be one less than the number of system processors.

Enabling Asynchronous I/O

Asynchronous I/O (AIO) is available with SCO UNIX. You can use AIO with databases built using raw devices, but AIO cannot be used with file systems. (Raw devices are recommended for all database devices for recovery purposes. See next section on "Raw Devices vs. File Systems").

During Sybase SQL Server installation, run the script *sco_kernel.scr*. This script adds AIO capability to your system by changing the /etc/conf/sdevice.d/aio file. You can find documentation on *sco_kernel.scr* in the document, *SYBASE SQL Server Installation Guide for SCO UNIX System V/386.*

Raw Devices vs. File Systems

In UNIX, the only way to guarantee that a write reaches the disk immediately is to define the device as a raw device or raw partition. Raw partitioning gives Sybase complete control over cache management. When the Sybase cache manager writes a page to disk, it is physically output to the disk at that moment (unless it is sent to the SMART Array Controller’s battery backed-up cache).

If you use UNIX file systems, when the Sybase cache manager writes a page to disk, this request is passed to the UNIX file manager. The page will go into UNIX cache, therefore the write does not go to disk at that moment but waits until the UNIX cache manager needs space or until the periodic flush is performed. Thus, Sybase cannot guarantee 100-percent recovery when using file systems because of the operating system intervention in the cache management. To guarantee

100-percent recovery, make sure that at least your log device(s) and master device are created as raw partitions. Sybase recommends creating all your database devices as raw partitions.

Memory Tuning

Initial Memory Recommendations for Sybase

Sybase recommends a minimum of 46 Megabytes of memory to install and run SQL Server. You might need to increase these initial memory requirements based upon the following factors:

- Number of users
- Complexity of queries
- Number of disk controllers
- Amount of total disk storage
- Number of Network Interface Controllers (NICs)
- Intensity of the workload in your environment

For each user, SQL Server allocates approximately 40 Kbytes for the users' stack area and an additional 5-10 Kbytes for the users' procedure cache. To estimate the minimum memory requirements for a 512-user system, multiply 512 times 50 Kbytes for a total of 25,600 Kbytes (25 Megabytes) and add this to the Sybase minimum requirement of 46 Megabytes for a total of 71 Megabytes.

Do not tune Sybase memory up at the expense of swapping. Swapping degrades system performance more than the advantages acquired by giving more memory to Sybase. (See the next section on checking for swapping.)

SCO UNIX Shared Memory Parameters

The amount of shared memory allowed in SCO UNIX must equal or exceed the amount of shared memory required for Sybase. The maximum amount of shared memory allowed in a system is determined by several tunable parameters.

The parameter **SHMMAX** sets the maximum size of a single shared memory segment. The maximum amount of shared memory that can be used by a single process is equal to the parameters **SHMMAX** times **SHMSEG** (maximum shared memory segment size times maximum number of shared memory segments). Check to see that only one shared memory segment is being allocated. (This is assuming no other applications on the system are using a segment of shared memory.) If more than one segment is being allocated, it is less efficient than having one large shared memory segment and you should increase **SHMMAX**. You can verify this with the command *ipcs*. For example:

```
> ipcs -b
IPC status from /dev/kmem as of Sun Jan 29 10:33:19 1995
T    ID    KEY         MODE        OWNER    GROUP QBYTES
Message Queues:
T    ID    KEY         MODE        OWNER    GROUP  SEGSZ
Shared Memory:
m    900 0x10043232 --rw-r----- sybase     dba       50745344
T    ID    KEY         MODE        OWNER    GROUP NSEMS
Semaphores:
```

NOTE: In this example, only one shared memory segment was allocated with a size of 50745344.

By increasing the amount of shared memory allocated, you are reducing the amount of memory available to non-server user processes. Be careful not to reduce this memory to a point where swapping occurs. You can detect swapping by noting the available space on the swap file with the UNIX command */etc/swap* -l or by noting swapping activity with *sar -q* or *sar -w*. For example:

sar -q 5 3

```
 10:09:54     runq-sz       %runocc       swpq-sz       %swpocc
 10:09:59     1.0           50
 10:10:04     1.0           100
 10:10:09     1.0           100

 Average      1.0           100
```

In this example, the swpq-sz and %swpocc columns are blank, indicating that no swapping is occurring. If these columns show a non-zero value, that means the system is swapping.

