Linear Scalability on Decision Support Systems: Cray CS6400

Brad Carlile, Cray Research, Inc., Business Systems Division

1 INTRODUCTION

Decision Support Systems (DSS) manipulate and analyze information to highlight previously un-explored relationships in large Gigabyte- or Terabyte- sized databases. They are emerging as an area of strategic value to many customers in the merchant RDBMS market who need to explore all of their data. In the past, lack of performance has prompted users to sample or to summarize data for DSS processing [1], however, sampling can hide valuable information. Insight can be gained by knowing the detail in the large database that summary data obliterates. Parallel processing enables completely scanning large database for this accurate detailed information in a reasonable time.

New powerful Solaris SMP systems, such as the 64-processor CRAY SUPERSERVER 6400 (CS6400), provide practical platforms that are scalable and flexible enough to handle large databases. The CS6400 is Cray's SPARC-based SMP System that runs the Solaris 2.4 operating system and is fully SPARC binary compatible. The combination of the CS6400 and the parallel features of Oracle7 provide scalable performance for DSS operations. This paper presents results that show near-perfect linear scalability on many of the basic tasks representative of parallel DSS queries (full table scans, nested loop joins, sort merge joins, index creation, etc.) using Oracle7 on the CS6400. These basic tasks are the components of more complex queries. Optimization of these components improves "real world" user performance. The key to high delivered performance on DSS is taking advantage of the important application characteristics and focusing on the important system parameters in the complete system.

2 DSS CHARACTERISTICS

An understanding of an application's characteristics is important when implementing a balanced system. Several characteristics of DSS dictate the need for different tuning strategies than are applied to traditional OLTP applications. Important aspects of DSS operation are the ad hoc nature of the queries, the parallelism that can be applied to queries and the data movement patterns within the hardware system. These aspects, when properly measured, can shed additional information on obtaining high performance on DSS operations.

The nature of DSS systems is to iteratively refine and define new queries based on the information gathered from previous queries. These queries are ad hoc and unpredictable. It is difficult to pre-plan for these types of queries since they are only executed once and a query may access millions or billions of rows [1]. With ad hoc queries there is no perfect data layout, especially when refreshing the database with inserts and updates imbalances the original data layout. For predictable performance on SMP systems, fine-grain distribution of data evenly across the disks provides equal access time to the data. Without this equal time access of data, bottlenecks can degrade performance by orders of magnitude and serialize the processing. Such performance problems are a defining characteristic of MPPs [6] [8]. Alternatively, high-performance SMP systems are very flexible and capable.

A characteristic of DSS applications that take advantage of parallel processing is the ability to divide a single query into sub-queries. Executing in parallel keeps both processors and disks active reducing execution time. In Oracle, these sub-queries execute on multiple "Query Servers" in parallel and provide results to a Query coordinator that combines results as required by the query. Parallelism, such as on the CS6400, provides a cost-effective approach to meeting typical DSS performance requirements on large databases.

Internal and external data movement is critical to performance and is often an overlooked characteristic of RDBMS operation. This is becoming more critical as the gap between processor speed, memory speed, and disk speed grows [3]. Optimal disk reference patterns are quite different in OLTP applications then they are in DSS applications. Typical OLTP disk accesses are totally random and are typically in the 2 Kbyte to 4 Kbyte size. The majority of OLTP transactions only need data from only a few rows in a few tables. The appropriate performance metric for these small reads and writes is IOs/second. In contrast, many important operations in the DSS applications tend to read many consecutive rows of a particular table (table scans, aggregates, group-bys, joins, etc.). To optimize IO for DSS, the disk accesses issued by the RDBMS should be very large, up to 1 Mbyte or more, and consecutive. Under these characteristics, the appropriate performance metric for these large reads is Mbytes/second. Within a single RDBMS, it is possible to tune it to implement OLTP transactions with small-sized IOs and to implement DSS-style queries with large-sized IOs. For instance in the Oracle RDBMS, the

"db_file_multiblock_read_count" parameter allows the DBA to set a larger read size for DSS style queries. Currently, we believe that delivered application disk performance for large DSS databases should be on the order of the hundreds of Mbytes/sec.

3 Measuring DSS Performance

During the tuning process it is necessary to establish the characteristics of a DSS application by measuring its performance. Appropriate performance metrics measure the important characteristics of a particular operation. In addition, they provide a reasonable prediction of performance when changing the dimensions of the database. Appropriate measures of DSS performance are MB/sec delivered during a query and the percentage of the job that is parallel. Some performance metrics do not allow accurate comparisons between different implementations and should not be used to predict performance.

The MB/s figure for a particular query will be a good predictor of the performance since a portion of most DSS queries consist of large consecutive IO operations. During these operations, entire rows move from disk to memory even when accessing a particular column of the row. The best characterization is the time it takes to move this data from disk and process it (MB/s=size of the table/time to scan the table). This disk transfer time typically dominates the computation. Given a particular query type, MB/s will be roughly constant on tables of different sizes.

An inappropriate performance metric is millions of rows/second. The problem with this measure is that the size of a row is highly dependent on the table design (a row may contain tens, hundreds, or thousands of bytes of data). For different size tables, full table scan times may be very constant in terms of Mbytes/sec whereas Mrows/sec varies by almost 3 orders of magnitude as is illustrated in the table shown below. Mrows/sec is an inappropriate performance metric given the inherent variability in row size.

