Running the SAR Application on a Cluster of PCs Connected with SCI using a HIC based SCI switch

Ame Maus¹, Torstein Strøm², Stein Gjessing³ and Lars Paul Huse⁴

Abstract—This paper describes the behaviour of a prototype SCI-switch in a real application. The switching fabric uses the IEEE 1394 HIC technology to switch SCI packets in a switch with up to seven SCI ports. In our lab we have tested a switch with up to three SCI ports. We report on the performance of a Synthetic Aperture Radar (SAR) application when it is run on a number of dual processor PCs connected by SCI and using our HIC-based SCI switch.

Keywords—Cluster Computing, SMP, ccNUMA, SCI, HIC, Interconnects, Switches

I. INTRODUCTION

When the number of processors in a symmetric multiprocessor (SMP) increases, the limited memory bandwidth becomes a bottleneck for the high aggregated bandwidth requirement. The Scalable Coherent Interface (SCI)[1]. It is designed as a high-speed bus replacement to provide good throughput and low latency communication in a scalable multiprocessor.

SCI is standardized for connecting ccNUMA systems, however SCI is more commonly connected to the I/O bus (typically a PCI-bus) to provide low latency, high bandwidth communication between workstations or PCs in a cluster [3][4]. In this paper we discuss the latter, SCI connected to a PCI bus.

Theoretically SCI scales well with up to 65 000 node (computer) ids, enabling building of large systems. The simplest SCI topology is a ring that connects all nodes, suitable for small systems. Like a bus, a ring can, however experience contention if it contains more than about 8 nodes. The spatial reuse of the SCI ring alleviates contention to some extent, but if the communication need of each PC is too high, the bandwidth of the SCI ring becomes too small. In current implementations, the bandwidth is in the order of 500Mbytes/sec [3]. Latency may also suffer, by and large not from the number of bypass and entry (input) buffers a message has to pass on the SCI ring, but from full bypass queues. For larger systems, several rings are connected using switches. The results are meshes, multicubes [6] or 2D torus. Such topologies are also designed to increase the bandwidth of the total system.

This paper describes a small SCI topology and its behavior in a real application. The topology consists of small rings connected by a newly developed SCI-switch. Internally the switch uses the IEEE 1394 HIC [9] technology to switch SCI packets in a switch with up to seven SCI ports. Due to supply limitations we have only been able to test a switch with up to three SCI ports. Earlier we have run micro benchmarks and reported on the performance of this SCI switch [11]. We have found latency and throughput numbers for different configurations and different sizes of synthetic user data.

In this paper we report on the performance of this SCI switch in a real application, a Synthetic Aperture Radar (SAR)-program, when it is run on a number of dual processor PCs connected by switched SCI.

The rest of this paper is organized as follows. In section 2, we give an overview of the SCI-HIC switch.

For sake of completeness and comparison, some old performance measurements are repeated in section 3. The SAR application is described in section 4, and the test results are presented in section 5. Finally, we conclude and discuss some further work in section 6.

II. THE SCI SWITCH

The switch is built from functional units that are illustrated in figure 1. The implementation is based on the LG-2 link controller chip from Dolphin Interconnect Solutions, and on two RCUBE chips from Tachys [9]. The RCUBE switches HIC packets, and it is this switching unit that is also used to switch SCI packets.

SCI packets are transferred between the SCI side and the HIC side using a dual-port static RAM as a temporary buffer. The total buffer capacity is 64 88-byte SCI packets.

One such 88 byte SCI packet contains 64 bytes of user data. Separate buffers are used for request packets and response packets, and separate buffers are used for packets going in opposite directions. The RAM buffer is 64 bit

![Fig. 1. Block diagram showing the functional units and the main data paths of the SCI to HIC bridge](image-url)
wide, governed by the width of the LC-2 backplane bus, BLINK.