Here is another example:

> sar **-**w

```
10:09:28    swpin/s    bswin/s    swpot/s    bswot/s    pswch/s
10:09:33    0.00       0.0        0.00       0.0        6
10:09:38    0.00       0.0        0.00       0.0        7
10:09:43    0.00       0.0        0.00       0.0        6
10:09:48    0.00       0.0        0.00       0.0        6
10:09:54    0.00       0.0        0.00       0.0        8

Average     0.00       0.0        0.00       0.0        6
```

NOTE: The blocks swapped in and out per second (bswin/s & bswot/s) are given in 512-byte blocks. The number of swap ins and swap outs per second is zero. This indicates that no swapping is occurring. If you see that swapping is occurring, reduce the memory size given to SQL Server (run the **sp_configure memory** command in isql), and check again until swapping ceases.

System 10 Server Configuration and Tuning Parameters

sp_configure

The default settings for the **sp_configure** parameters are usually sufficient for running SQL Server. If you need to alter these parameters, do so with care. Some **sp_configure** values take effect dynamically as you change them. Others require you to stop and restart SQL Server to take effect. If SQL Server determines that you are setting a value outside of its pre-determined normal guidelines, you might have to issue a *reconfigure with override* followed by a *checkpoint* to set the new value. In all cases, use ISQL to ensure that the **sp_configure** run_value and config_value of the items you changed match before letting additional users onto the system.

It is possible to configure these parameters so that you cannot start SQL Server. If this happens, you can reset *all* of the **sp_configure** parameters to their default values using the **buildmaster** routine found in the Sybase *bin* directory. The -d option is followed by the full path name of your master device. For example:

```
buildmaster  -d /dev/master  -r
```

You might not be able to start your database if the number of devices that your database requires exceeds the default number of 10 devices. If this is the case, use the command line switch -ycnvdisks=nn, where nn is the number of database devices you need. For example, if your database uses 25 devices, the reset command line looks like the following:

```
buildmaster -d /dev/master  -ycnvdisks=25
```

To get a listing of all the **buildmaster** parameters and their current values, execute the following command:

```
buildmaster -d /dev/master -yall
```

sp_configure Recovery Interval

Recovery interval determines how often the server should do a checkpoint. During a checkpoint, the SQL Servers data cache area is forcibly written to disk, during which time all other database activity is suspended. Immediately after a checkpoint, user response times are slightly faster than normal until the data cache area becomes filled. Once the data cache area fills, user response times slow down to "normal" levels because of the necessary disk access and memory management.

Leave the recovery interval setting at its default value unless you are willing to take the risk of setting it to a higher value. If you set the recovery interval too long, the user response times deteriorate and become intolerable when a checkpoint occurs. If you set the recovery interval too short, it wastes valuable system processor cycles and generates excessive disk I/O.

sp_configure User Connections

Set this parameter at the minimum value you can run with. Setting this parameter too high wastes memory and increases the size of the table that Sybase must scan when looking for new user logins or existing users logging out.

NOTE: For each user connection, SQL Server allocates approximately 40 Kbytes for the user's stack area and an additional 5-10 Kbytes for the user's procedure cache. You might have to readjust the **sp_configure memory** value, depending upon the number of user connections your environment requires.

sp_configure Memory

Leave this setting at its initial default value. If you have a large number of active user connections or if users cannot connect to the database, you must increase this value. The value expressed by **sp_configure** is in 2-Kbyte pages. This memory area is used to store the data and procedure caches. Refer to the sections in this document on, “Initial Memory Recommendations” and “sp_configure User Connections” for more details on determining the proper value for this parameter. You can further tune this setting by adjusting this value up or down for best user response times. In general, more memory allocated to Sybase gives better performance, up to the point where swapping starts to occur.

One method to improve performance would be to set **sp_configure memory** large enough for heavily used tables and indexes to fit into the data cache area. Be careful in doing this, because setting this value higher than necessary might cause a *loss* of performance, not a gain. SQL Server might spend too much time trying to manage the data cache memory area instead of using it. If you set the value for memory higher than the amount of memory available to the server, you cannot start SQL Server. If this occurs, use the **buildmaster** routine with the **-**r option, as described previously, to reset the value and restart SQL Server.

sp_configure Procedure Cache

This setting is a percentage of the memory allocated to SQL Server that is reserved for caching of stored procedures. The initial default setting of 20 percent should be sufficient for most database environments. You might want to experiment running with a lower percentage for procedure cache, to leave more memory for data cache. On the other hand, if you run many different procedures or ad hoc queries, you might want to increase this value. The procedure cache is not only used to store the compiled stored procedures, but also to compile queries. The cache is also used during the creation of stored procedures.