On parallel systems, a useful DSS performance metric is scalability. Caution should be exercised when discussing scalability. Scalability figures can be artificially inflated by crippling single processor performance and optimizing parallel performance. It is important to test the application with the appropriate tuning parameters for both parallel and sequential executions. It is best to be suspicious of scalability calculations based on best parallel runs against initial (un-tuned) single processor runs.

We suggest that the best manner to look at scalability is the speedup on a particular number of processors or the percentage of parallelism. The only manner to accurately compare system scalability is to use actual performance. The maximum number of processors on a system does not determine its scalability. The percentage of an application that is parallel and the overheads involved in using multiple processors limit delivered parallel performance. Parallel performance beyond a given number of processors can be estimated using a formula based on Amdahl's law [3]. This estimate involves determining the percentage of a job that is parallel and predicts the speedup for a given processor count using "percent parallel".

To determine the percentage of a job that is parallel (P= percent parallel), a one-processor run and a 40-processor run (full table scan with aggregates) will be used to calculate percent parallel.

$$P = (1/observed\_speedup-1)/(1/n-1) \quad (1)$$

where n is the number of processors and speedup is the observed speedup. For example, if a 40 processor can get a 39.083x speedup over one processor, then the percent parallel is: P = (1/39.083-1)/(1/40-1) P =.99939 or 99.939% parallel. To predict other speedups, use the following formula:

$$predicted\_speedup=1/(P/n+(1-P)) \quad (2)$$

where P is the percent parallel and n is the number of processors. Using the example above (P=.99939), we get the following table, which shows good agreement with the actual results.

Prediction using Amdahl's Law			
N	Calculation	Predicted Speedup	Actual Speedup
1	*actual data used*	1.0	1.0
8	1/(.99939/8+(1-.99939))	8.0	8.7
16	1/(.99939/16+(1-.99939))	15.8	14.4
24	1/(.99939/24+(1-.99939))	23.7	23.4
32	1/(.99939/32+(1-.99939))	31.4	32.3
40	*actual data used*	39.1	39.1
56	1/(.99939/56+(1-.99939))	54.2	Est.
64	1/(.99939/64+(1-.99939))	61.6	Est.

Potential errors in prediction may arise from the following areas:

- Performance limitations due to application characteristics or by the IO or memory bandwidth of the system.
- Changing the size of a problem will usually increase the time spent in a parallel region (it is very difficult to use the above estimation for a different problem size).

Comparison of MRows/Sec and Mbytes/sec				
Rows	Bytes/Row	Seconds	Rows/Sec	Mbytes/Sec
2,000,000	2000	66	0.03 Mrows/s	61 MB/s
5,000,000	200	16	0.30 Mrows/s	63 MB/s
10,000,000	50	9	1.20 Mrows/s	57 MB/s
200,000,000	150	470	0.42 Mrows/s	64 MB/s

- Estimates can be very low if more parallelism exists by tuning the code (improving performance of serial section or making more of it parallel).
- Estimates can be very low if using more processors increases the percentage of a job that is parallel. (cache effects such as interference and memory layout).

In the above example, the percent of parallelism was a good predictor of performance. The different DSS queries measured in this report are approximately between 98.00% and 99.93% parallel.

Another performance metric used by some is percent speedup (*actual speed/perfect speedup*). This is not an accurate manner to report scalability since it varies with the number of processors. In the table above, the percent parallel was roughly constant. If we looked at the erroneous percent speedup, we would see that this figure varied between 100% and 97% and this figure will decrease as the number of CPUs grow. This measure does not give an accurate view of the performance and should not be used.

4 IMPORTANT SYSTEM PARAMETERS

A DSS system consists of many components. With respect to performance, the order of importance of these components is the RDBMS, operating system, delivered disk bandwidth, delivered memory bandwidth, and finally processor speed. There are many interdependencies between various aspects of these performance components.

RDBMS

The implementation of the RDBMS is critical for database performance. This is especially critical for parallel-processing performance. In general, queries can be decomposed to run in parallel in a variety of ways. A query optimizer needs to be intelligent enough to determine an appropriate parallel query plan of execution. It should effectively choose between options such as using table scan versus index reads or a particular implementation of table joins. An efficient implementation balances query decomposition for the system configuration and database table sizes. During query execution, multiple processes need to be efficiently used and coordinated. In addition, the RDBMS needs to be optimized for memory management and to properly utilize its cache (System Global Area or SGA). Issues related to RDBMS performance include shared resource utilization, locking issues, IO strategies, and efficient code. Oracle incorporates all of these features in a shared-everything architecture.

Important tunable Oracle parameters are located in the *init.ora* file. Parameters of interest to DSS workloads involve SGA size (up to 2 GBytes), database block size, multi-block read size, query server sort area size (not limited by SGA size), and the number of query servers. The DSS Benchmark section below discusses some of these in more detail.

