draft 20 March 2000

The Use of Low-cost SMPs in the Atlas Level-2 Trigger

Mannheim, MSU and CERN

(R.Bock, A.Bogaerts, Y.Ermolin, A.Kugel, R.Lay, P.Werner)

Low-cost SMP (Symmetric Multi-Processor) systems have become generally available since 1998; they provide substantial CPU and I/O capacity along with a memory that is shared by all processors. We have investigated two areas of application in the Atlas level-2 trigger:

A. Consideration of an Active Readout Buffer (ROB) Complex, grouping multiple Readout Buffers,

B. Measurements on a ccNUMA (Cache Coherent Non-Uniform Memory Architecture) system for handling all Readout Buffers of a full (sub-) detector.

Other application of the same systems in the ATLAS data flow are conceivable, like as event flow supervisor or as powerful processing node. Some of our measurements can be mapped immediately on such applications.

A. Active Readout Buffer Complex

This work is based on the original idea to build detector-adapted computing stations with multiple processors, all having access to all ROBs of a detector, using commercial components from the HPCN market: several multi-processor boards with proprietary interconnect, all working under a shared-memory paradigm.

The availability of truly commercial components, today limited to 4- or maximally 8-processor systems, has reduced the goal to a multi-ROB station; we call it ’active’ because the processors actively contribute to alleviate the critical traffic over the general network.

Two activities were proposed in September 99 and partial results were presented at the Atlas TDAQ workshop in December 99:

- *modelling* the active ROBC to assess the effect on general level-2 traffic for different detectors; a report is available (R.K.Bock, D.Francis, J.Vermeulen, S.Wheeler, ATLAS-COM-DAQ/99-020); the encouraging conclusions of this report are summarized below.
- *measuring* the internal performance of a present-day commercial SMP; our results were obtained on an Elonex 4-processor PC with two independent PCI buses. Measurements concern the limits of traffic when run with multiple ROBs, and the limits of communication between processors, for programs communicating at high level.

A.1 Conclusions from modelling
(reproduced from ATLAS-COM-DAQ/99-020)

- under the assumptions of the report (up to 16 ROBs accommodated per AROBC, no internal performance limits), the number of ports for the general network is reduced by an average factor of 11.4;

- the reduction factor for the total number of messages exchanged over the network is 4.3;
- all required performance numbers for communication of the AROBCs with the general network stay within technology limits, for frequencies and bandwidth needed for level-2 (<15 kHz, < 20 MB/s), using the computing capacity in the AROBC for pre-processing and for compacting the data;
- a bandwidth problem arises for the Event Builder (EB) traffic over the network connection, if data compaction for the EB is excluded (although the problem is alleviated by the larger packets, allowing to reach higher performance);
- there is no problem in providing enough computing capacity in AROBCs to execute pre-processing, for all detectors;
- the possibility of using AROBCs for feature extraction is not obvious, but for some detectors this may be a good solution if the farm processors and the AROBC processors are software compatible.

A.2 Measuring external I/O limits in a commercial SMP system

A PC server based on a four-processor board (Intel SC450NX) was purchased in October 1999 (4 x 550 MHz Pentium Xeon III, 2 PCI buses of 32 bits/33 MHz, 7 free PCI slots, 512 MB shared memory, 19 GB disk).

Standard commercial Micro-Enable boards were used as ROBs (Slinks are available but were not used).

All programs were developed on standard PCs:

- the commercial Micro-Enable drivers were adapted to the multi-processor / multi-bus environment,
- a high-level interface was written to handle requests for ROB data (by event number), and to provide a polling mechanism,
- an application program was written in C++ to perform the basic measurements,
- multi-threaded applications allow the system to distribute the tasks to different processors.

A.2.1 Maximum access rate and bandwidth

The goal was to measure the aggregate bandwidth for multiple ROBs on multiple PCI buses. Results are summarized in the following figure.

A.2.2 Limits set by the Memory Bus

The I/O rate does not only depend on PCI performance: we have measured that the total available PCI bandwidth drops by about 20%, when loading the memory bus with a read/write flow of 130+130MB, i.e. it goes from the (unidirectionally) measured 160 to 130 MB/s, for 1kB Rob fragments. The memory bus of the four-processor Intel board is a known bottleneck, so this is not unexpected; we hope that future SMP boards will have a better behaviour in this respect.