SCO UNIX Scheduling

To improve UNIX sheduling for database servers in an MPX environment, set the following parameters in the file /etc/conf/pack.d/crllry/space.c and relink the kernel:

Parameter	Description
preemptive=0	This turns off the ability for higher priority processes to interrupt the currently running processes on that processor.
load_balance=0	This turns off the ability for a higher priority ready-to-run process to interrupt the currently running process on another processor.

With preemptive=1, the default value, the scheduler will check the run queue when a process transitions from kernel to user mode to see if there are higher-priority processes which are ready to run. If so, the currently running process can be preempted and relinquish the system processor to allow that process to run.

With load_balance=1, also the default value, the scheduler checks the other system processors for lower-priority running processes and forces one of them to relinquish the processor in preference to the higher-priority process.

Compaq testing has revealed that when a Sybase process is preempted and resumes there is usually only a small amount of time before that processes relinquishes the processor for a resource request, such as an I/O request. This means that two additional context switches have occurred unnecessarily. With preemtive and load_balance turned off processes run to completion.

With preemtive and load_balance turned off there is the danger of a processor-intensive process running through to its full timeslice (1 second, by default) thus starving out all of the other processes. (This will only be the case if there are applications other than Sybase running on your system. There will not be any starving process problems when running Sybase alone.) To protect against this, you can set the kernel parameter MAXSLICE to 3. This causes a process to relinquish the processor after 3 clock ticks (each tick is 10ms).

SCO UNIX User Capacity Parameters

There are several areas of tuning necessary based on the number of users you want to connect. The Sybase parameter, **user connections**, is explained in the section, "System 10 Server Configuration and Tuning Parameters". There are two operating system parameters that deal with user connections, **MAXUP** and **NPROC**. The parameter **MAXUP** specifies the maximum number of processes allowed on the system on a per-user basis. The parameter **NPROC** specifies the maximum number of processes allowed on the system. **NPROC** should be at least 50 greater than **MAXUP** to allow for other operating system and user processes to run. **MAXUP** and **NPROC** are both tuned automatically for you during Sybase installation when you run the *sco_kernel.scr* script. MAXUP gets set to 200 and NPROC to 1200. These values are set high so that most systems will run fine with them. If you have an extremely small system, for example a 16K memory system, you may need to tune down these parameters and other kernel tunables to create a smaller kernel. This is a rare case.

SCO MPX Supplement

Adding the SCO MPX Supplement on a Proliant 4000 gave the following performance improvements when running a complex OLTP benchmark test. The results show that for improved system performance, the MPX supplement should definitely be installed on a multi-processor system.

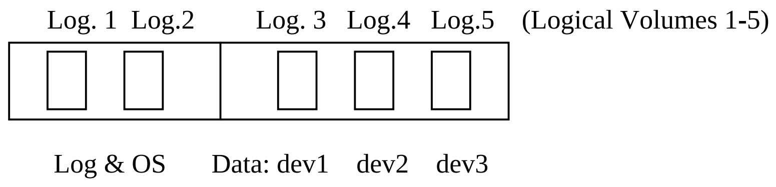
Configuration	Performance improvement with MPX
2-100mhz Pentium processors with 512M RAM	16%
4-100mhz Pentium processors with 512M RAM	33%

Data Striping with Compaq Fast SCSI-2 Controller vs. the Compaq SMART SCSI Array Controller

The SMART Controller can perform hardware striping of data in 16-Kbyte blocks on the drives. This is a great advantage over the Fast SCSI-2 Controller. The best way to illustrate this advantage is with an example. Let's assume you have a system with five 2-Gigabyte drives. The first two drives are mirrored and hold only the operating system and log files. The other three drives hold data only and are configured for 'No Fault Tolerance'. Let's look at how each controller handles data striping in this scenario.