Operating System

The operating system (OS) can limit the efficiency and scalability of RDBMS performance. Efforts by Cray and Sun [4] [2] allow the RDBMS to exploit performance features of Solaris 2, such as the multi-threaded architecture of the Solaris kernel, asynchronous IO, soft processor affinity, efficient OS striping, and enhancements to memory management for large memory systems. The combination of these efforts and others provides a system that is responsive and efficient when executing parallel workloads.

There are very few tunable OS parameters since the operating system is tuned for RDBMs workloads. Parameters that may be of interest to some workloads are scheduler classes, increased semaphore limits, and the file system flusher. Providing application scalability has been a design requirement of Solaris for many years.

Disk Bandwidth

The CS6400 has delivered over 265 MB/s on a moderate size disk configuration (90 Elite3 disks). This is more than some vendor's memory bandwidth. The program tested in this case issued continuous reads with no computations. This characterizes the maximum realized disk bandwidth for this configuration. IO-bound applications are likely to deliver less than this figure due to additional required application processing. Delivered disk performance on a particular disk configuration provides a much better baseline for performance estimation than system maximums listed on spec-sheets.

In order to obtain sufficient I/O for DSS applications, disk reads must be of a sufficient size to maximize bandwidth as opposed to the small I/O's typically used for OLTP applications. A simple manner to optimize the performance of large IOs is to use a Volume Manager (Solstice DiskSuite, Veritas Volume Manager, etc.) to stripe the data across the disks. Disk striping can also be viewed as a way to optimize data layout on disks. The fine-interleaving of datablocks across the disks has the advantage of naturally distributing inserts and updated data throughout the disk system. Ad hoc queries are naturally optimized. There is no processor dependence on data layout, simplifying performance tuning.

Fine-grain disk striping (64k to 1M) can increase disk bandwidth and minimize disk hot-spotting. Fine-grain striping is most generally applicable to a wide range of query types. Alternatively, course-grain disk striping (concatenated disks) may give higher performance for only certain queries. For example, the CS6400 with a moderate size disk system has delivered over 110 MB/s on a disk-resident database using Oracle.

Memory Bandwidth

Memory Bandwidth of a system is also a major contributor to system cost. Efficient use of the available bandwidth is critical. The CS6400 delivers over 1100 MB/s of memory bandwidth. Delivered bandwidth for cached data is much higher.

Memory size can also be an issue for DSS performance. Data that does not fit in the physical memory of the system will reside on the disk. The CS6400 supports up to 16 GBytes of physical memory. This allows for a large SGA and large sorting area. Swapping and paging are not generally an issue with the large

configurations of the CS6400. The large memory also provides the ability to cache indexes or other randomly read "hot" tablespaces. One manner to cache these tablespaces is to put them in a Unix File System (UFS) and let the Unix file caching mechanism buffer data from these tables. Using this method, it is possible to effectively cache randomly accessed tables that approach physical memory size. This has increased performance for some workloads.

Processor Speed

Processor performance can be measured by using the CPU speed or by using "cache-friendly" benchmarks such as SPECint92. Processors designed to deliver high SPECint92 performance focus primarily on processor cache-to-processor issues. These benchmarks do not even stress the processor-to-memory issues that are important to most applications. For these reasons, SPECint92 results can be very misleading when they are used to gauge performance of "non cache friendly" operations more typical in RDBMS code. DSS performance is more likely to be determined by IO speed or the efficient implementation of the RDBMS.

Processor speed is important, but it must be balanced with processor-to-memory bandwidth and efficient IO. These are factors that SPECint92 does not measure. Recent data [5] [7] suggests that delivered MB/s is a better estimate than SPECint92 for DSS performance. Even though this data is not directly comparable, it can be instructive. The table below shows that a DEC 7000 (275 MHz) has a much higher SPECint92 rate than the CS6400 (60 MHz) but has a much lower delivered performance on DSS operations. Results later in the paper show that the CS6400 performance will scale with more processors. Digital has not published any data on parallel DSS performance.

	CS6400 (1 processor) [5] SPECint92=89	DEC 7000 (1 processor) [7] SPECin92=180
Full Table Scan	4.1 MB/s	3.4 MB/s
Multi-Table Join	1.3 MB/s	0.3 MB/s
Index Creation	0.9 MB/s	0.1 MB/s

As shown above, the CS6400 is 1.2x to 9x faster than the DEC 7000 on DSS operations, however this would not be predicted by the SPECint92 rating (or CPU MHz). SPECint92 should not be used to predict DSS performance.

In addition, Stephen Brobst [1] has proposed a constant based on SPECint92 to estimate performance that a processor should have for DSS operations. His constant is (10 SPECint92/Disk spindle). This number comes from experience on the Teradata system (486, SPECint92 = 32) which gets maxed out at 3 disks or 10 (SPECint92/Disk spindle). Due to the variability of this constant on different systems and its reliance on SPECint92, it is not recommended that this measure be used to compare systems or estimate requirements for a balanced system.

