Scalable Computer Architectures for Data Warehousing

By Mark Sweiger
President and Principal
Clickstream Consulting
msweiger@clickstreamconsulting.com

March 2002

Executive Summary

Choosing the appropriate computer architecture for your data warehouse is one of the most crucial infrastructure decisions you can make when building and deploying business intelligence applications. Many data warehouses have failed due to scalability problems that have their root causes in the underlying hardware architecture, not in other components like the database software and query tools, which are often erroneously blamed for all scalability problems.

While data warehouse infrastructure decision-makers tend to concentrate on technical details like individual component speeds and costs, they often neglect the key to scalability, which is the quality of the data warehouse machine's connection model. Choosing the fastest individual components or the lowest cost machine doesn't matter if the resultant system cannot possibly scale due to an inferior interconnect topology.

This paper compares five different machine architectures: the Sun Fire™ 15K Server, the NCR/Teradata 5250, the IBM p690 Regatta, the IBM RS/6000 SP, and the HP Superdome. The connection model of each machine is presented in a visual format, and the strengths and weaknesses of each architecture are revealed. The machine architectures that enable uniform data access times and high bisectional bandwidth are superior—those that do not are inferior. Each machine architecture is graded according to these characteristics, and the Sun Fire 15K is the only system that has both uniform data access times and a large bisectional bandwidth. Its unique, tightly-coupled, symmetric crossbar interconnect is the key to its superior scalability for parallel SQL data warehouse workloads.

The audience for this paper includes the management decision makers and the technical architects in charge of making hardware infrastructure choices for data warehousing applications.

1 Introduction

Data warehousing workloads are by far the largest and most computationally intense business applications at most enterprises. Consisting of huge time-history databases spanning hundreds of gigabytes to multiple terabytes of disk storage, data warehouses stress typical information system architectures to their limits. It is no accident that the rise of data warehousing coincided with the rise in large-scale parallel computers and parallel database software. Most medium to large-size data warehouses could not be implementable without larger-scale parallel hardware and parallel database software to support them.

But simply having a large parallel computing complex is not enough to guarantee robust implementation of a data warehouse. While the choice of data warehouse infrastructure components often concentrates on application-level software like query tools, the crucial component for success lies at deeper levels of the infrastructure—in the parallel machine architecture upon which all the layers of software are built.

Unfortunately, choosing the appropriate machine architecture for a particular data warehouse workload is often the result of one of two flawed decision-making paths:

- *The Fatalism Path:* The enterprise hardware vendor is already "X", and therefore the enterprise will use this hardware whether it is appropriate or not.

- *The Speeds and Feeds Path:* Using an alchemy-like process of analyzing individual components, like the speeds of processors, memory and disk drives, designated infrastructure specialists within the organization determine which machine has the "fastest" set of components for the "lowest" cost.

While few would argue that *Fatalism* is an appropriate way to choose a data warehouse machine architecture, the *Speeds and Feeds* method has a large following. While *Speeds and Feeds* are important, they are not the most critical factor in choosing an appropriate data warehouse machine architecture. The most critical factor, by several orders of magnitude, is one that rarely gets any attention at all, namely, the *connection model* of the computer.

Why is the machine connection model so important?

Most data warehouse infrastructure decision-makers concentrate on technical details like component speeds and costs, but they neglect the overriding issue—the quality of the data warehouse machine's connection model. Choosing the fastest individual components or the lowest overall cost doesn't matter if the resultant machine cannot possibly scale due to a bad connection model. The remainder of this white paper will explain this important concept in more depth, and show which commercially available machine has the best connection model.[1]

2 Asymmetric Architectures Prevent Scalability

Data warehouses depend heavily on fast parallel SQL operations, like parallel scans, sorts, joins, inserts, and deletes, as well as other parallel operations like parallel create index and parallel aggregate creation. The software algorithms that run these operations attempt to make as much use of the underlying parallel hardware as possible, utilizing multiple processors, multiple areas of memory, and multiple disk drives to get the operation done as quickly as possible.

Depending on the machine architecture, the data processed by parallel SQL operations has to reside in at least some of four possible containers—local memory, local disk, remote memory, and remote disk. In general, the threads of parallel operation execution take data from disk drives and funnel it through memory where it then can be processed by the machine's processors. For any given parallel operation, the underlying workload has to be evenly divided among the threads of execution to get the fastest possible execution time. If the workload is skewed toward only a few of the threads of execution, these threads will bear most of the burden of the workload, and parallel execution will be slowed.

The main cause of skewed workloads is skew in the underlying data being processed. If one thread has to process 10 times as much data as the other threads of execution, that thread will take 10 times as long to do its work and it will dominate the total parallel operation execution time.

The problem of skew is not confined to computers and was first encountered during industrial automation in factories. Optimal levels of parallelism in computers are achieved using a paradigm similar to that used to eliminate skew in industrial division-of-labor production problems. Consider the following division-of-labor example of factory workers finishing sheets of metal on three parallel assembly lines:

[1] For more supporting information on machine connection models and scalability, please read the paper, "Optimal Machine Architectures for Parallel Query Scalability", which was published by this author in Chapter 3 of the book "Building, Using and Managing the Data Warehouse", edited by Ramon Barquin and Herb Edelstein, Prentice Hall 1997.

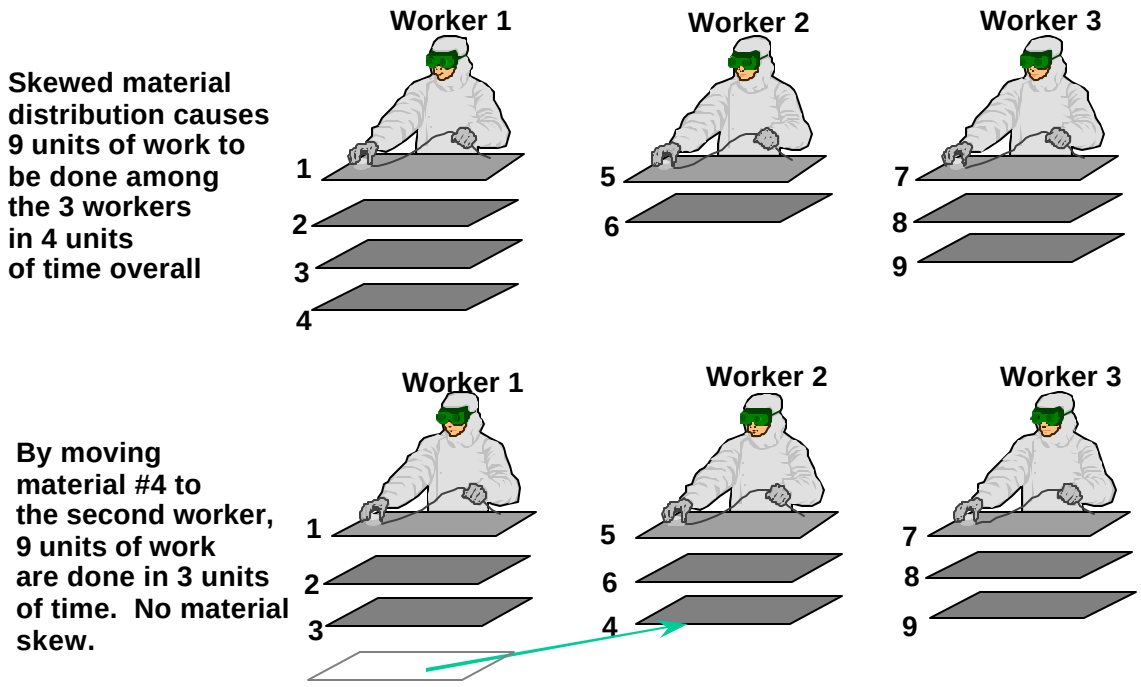

Figure 1: Uniform material distribution gets the most work done in the shortest period of time.

Initially, Worker 1 has four sheets of material, Worker 2 has two sheets of material, and Worker 3 has three sheets of material to process. Because Worker 1 has the most sheets, the total time for all the workers to process all the material is dominated by the time Worker 1 takes to finish his four sheets. The other workers go idle as Worker 1 completes his work. But by redistributing the raw material to the workers evenly, as shown in the second part of the diagram, material skew is eliminated. With uniform material distribution, all nine units of work are completed, in parallel, by the workers in only three units of time, compared to four units of time when the material distribution was skewed. We should note that if sheets of metal were data and the workers were CPUs, the result would still be the same.

Another cause of skew is latency. A nearby item can be processed quickly, with no time wasted getting it. But if an item is far away, in a remote material warehouse or a remote computer node, there can be considerable latency in getting the item to point of processing. Again we illustrate this with an industrial division-of-labor example:

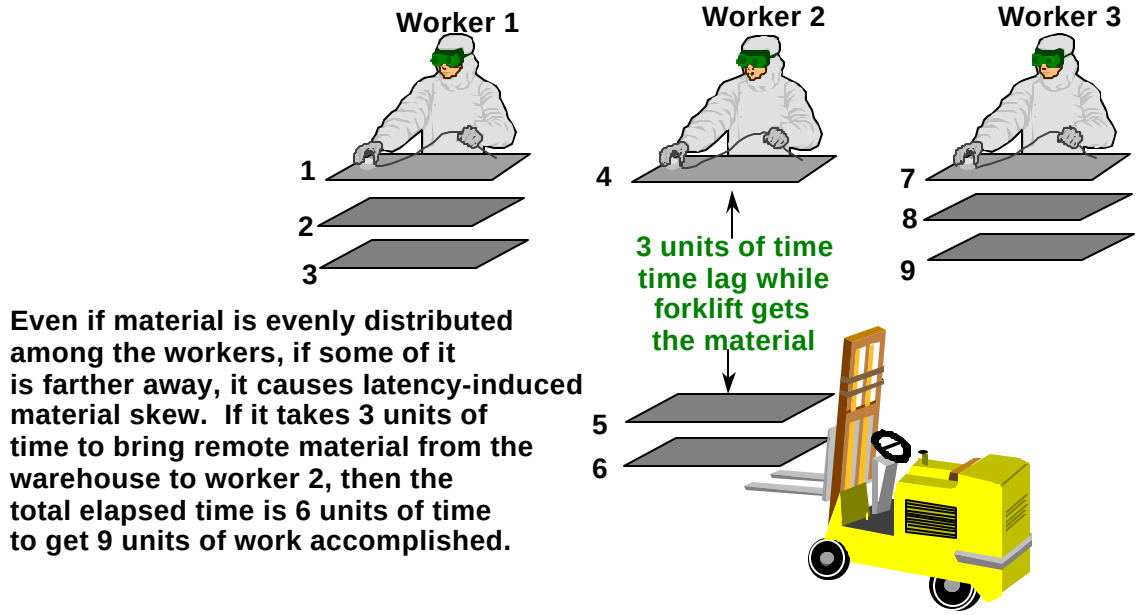

Figure 2: Latency caused by material in a remote location reduces productivity.

Because the material Worker 2 needs is far away, and needs to be forklifted from the warehouse to Worker 2 with a delay of three units of time, this additional latency adds three units to the completion time for the total workload. Again, if the sheets of metal were data and the workers were CPUs, the result would still be the same.

