

Preliminary Simulations of a SCI based Clustered Database Machine

Haakon Bryhni and Stein Gjessing

Department of Informatics, University of Oslo, 0316 Oslo, Norway

Antonio Schinco, Olivetti Servers R&D, 10010 Scarmagno, Italy

Abstract

High performance database machines can be built by connecting several traditional database machines into a clustered database machine (CDM). We outline a CDM architecture and present preliminary simulation results based on the shared memory multiprocessor standard Scalable Coherent Interface (SCI - IEEE Std. 1596-1992). Synthetic workloads have been calculated, using the characteristics of TPC-B and frequently used operating system operations. The results of the simulations are system bus load and interconnect throughput and latencies as a function of the applied traffic. The simulations are performed to support current activities in building working prototypes of SCI-based NUMA Database Machines.

1 Introduction

Multiprocessing is the obvious way to increase processing performance, also for database machines. A small number of processors can cooperate on a common system bus, and share the transaction processing load. When the number of cooperating processors grows beyond the number that can productively share the same bus (4-8), serious new problems arise. This paper reports on research supporting the construction of a database machine where several 4-CPU bus-based database machine nodes are connected using the IEEE standard 1596-1992 - Scalable Coherent Interface (SCI) [1, 2] to form a Clustered Database Machine (CDM). SCI defines a protocol for cache coherent shared memory, using a point-to-point, 16 bits wide interconnect with unidirectional links between nodes.

The aim of our work is to estimate the performance of a four node, 16 processor database machine cluster. In this work, the standard database benchmark TPC-B has been used to obtain a traffic mix [3]. Ideally, the transaction processing performance should increase linearly with the number of processors in the system. Obviously, this is not the case due to sharing, synchronization and the latencies of a NUMA computer. We want to investigate how the performance scales when bus-based multi-processors are interconnected in a NUMA directory-based cache coherent environment. Aim of the work is also to give feedback to the operating system designers regarding how much extra traffic the system can tolerate. We estimate what traffic the operating system will introduce, and vary the load parameters somewhat in order to see how robust our results are to variation in operating system traffic.

In the following we first describe the architecture of the clustered computer and our simulation model. Then we describe how we calculated a starting point for our simulations of the traffic between the nodes. We discuss the simulation results, and finally we conclude and explain how we will continue this research.

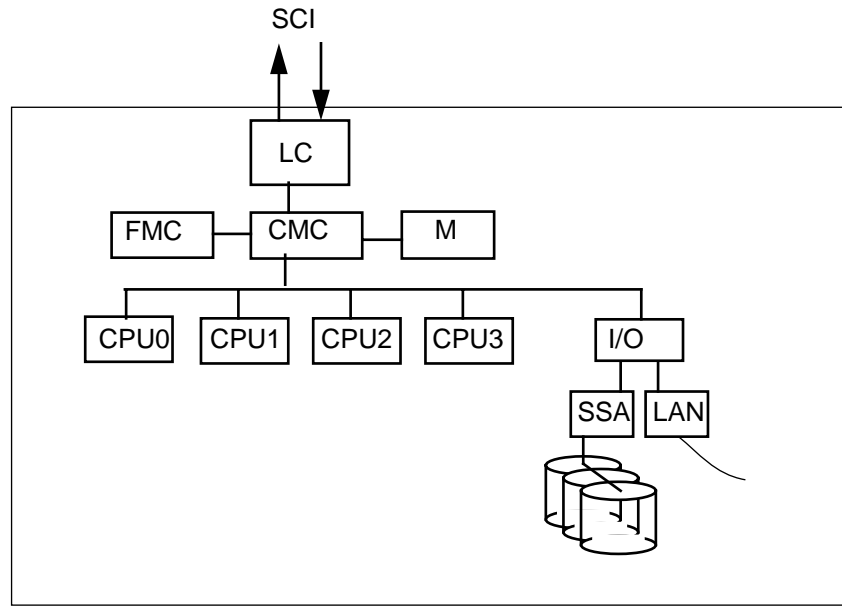


Fig 1. CDM architecture (one node shown)