5 DSS BENCHMARK

To demonstrate the high scalability that Oracle provides on powerful SMP systems, several queries representative of important DSS operations were executed with a CS6400 system. The benchmark consisted of tuning the database for optimal performance and measuring various system functions and scalability with different numbers of processors. The CS6400's configuration consisted of forty 60 MHz SuperSPARC processors, 1240 MB of physical memory, and ninety (2.9 GB) disks. Oracle version 7.2.1 was used together with Sun's Online DiskSuite Version 4.0.1 (beta) which provided perform machine (OS) striping for the data files. The data tablespace was evenly spread out across 72 disks using a stripe size of 64K. The SORT_AREA_SIZE was 20 MB per query server process, DB_BLOCK_SIZE was 8K, and the number of DB_BLOCK_BUFFERS was 6400 (approximately 52 MBytes). The data tablespace had a PCTINCREASE of zero. The tuning process only involved adjusting Oracle init.ora parameters.

A 5 million row table with 16 columns and a 204 byte average row length was used to test the Oracle7 parallel features. A single Oracle instance was used with a single user issuing the queries sequentially. The amount of table data was constant during the execution of the following tests:

Table Scan with Aggregates (SCAN). This operation scans the table and performs an aggregation on each of the sixteen columns in the table. Aggregation functions used were min, max, avg, sum, and count.

Sort-Merge Join (SMJ). This entails joining the table to itself. Here the table is scanned twice where each scan chooses half the number of rows resulting in 5 million rows being grouped by different criteria, resulting in 1,000 evenly distributed groups that are ordered. The sort-merge join is selected over nested-loop join by the use of an optimizer hint.

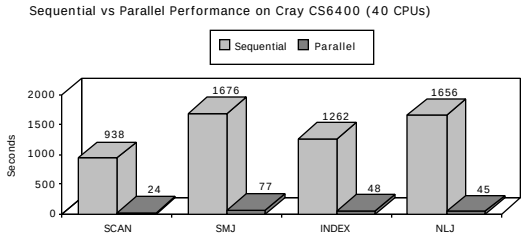
Parallel Indes Creation (INDEX). For this operation, a set of query servers scans the target table, and passes the row IDs and column values to another set of query servers that sorts the index entries. These sorted entries are finally passed to the query coordinator process that builds the index. For this test an index was built on a single numeric column in the table.

Nested-Loop Join (NLJ). This operation scans the table with 2% of the rows selected to join with another instance of the same table. The join is 1:1 so 100,000 index lookups are done on the index created by the index creation test. The joined rows are then filtered by the predicate in the inner table returning 10 rows. The nested-loop join is selected over sort-merge join by the use of an optimizer hint.

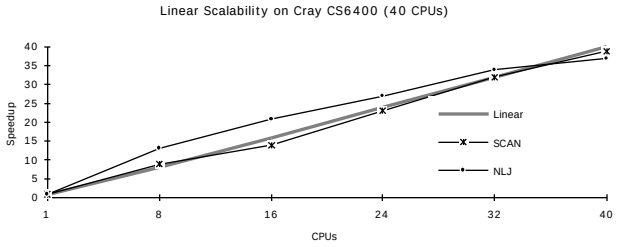
Results

The SCAN operations using 40 processors took only 24 seconds to complete while one processor took approximately 16 minutes to complete. This is a speedup of 39 times compared to a non-parallel operation with a single CPU. The NLJ operation showed a 37x speedup, reducing the query time from half an hour to 45 seconds. The speedup for the SMJ operation is also significant since the initial scan as well as the subsequent join and group-by operations are parallelized. In each of these cases, all processors are effectively used. Even more performance is expected with an expanded IO system with more disks and

controllers to increase the delivered IO bandwidth for these IO intensive operations.



These benchmark results demonstrate near-linear scalability. Analysis of these queries show an average of 99% parallel, which as discussed above, predicts that all 64 processors can be effectively used on the CRAY CS6400 system. The graph below compares actual results with perfect linear scaling with the number of processors. During these runs, only the number of query servers was varied with other parameters kept constant. Optimal performance was obtained at 1 processor and 40 processors, not optimizing the performance for other processor configurations.



Another feature explored is parallel loading of the database. For large databases this can be an important time-consuming operation. Multiple processors will greatly accelerate this operation. Parallel Load involves reading multiple data files from disk and using the multiple concurrent direct loader sessions to write this data simultaneously to the same table in the database. Performance on Parallel Load is IO bound and additional performance improvement can be obtained by increasing the number of disks and controllers.

Updated and additional results will be made available at the conference presentation.

6 CRAY SUPERSERVER 6400 SYSTEM

The CS6400 is an enterprise-class application or data server for a wide range of tasks such as on-line transaction processing (OLTP), decision support systems (DSS), on-line analytical processing (OLAP), or data warehousing. The result of a technology agreement between Cray Research and Sun Microsystems, the CRAY CS6400 is a binary-compatible upward extension of Sun Microsystems' product line. Its full compatibility with Sun Microsystems' Solaris operating system guarantees the availability of the largest set of third-party solutions in open systems. Large configurations of this SMP system can simultaneously support sixty-four processors, 16 Gigabytes of physical memory, and 10 terabytes of online disk storage. The CS6400 also has the capacity to combine DSS and online transaction processing (OLTP) job mixes on the same platform. The CS6400 also provides processor partitioning to segregate these workloads for flexibility in system management. In addition to DSS scalability, the CS6400 has also shown excellent OLTP Scalability. It leads in the industry in TPC Benchmark™ B Results with a performance of 2025.20 tpsB and leads in price/performance with $1,110.14 per tpsB (result date: 6/4/94).