Suppose it were possible to give Worker 1 and Worker 3 more local work by putting three more items into each of the local assembly lines of these workers:

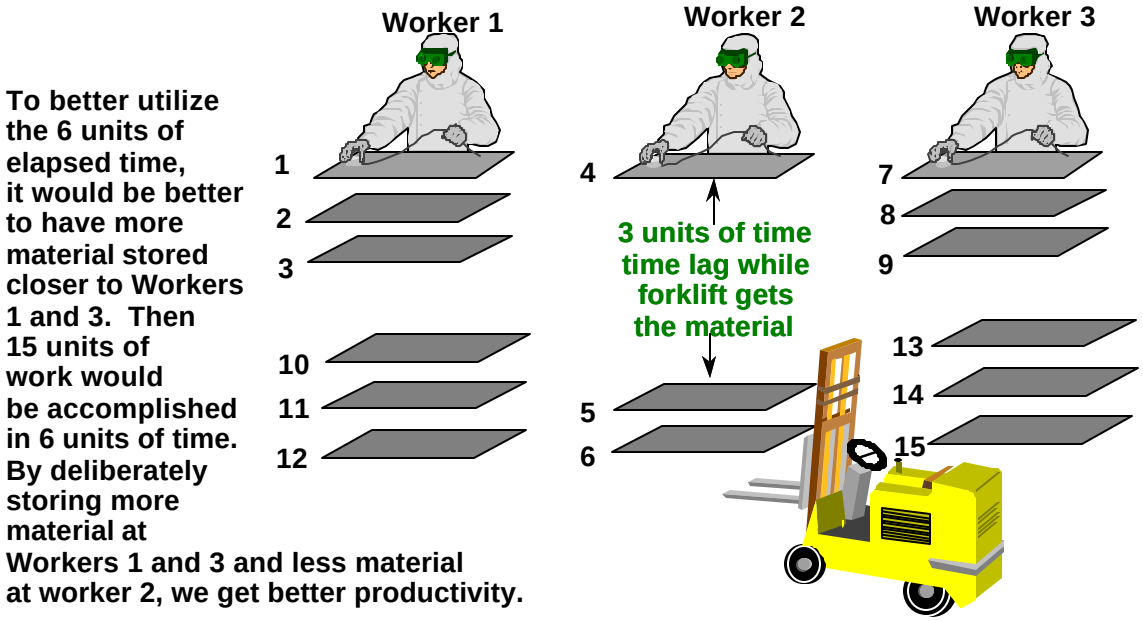

Figure 3: Adding material to the lines of Workers 1 and 3 increases productivity, but it is not optimal.

Adding the extra material keeps Workers 1 and 2 busy while the forklift goes to the remote warehouse to get more material for Worker 2. But the total production is still not optimal due to the latency in getting the material for Worker 2.

Productivity can be optimized by completely eliminating latency and, therefore, material skew as shown in the following diagram:

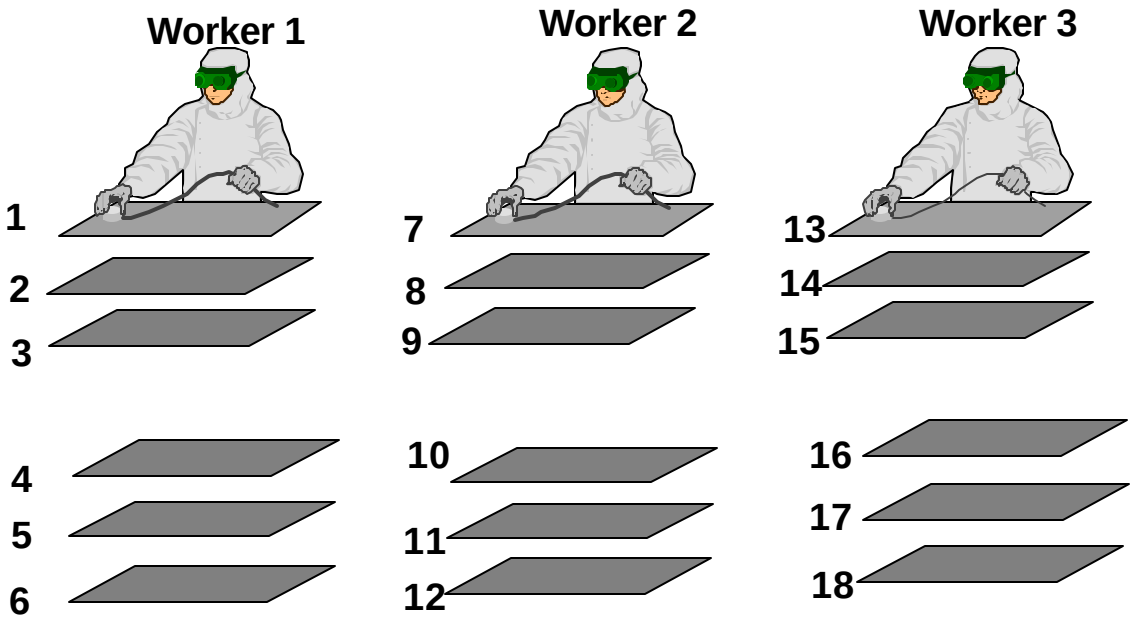

Figure 4: By uniformly distributing material among the workers without latency, 18 units of work are completed, three more than in the previous, latency skewed example.

As you can see, skew, whether resulting from asymmetric distribution of material across assembly lines or from latency getting material to production lines, decreases the efficiency of parallel production in any industrial application. The same is true inside a computer, which is analogous to a factory in which the workers are the CPUs, the assembly line is the local memory and the local disks, and the warehouse is the remote memory and the remote disks. The lessons of industrial production show us that uniform distribution of material between workers with a minimum and uniform latency to get the next item for processing produces the maximum level of production. The same is true of computer hardware platforms running parallel data warehouse software.

3 Symmetric Architectures Enable Scalable Data Warehouses

The loss in productivity caused by latency-induced material skew in industrial applications is significant, but the latency-induced loss of performance in computer architectures is much more significant. Consider the huge magnitude of the differences in access times for the various containers used by hardware architectures, shown in the figure below:

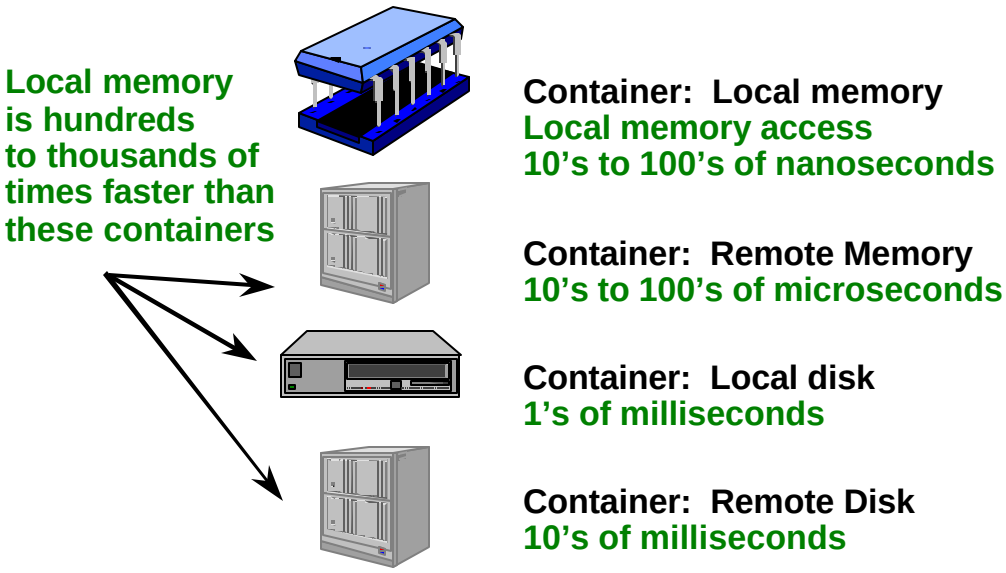

Figure 5: In our division of labor example, material skew, caused by the need to forklift material to Worker 2's assembly line, caused a delay of three time units. But in computer systems, the delays between the various containers for data are much more significant.

Because of the enormous relative differences in latency between the components in a typical computer, the corresponding industrial factory latency of a few minutes to get some material from the warehouse would translate into a delay 100 or 1,000 times that size inside a computer. It is as if a factory wait of 30 minutes for a forklift turned into 30 x 100 =3,000 minutes (50 hours), or 30 x 1,000 = 30,000 minutes (nearly three weeks) for data from a remote node or disk. This is why data skew, and its relationship to the corresponding machine connection model, totally dominates a machine's price/performance equation.

It is very important to understand that latency-induced data skew is caused by the machine architecture itself. The relative distance between platform components is a defining characteristic of a hardware architecture, and those architectures with local and remote nodes have the widest variation in access times and therefore the highest level of *machine architecture induced data skew.* Many hardware vendors will claim that machine architecture induced data skew is a software problem that is solvable by having database administrators reload database data in various ways to mitigate skew. Reloading the data not only won't help, but it is simply a smokescreen to divert attention from the hardware skew that is inherent in the particular machine architecture.

Which machine architecture prevents data skew?

Symmetric multiprocessor (SMP) machine architectures eliminate data skew because they support uniform access times between all the components of the machine by definition. A classic SMP machine architecture is shown in the figure below:

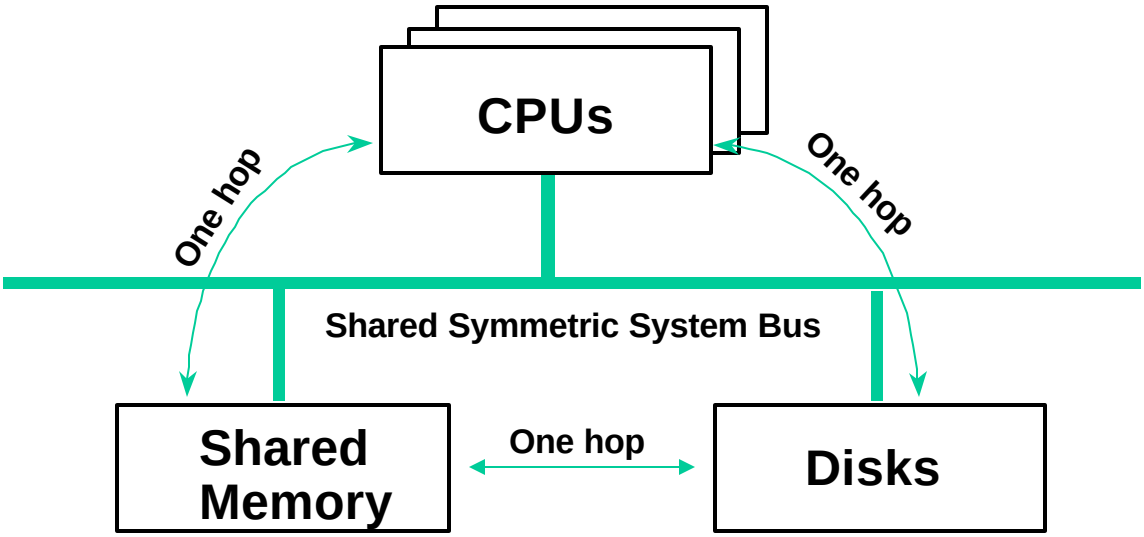

Figure 6: All components are equidistant in classical SMP architectures.

Symmetric multiprocessors have always had uniform access times between system components, and this is why SMPs have led in scalability, whether the workload was online transaction processing or parallel SQL data warehouse operations. Classical SMPs are characterized by a direct connection model where all components are equidistant and directly connected to one another using a single system bus. In this machine architecture, variations in access times are a result of individual differences in component speeds and nothing more. Disks are the slowest components on an SMP, and a large memory database page cache, as well as data layout strategies like disk striping and key-range and hash partitioning, are used to mitigate their slow access times.