The RCUBE is primarily an 8x8 serial switch, but on two ports there is an option to select a parallel interface instead. Each parallel interface consists of two 9-bit ports, one for input and one for output. Eight bits hold the character value, and the ninth bit separates data characters from control characters. These two eight-bit interfaces are used to connect the RCUBE to the dual port RAM, and hence a 64-to-8 multiplexing and demultiplexing is performed here. Using the parallel interface instead of the serial interface between the RAM and the RCUBE means that we can use a slower clock here. The six serial ports are used to connect up to seven such switching units to make up a total switch. Using both RCUBE A and RCUBE B (see figure 1), two links between each unit can be set up, hence doubling the switch throughput.

When taken off the SCI ring at the entry to the switch, the SCI packet is embedded as data in a HIC packet. A HIC packet consists of a short destination address header and data. Hence, in our case the data is simply the total SCI packet. The HIC destination address is simply the SCI destination address, so the SCI routing decision is left to the HIC router (the RCUBE).

The maximum data rate on the SCI ring is 500 Mbytes/s using a 125 MHz SCI clock (clocking on both edges). Maximum data rate on the BLINK is 800 Mbytes/s using a 100 MHz clock. The switch is designed to be able to transmit two 64 bytes SCI packages at 80 Mbytes/sec each. Because the HIC components are not running at full speed, and because other logic components that we have used in the prototype are not currently working as fast as we designed them for, the speed of one SCI packet over an HIC-link is presently 20 Mbytes/sec, and we are not able to utilize the possibility of transmitting two packets in parallel because the lack of RCUBE chips.

III. PREVIOUS RESULTS

A switch will always incur some extra latency, but an order of magnitude longer than the latency through a bypass buffer, should not be exceeded. In [11] we report on latency and throughput microbenchmarks for our switch. In figures 2 and 3 we recall these results.

In the figures performance of four configurations are compared: One ring (i.e. no switch) with two and four nodes, two rings with one node on each ring and three rings, also with one node on each ring. When we use two and three rings, it is of course our switch that connects them.

Figure 2 shows latency comparisons. The mean latency on a ring is the same for a ring with two and a ring with four nodes. The latency starts at 5 microseconds for the smallest 16 bytes packets. Packets up to 63 bytes are transferred with a multiple of such small 16 bytes packets. The SCI interface card is not optimized for such small packets, hence latency increases very sharply up to 63 bytes (the same "abnormal" behavior can be seen in the cases where the switch is used too). Figure 2 shows that latency increases with $\approx 10\mu s$ for every 350 bytes increase in packet size in the unswitched case.

When the switch is used, the smallest possible latency is increased to about 8 microseconds. Except for some peaks that we are unable to fully explain, the latency increases with almost 30 microseconds for every 350 bytes of increased performance.

Figure 3 shows throughput performance numbers. We see that unswitched systems sustain a throughput in the order of 40 Mbytes/sec, while a switched system achieve approximately 10 Mbytes/sec in an end-to-end user application. The bottleneck is obviously the switch. The main reason that we achieve only 10 Mbytes/sec of user data throughput, while the HS link operates at 20 MBytes/sec, is the combined overhead of the SCI and HIC protocols.

IV. THE SAR APPLICATION

Scal's cluster implementation of the Extended Exact Transfer Function (EETF) Synthetic Aperture Radar (SAR) algorithm for data from the ERS-1 satellite is used as a practical example of SAR processing [16]. ERS-1 was launched in 1992 by European Space Agency and the radar images is used for e.g. detecting ship routing and oil spill, forecasting sea state and monitoring agricultural produc-
tion, sea ice and wind state. The SAR processing implementation is based on the original algorithm developed by Dr. Knut Eldhuset at Norwegian Defence Research Establishment [12]. The tested software is an enhanced version of the production code for the Scalari machine at Tromsø Satellite Station, specially adapted for the x86 architecture.