Fast SCSI-2 Controller Example

The SCSI-2 Controller requires that you configure each of the drives as a separate EISA logical volume. For the example, that means you have a total of five logical volumes. To stripe data on the three data drives, you must use software striping. You can perform this using the Sybase **create database** command. You must first use the Sybase **disk init** command to initialize a device for each of the three logical data volumes. You then use those devices in the **create database** command to stripe the data by using fragments. The following gives an example:



```
ISQL
/* Initialize the three devices you have made partitions for through UNIX utilities */
1> disk init name = "dev1",
2>        physname = "/dev/syblink/dev1",
3>        vdevno = 5,
4>        size = 921600                    /* 1 million-2KB pages is max size for disk init.
                                           That is 1953.125 MB which is < 2GB */
5> go

1> disk init name = "dev2",
2>        physname = "/dev/syblink/dev2",
3>        vdevno = 6,
4>        size = 921600
5> go
1> disk init name = "dev3",
2>        physname = "/dev/syblink/dev3",
3>        vdevno = 7,
4>        size = 921600
5> go
```

```
/* Create the database with fragments, striping across  the three devices */
1> create database mydb on
2>         dev1 = 180, dev2 = 180, dev3 = 180,
3>         dev1 = 180, dev2 = 180, dev3 = 180,
4>         dev1 = 180, dev2 = 180, dev3 = 180,
5>         dev1 = 180, dev2 = 180, dev3 = 180,
6>         dev1 = 180, dev2 = 180, dev3 = 180,
7>         dev1 = 180, dev2 = 180, dev3 = 180,
8>         dev1 = 180, dev2 = 180, dev3 = 180,
9>         dev1 = 180, dev2 = 180, dev3 = 180,
10>        dev1 = 180, dev2 = 180, dev3 = 180,
11>        dev1 = 180, dev2 = 180, dev3 = 180
...
> go
```

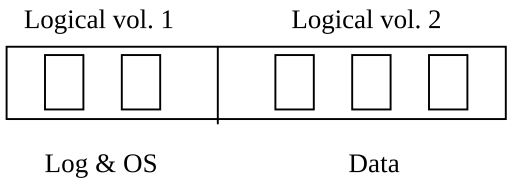
```
/* Create one segment spanning all three devices in order to load table data onto that segment.
That way the data will span across all three devices. */
1> exec sp_addsegment Seg1, dev1
2> go
1> exec sp_extendsegment Seg1, dev2
2> go
1> exec sp_extendsegment Seg1, dev3
2> go
```

This method works, but is more tedious and complicated than the next example with the SMART Controller.

SMART Controller Example

The SMART Controller allows you to create a logical volume that spans up to 14 drives. The UNIX size limit for a raw device partition is 2 Gigabytes. If you are using all 2-Gigabyte drives, you could take better advantage of disk space by creating two logical volumes of seven drives each because there are seven partitions allowed per logical volume. If you want to spread your data across 14 drives in one logical volume to achieve faster I/O on heavily accessed tables, you may do so. However, you will only be able to make use of 14 Gigabytes out of the available 28 Gigabytes.

In our example with three data drives, we can make one logical volume. Now we can use the data striping capability of the SMART Controller to do the work for us:



```
ISQL
/*  Initialize three devices you have made partitions for through UNIX utilities.
If your data size will be less than 2GB, you only need one 2GB partition and one device. */
1> disk init name = “dev1”,
2>          physname = “/dev/syblink/dev1”,
3>          vdevno = 5,
4>          size = 921600
5> go
1> disk init name = “dev2”,
2>          physname = “/dev/syblink/dev2”,
3>          vdevno = 6,
4>          size = 921600
5> go
1> disk init name = “dev3”,
2>          physname = “/dev/syblink/dev3”,
3>          vdevno = 7,
4>          size = 921600
5> go

/* Now the create database command is simple.  The SMART Controller stripes your data across
the devices in 16-Kbyte fragments.  This is much better than the previous example. */
1> create database mydb on
2>          dev1 = 1800, dev2 = 1800, dev3 = 1800
...
> go
```

```
/* Now create one segment spanning all three devices in order to load table data onto that
segment. That way, the data spans across all three devices. */
1> exec sp_addsegment Seg1, dev1
2> go
1> exec sp_extendsegment Seg1, dev2
2> go
1> exec sp_extendsegment Seg1, dev3
2> go
```

Using the SMART Array Controller hardware striping capabilities increases performance and makes it easier for the System Administrator to maintain the database.