It is striking that of the four major vendors of data-warehouse-scale computers—Sun Microsystems, NCR/Teradata, IBM and HP—only one vendor uses a uniform access time SMP architecture for its largest data warehouse oriented systems. All the other vendors use different asymmetrical connection models that have inherent machine architecture induced data skew. In the next four sections we will explore the connection models of Sun, NCR/Teradata, IBM and HP, and expose the strengths and weaknesses of each architecture.

4 Sun Microsystems Sun Fire 15K Server

Sun Microsystems is the only data warehouse computer vendor offering an SMP architecture system with uniform access times throughout its product line. Its largest machine, the Sun Fire™ 15K Server, is a highly scalable system suitable for the biggest data warehouse applications. Instead of the single-bus classic SMP architecture, the Sun Fire 15K Server uses an innovative crossbar centerplane, called the Sun™ Fireplane. A diagram of the Sun Fire 15K crossbar topology architecture is shown below:[2]

[2] The material in Section 4 is derived from the "Sun Fire 15K, Detailed View" paper, http://www.sun.com/servers/highend/sunfire15k/details.html

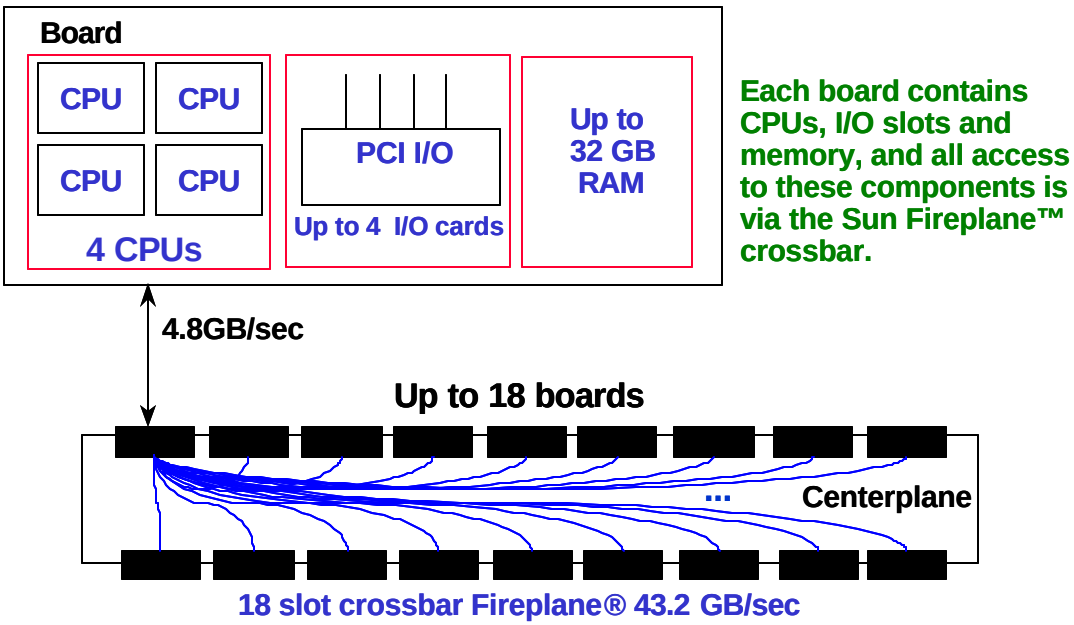

Figure 7: The Sun Fire 15K crossbar-topology centerplane SMP

With a crossbar-topology centerplane, every board is directly connected to every other board, meaning that no interconnect requests have to share the same bus, as is true in classic SMP architecture machines. The crossbar connection model for the first slot is shown in the diagram, with 17 direct connections to each of the 17 other boards. The connection model for all the other slots is similar and all boards are a symmetrical distance from one another.

The industry-standard mechanism for calculating the bandwidth of a system's interconnect is to determine its bisectional bandwidth. When a system interconnect is bisected, exactly half the computing complex lies on one side and exactly half lies on the other side of the bisection. The bisectional bandwidth is calculated by determining the total available interconnect bandwidth between the two halves of the bisection. Furthermore, if the two halves of the machine have the same bisectional bandwidth no matter which way the bisection is performed, then the machine has a symmetric architecture.

The Sun Fireplane is a symmetric crossbar, with 18 slots, nine on each side of any bisection. The clock rate of the crossbar interconnect is 150 MHz and the crossbar is 32-bytes wide. To get the bisectional bandwidth, we multiply the number of slots times the clock rate and the byte-width of the interconnect. We then take this quantity and halve it to get the bisectional bandwidth. This measurement can be thought of as the potential bandwidth available for nine slots on one side of the crossbar to continuously communicate with all nine slots on the other side of the crossbar at maximum speed. So, the bisectional bandwidth of the Sun Fireplane is:

(18 slots * 150 MHz * 32-bytes) / 2 = 43.2GB/sec

The crossbar-topology Fireplane is a very significant innovation in interconnect design compared to the single bus classic SMP. In a single bus SMP, all the boards share the bandwidth of the single bus, quickly overwhelming the bus with more traffic as more boards are added. Furthermore, since all boards use the same bus, only one bus access can occur at a time, meaning

that other potential parallel bus traffic from other boards has to wait while the first traffic finishes. Even worse, different kinds of bus traffic, especially memory and I/O address calculations, place hold cycles on the bus, causing empty bus cycles that reduce the available bus bandwidth.

The Sun Fire 15K crossbar-topology interconnect solves these problems on every level. The interconnect design consists of three identical crossbars that segregate address, response, and data traffic onto their respective crossbars, as shown in the figure below:

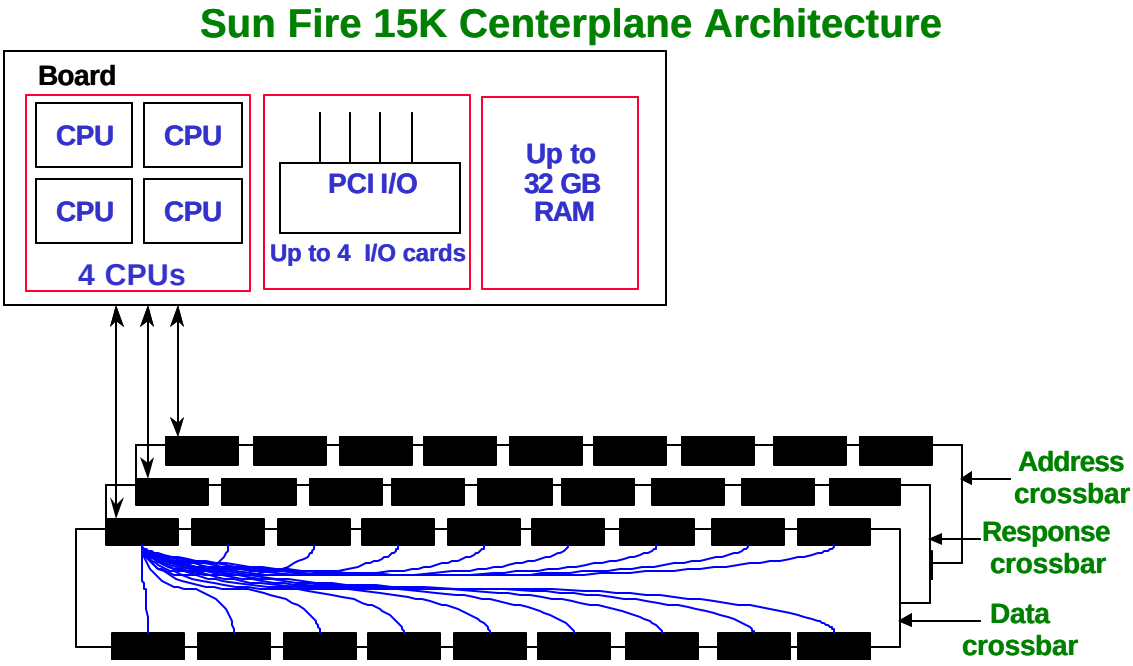

Figure 8: The Sun Fire 15K centerplane consists of three identical crossbars that segregate otherwise contentious address, response, and data traffic, maximizing bandwidth.

The three separate crossbars allow address, response, and data traffic to proceed unimpeded in parallel. The crossbar topology means that every possible type of transfer has an unblocked connection, maximizing parallel throughput of the interconnect.

The maximum configuration of the Sun Fire 15K Server for data warehouse type workloads is as follows:

- 72 processors (4 CPUs/board x 18 slots)
- 576 gigabytes of memory
- 72 PCI I/O slots
- 43.2 GB/sec of sustainable centerplane bandwidth

The Sun Fire 15K Server can hold up to 34 more processors by using unneeded I/O-assembly slots, but data warehouse workloads typically need all the available I/O slots to connect their massive disk farms and spread the resultant I/O over as many slots as possible.

The Sun Fire 15K Server has one other huge advantage for data warehouse workloads—its superior ability to dynamically redistribute data during parallel SQL operations. In order to understand this, we need to understand how parallel SQL works.

Even the simplest parallel queries can be adversely affected by the inability to dynamically redistribute data during execution. Consider the following query:

SELECT * FROM TableA ORDER BY Column2

This simple query parses into the three-node query parse tree shown below:

SQL query parse tree for "SELECT * FROM TableA ORDER BY Column2"

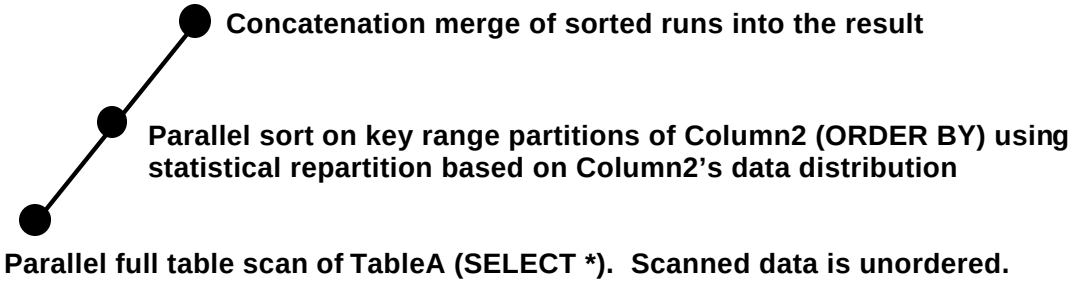

Figure 9: Query parse tree for a parallel query that selects all the rows in TableA and orders them by the key values in Column2.