The processing is divided in a master process and a number of processing agents. The master process fetches data from disk, scatters the raw-data to the agents, gathers the resulting image from the agents and saves it to disk, i.e. only data movement. The master pre-fetches raw-data to the agents throughout the processing in order to reduce input-data waiting time to a minimum. The communication between the master and agents are done over SCI using ScaMPI [19]. ScaMPI is Scalari’s implementation of the MPI (Message Passing Interface) [18], which is a well established communication standard.

The agents do the calculation of the image from the raw-data. The calculation can roughly be divided into various forward and inverse FFTs (Fast Fourier Transform) with a filter multiplication and three correction phases. The processing is done in an azimuth-range coordinate system in 35 overlapping blocks in azimuth direction. The raw-data file is 193.6 MBytes, and each block is 7.02 MBytes i.e. a total of 245.7 MBytes is transferred from the master to the agent (assuming single agent). Each block is processed as four 1k x 2k (azimuth x range) sub-blocks. Table I shows the processing requirement for one sub block. Total processing requirement for one image on a single agent is 94.2 Gflop. Before processing of a block parameterized results from the calculation of the boundary block must be available or estimated. Estimating parameters of a block is similar to processing it and requires access to the complete raw-data block. Reuse of the data in the master for the blocks being processed twice is complicated since it is the first block for the agent estimating parameters and the last block for the agent doing the actual processing. No internal reuse of raw-data between agents is therefore performed, i.e. the volume of data fetched from disk increase with the number of agents. The resulting ESA CEOS formatted product file is 60.1 MBytes. The application has been fully parallelized and transformed to reduce memory traffic (good cache utilization) and improve disk I/O speed (Master/Agents decomposition).

The master/agent communication in the SAR application introduces the master as a hot sender/receiver. As described in [17] this does not introduce limitation on the communication link for SCI based solutions for up to 8 agents. Since the agents do not communicate with each other, no other inter-node traffic is generated. The processing time without disk I/O is dominated by the calculation in the agent.

Achieving fast disk I/O is essential to get good performance of the current SAR processing. Just reading 200 MBytes (7 agents) and writing 60 MBytes at 4 Mbytes/sec to/from disk (not unrealistic single disk performance) requires 80 s. The master always prefetch raw-data to the agents. Since the prefetch in the master overlap in time with the processing in the agents, but for the first block to each agent, a minimum of time (usually none) is lost while fetching data from disk.

While application performance is calculated from the single agent communication and calculation requirements, the real requirements increase with the number of agents. To start the processing the agent need the processing result of the previous block, i.e., for each new agent the processing increase. The real communication and computation volume is therefore given by:

\[
\text{Transfer volume} = 7.02 \times (34 + \text{agents}) + 60.1 \text{MBytes}
\]

\[
\text{Calculation} = 4 \times (34 + \text{agents}) \times 672.9 \text{Mflop}
\]

To achieve real time SAR processing (processing the signal from the satellite as it passes without temporary storage) the processing has to be done in 15 seconds (25 Gflops) [13][14]. Meeting these requirements for a dedicated SCI based workstation cluster is detailed in [15].

V. TEST RESULTS

The test equipment consists of one Scalari cluster, containing 8 PC's (440BX chipset) with dual Pentium II 400 MHz processors, 128 MBytes of RAM, running SUN Solaris 2.6. Each computer has one Dolphin Interconnect Solutions CluStar PCI-SCI card.

One node is dedicated to the master process, and from 1 to 7 nodes are used as computing nodes.

![Diagram](image)

**Fig. 4. Using the switch to connect 6 processing nodes**

In figure 4 we see that in our test configuration, the switch connects three SCI rings. Each ring consists of uni-directional links (dotted lines) between the nodes in the
ring. On one ring the master is the only node (apart from the switch), on the two other rings there are three nodes (the processing agents) in addition to the switch.