RAS features are a critical part of the design of the CS6400. There is nearly complete redundancy of system components in the CS6400. This includes multiple redundant system buses, N+1 power supplies, dual pathing, RAID devices, disk mirroring, etc. The CS6400 also offers fail-over, hot swap of system boards, dynamic reconfiguration (and expansion), and automatic reboot. A separate service processor including monitoring software (with call home on unplanned reboots) and remote diagnostics.

The speedup factors obtained are the result of joint engineering efforts by Oracle, Cray, and Sun in exploiting the performance features of Solaris 2, such as the multi-threaded architecture of the Solaris kernel, asynchronous I/O, and efficient OS striping. Likewise, the hardware strengths of the CRAY SUPERSERVER 6400 that facilitate good scalability include the quad XDBus bus architecture, fast SCSI controllers, and larger CPU caches to hold frequently referenced data and instructions. Oracle will exploit faster CPUs with larger caches to deliver even bigger performance boosts for future generations.

The SMP architecture allows DSS queries to be optimized for parallel operations, while avoiding the MPP performance and administration problems. MPP performance can be very depen-

System Components	Configurations	Specifications
Number of Processors	4-64 SPARC	60 MHz SuperSPARC
Memory Size	16 Gbytes	SMP, Shared Memory
System Bandwidth	1.7 GB, 4 XDBuses	55 MHz
I/O Channels	16 SBuses	800 MB/s
Bus Controllers	64	Full Coherency
Online Disk Capacity	10 Tbytes	Using 9 GB disks
Operating System	Solaris 2.4	SVR4, Solaris Enterprise Server

dent on data layout. On MPPs, the user has the choice between executing high-performing "good" queries and slow-performing "bad" queries. This has the drawback of potentially "training" users what queries not to submit. In these respects, MPPs are more difficult to tune and to administer. Even on an MPP that uses a shared disk strategy, there can be other problems on an MPP due to coordinating the various IO requests from within the MPP.

7 CONCLUSION

Performance and scalability are particularly important for DSS applications. The CS6400's SMP design allows commercial DBMSs to effectively use its large configuration of processors. Large configurations of the CS6400 provide excellent scalability on DSS operations using the Oracle7 shared-everything implementation. The characteristics of DSS operations allow IO optimizations that deliver high bandwidth. The efficient implementation of the RDBMS on the C6400 provides near-linear scalability while maintaining all of the advantages of SMP systems. These effects are effectively demonstrated using the MB/s and percent parallel metrics. Past limits to SMP scalability are avoided by providing sufficient performance at every level in a balanced system.

8 REFERENCES

[1] S. Brobst, "An Introduction to Parallel Database Technology", VLDB Summit, Miller Freeman, Inc, 1995.

[2] A. Cockcroft, *Sun Performance and Tuning*, SunSoft Press, A Prentice Hall Title, 1995.

[3] J.L. Hennessy and D. A. Patterson, *Computer Architecture A Quantitative Approach* (Morgan Kaufmann Publishers, San Mateo CA, 1990).

[4] D. McCrocklin, "Scaling Solaris for Enterprise Computing", Cray User's Group, 1995.

[5] *Oracle and Cray Superserver 6400 Linear Scalability*, Oracle Corporation, May 1995.

[6] "Open Computing & Server Strategies", Fourth Quarter Trend Teleconference Transcript, META Group, Dec 13, 1994.

[7] J. Scroggin, "Oracle7 64-Bit VLM Capability Makes Digital Unix Transactions Blazingly Fast", Oracle Magazine, Vol IX, No 4, July/August 1995, pp 89-91.

[8] C. Stedman, "What you don't know... ... will hurt you", Computer World MPP & SMP special Report, March 27, 1995, supplement pp 4-9.

9 AUTHOR INFORMATION

Speaker's Biographical Sketch

Brad Carlile is a Performance Analyst at Cray Research, Business Systems Division. He is responsible for analyzing and characterizing real-world workloads. Background includes work on eight distinct shared and distributed memory parallel architectures on a wide variety of commercial and technical applications. His current focus is DSS performance issues.

Contact Information

Brad Carlile
Cray Research, Business Systems Division
8300 Creekside Ave,
Beaverton, OR 97008
bradc@oregon.cray.com
(503)520-7622 (voice)
(503)520-7724 (fax)

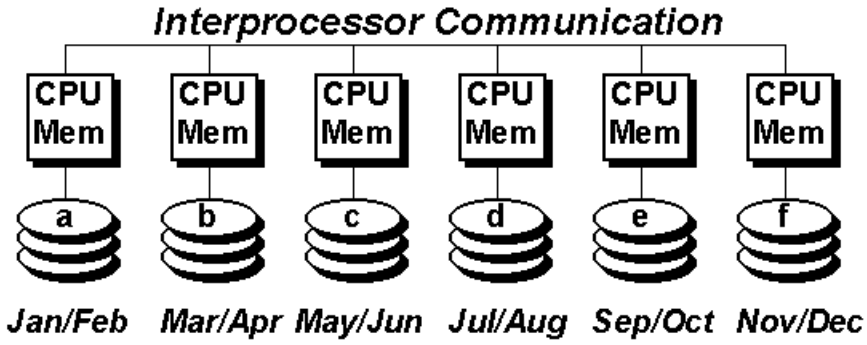
Overview

- Scalability Has to be Designed into System
- Cray CS6400 - "The SMP That Scales"
  - OLTP: On-line Transaction Processing
  - DSS: Decision Support
  - Multimedia
  - Numerically Intensive
- Scalability and Performance on DSS & OLTP
  - Parallel Query Delivers High DSS Performance
  - High Capacity for Terabyte Data Warehouse
  - Parallel Server Delivers High OLTP Performance

