

Measurements on a commercial Active Rob Complex

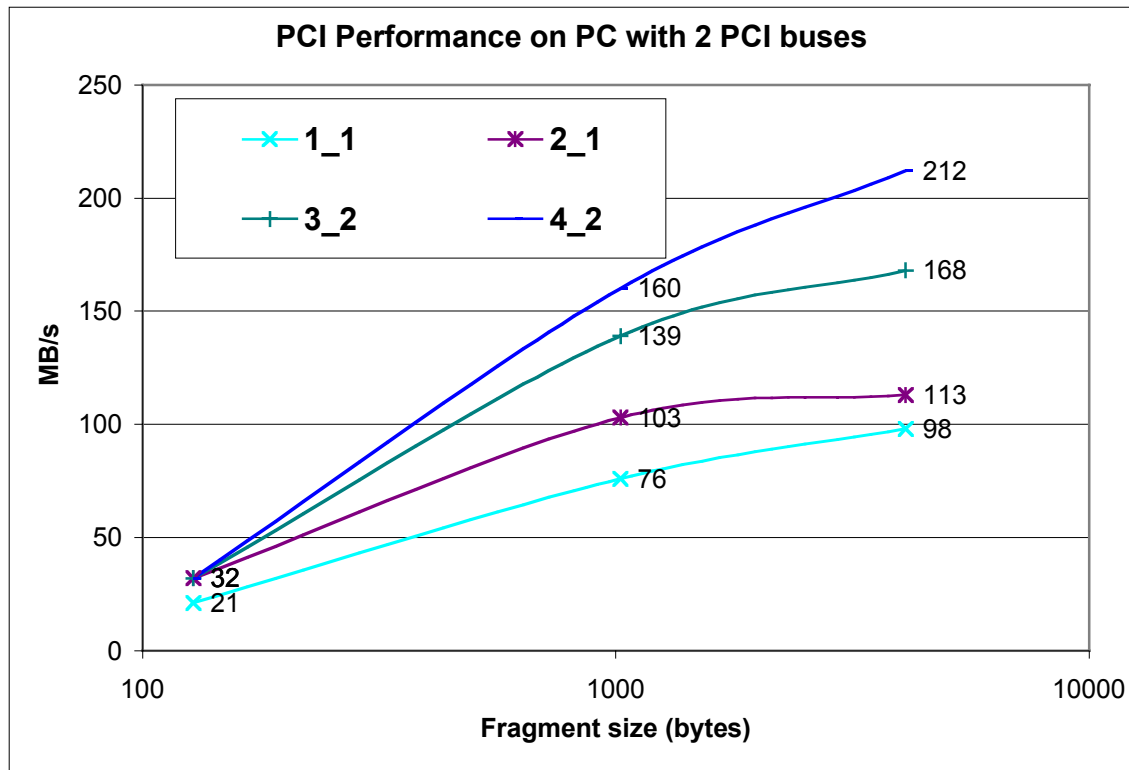
R.Bock, Y.Ermolin, A.Kugel, P.Werner

This note summarises preliminary results of measurements done on an Active Rob Complex, AROBC, composed of MicroEnable boards as Robins in an Elonex 4-processor PC, with 7 PCI slots spread over 2 independent PCI buses. The measurements were done at CERN and in Mannheim in November and December 1999. The results were partially presented (JV) at the ATLAS TDAQ workshop in Beatenberg, December 1999.

Note that the communication with ROB clients, e.g. steering processes, via the event data network is so far ignored in this study.

1) Maximum access rate and bandwidth

The primary goal was to measure the aggregate bandwidth for multiple Robins on



multiple PCI buses.

Figure 1 Bandwidth for 1-4 Robins on 1-2 PCI buses

Figure 1 shows the bandwidth as a function of the data size for 1 to 4 Robins; two PCI buses are used in the case of three or four Robins. The rate increase for the second PCI bus, i.e., from two to four Robins is 88%.

2) CPU time available for algorithms

Figure 2 shows the event rate as a function of CPU time (in μs) used per event. All events are composed of 4 Robs with 1kB/Robin, i.e., 4kB/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.