The query parse tree is executed from the bottom up. Suppose the degree of parallelism is nine. The optimizer starts nine parallel scan processes to do a full table scan of the table and nine parallel sort processes to sort the resulting output. They are arranged in a producer-consumer pipeline as shown in the diagram below:

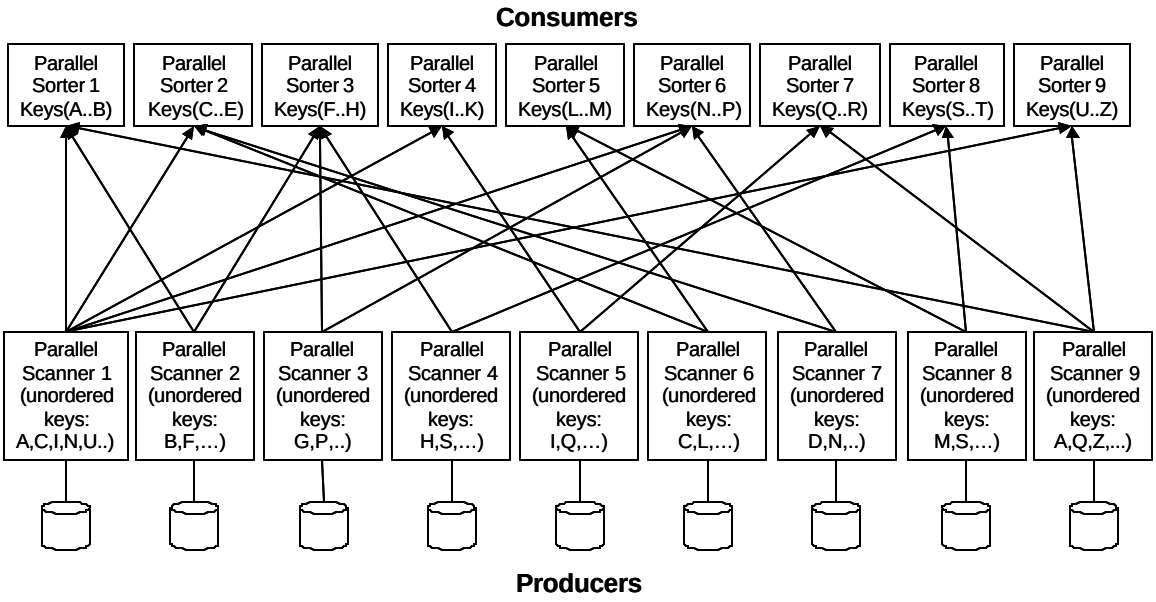

Figure 10: Parallel SQL producer/consumer process pipeline with dynamic data redistribution

Data in the TableA is not ordered by the keys in Column2, which for the sake of simplicity in this example are the letters of the alphabet [A..Z]. The data can be in any order, and the first few keys of each segment of the database are shown in the scanner process boxes above. Because the degree of parallelism is nine, the nine scanning processes divide the table into nine segments of approximately the same number of rows. For example, if there were 90,000 rows, each scanning segment would contain about 10,000 rows. As the scanners read the rows in TableA, they do a statistically significant sampling of the key values to determine the data distribution of the keys in Column2. This sampling determines which set of key ranges on Column2 divides the table into equal-sized buckets of 10,000 rows with those key ranges. The parallel sort processes then begin to consume the rows scanned by the scanning processes according to this key-partitioning scheme. As you can see in the diagram above, keys can go anywhere depending on their key values. This process is called *dynamic data redistribution*. Once all the keys are scanned and sorted, the sort buckets are concatenated according to their key order and that is the query result.

Do you notice the similarity of the data paths between the producer/consumer pipeline and the crossbar interconnect of the Sun Fire 15K Server?

The scanning producer processes can conceivably send rows to any of the consumer processes, depending on their Column2 key values. If we were to draw in the data redistribution lines for all the keys in the table, we would eventually connect every point on the producer side with every point on the consumer side. This connection model is very nearly a crossbar, and it becomes a crossbar if we add interprocess coordination messages between the various scanner processes and the various sorter processes. Below is a diagram of this situation, inserting the Sun Fire 15K crossbar centerplane as the data redistribution mechanism between the producer and consumer processes:

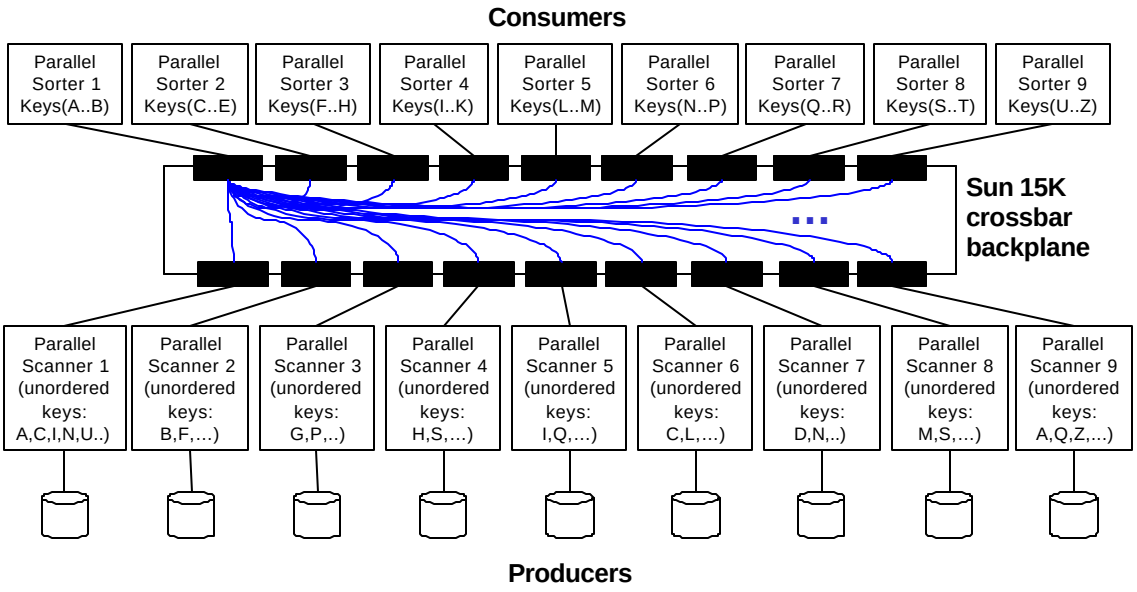

Figure 11: Dynamic data redistribution using the Sun Fire 15K crossbar centerplane.

Using the Sun Fire 15K crossbar, parallel scanner processes can send their rows to any sorter process in one nonblocking hop across the centerplane. And since there can be a huge 576 GB shared memory backing the parallel SQL database software, rows already in memory do not actually need to be physically moved from scanning to sorting processes. Only the memory address of the rows needs to be sent from one process to the other, further increasing dynamic data redistribution efficiency.

Most parallel SQL operations are significantly more complex than the one shown above. There may be joins between tables, WHERE clauses, multiple sorts on different keys (ORDER BY, GROUP BY, …), and other types of complex parallel operators that need to do dynamic data redistribution multiple times during the course of operation. The more effective the computer architecture is at dynamic data redistribution, the better it is able to maintain the same degree of parallelism throughout all the steps of the operation. The Sun Fire 15K connection model is optimal for data-redistribution-heavy data warehouse workloads.

5 NCR/Teradata

NCR is the only major computer vendor whose entire server product line is devoted solely to data warehousing and business intelligence applications. This may seem odd, since most server-style computers are general-purpose machines that can do a wide range of computational tasks, going well beyond data warehousing. The major reason why these servers are data warehouse specific is their unique connection model, which does not support general-purpose workloads. This connection model, a loosely-coupled banyan network called the BYNET®, is a legacy of NCR's acquisition of Teradata in 1991. Teradata® still operates as a separate NCR division, creating the proprietary database software and associated services for the data warehouses and business intelligence applications that run on the NCR WorldMark™ series servers.[3]

[3] The NCR 5250 series machine information in Section 5 is derived from the NCR/Teradata 4851/5251 and 4855/5255 Product Guide and Site Preparation Manual, Release 1.2, B035-5551-111A, November 2001.

NCR WorldMark servers consist of collections of 4-processor SMPs connected by a banyan network. The largest model, the 5250 series, can have up to 128 nodes with 512 processors. A diagram of a fully configured 5250 series is shown below:

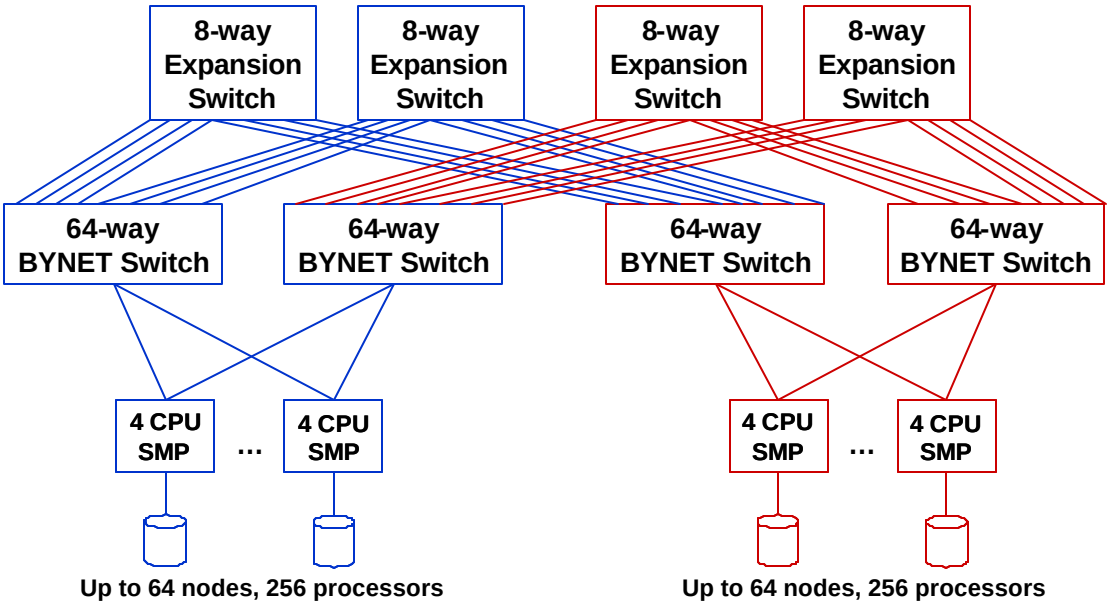

Figure 12: NCR/Teradata 5250 Series Connection Model

Each 4-CPU SMP uses 700MHz Intel Xeon processors, and each leg of the Fiber Channel based BYNET interconnect runs at a the NCR-specified rate of 120 MB/sec, somewhat less than the potential rate of 132.8 MB/sec rate of the Fiber Channel medium. Each node has two BYNET connections that go to dual redundant 64-way BYNET switches. The butterfly-like banyan interconnect topology means that the redundant connections also produce a doubling of the number of nodes that can be connected using the BYNET, to a maximum of 128 nodes redundantly attached over two 64-way switches. The second BYNET connection at each node is used only as an alternate transmission path, which means that the total available interconnect bandwidth at any node is the equivalent of one connection, or 120 MB/sec. This also means that the total interconnect bandwidth available for a complex of 256 processors is 64 * 120MB/sec or 7.68 GB/sec.