Decision Support Systems

- Data Mining: Trying to find Interesting Gems in Data
- Data Warehouse: Enterprise-Wide Data Repository
- Decision Support System:
  - HW & SW for Querying Database for Data Relationships
  - Often Scan Large Amounts of Data
  - Ad Hoc Questions posed to the Database
  - Results Use in Decision Making Process
  - Markets:
    - Retail
    - Manufacturing
    - Financial
    - Government
    - etc...

MPP Disk Partitioning Problems



- Static Partitioned Data Only Helps Some Queries
  - CPUs & Disks Fully Utilized on Montly Averages
  - Christmas Sales Queryies (Nov/Dec), Only Use One Disk!
  - Data Can Only be Passed on Slow Interprocessor Links!

SNDC.DSSscale.bw.R0.1 4

SMP Performance Solutions

- Scalability & High Performance!
- Disk Striping Optimizes Data Layout
  - Updates & Incremental – No Problem
  - Ad Hoc Queries – Natural
- Shared Everything
  - No Processor Dependence on Data Placement
- Merchant Databases – Standard Software
- Thousands of Applications

SNDC.DSSscale.bw.R0.1 5

DSS & Data Warehousing Requirements

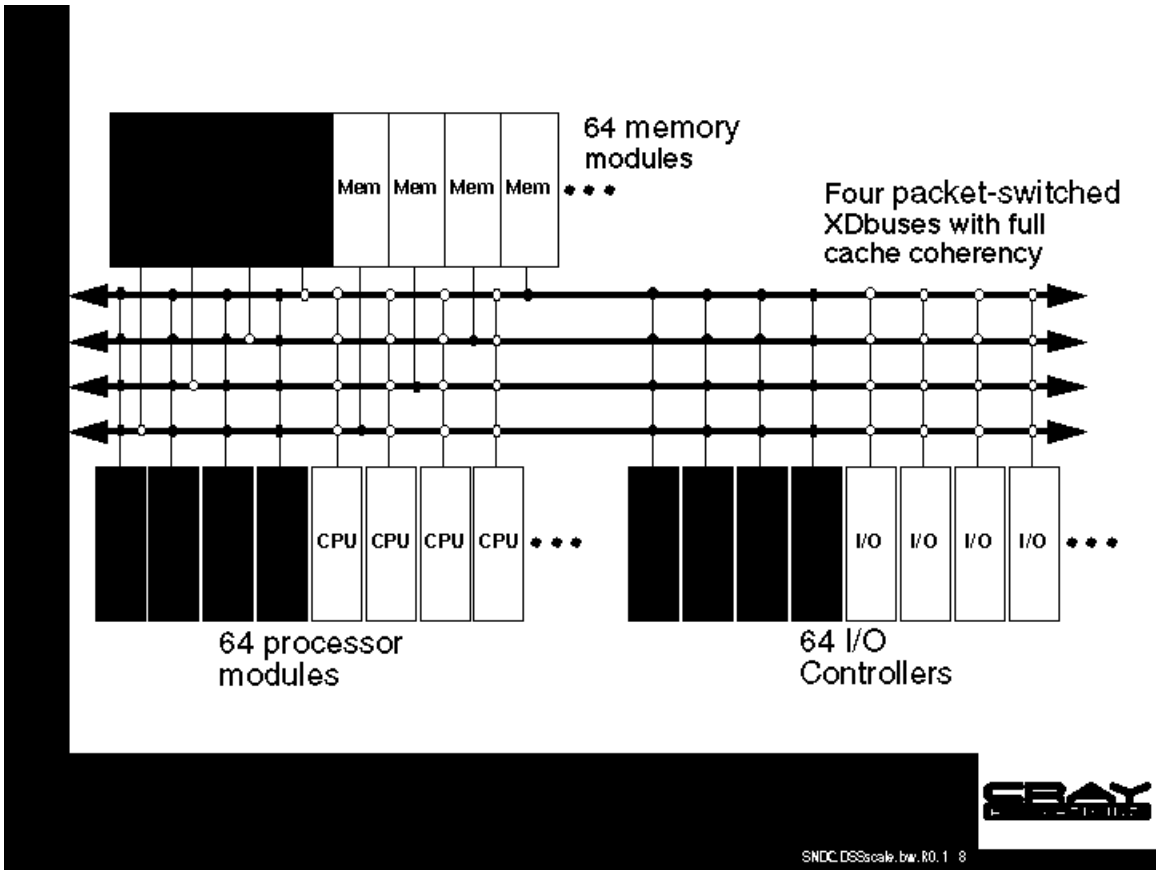
- Rapid Growth of the Data Warehouse
  - CS6400: 10 Gigabyte Pilot to Multi-Terabyte Production
- Need Fast Data Move from Disk to CPU (100's MB/s)
- Ad Hoc Queries -*NO Perfect Data Layout!*
- Updates and Inserts to "Refresh" Database
- Large Sorts - Gbytes of Physical Memory
- Fast Table & Index Access - Gbytes Physical Memory
- Data Mining Tools
- Often Mainframe Connectivity Required