2 The database machine architecture

Figure 1 gives an outline of the clustered database machine (CDM). The four processor nodes are all connected to one SCI ring. Each node contains four processors, an I/O controller and a memory system. The processors are fast standard microprocessors. The I/O subsystem contains disk controllers and LAN connection. The disk attachment is done with SSA, Serial Storage Architecture, which is an evolution of SCSI to interconnect large amounts of hard disk units. The memory subsystem contains an interface to SCI (i.e. to the other nodes in the system), a Far Memory Cache (FMC), a main Memory and a Cache/Memory Controller (CMC). All memory requests go to the CMC, that, depending on the address, either looks it up in the Memory or in the Far Memory Cache. In the latter case, if data is found, the access constitutes an FMC hit. If data is not found in the FMC, the CMC generates an SCI transaction destined for the home of this address. Under stable conditions the FMC is full. Hence a little used line has to be rolled out of the cache whenever a new one is inserted. This also adds traffic to the SCI ring. The CMC also services incoming SCI traffic. Read and write requests from other nodes are handled, as well as tag update requests both to Memory and to the Far Memory Cache.

3 Architecture of the Simulator

The simulator contains a detailed model of the SCI interconnect and the interface between each node and the interconnect. The interface includes input and output request and response buffers. The sizes of these buffers are determined by parameters. The interconnect model includes the bypass FIFO, input and output MUXes. The signal propagation delay between each node is set by a parameter, as is the node bypass latency. The input elasticity buffer (that takes care of the difference in the incoming clock signal and the on chip clock signal) is not modelled. A switch (connecting several SCI rings) is also modelled, but is not used in the work reported in this paper. We model a one ring, four node, tightly coupled multiprocessor with high speed interfaces. We have set the signal propagation latency between each node to 1 ns, and the bypass delay for each node to 8 ns.

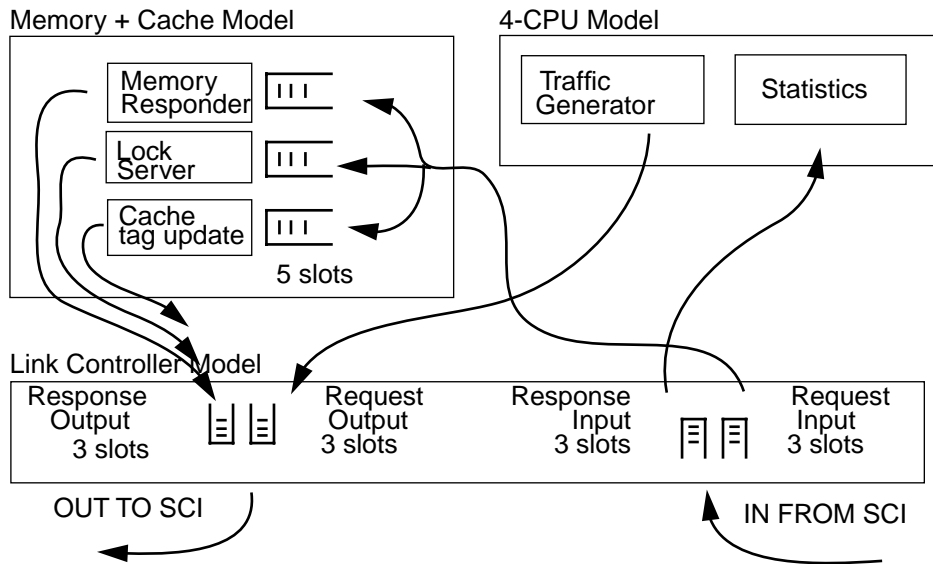


Fig 2. Simulator Architecture

The model is event driven, the main event being the head of a packet arriving at a node. A routing decision will either force the packet off the ring, or let it bypass the interface to the output link. Another event is triggered when the tail of the packet leaves at the output link. Then either a new packet is sent from the bypass FIFO, a packet is sent from one of the two output buffers or the output link becomes idle. Idle symbols are only modelled when the polarity of the go-bit contained within, changes.