A.2.3 CPU time available for algorithms

Four types of threads are used in the test program:

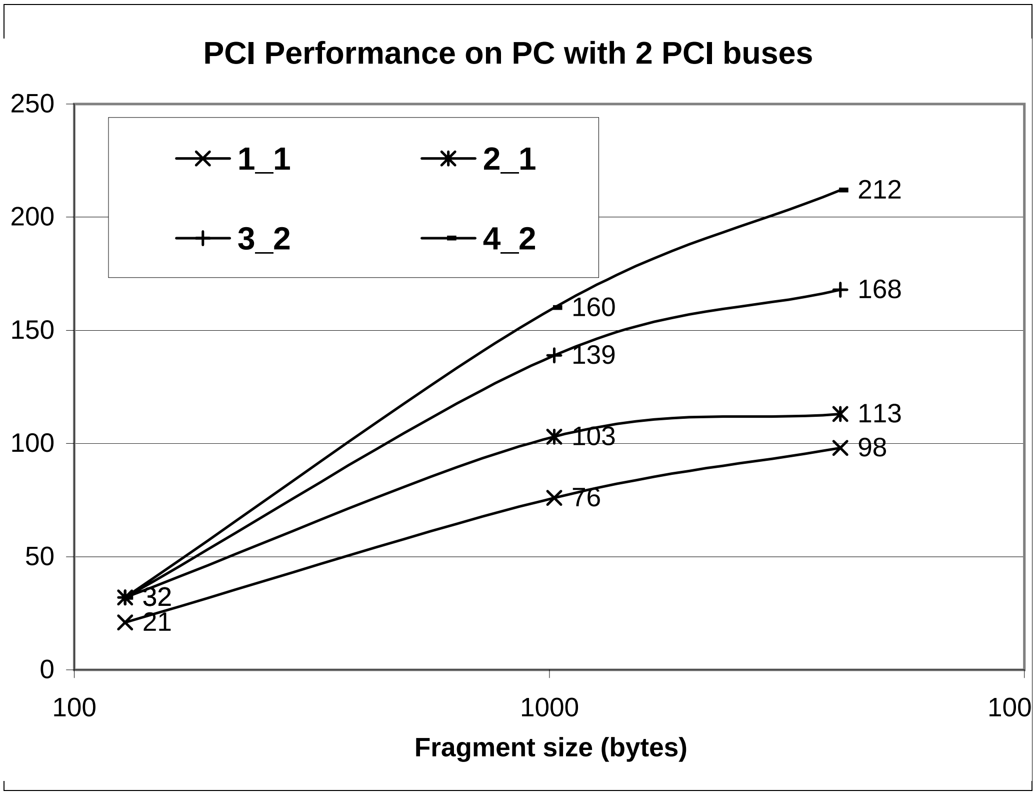

Figure 1 The bandwidth as a function of the data size for 1 to 4 ROBs; two PCI buses are used in the case of three or four ROBs. The rate increase for the second PCI bus (i.e., going from two to four ROBs) is 88% for large packets, 55% for our preferred packet size of 1 kB

- One *RequestThread* generates the requests for the *requestQueue*. It controls the number of outstanding requests, and represents requests to the Rob Complex coming from outside (e.g. steering processors).
- One *CollectionThread* that reads the *requestQueue* and collects all event fragments (ROBs) needed for the event. Collected events are put on the *eventQueue*. This is the *Fragment Collection* task.
- One or more *WorkerThreads* read from the *eventQueue* and spends a certain amount of CPU time, “algorithm”, on each event before the LVL2Result is put on the *resultQueue*. This represents the *Preprocessing* task.
- One *ResponseThread* reads the *resultQueue*. This represents answers from the Rob Complex to outside (e.g. steering processors).