SNDC.DSSscale.bw.R0.1 6

CS6400 System Specifications

Number of processors	*4 - 64 SPARC*	*60 Mhz*
Memory Size	*16 GBytes*	*SMP, Shared memory*
System Bandwidth	*1.7 GB/s, 4XDBuses*	*55 Mhz*
I/O Channels	*16 Sbuses*	*800 MB/s (total)*
Bus Controllers	*64*	*Full Coherency*
On-Line Disk Space	*10 Tbytes*	*Using 9 GB Disks*
Operating System	*Solaris 2.4*	*SVR4, Enterprise*

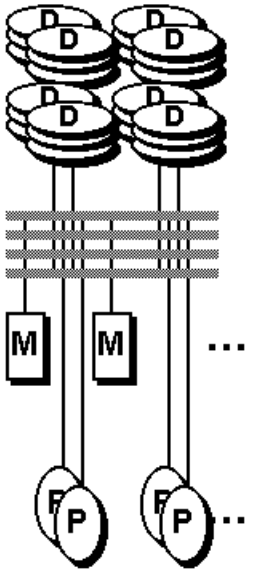
SNDC.DSSscale.bw.R0.1 7



Bandwidth Is Key to Scaling!!!

"Follow the Disks"

- Demonstrated over 265 MB/s Disk IO
  - Disk to Processor
  - More Than Many Competitors' Bus Bandwidth
- Measured 1100 MB/s Bus Bandwidth
  - True Shared Memory
  - Small Uniform Latency
- Up to 16 GB Coherent Physical Memory
  - Swapping and Paging not an issue
  - Large Shared Memory Area and Buffer Pool
  - Large Sorting Areas for Database
- 7680 Integer MIPS



SNDC.DSSscale.bw.R0.1 9

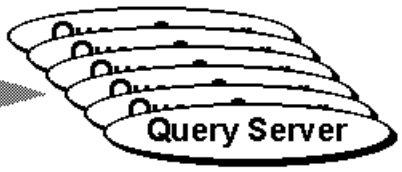
DSS Benchmark Setup

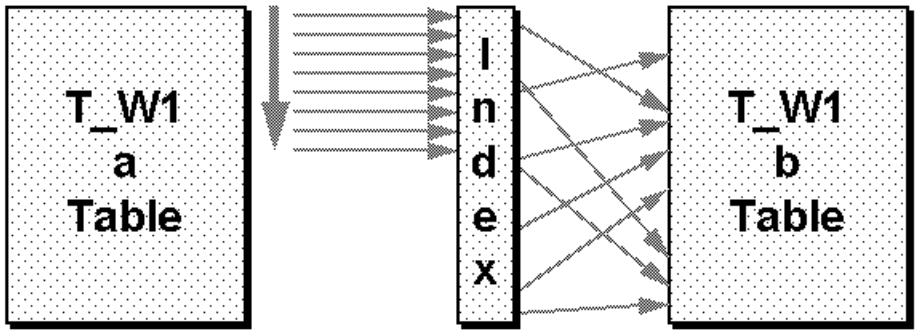
- Tested Components of Complex Ad hoc Queries
- 1 GB Table
  - 5 million rows, 204 Bytes/row
  - Modeled after "Wisconsin" Data
- Data Table
  - 72 Elite3 Drives (3 Drives/controller)
- Index Tablespace
  - 12 Elite3 Drives (3 Drives/controller)
- Oracle 7.2.1
- Volume Manager: OnLine Disk Suite 4.0.1
  - Disks Striped at 64K

Nested Loop Join

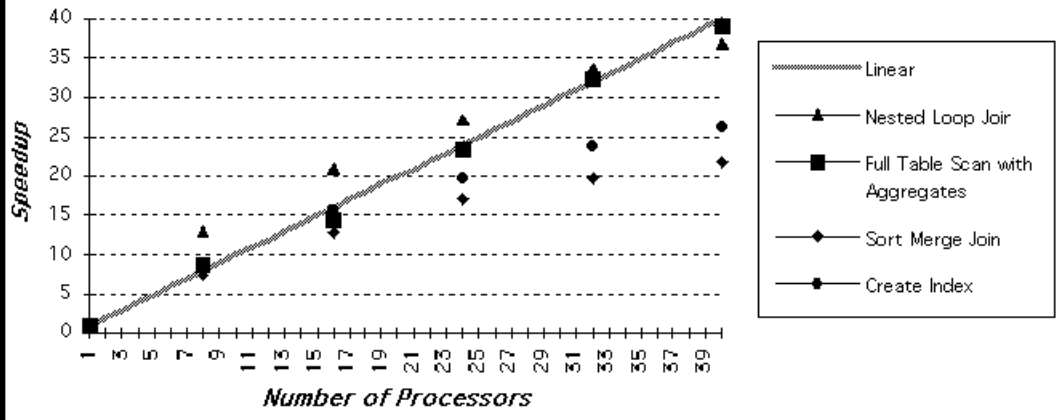
Query Coordinator

```
Select a.uniq1 b.uniq2 b.str
from t_w1 a, t_w2 b
where a.h < 2
  and a.uniq2 = b.uniq1
  and b.tenthous=1;
```





Query Scalability
22x to 39x Speedup on 40 CPUs!