Arbitration for access to the SCI ring is modelled in detail, as is the go-bits that decides when a packet can be inserted onto the ring from an output buffer. So even though the simulator is packet based and fast (it uses relatively short real time to simulate a lot of SCI traffic), it executes the SCI standard down to symbol level accuracy.

The model of each node is much simpler. The internal structure of the nodes is not included. What is left is an SCI traffic generator between the Cache/Memory Controller and the SCI link. This traffic generator is programmed to output memory requests that miss in the FMC onto the SCI ring. The simulated data flow between nodes are illustrated in Figure 3. The traffic generator issues memory requests that are put in the request output buffer. In this work we simulate all SCI input and output buffers with room for 3 packets. Rollout requests are also placed in the request output buffers. The request will arrive later at the request input buffer (lower right corner of Figure 2) of another node. Depending on the kind of request, it is served by one of the modules in the upper left corner of the figure (Memory Responder, Lock Server or Cache Tag Update server). These responders issue a response packet that is put in the response output queue (lower left corner). The simulator makes certain that the packet arrives back at the requesting node in the response input queue, and is finally consumed by the statistics module.

4 Simulation Plan

Our goal is to predict the performance of the computer when it executes database applications. We have chosen TPC-B as a sample application, since this benchmark is frequently used to compare the performance of database machines.

Even if we assume the TPC-B workload, the underlying operating system is not yet written. Hence we do not know the overhead incurred by the operating system when TPC-B is run on the CDM. What we do is to extrapolate what we know about operating system overhead in a bus based multiprocessor. We assume that the profile of the operating system traffic will contain a lot of references to shared data and that this data is protected by locks.

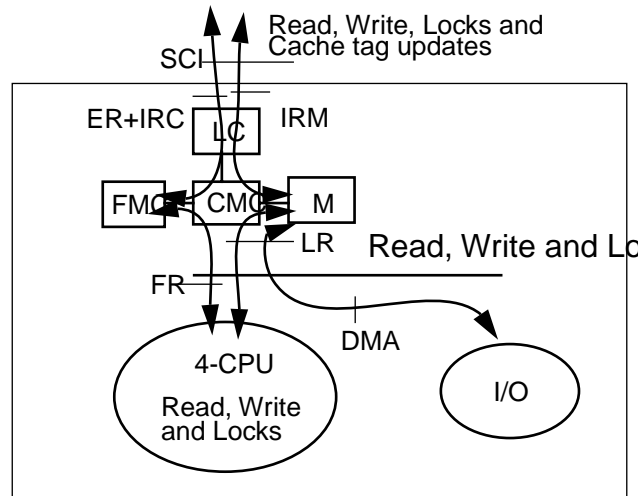


Fig 3. CDM Traffic flow

A TPC-B transaction consists of

- read/write/read operations on a cache line A,
- read/write/read/write operations on a cache line B,
- and only read operations on a cache line C.

In addition there is a 2K Byte Disk Read and a 2 K byte Disk Write operation to and from the database cache.

Traffic is caused both by the execution of transactions and by the operating system. Analyzing the TPC-B operations and adding shared data and lock operations from the operating system, we can find an initial ratio between reads, writes, and locks on the system bus. This is shown on Figure 3 as FR (Far References) and LR (Local References). Local References are either DMA or CPU traffic. The Far References that are not hitting in the cache, is emitted onto SCI as External References (ER). IRC and IRM are Incoming References to Cache and Incoming References to Memory, respectively. The incoming references are produced by the traffic generators of the other three nodes. The SCI traffic generated by this node is ER.

Our simulation plan is divided into five separate steps:

1. Find the traffic mix (reads/writes/locks) on the local bus.
2. Find the SCI traffic mix that corresponds to this local traffic mix (i.e. ER).
3. SCI throughput and latencies corresponding to this traffic mix are found by simulations
4. The simulation results (throughput and latencies) are used to calculated back to local bus throughput.

Since the actual behaviour of the operating system is not known, we have repeated step 1 through 3 above for the different traffic mixes:

1. Traffic dominated by Database traffic. (Data shared and Data not shared)
2. Mixture of Database and Operating system traffic.
3. Traffic dominated by Operating System traffic.