<u>us/Ev</u>	<u>MB/s</u>	<u>KEvs/s</u>	<u>useful time/totaltime</u>	<u>us eff/Ev</u>	<u>usI/O</u>
100	55	13.8	35% $(0.1 \cdot 13.8 \cdot 100) / 4$	289	189
200	49	12.2	61% $(0.2 \cdot 12.2 \cdot 100) / 4$	327	127
300	37	9,2	69% $(0.3 \cdot 9.2 \cdot 100) / 4$	435	135
400	30	7.5	75% $(0.4 \cdot 7.5 \cdot 100) / 4$	532	132

Table 1. Four WorkerThreads consuming us/Event us for each event

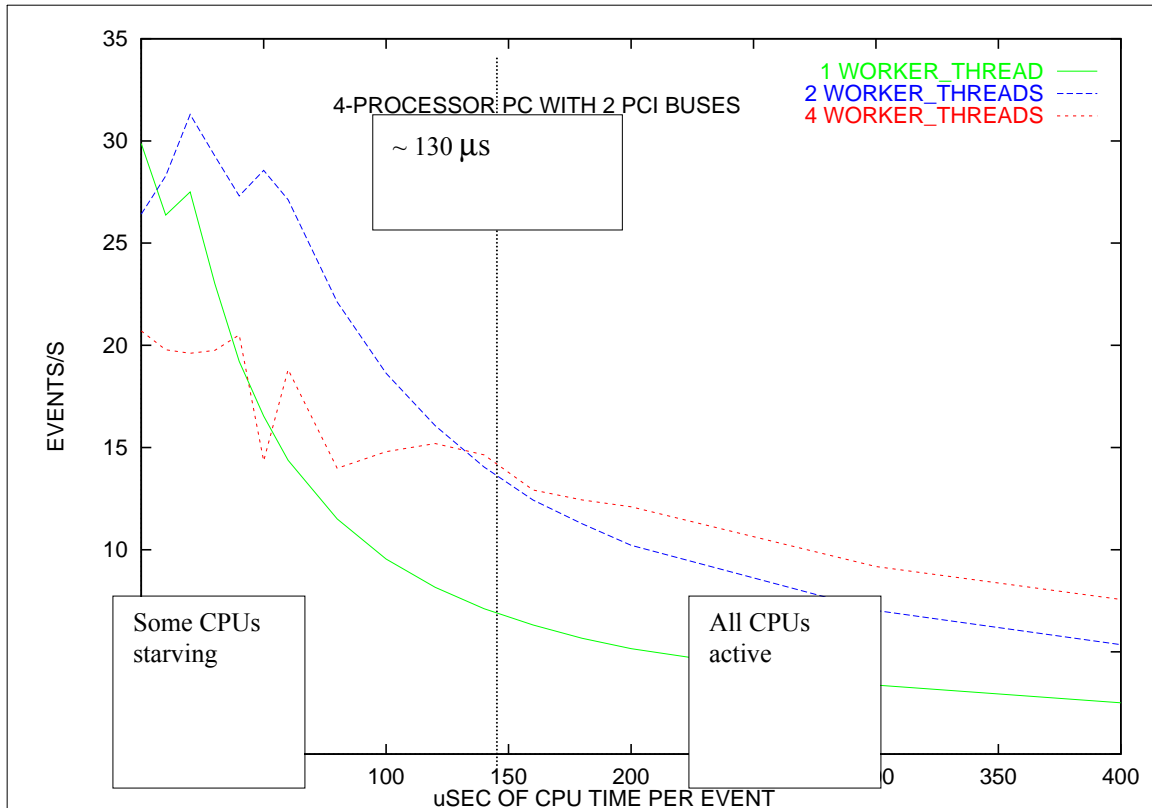


Figure 2 Throughput dependence on CPU utilisation

The measured points in figure 2 were at 0, 10, 20, 30, 40, 50, 60, 80, 100, 120, 140, 160, 180, 200, 300, 400 $\mu\text{s}/\text{event}$.

Four types of threads are used in the test program:

- 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 Robin fragments 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).

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 400 μ s 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 overheads *plus CPU idle time in the regime where the system is limited by I/O*. It reaches a stable value of ~130 μ s in the regime where all CPUs are occupied, i.e. when the CPU time/event reaches a value of 180 or more μ s/event. Below this value (the left side of the plot in Figure 2) 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 50 μ s per event corresponding to an event rate of 20-30kHz. Since the input from the Robins is handled by one CPU (the *Collection Thread*) 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 μ s/event give consistent utilisation results. The behaviour for lower CPU/event values, the results seem somewhat erratic and we ascribe this to 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.

3) Limits set by the Memory Bus

The I/O rate does not only depend on PCI performance: when loading the memory bus with a read/write flow of 130+130MB/s the total PCI drops by about 20%, viz. 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 expected; we hope that future SMP boards will have a better behaviour in this respect.

4) Conclusions

- These preliminary measurements made on an SMP system with MicroEnable as fast Robins show that an AROBC with significant pre-processing and concentration capabilities can be set up from commercial components alone (the exception being the driver for the MicroEnable board). 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.
- For the canonical 1KB fragment size, an aggregated bandwidth of 160MB/s has been measured.
- For pre-processing tasks needing 200 μ s or more of processing time per event, a very reasonable CPU availability for user tasks is achieved, up to 75% of totally available CPU time. For little CPU time spent per event, the behaviour of multi-threaded applications needs further investigation. This should be done within the whole TDAQ context.