- **# Query Servers Twice Number of Processors**
- **Degree of Parallelism Held Constant**

Scalability Predictions Based on Amdahl's Law

- P = (1/observed_speedup-1)/(1/n-1)
- predicted_speedup=1/(P/n+(1-P),
- where n=#CPUs

N	Calculation	Predicted Speedup	Actual Speedup
1	*actual data used*	1.0	1.0
8	1/(.99939/8+(1-.99939))	8.0	8.7
16	1/(.99939/16+(1-.99939))	15.8	14.4
24	1/(.99939/24+(1-.99939))	23.7	23.4
32	1/(.99939/32+(1-.99939))	31.4	32.3
40	*actual data used*	39.1	39.1
56	1/(.99939/56+(1-.99939))	54.2	Est.
64	1/(.99939/64+(1-.99939))	61.6	Est.

World Record: 1.6 Terabyte Database

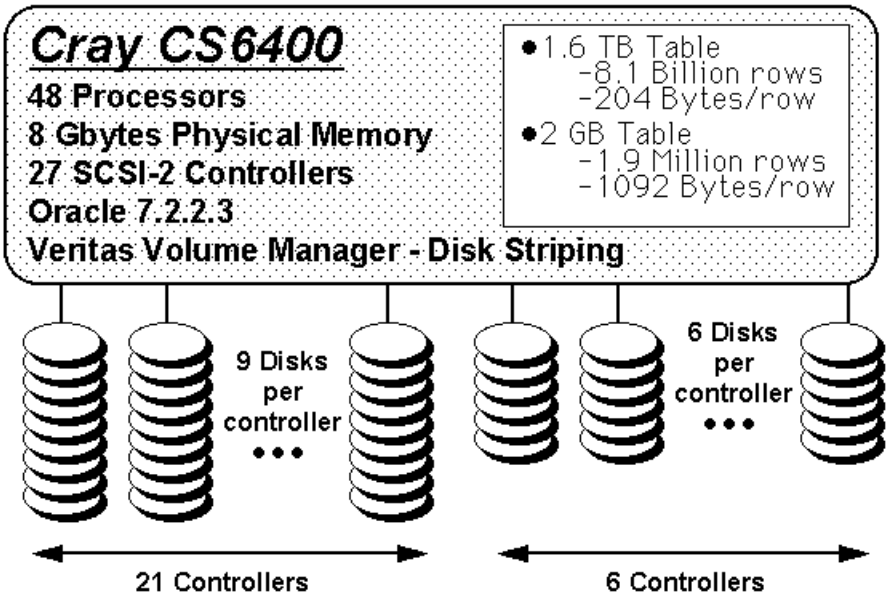
"Only the Biggest Baddest SMP can Handle a Mountain of Data"

- Largest Oracle Databases!
  - Query on 1.6 Terabytes Data.
- 8.1 Billion Records (rows)
- 1.6 Terabytes=1651 Gbytes
- Largest Table: 1.5 Terabytes of User Data
- 48 Processors

- *1.6 Terabyte book would be 55,958 feet high*
  - *80 char/line, 66 lines/page, 2 pages/sheet, 250 sheets/inch*
- *Mount Everest is 29,208 feet above sea level!!!*

SNDC.DSSscale.bw.R0.1 14

1.6 Terabyte Configuration



SNDC.DSSscale.bw.R0.1 15

Data Warehouse Performance
1.6 Terabyte Cray/*Oracle7*

- Full Table Scan
  - 17 Aggregates on 8.1 Billion Rows
  - 9 hours 30 minutes – Would take Weeks on Mainframe
  - Volume Managers Provide Performance and Ease of Use

- Nested Loop Join
  - Joined 1.6 Tbyte Table to 2 GByte Table
  - 10 hours – Would take Weeks on Mainframe
  - Index on Smaller Table
  - 840 Rows Returned – Executive Decision Support
  - Volume Managers Provide Performance and Ease of Use

SNDC.DSSscale.bw.R0.1 16

High Performance Database
"*Finally a Powerful SMP System*"

- High Capacity Data Warehouse
  - 1.6 Terabyte Demonstration, High Scalability
  - Up to 10 Terabyte Disk Capacity
  -
- Excellent Decision Support Scalability
  - Full Table Scan w/ 16 Aggregates: 39x Speedup, 40 CPUs
  - Nested Loop Join: 37x Speedup, 40 CPUs
  - 99.77% Parallel on Wide Variety of Queries
  -
- Excellent OLTP Scalability
  - Industry Leading TPC Benchmark™ B Results
  - Performance: 2025.20 tpsB
  - Price/Performance: $1,110.14 per tpsB

SNDC.DSSscale.bw.R0.1 17