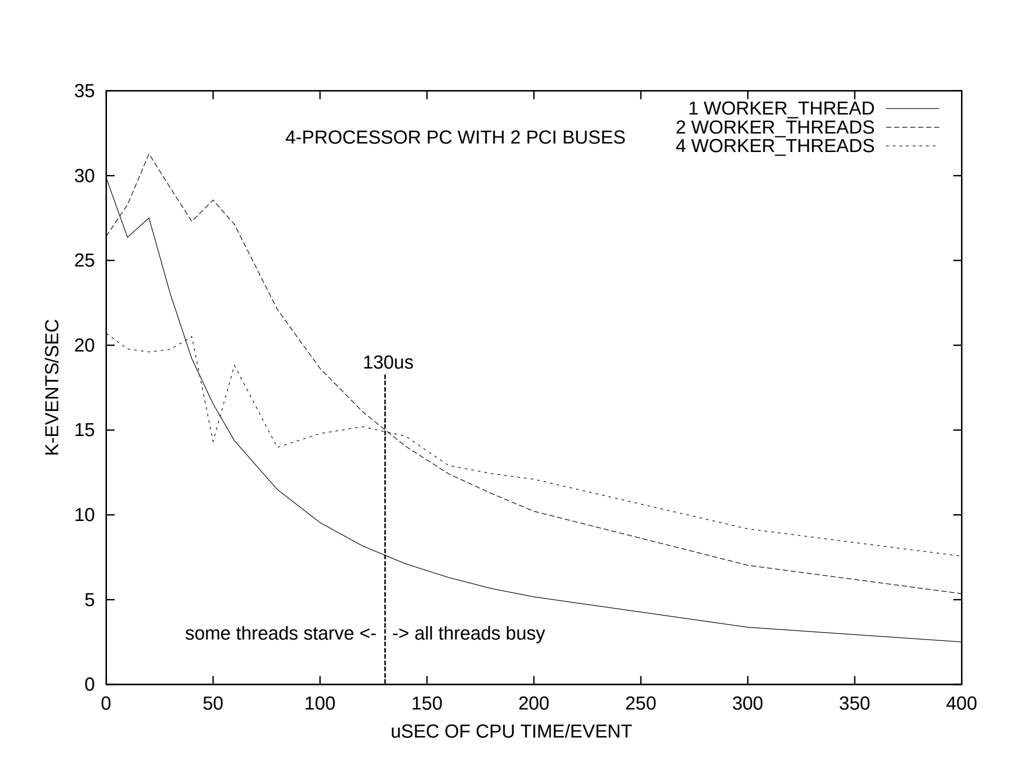

Figure 2 **Throughput dependence on CPU utilisation**. The event rate is shown as a function of CPU time (in µs) used per event. All events are composed of 4 ROBs with 1kB/ROB, i.e., 4kB/event fragment. The measured points in figure 2 were at 0, 10, 20, 30, 40, 50, 60, 80, 100, 120, 140, 160, 180, 200, 300, 400 µs/event.

Table 1 shows the percentage of total CPU utilisation by the worker threads (running the "algorithms"), for different algorithm process time with *four concurrent worker threads* processing the events. .

Table 3 Four WorkerThreads consuming different CPU times per event

µs/event	MB/s	Kevents/s	useful_time/ total_time	µs_eff	µs_io
100	55	13.8	35%	289	189
200	49	12.2	61%	327	127
300	37	9.2	69%	435	135
400	30	7.5	75%	532	132

As the CollectionThread is actively polling the MicroEnable boards using ~one processor, the observed maximum of 75% CPU, achieved for a CPU time/event of 400µs is the expected maximum. The last column shows the difference between the effective time/event and the CPU time used per event. This time represents the I/O time spent in the Request-, Collection- and Response-Threads plus thread switching overhead, plus CPU idle time in the regime where the system is limited by I/O. It reaches a stable value of ~130µs in the regime where all CPUs are occupied, i.e. when the CPU time/event reaches a value of 180 or more µs/event. Below this value, the I/O rate is insufficient to keep all worker threads active.

For zero CPU usage time the system is completely dominated by I/O time which varies between 33 and 50µs per event corresponding to an event rate of 20-30kHz. Since the input from the ROBs is handled by one CPU (the CollectionThread) the remaining CPUs are mainly idle. Adding more worker threads only increases the competition for scarce events resulting in additional overheads. Figure 2 clearly shows the detrimental effect of adding more worker threads for zero computation time as well as the cross-over points where additional CPUs start to become effective.

CPU times of 180 or more µs/event give consistent utilisation results. The behaviour for lower CPU/event values, the results seem somewhat erratic and we ascribe this to the thread switching and locking mechanisms, not fully understood. We conclude that the problems of multi-threaded behaviour on SMP nodes (and on multi-node systems in general) need to be studied in the context of the ATLAS TDAQ in general.

A.3 Conclusions from the preliminary measurements

- these preliminary measurements made on an off-the-shelf SMP system using as ROBs the commercial MicroEnable board (purchased from Silicon Software) show that an AROBC with significant pre-processing and concentration capabilities can be set up from commercial components alone (the exception being the adaptation of the driver for the MicroEnable board). Today's commercial multi-processor systems are packaged in a way that makes their integration into a system of farm processors a trivial task;
- simulation results reported earlier (see ATLAS DAQ note 99-020) show that the impact of such systems in the ATLAS level-2 trigger is substantial: the overall traffic is alleviated in frequency and bandwidth, optimisation of drivers is localised, detector-specific tuning is facilitated;
- our measurements further show that the I/O capacity available in SMP systems can largely be put to use by the commercial interface cards we used for ROBs;
- measurements of internal communication at application level have also shown that the substantial CP capacity available in SMPs can largely be made available to user programs in situations approaching those of ATLAS level-2 traffic. Not all obtained data points are fully understood, however, and more work is required to show the effect of the operating system in managing high-level communication (thread switching);
- SMP systems of the type evaluated are available from several manufacturers; this fact and their competitive pricing make them a serious candidate for the active ROB complex application in Atlas: an active ROB complex must thus be considered an important element in the overall architecture discussion, independently of other options (e.g. technologies, one or two networks, VME-based readout crate or PCI cards directly in a PC).