Two of these paired 64-way node complexes can be connected using four 8-way expansion switches, which is shown in the top tier of the diagram above. The resultant machine has twice as many nodes, 128, and twice as many processors, 512. It does *not*, however, have twice the bisectional bandwidth, which is a physical requirement for linear scalability. Look at the following diagram, which shows the bisection of the interconnect:

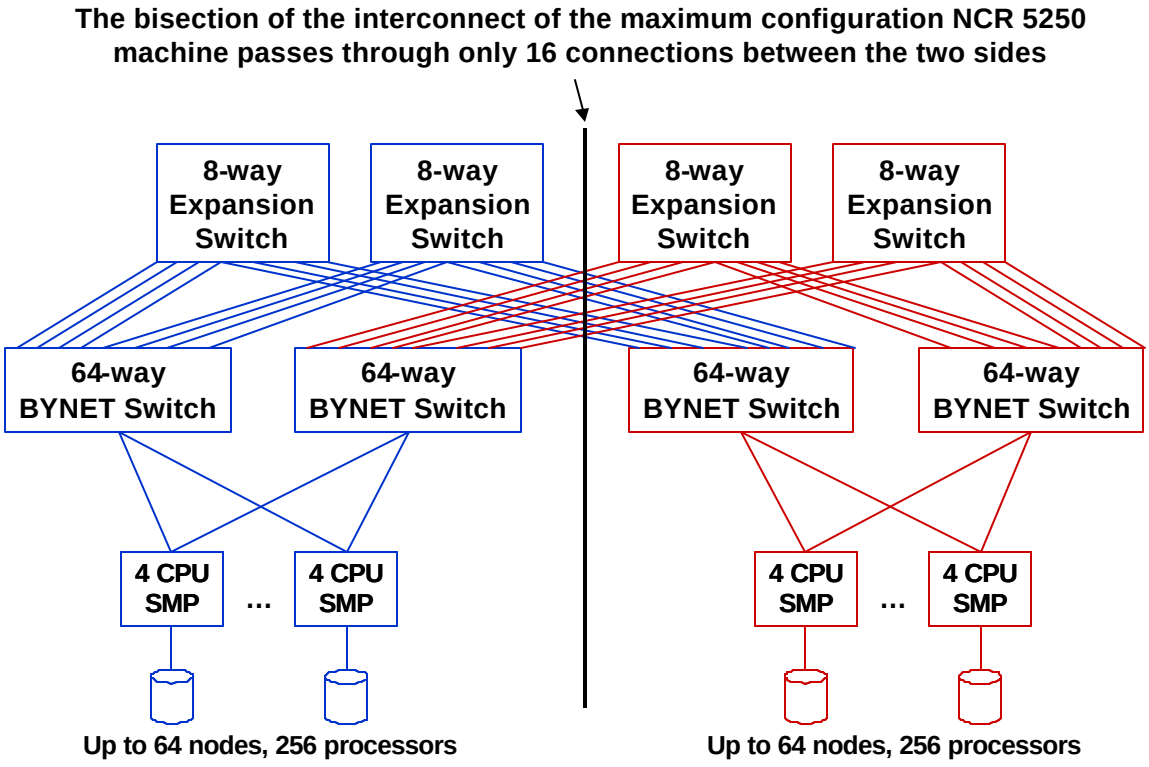

Figure 13: Bisection of the interconnect of the maximum configuration NCR 5250 Series machine

As you can see, the bisection cuts the machine in half and goes through only 16 interconnection lines. This means that the two massive halves of the system, each with 256 processors have only 16 * 120 MB/sec or 1.92 GB/sec of bisectional bandwidth between them, considerably less than the 7.68 GB/sec of interconnect bandwidth available within each half. This skinny bisectional interconnect will become a bottleneck in any parallel SQL operation that spans both sides of the computing complex and needs to do even a relatively small amount of dynamic data redistribution. Given the high likelihood of data redistribution in most query workloads, this is a major architectural problem for this interconnection model.

This interconnect also has the usual message-passing latency problems associated with loosely-coupled architectures. The maximum configuration would take three hops across three switches to get a row from one side of the machine to the other side. Message passing induces extra latency compared to tightly-coupled shared memory access models, where the equivalent to message passing is, at most, a shared memory copy operation, or, more often, simply passing an address in shared memory between cooperating threads of parallel execution. The magnitude of the time difference between passing an address in shared memory versus sending large blocks of data through three levels of interconnect adds on the order of 1,000 to 100,000 times more latency, depending on the size of the data being transferred. This cannot possibly improve throughput.

Another issue is the asymmetric nature of the BYNET interconnect model. Depending on where data is located, it may take no hops (the producer process and consumer process are local to the SMP node), one hop (the producer process and consumer process are on the same 65-way switch), or three hops through the BYNET (the producer process and consumer process are on

opposite halves of the maximum configuration) to get to its destination. This means that various threads of execution participating in a parallel SQL operation will encounter different latencies depending on data location, and this fact will cause non-uniform access times resulting in data skew and poor parallel performance.

Although Teradata literature makes many claims about their massive interconnect bandwidth, and how the BYNET is not a bottleneck, the design of Teradata database software does not credibly support these claims. There are many unique characteristics to this software, and some of the highlights include:[4]

- All database tables must have a hashed data organization. The underlying data access nodes, called Access Module Processors or AMPs, hash the rows according to a primary key across all the AMPs defined for a particular database. While hashing is a good randomizing data organization, other useful data organization choices, including unordered and clustered row organizations are not supported as they are in other open systems RDBMS software like Oracle and DB2. In the Teradata environment, hashing is an enforcement mechanism designed to prevent data skew, but even that is thwarted by the asymmetric connection model of the underlying hardware.

- Parallel joins, especially outer joins, are problematic using Teradata software. Joins are one of the most common relational operations, and parallel joins often require dynamic data redistribution capabilities to maintain a high degree of parallelism during their operation. While equi-joins between tables *on the primary hash key only* will perform well in the Teradata architecture, joins on non-hash key columns and outer joins (which answer valid business questions like "Which customers did *not* buy products from product line X"), do not do well on the banyan interconnect. This is because non-primary key equi-joins, and outer joins require massive data redistribution to perform the operations efficiently in parallel. To counter this problem Teradata has created so-called join indexes, which are essentially pre-computations of these joins that look a lot more like adjunct tables than indexes. Other RDBMS software that runs on other symmetric architectures simply does not have this problem.

- Teradata software is not portable. One of the advantages of using open systems RDBMS software for a data warehouse is that one can move from one machine vendor to another and take the same RDBMS software to the new system. Even though Teradata software is compliant with many parts of the SQL standard, it cannot run in external vendor environments, with one exception. Teradata is not portable in the Unix environment, because it requires the MP-RAS version of NCR Unix SVR4. This version of Unix has hundreds of extra system calls that specifically support low-level Teradata database operations, like key comparisons and message passing, which are a legacy functionality of Teradata's old TOS operating system. These system calls are not present in other vendors' versions of Unix, so Teradata software does not port to them. Regarding the portability exception mentioned above, Teradata has ported its software to Windows 2000 (NT), but only for systems up to 16 nodes, far smaller than the 128-node system discussed here. Presumably, Teradata had to rewrite the RDBMS software to use standard Windows 2000 functionality, since it would not have all the MP-RAS-style system calls.

[4] The Teradata software information in this section is derived from Teradata RDBMS SQL Reference-Volume 1 Fundamentals, V2R4.1, B035-1101-061A, June 2001, and Teradata RDBMS Database Design, B035-1094-061A, July 2001.

In short, the Teradata database software does everything it can to avoid using the bandwidth-constrained BYNET, especially operations that could cause large-scale dynamic data redistribution. This is perfectly logical given the asymmetric, loosely-coupled connection model of the underlying computer. But accommodating this exotic connection model has driven NCR and Teradata down a proprietary implementation path that makes them the least open of the data warehouse system architectures, with all the negatives that implies.

6 IBM p690 Regatta and RS/6000 SP

IBM makes two machine architectures that are sold into the data warehousing market, the more tightly-coupled IBM pSeries servers, the largest of which is the newly released IBM eServer pSeries™ p690, and the loosely-coupled, massively parallel RS/6000® SP™. Both will be discussed here.

6.1 IBM eServer pSeries 690 System

The IBM eServer p690 is the latest and largest tightly-coupled pSeries Unix system sold into the data warehouse marketplace. Although IBM literature says this system is a new topology "distributed switch" system, further analysis reveals that it is a more prosaic non-uniform memory access system interconnect.[5]

The existing IBM p690 literature also makes many references to the high bandwidth of the system. While its components are fast, to date the literature does not specify bandwidth figures for the interconnect components of the system, referring only to clock rates and data widths, leaving the bandwidth calculations to the reader. We will do the math for you in this section.

Five years in the making and code-named Regatta, the p690 has the multiple-ring, asymmetric memory access architecture shown in the diagram below:

[5] The material in Section 6.1 is derived from the IBM white paper, POWER4 System Microarchitecture, October 2001 http://www-1.ibm.com/servers/eserver/pseries/hardware/whitepapers/power4.html

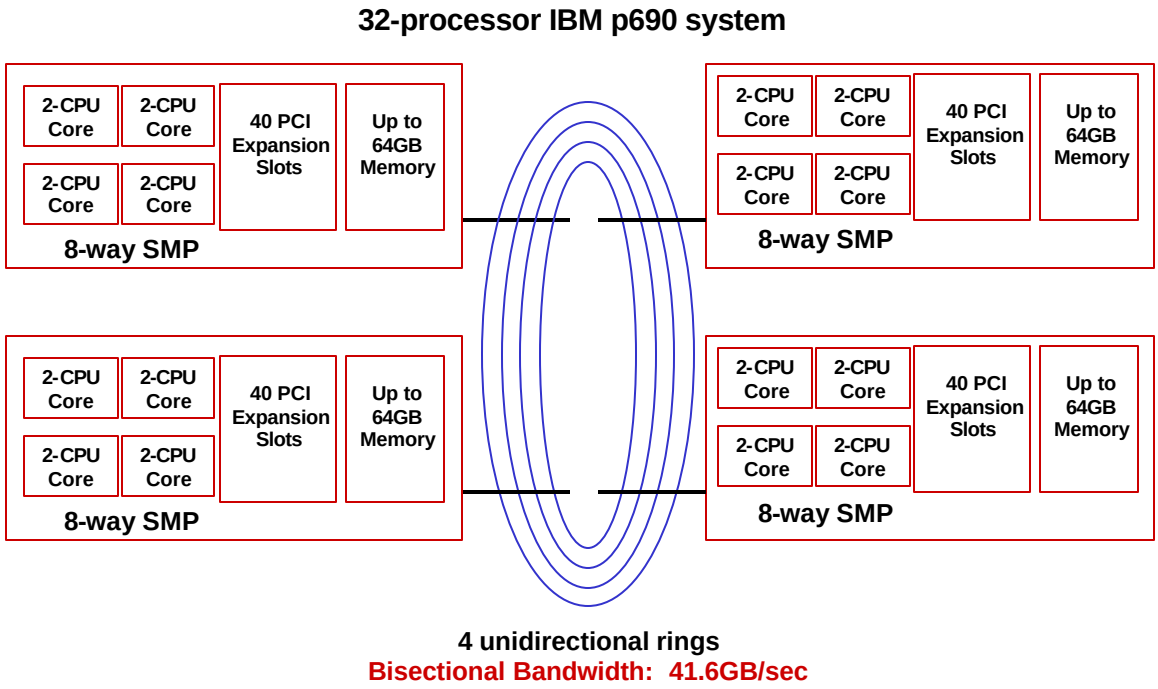

Figure 14: IBM p690 Architecture

The p690 consists of four 8-way SMPs each having up to 64GB of memory and 40 PCI I/O expansion slots. The 8-way SMP on a board is called a Multi-Chip Module (MCM). Each of the four SMPs are connected by four 8-byte wide rings, running at one half the processor speed of 1.3 GHz. If we bisect the ring topology interconnect, we cut through the ring lines eight times, four on one side, four on the other. As a result, bisectional bandwidth for the system interconnect is:

(8 * 8 bytes * 1.3 GHz) / 2 = 41.6 GB/sec

All the traffic between the SMPs travels across the unidirectional quad-ring interconnect. Latency varies depending on how distant the sender and target MCMs are from one another, anywhere from one to three hops around the ring.

While the more tightly-coupled IBM p690 ring-interconnect is an improvement over the loosely-coupled MPP systems like the NCR 5250, it still suffers from significant non-uniform data access times that cause machine architecture induced data skew during parallel SQL operations. Because of this, the non-uniform p690 is not a good substitute for a uniform, crossbar centerplane SMP, like the Sun Fire 15K Server, which shows much better scalability characteristics.