For the remainder of this paper, we will primarily discuss simulation results using traffic mix 2 (DB and OS traffic), but briefly show that the interconnect is not very sensitive to varying traffic mix.

The result of our calculations for SCI traffic (traffic mix 2) when we combined the memory reference traffic from DB and OS operations is shown in Table 1.

Table 1: DB and OS traffic

Operation	Frequency per transaction	Percentage
read	4	21
write	2	11
rollout	2	11
locks	11	57

The frequency of operations per transaction was derived from the characteristic operations of TPC-B and the characteristic operation of the operating system (protecting shared data by locks).

Corresponding traffic mixes for traffic type 1 was read 25, write 17, rollout 8 and lock 50% (Data not shared) and read 21, write 14, rollout 19, lock 50% (Data shared). For traffic type 3 we used read 20, write 0, rollout 20, lock 60% (OS dominated traffic).

5 Simulations

The described traffic load mix was fed into the simulator. The read and write operations are on 64 byte cache lines, while cache-tag and lock operations are assumed to work on 16 byte quantities. Each simulation ran until we had results within a confidence interval of 99,6%.

Figure 4 shows number of operations per second vs. latency for read operations, cache tag operations and lock operations. Notice that the latencies starts to increase for each type of operation at different numbers, but these differences corresponds to the differences in the load mix.

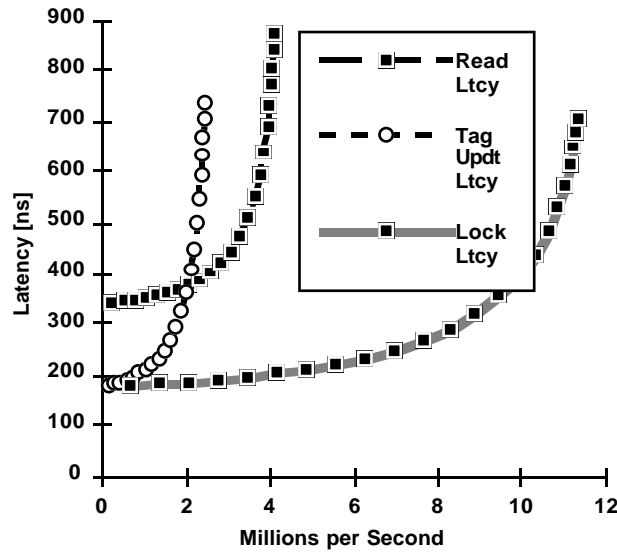


Fig 4. Throughput vs. Latency (DB+OS)

Figure 5 shows that the read latency starts to increase more dramatically at 200 MByte/sec read throughput for the system as a whole, or 50 MByte/sec read throughput for each node. The read latency is then at 400 ns, which is acceptable as far memory latency in a NUMA architecture.

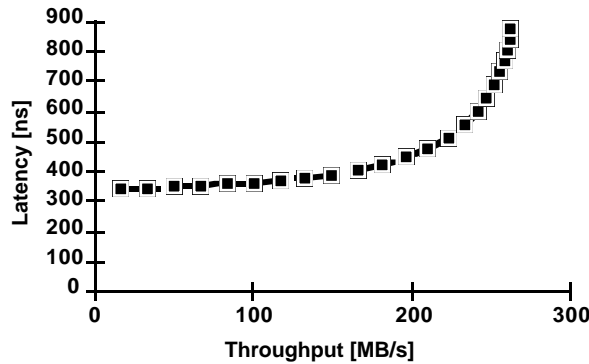


Fig 5. Read Throughput vs. Latency

Using this point as an entry to the results shown in Figure 6, we find the corresponding total throughput of 400 MB/s for the four nodes.

Figure 7 shows how the SCI read throughput is robust to the traffic mix. Notice that it is not a dramatic difference in read throughput between the traffic mix that contains only data base operations (data not heavily shared), and the traffic mix that contains only operating system traffic (data heavily shared).