B. Measurements on a ccNUMA architecture

In the beginning of 1999 we assessed the performance of an SCI ccNUMA architecture. The motivation for this was:

- a large machine with many I/O slots may considerably simplify the data collection and subsequent processing (“feature extraction”) for an entire sub-detector

- SCI based ccNUMA architectures built from COTS components scaling up to a reasonably large size are components were commercially available
- the software effort for testing this hypothesis is small

The AViiON25000 was generously made available by Data General for the test.

B.1 Test Setup

A shared memory version of the ATLAS communications benchmark suite, http://www.cern.ch/RD11/combench.html, was adapted for this machine. Each node of the AViiON25000 consists of a quad Pentium Xeon board with two PCI buses giving 13 PCI slots per node. Apart from the usual two level of caches each node is equipped with a 16 Mbytes third level SCI cache. Up to 16 nodes can be interconnected by a dual SCI ring of 500 Mbytes/s each to form a 64-processor ccNUMA system. Cache coherency at the system level is maintained by the IEEE Coherent SCI protocols. Tests were run between a pair of CPUs on the same board (“near nodes”) which does not involve SCI and between CPUs on different nodes (“far nodes”) which provokes transfers over the SCI rings subject to the coherency protocols. In both cases latency and throughput were measured. The following tests were run:

- ping pong: one node sends data to another node and waits for the data to be returned.
- by-directional ping-pong: both nodes send and return data to each other

other test were available but not used, see conclusion s below.

As data coming in from detectors is normally not cached care was taken to eliminate possible cache effects by placing source and destination data in successive memory areas within a 16 Mbyte block (far larger than the second level cache).

Furthermore, SCI cache coherency protocols utilise invalidation. Consequently, data sent to another node does normally not leave the local SCI cache until used by the far node. It was therefore necessary to touch the data on the receiving end to provoke actual data transfers.

B.2 Results and Conclusion

The results for uni- and bi- directional ping-pong for different message sizes are shown in table 4 and table 5 respectively.

- The results for communication between near nodes is without surprises: the bandwidth for large messages and two sided communication approaches the CPU to memory bandwidth of ~ 125 Mbytes/s, consistent with memory copy operations (the CPU to level-2 cache bandwidth is much larger, ~ 600 Mbytes/s). For one-way ping-pong, the bandwidth is halved as expected.
- The SCI bandwidth is surprisingly low, ~ 14 and 24 Mbytes/s for one and two way ping-pong respectively for one pair of far nodes. It is caused by the long latencies associated with the SCI cache coherence protocols: 2-3 SCI packets are required to

invalidate the far SCI cache (when the message is produced on the near node) and 1-2 to transactions later when the data is touched on the far node. Clearly, the message-passing paradigm based on writing data in the memory of a far node does not work well with the SCI coherency protocol.

- Large SMP systems of the type evaluated are not commonly available on the market, and they presently are expensive. We do not, at the present time, propose to pursue studies with the goal of using them in Atlas.

Table 4 Bandwidth and latency for uni-directional ping-pong for different message sizes on DG AVION 25000

	Bandwidth (MB/s)		**Latency (μs)**	
	Near-node	**Far-node**	**Near-node**	**Far-node**
4	1.5	0.15	2.6	26
16	5.6	0.51	2.7	30
64	20	2.0	3.1	30
256	37	5.6	6.6	44
1024	57	9.8	17	100
4096	52	13	75	306
16384	62	14	250	1150
65536	69	14	901	4610

Table 5 Bandwidth and latency for bi-directional ping-pong for different message sizes on DG AVION 25000

	Bandwidth (MB/s)		**Latency (μs)**	
	Near-node	**Far-node**	**Near-node**	**Far-node**
4	2.1	0.20	1.9	19
16	7.8	0.75	2.0	20
64	26	2.9	2.3	21
256	57	8.3	4.3	29
1024	89	16	11	59
4096	89	22	44	175
16384	125	24	125	651
65536	125	23	500	2700