6.2 IBM RS/6000 SP

The RS/6000 SP is a crossbar-switched, loosely-coupled, massively parallel system. The system consists of 2 to 128 nodes. In the Unix environment, standard SP nodes come in three varieties named after their physical size characteristics—Thin, Wide, and High—with Thin being the smallest and High the largest node. IBM pSeries tightly-coupled multiprocessor machines can also be attached to the SP switch elements through their I/O expansion slots, a strategy that is

usually used for high-availability clusters more than for scalable data warehousing, because of the relatively low bandwidth of I/O slots.[6]

The SP crossbar switch element has recently been upgraded to the Switch2®, which more than triples the old per-port SP Switch transfer rate 300MB/sec to 1GB/sec for the Switch2. The increased bandwidth is mostly a result of the new ability to connect Switch2 adapters directly to a node's bus, rather than the old method of having an indirect connection through an I/O expansion slot. The only type of node that currently supports the faster SP Switch2 switching element is the High node, and both are depicted in the figure below:

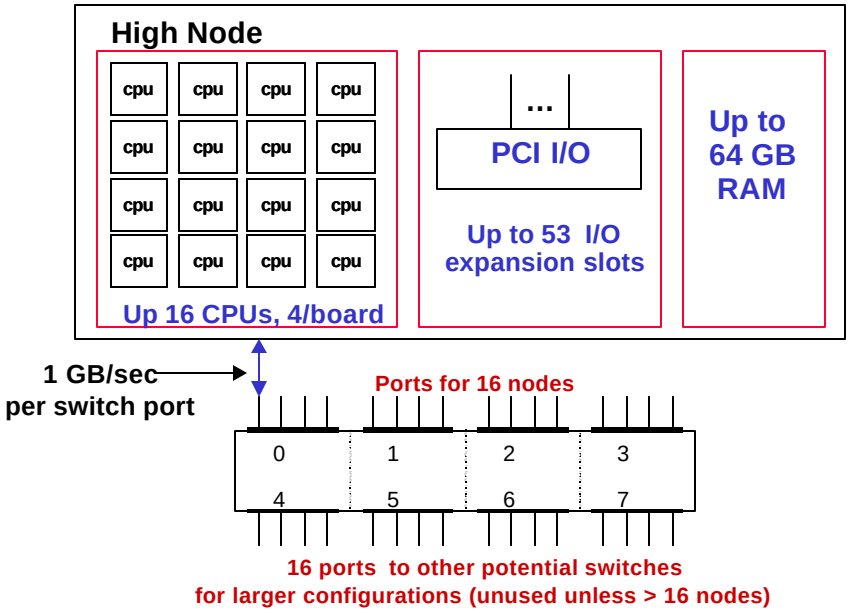

Figure 15: A single SP Switch2 switching element with the new High node

The new High node can have up to 16 processors, 64 GB of RAM, and 53 PCI expansion slots with the SP Expansion I/O units. These nodes are connected to 32-port crossbar SP Switch2 switching elements, with 16 ports used for nodes and 16 ports used to connect to other potential switching elements. There can be a total of 128 nodes spanning 16 switches in a standard SP configuration.

Each Switch2 switching element is comprised of eight 8-way switching chips, configured as a crossbar. Each chip has four external ports and four internal connections to the four switching chips on the other side of the switching element. The chips are numbered 0-7 in the diagram above, and the 0-3 ports are always used for nodes, and the 4-7 ports are always used for connections to other switches.

SP configurations start with a single crossbar switching element, which can support 16 nodes, with the other 16 ports unused for additional switching element connections. Each of the 16 switch ports transfers data at 1GB/sec, and if we bisect this 16-way crossbar, we get a bisectional bandwidth of:

[6] The information in Section 6.2 is derived from the IBM RS/6000 SP Manual "Planning, Vol. 1, Hardware and Physical Environment", GA22-7280-10

(16 * 1GB/sec) / 2 = 8GB/sec of interconnect bandwidth for 16 nodes on a single switch

By adding additional switches, the number of nodes increases in groups of 16, up to five switches. We show these configurations in a series of diagrams, starting with 32 nodes spread over two crossbar switches:

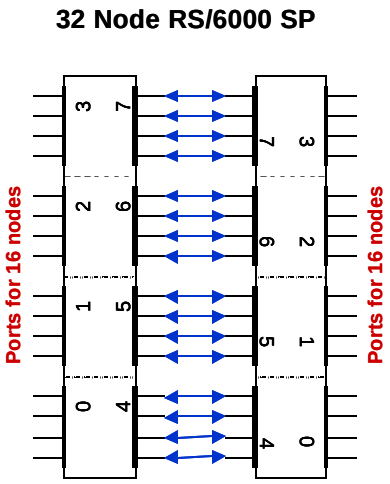

Figure 16: 32-nodes with two Switch2 crossbar switches

The next diagram shows 48 nodes spread over three crossbar switches:

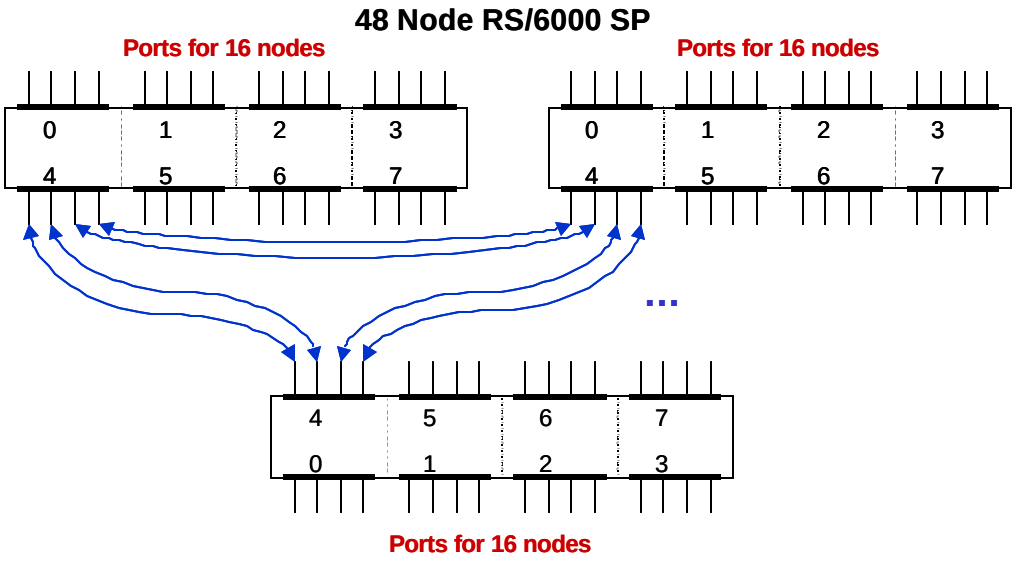

Figure 17: 48-nodes with three Switch2 crossbar switches

In order to better illustrate the triangular topology of the three-switch connection model, we only show the connections for the ports of switch chip 4 in the figure above. The connection topology is the same for the other switch chips, 5-7.

The next figure shows 64 nodes spread over four switches:

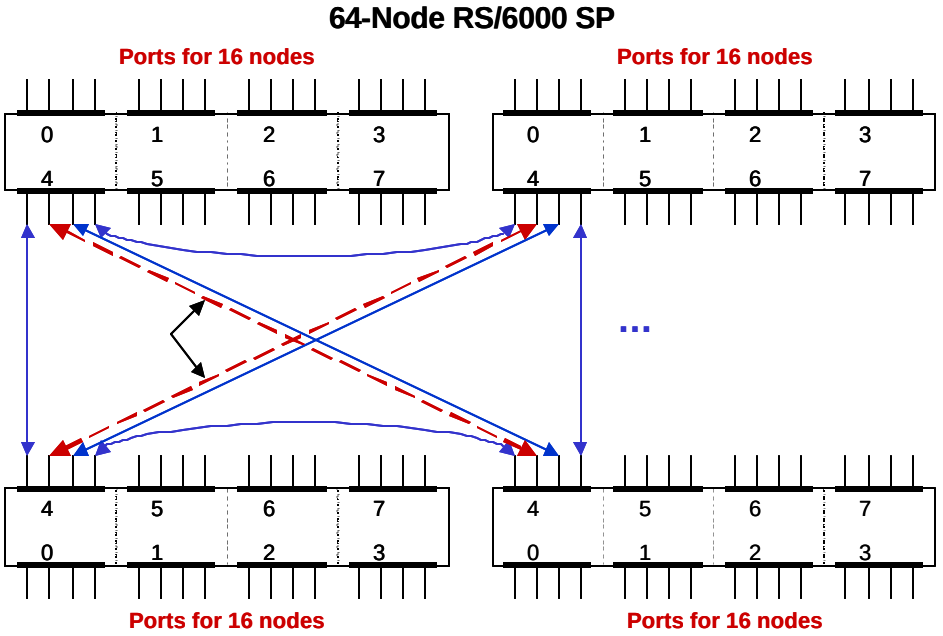

Figure 18: 64 nodes spread over 4 crossbar switches.

The four-switch model has enough switching ports to allow an asymmetric bandwidth connection model, which is shown above by the two additional, dotted-line crosswise connections. In the interest of preserving a symmetric architecture, these two extra connections would not be added for data warehouse workloads.

The next diagram shows 80 nodes spread over five crossbar switches:

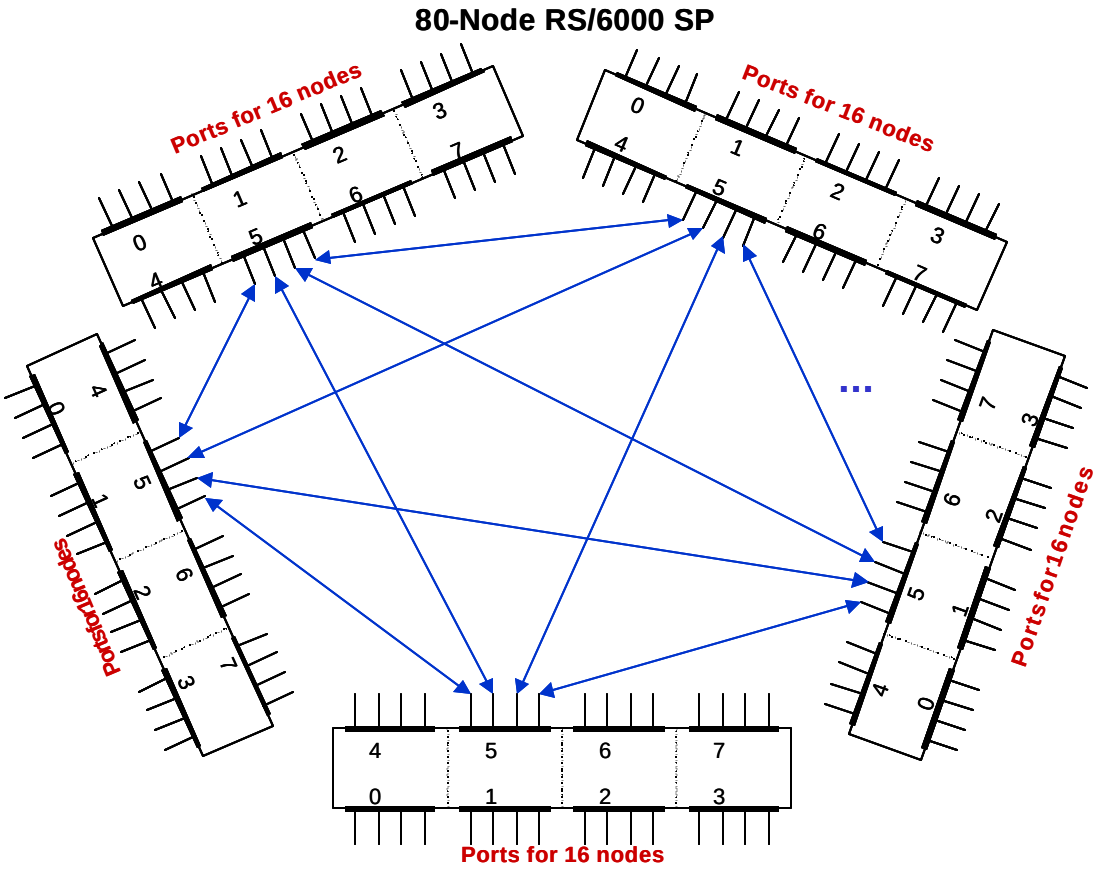

Figure 19: 80 nodes spread over 5 crossbar switches

We show only the pentagonal connection model for switch chip 5, but the topology is identical for the other switch chips, 4, 6 and 7.