6 From SCI-traffic to Processor Bus traffic

The result of our simulations show how much traffic the SCI interconnect can sustain. Depending on how large portion of the local bus traffic is going to the SCI interconnect, we can calculate back, and find the processor bus traffic corresponding to the SCI traffic. Our parameters are the

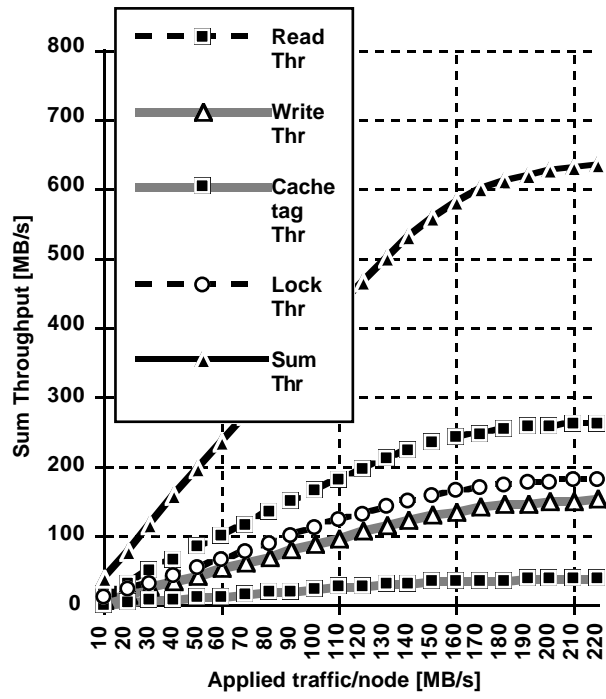


Fig 6. Sum Throughput (DB+OS Traffic)

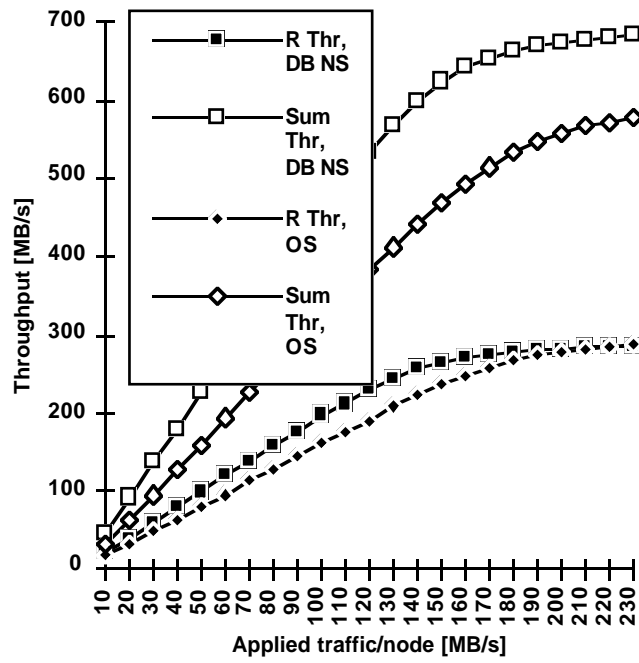


Fig 7. Comparison of traffic mixes (OS)

locality of the shared data (percentage of Far References, P_{FR}) and the Cache Hit Rate (CHR) in the Far Memory Cache.

If all references are equally spread out in memory of the four nodes, the percentage of Far References is 75% on each node. If there is some locality in memory assignments, then 50% seems like a reasonable number. If more locality can be obtained, then we can get the ratio of Far References down to e.g. 25%. Hence, we vary the percentage of Far References between 25%, 50% and 75%.

Another major uncertainty is the hit rate in the Far Memory Cache. Due to the fact that some data are heavily shared, and that database data are used very little by each transaction, we fear that the hit rate will be low. We have executed simulations with three different cache hit rates: 25%, 50% and 75%.

The model used to calculate back is given below, where X_R = Read Throughput, P_{FR} = Probability of Far References and CHR = Cache Hit Rate.:

1. Bus traffic component $X_F = X_R / P_{FR}(1-CHR)$

This component constitutes the local bus traffic related to external references.