If you have more than one logical volume and you need to spread a table across volumes to get faster I/O performance, you must use Sybase software striping, through the **create database** command, as shown above. Then you must create and extend a segment to span a device on each of the logical volumes on which you want to spread the data. That way, the table data is loaded sequentially across volumes, according to your fragment sizes.

The SMART Controller still uses its hardware striping capabilities within each logical volume during creation of the database and loading of the table data. Each drive in a logical volume gets 16- Kbytes of data alternately and sequentially, until the total fragment size that you defined in **create database** is reached, or during data load, until the all data for the table is loaded. Therefore, by using a combination of SMART Controller hardware striping and Sybase software striping, the data is striped evenly across drives in a volume, as well as across volumes.

Network Characteristics of a SQL Server Environment

A client workstation usually assembles a group of SQL commands and submits them for execution by the database server. The server processes the commands and returns the resultant data.

Rather than having the client workstation send a huge grouping of SQL commands, profile the queries. Determine if any of the queries are candidates for conversion to stored procedures. A stored procedure is a grouping of 'standardized' SQL query commands that are pre-compiled and placed into the procedure area of the database by the System Administrator. The stored procedure can then be referenced by name for execution. Using stored procedures for most of the standard DBMS activities reduces the amount of network traffic and uses less server processor resources to process the query.

Compaq Insight Manager

Insight Manager is a Microsoft Windows-based utility that uses SNMP in conjunction with operating system and driver Agents on the server to report hardware failures and system degradation due to a hardware problem. You can configure Insight Manager to page the System Administrator if a component is failing. Using the Insight Manager pre-failure warranty allows you to replace hardware components under warranty before they fail. Insight Manager monitors system hardware and a few operating system components.

Conclusion

The information in this paper is not a complete tuning guide but a supplement to other tuning information provided by Sybase and SCO. To achieve an optimal configuration, there are several factors to consider. Tuning the application, tuning the hardware, tuning the OS, and tuning the network are all areas that must be carefully planned and performed. The tuning process is iterative and can be performed several times to achieve the most optimal performance possible.

We hope that the information provided in this paper helps in this tuning process. The information given is based on experience in tuning Sybase on SCO UNIX Open Server 3.0, however, each configuration is unique. Although all of the hints given here have been tested extensively, do not presume that tuning a specific parameter always gives the desired result.

We welcome feedback on your configurations and experiences to improve our information products in the future. Please send us any comments or suggestions on the attached form, attaching additional sheets if necessary. This helps us tailor future information products to your needs, and enables us to make future revisions of this document and related new information products available to you.

References

Sybase SQL Server System Administrator Guide, Document ID: 32500-01-1000-02

Sybase Architecture and Administration, John Kirkwood, Ellis Horwood Limited, 1993

User Registration/Evaluation Form

Please fill out and return to us this registration/evaluation form to help us keep you up to date with future revisions of this document and related new information products. Your effort will help us improve the quality of the future information products.

Name: ______________________________
Title: ______________________________
Company: ______________________________
Address: ______________________________
____
____
Phone Number: ______________________________

Please evaluate the quality of this document:

	Excellent				Poor
Technical Accuracy	1	2	3	4	5
Organization	1	2	3	4	5
Clarity	1	2	3	4	5
Completeness	1	2	3	4	5
Timeliness	1	2	3	4	5

Please indicate the type of environment you have at your site:

Operating Systems	RDBMS	Processing Type
☐ SCO Unix	☐ Microsoft SQL Server	☐ On-Line Transaction Processing
☐ Microsoft Windows NT	☐ Sybase System 10	☐ Decision Support
☐ IBM OS/2	☐ Oracle 7	☐ Batch Processing
☐ Novell NetWare	☐ Other:	☐ Other:
☐ Novell UnixWare		
☐ Other:		

Please indicate the type of information you would like us to provide in the future:

Topic	Operating Systems	RDBMS
☐ Configuration and Tuning	☐ Microsoft Windows NT	☐ Microsoft SQL Server
☐ Capacity Planning	☐ Novell NetWare	☐ Sybase System 10
☐ Integration Information	☐ IBM OS/2	☐ Oracle 7
☐ Competitive Analysis	☐ Novell UnixWare	☐ Other:
☐ Systems Management	☐ SCO Unix	
☐ Other:	☐ Other:	

Additional Comments:

Return to:
Database Engineering
Compaq Computer Corporation
MailCode 090803
20555 SH 249
Houston, Texas 77070