At this point in the connection model, there is only one connection to another switch for each port on a switch chip (chips 4-7). If we were to attempt to symmetrically connect more nodes with more switches, every switch chip would no longer have a direct connection to all of the other switching elements, breaking the crossbar topology. Therefore, 80 nodes spread over five switches is the largest possible configuration without the introduction of another level of crossbar switching. This two-level topology, for configurations between 81 and 128 nodes, is shown in the figure below:

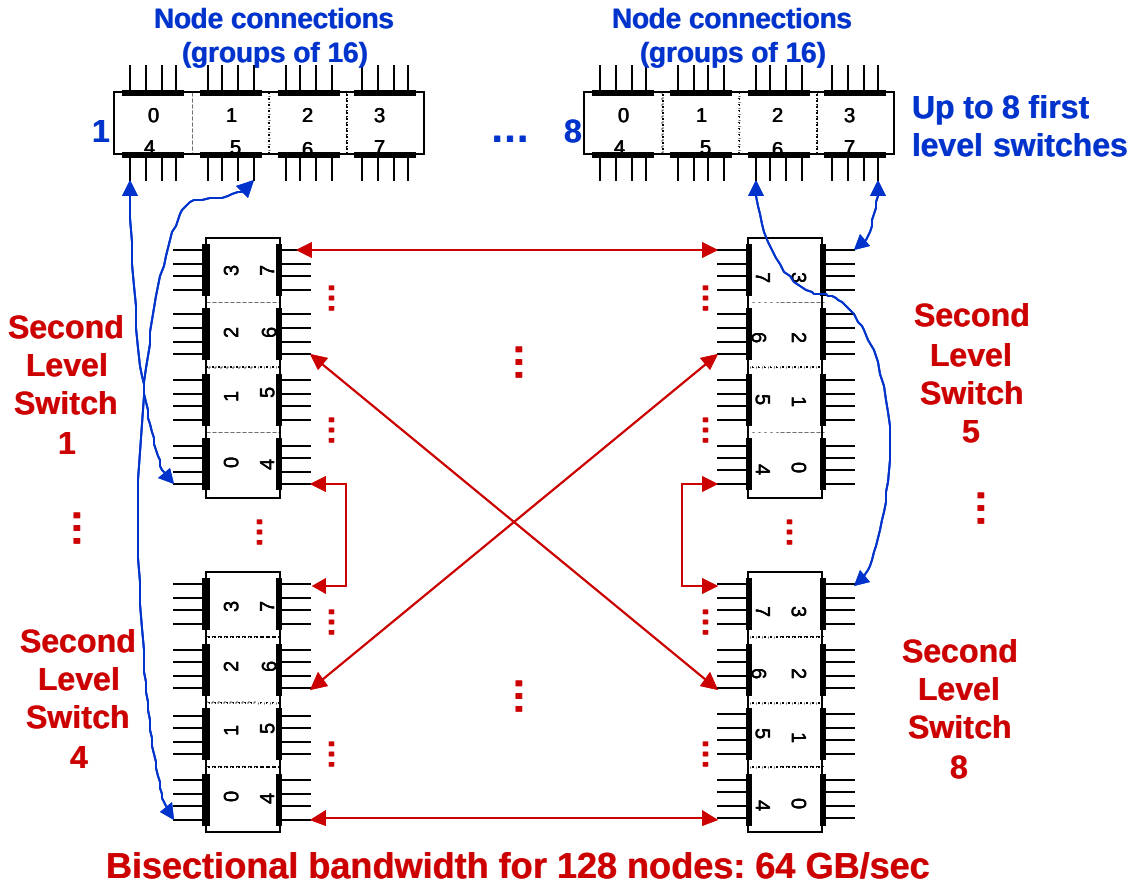

Figure 20: 128 nodes spread over 16 crossbar switches in a two-level topology

The maximum 128-node configuration could have up to 2048 processors and 8 terabytes of memory, with 64GB/sec of bisectional bandwidth. While the raw numbers are certainly impressive, the problem lies with the loosely-coupled interconnect. There are four levels of latency in an SP:

1. The intra-node SMP bus latency.
2. The latency introduced by a single switch for 2-16 node configurations (level 1 of loose coupling).
3. The latency introduced by two communicating switches for 17-80 node configurations (level 2 of loose coupling).
4. The latency introduced by four communicating switches, in the 81-128 node configurations (level 3 of loose coupling).

As the node configurations get larger, the levels of latency increase, with configurations of greater than 80 nodes passing messages through up to four communicating switches. These extra hops introduce increasing latency, and drive up the probability of non-uniform access times. The potential for machine architecture induced data skew is very high.

The loosely-coupled architecture also introduces another big problem for data warehouse workloads, namely, the necessity of using distributed database software. Having 128 nodes

implies having128 copies of the database software, complicating database administration and data warehouse extract, transformation and load. Parallel SQL must also execute in a distributed database environment, meaning that none of the tightly-coupled efficiencies of shared memory can be exploited. In a shared memory, parallel SQL producer and consumer processes can share data by sharing address pointers to blocks of data in shared memory. In loosely-coupled, multiple database environments, data must be passed, in its entirety, across multiple switch levels to reach its destination where it can be processed. This is extremely slow and eats up tremendous amounts of bandwidth.

7 HP Superdome

The HP Superdome is HP's primary offering in the data warehousing marketplace. HP's largest system, it consists of up to sixteen 4-processor SMPs connected together in a non-uniform, two-level crossbar topology using up to four 8-way crossbar switches.[7]

The 4-CPU SMP Superdome node architecture is shown in the figure below:

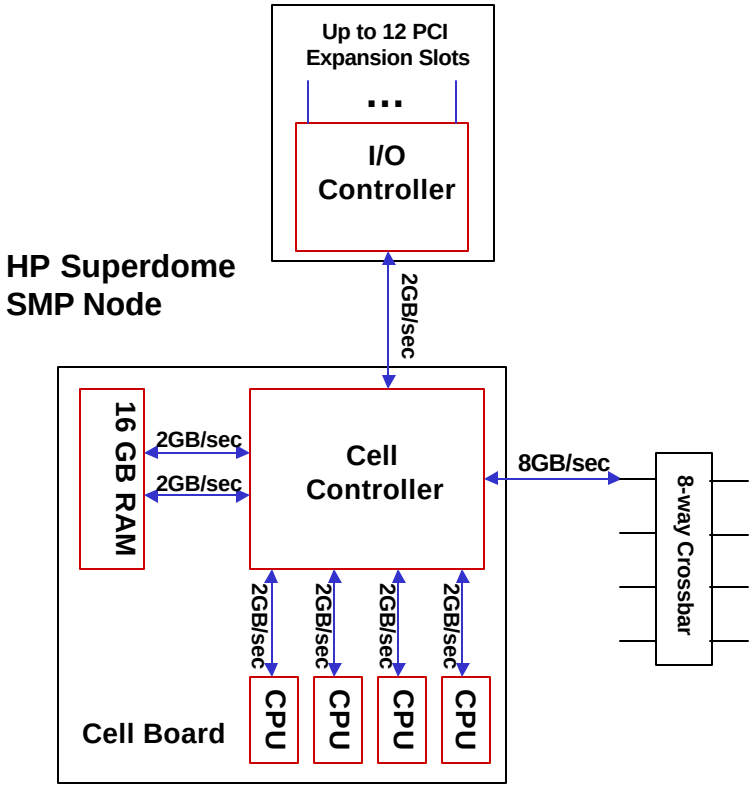

Figure 21: HP Superdome 4-CPU Node Configuration

HP calls the individual SMP nodes "Cells", after the Cell Controller, which connects up to 16 GB of RAM, four CPUs and 12 PCI I/O expansion slots. Each SMP architecture Cell has 8GB/sec of CPU bandwidth, 4GB/sec of memory bandwidth, and 2GB/sec of I/O bandwidth. Each Cell can have one 8GB/sec connection to other Cells via an 8-way crossbar switching element.

[7] The material in Section 7 is derived from the IDC White Paper "Superdome, Hewlett-Packard Extends its High-End Computing Capabilities", Christopher Willard, PhD.
http://www.hp.com/products1/servers/scalableservers/superdome/infolibrary/whitepapers/superdome_idc.pdf

The Superdome comes in configurations of one or two main cabinets, each containing two crossbar switches. The one cabinet configuration is shown in the figure below:

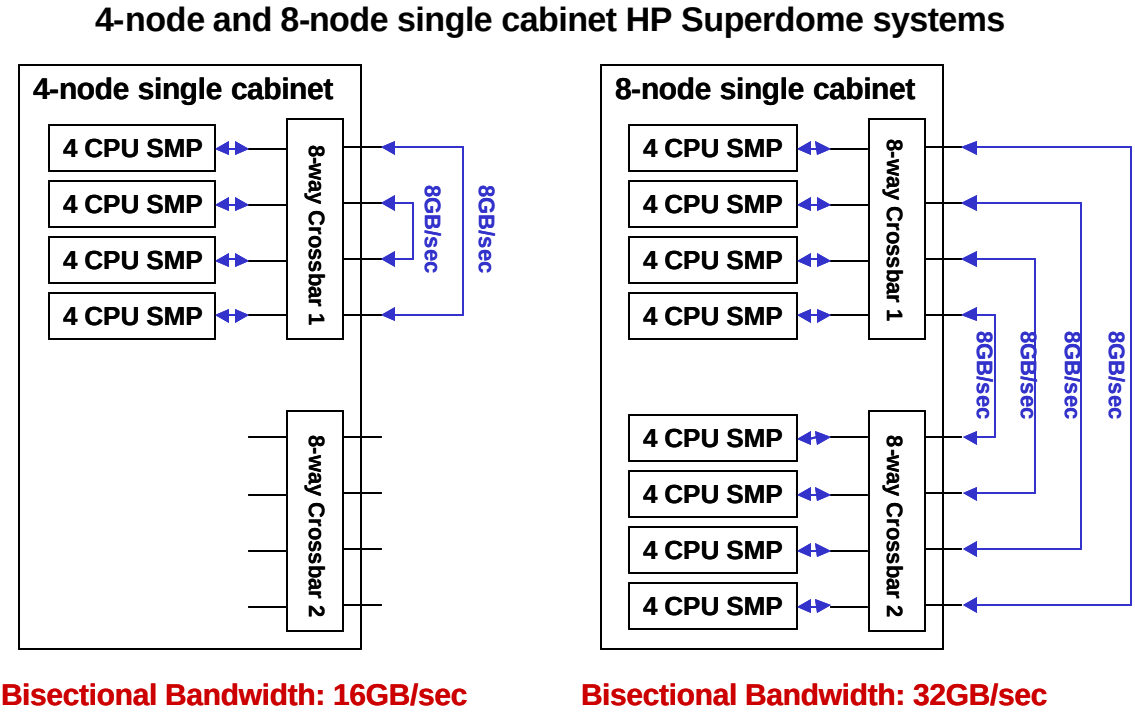

Figure 22: 4-node and 8-node single cabinet HP Superdome configurations

The crossbar switches are built into the backplane of the Superdome cabinets, so you get two per cabinet no matter how many nodes are used to populate the cabinet. In a 4-node configuration, shown on the left side of the figure above, the crossbar switch connects four SMP nodes as shown, for a bisectional bandwidth of 16GB/sec. In an 8-node configuration, two crossbar switches are connected as shown, giving 32GB/sec of bandwidth.