2. Bus traffic component $X_L = X_F/P_{FR} * (1-P_{FR})$

This component constitutes all local references (i.e. hits in local Memory or in the Far Memory Cache).

3. $X_{BUS} = X_F + X_L$

Total local traffic is the sum of far and local references.

As an example, for $X_R = 50$ MB/sec (as simulated), with $CHR = 25\%$ and $P_{FR} = 75\%$, $X_F = 88.8$ MB/sec and $X_L = 29.6$ MB/sec, gives $X_{BUS} = 118$ MB/sec.

The calculated numbers are presented in Figure 8. We assume that each SCI read or write is 64 Bytes and that each lock and cache tag operation is 16 bytes.

Resulting Maximum Local System Bus Traffic				
CHR	25%	50%	75%	Far Ref
[MB/s]	1070	1600	3200	25%
	267	400	800	50%
	118	178	355	75%

Fig 8. From SCI to Local System

We assume that the local bus is a a modern high speed shared bus, that can sustain around 300 MB/sec. We see that when the number of Far References is as low as 25%, then the SCI interconnect is not a bottleneck in any case. For Far References of 50% we see that if the cache hit is good, SCI is still no bottleneck. The cases that balances best between local bus and SCI traffic is the 50% Far References combined with 25% Cache Hit Rate, and 75% Far References combined with 75% Cache Hit Rate. When the number of Far References is high (75%), and the hit rate is low (25% or 50%), the results show that the bus is underutilized, i.e. SCI is a bottleneck.

7 Conclusions and further work

The main results shown in this article are derived from simulations of a specific traffic mix. However, varying the traffic mix does not change our results significantly. We conclude that the SCI interconnect is robust towards different ratios between read/write/locks and cache line operations.

We have shown that a four node SCI CDM with operating system and data base traffic mix can sustain 400 MB/sec total SCI throughput. Half of this traffic is overhead related to locks, roll outs in the far memory cache and other cache administration operations. The other half (200 MB/sec) is read operations. Read operations are performed with a latency of 400 ns or less. This gives a effective read throughput of 50 MB/sec per node.

Our simulations have shown that the performance of the four-CPU nodes are in balance with the SCI interconnect when there are 75% far references and the Cache Hit Rate in the Far Memory Cache is 75%. The system is also in balance when there are 50% Far References and the Cache Hit Rate is 50%. There are obviously numbers between these two cases that will also keep the system in balance. And for lower SCI traffic numbers, the local systems will of course still run at full speed. It is only for very high ratio of Far References (75%), combined with very low hit rate in the Far Memory Cache (50% and less) that global traffic will slow down the local bus in terms of throughput.

The high percentage of Far References used in our simulations, supports the view that a shrink wrapped operating system for a 16 processor multiprocessor could possibly run on our 16 processor CDM as well. Such an operating system would not be aware of the NUMA, but still our findings indicates that the operating system could work without modifications. However, we must not forget that our conclusion is based mostly on throughput numbers, and that the effects of longer latencies are not yet well understood. A question that needs more investigation is how longer latencies impacts database performance.

So far, we have used global vs. local bus balance arguments to find the performance of the database machine. Obviously, we would like to state exact TPC-B transaction processing numbers. In order to predict TPS performance, we need an executable model of TPC-B which is not available to us at this point. Our plan is to predict TPS performance on the 4-CPU processing node using performance evaluation techniques developed by Olivetti and at the same time obtain a more detailed traffic model for the local traffic. The SCI simulator used in this work is then used to scale the results to the full clustered database machine.

Acknowledgments

We acknowledge Dr. Øivind Kure and his staff at Telenor Research for helpful comments to our initial simulations. We are grateful for support from the Norwegian Research Council.

References

- [1] IEEE Std. 1596-1992, "The Scalable Coherent Interface"
- [2] D.Gustavson, "The Scalable Coherent Interface and related Standards Projects", *IEEE Micro*, Feb. 1992, pp. 10-22
- [3] J. Gray, "The Benchmark Handbook for Database and Transaction Processing Systems", The Morgan Kaufmann Series in Data Management Systems, San Mateo, California, USA, Morgan Kaufmann, 1991.