As was pointed out in [11], our switch has problems when we try to send SCI packets through it back to back. Due to lack of resources (finance and people), we have not been able to locate and correct this problem. Therefore, all tests using the switch must take care that the application waits a while between the sending of packets. In the MPI implementation, buffers that are larger than 64 bytes are transmitted as several 64 bytes SCI-packets back to back. In order to be sure not to send SCI-packets back to back, we have to limit the MPI buffer size to 64 bytes. From figure 3 we see that the switch is able to transmit about 2.5 MBytes/s in 64 bytes chunks, while the figure also shows that with larger chunks it is able to transfer almost 12 MBytes/s. Hence we must be prepared to see a degradation in performance when we run programs that communicates using MPI with buffer size restricted to 64 bytes.

In figure 6 the speed-up is plotted. From both figures 5 and 6 it is possible to see the doubling in speed in all three cases when a second processing node is added.

We have not run the switched configuration with different number of processing nodes in the two rings, hence the last two cases for the switch are a total of 4 and 6 compute nodes. From figure 6 we see that in the unswitched cases the performance is increasing all the way up to 5 and 6 processing nodes. In the switched case, however, there is a degradation in performance going from 4 to 6 nodes. This is caused by the fact that the communication overhead become larger than the performance gain caused by the added processors. Such a negative speed-up is common when the task is split up in too many sub-tasks, compared to the communication overhead.

As we have pointed out earlier, SAR is an almost embarrassingly parallel task. However, because the problem set is not very large, the limit to good performance gain is reached relatively fast. From figure 6 it is seen that the speed-up from 4 to 5 nodes is less than 10%. In this case the communication throughput is good, about 50 MBytes/sec (figure 3).

So when the communication speed is reduced from 50 MBytes/sec in the un-switched case, to 2.5 MBytes/sec in the switched case, it is reasonably that the performance gain from 4 to 6 nodes in the switched case is negative.}

![Graph 5](image5.png)

**Fig. 5.** Running time for the SAR application on Scalp's cluster compared to a switched topology

![Graph 6](image6.png)

**Fig. 6.** Speedup for the SAR application on an increased number of nodes both on Scalp's cluster and on a switched topology

As stated previously, the SAR application is I/O-intensive. There is a possibility that the disk-performance is a limiting factor when the number of processing nodes increases. We have observed that some transfers of raw data from the master to a processing node suffers from low bandwidth, i.e. below 1 Mbyte/sec, but again the next transfer is back to normal (>20 MBytes/sec). This might indicate that the master has not been able to prefetch the requested block.

VI. CONCLUSIONS AND FURTHER WORK

We have shown that the SCI switch based on HIC is actually working in an real application. By restricting the
size of the MPI buffers we have been able to work around limitations of the prototype.

In small configurations, we have shown that switched systems can give the same speed up as unswitched systems. For larger systems the speed up is less. Our application has one data producer, all the other computers are consumers. The work is spread out, with some overlap, among the consumers. For a given problem size we have shown that the system does not scale beyond 4 consumers.

A faster switch must be developed if we shall be able to reach the goal of a run time of 15 seconds in a switched system. A faster producer must also be developed, that is one that is able to stream data efficiently to many consumer simultaneously, I/O bandwidth from the disk to the master must also be measured and probably increased.

In order to gain more insight into the performance of communication based on SCI in a switched environment, we will setup new experiments. In future work, we will use different problem sets, to see how speed up is dependent on this parameter. We will also run several SAR application in parallel, hence having more than one producer. This will make the traffic through the switch more uniform, and hence we might be able to utilize the good characteristics of the switch better, because the switch is designed so that one traffic flow through the switch shall influence very little on other flows.

We have proved that the SCI-HIC switch is actually working in an real application, however the switch has limitations which has to be removed in a useful, industrial version. The prototype is running at a quarter of maximum clock speed, and only one HIC link is used, while two could have been used for one end-to-end communication, sending two SCI-packets through the switch in the same direction at once.

VII. ACKNOWLEDGEMENTS

The authors would like to thank the researchers at SINTEF, in particular B. Haalen, who designed the switch prototype. The authors are grateful to their colleagues at the University of Oslo and at Scali and the anonymous referees.

REFERENCES