The 16-node, two cabinet, four crossbar switch configuration is shown in the diagram below:

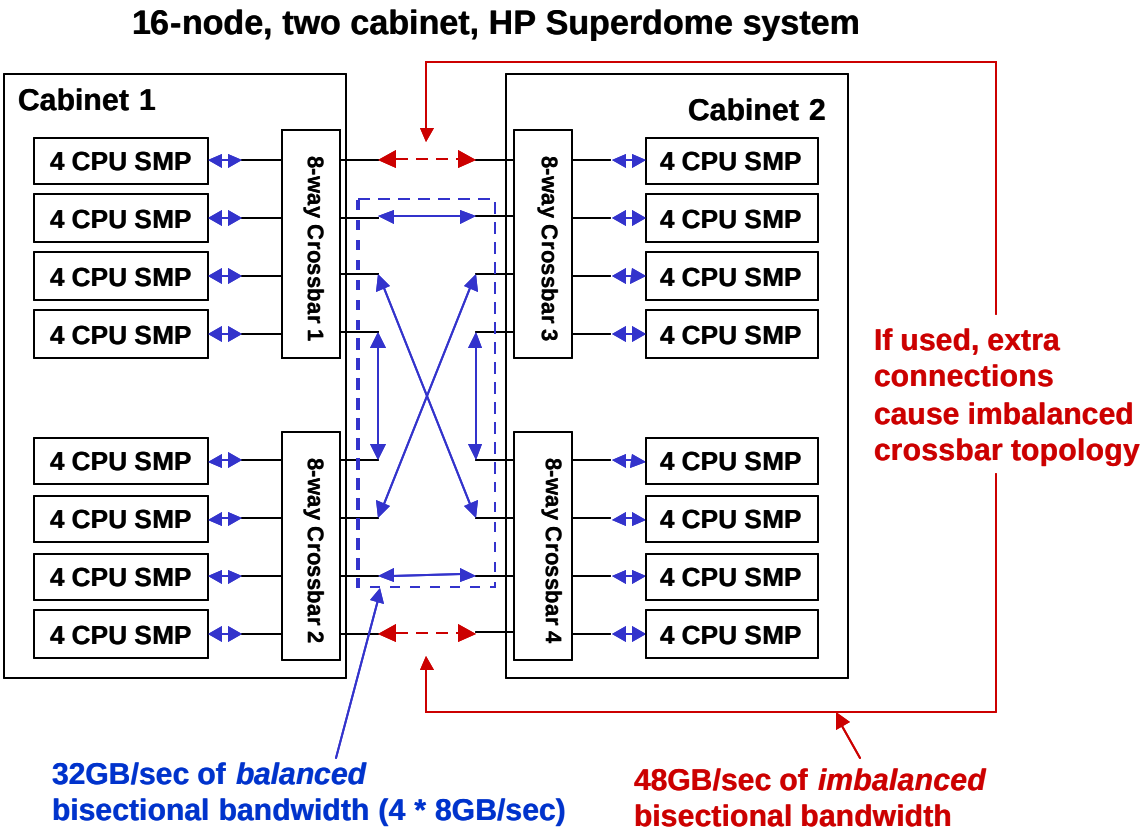

Figure 23: 16-node two cabinet HP Superdome system

The 16-node configuration uses each of the 4-crossbars in the two cabinets. While the HP literature claims a linearly scalable 64GB/sec of "aggregate" bandwidth, only 32GB/sec of balanced bisectional crossbar bandwidth can be achieved with the 4-switch topology, which is the same bisectional bandwidth as with two switches. HP comes up with the 64GB number by adding up the transfer rates of each of the eight potential connections shown in the diagram above. But this is *not* the bisectional bandwidth and if all eight connections were used, the machine topology would become imbalanced. Only six connections can be symmetrically configured as shown in the dashed-line box in the center of the diagram. No matter how the machine is bisected (horizontally or vertically), only four of the symmetric connections cross the bisection, which, at 8GB/sec each, is 32GB/sec of bisectional bandwidth.

Adding the two additional possible connections creates an imbalanced topology that has twice the bandwidth between crossbars 1 and 3 and crossbars 2 and 4 as it has between crossbars 1 and 2, 3 and 4, 1 and 4, and 2 and 3. This imbalanced topology would cause parallel SQL operations to complete twice as quickly for those producer and consumer processes that were communicating between crossbars 1 and 3 and crossbars 2 and 4. In order to preserve architectural symmetry and prevent machine architecture induced data skew, the two extra connections shown by the dashed lines need to be left unused, which means the balanced bisectional bandwidth of the maximum-sized 16 node Superdome machine is only 32GB/sec.

The HP Superdome has two potential levels of non-uniform interconnect latency:

1. First-level switch latency, for two to four nodes
2. An additional second-level switch latency, for five to 16 nodes

The HP literature quotes the average load-use memory latencies as follows:

- Across a first-level crossbar: 266ns
- Across an additional second-level crossbar: 360ns, or more than 35% greater than across a single crossbar.

These non-uniform interconnect access times, caused by the two-level crossbar, will inevitably lead to machine architecture induced data skew, as data is redistributed over the Superdome nodes during a parallel SQL operation. The Sun Fire 15K Server, on the other hand, has a completely uniform 18x18 crossbar interconnect that does not suffer from machine architecture induced data skew.

Although the more tightly-coupled HP Superdome and IBM p690 architectures are an improvement over the huge differences in latency found in loosely-coupled architectures like the NCR/Teradata 5250 Series and the IBM RS/6000 SP, the HP Superdome and the IBM p690 still have a high degree of architectural asymmetry that significantly impairs data warehouse scalability.

8 Conclusion

There are three critical questions to ask yourself when analyzing a particular machine architecture for scalability:

1. What is the topology of the machine's connection model?
2. Does the machine's topology guarantee uniform data access times, preventing machine architecture induced data skew?
3. Does the connection model have very large bisectional bandwidth, maximizing the amount of dynamic data redistribution possible during parallel SQL data warehouse operations?

With these questions in mind, let's compare the architectures of the five systems we analyzed in this paper—the Sun Fire 15K Server, the NCR/Teradata 5250, the IBM p690, the IBM RS6000/SP, and the HP Superdome:

Machine	# of Nodes	# of Processors	GB of Memory	System Connection Model (Topology)	Uniform Global Data Access Times?	Bisectional Bandwidth (GB/Sec)	Asymmetry of the Interconnect
Sun Fire 15K	1 tightly-coupled node	72*	576	Single-level Crossbar	Yes	43.2	None
NCR 5250	128 loosely-coupled nodes	512	512	Loosely-coupled Banyan MPP	No	1.9	Very High
IBM p690	4 tightly-coupled nodes, Single OS image	32	256	Four Non-uniform Rings	No	41.6	Medium
IBM RS/6000 SP	128 loosely-coupled nodes	2048	8192	Loosely-coupled Crossbar MPP	No	64.0	High
HP Superdome	16 tightly-coupled nodes, Single OS image	64	256	Non-uniform 2-level Crossbar	No	32.0	Medium

* The Sun Fire 15K Server can have up to 34 more processors by using unneeded I/O-assembly slots, but data warehouse workloads typically need all the available I/O connectivity to attach their massive disk farms and spread the resultant I/O over as many controllers as possible.

If you find yourself thinking that the IBM RS/6000 SP looks best because it has the most processors and memory, you have completely missed the point of this paper. The most important discriminators are in the shaded areas on the right side of the table.

Looking at the above table, the only machine architecture that has uniform global data access times is the crossbar bus topology of the Sun Fire 15K Server. Furthermore, the Sun Fire 15K Server has one of the highest bisectional bandwidth figures, at 43.2 GB/sec. Also note that the Sun Fire 15K Server is a single node system, which has many benefits, including a completely shared memory with uniform access times throughout, and a single, easy to manage image of the database software. This machine will scale well throughout its configurable range of processors and memory.

Other machines in the table do not fare as well. The NCR/Teradata 5250 Series system has the least-symmetric connection model, which makes its access times the least uniform. The bisectional bandwidth of the largest system is a surprisingly low 1.9GB/sec. As explained in the NCR section of this paper, this machine will have a great deal of difficulty doing non-primary key joins and outer joins, because the interconnect does not have enough bandwidth to do the necessary dynamic data redistribution to maintain the maximum level of parallelism throughout these operations. Proprietary Teradata software and specialized SVR4 MP RAS operating system support also make this the most closed of the machine environments we discussed in this paper.

Despite the hype, the IBM p690 turns out to be a ring topology system, with non-uniform system interconnect latencies. At only 32 processors and only 256 GB of memory, it is not a particularly powerful system for data warehouse applications.

On the other hand, the IBM RS/6000 SP is massive in scale with a maximum configuration of 128 nodes, 2048 CPUs, and 8192GB of memory, but it has a rather diminutive 64GB/sec of bisectional bandwidth for this huge loosely-coupled MPP computing complex. A comparison with the 72-processor Sun Fire 15K Server highlights this deficiency. With 28.4 times the number of CPUs in the Sun machine, it would stand to reason that the IBM RS/6000 SP would also need at least 28.4 times the bandwidth to support a similar data warehouse workload scaled up to this massive size. This would also imply the need for 28.4 times Sun's 43.2GB/sec of

bisectional bandwidth, or approximately 1.2 *terabytes* per second of bisectional bandwidth. At only 64GB of bisectional bandwidth, the IBM SP is short of this mark by a factor of about 19 times. Couple this deficiency with the four levels of non-uniform access times in the largest configuration, and it is easy to see why the IBM SP has trouble scaling data warehouse workloads.

Finally, while the HP Superdome is probably the closest system to the Sun Fire 15K Server in an architectural sense, there is still a wide gulf in its ability to scale, caused by its non-uniform, multi-level crossbar switching scheme. Furthermore, its bisectional bandwidth does not linearly scale as the system expands in size, making the Superdome a poor substitute for the Sun Fire 15K Server.

Mark Sweiger is President and Principal of Clickstream Consulting, a boutique consultancy specializing in the design and implementation of large-scale clickstream data warehouse systems. He is a noted expert in data warehousing and machine architecture topics, and a speaker at the Data Warehousing Institute and other industry conferences. He is also the principal author of the new book, *Clickstream Data Warehousing*, by Mark Sweiger, Mark Madsen, Jimmy Langston, and Howard Lombard, published by John Wiley & Sons, 2002.

Clickstream Consulting
1587 Shasta Ave., Suite 100
San Jose, CA 95126

408-283-0551
408-396-9500

www.clickstreamconsulting.com

Read our new book, *Clickstream Data Warehousing* (Wiley, 2002).
Book Web Site: www.ClickstreamDataWarehousing.com