# Abstract Communication Benchmarks on Parallel Systems for Real-time Applications

J. Apostolakis [a], R. Hauser [a], C. Hörtnagl [b] K. Koski [a], I. Legrand [c], R. van Wijk [d], I. Zacharov [e]

[a] *CERN, Geneva, Switzerland*

[b] *University of Innsbruck, Austria*

[c] *DESY, Zeuthen, Germany*

[d] *NIKHEF, Amsterdam, Netherlands*

[e] *Silicon Graphics Inc., Switzerland*

We present a set of abstract communication benchmarks designed to test the basic communication capabilities and more advanced communication schemes of various commercial parallel computer systems. The benchmark package consists of seven programs measuring basic communication parameters and three benchmarks relevant for our real-time application, a (second-level) trigger system for the ATLAS experiment.

*Key words:* Networking; Network performance; Switches; Massively Parallel Systems

## 1 Introduction

Future accelerators such as the LHC at CERN will produce an enormous amount of data which has to be partly processed on-line in the trigger system. The Atlas [1] detector foresees a three level trigger system, with the first level implemented in custom hardware and the third level as a farm of commercial workstations or PCs. It would be desirable, that the second-level trigger, which has to keep up with an event rate of about 100 kHz, can be implemented with as many commercial and off-the-shelf components as possible. To reduce the overall data volume of about 100 Gbyte/s, the second-level trigger uses a level-1 guided *region of interest* concept, where only a small subset of the event data (a few kBytes per event) is required. Several options for the second-level trigger are currently under study, from FPGA based hardware to farms of commercial computing systems [2].

## 2 Abstract Communication Benchmarks

Experience so far shows that the real problem will not be the raw bandwidth, but the overhead associated either with the given switching technology or the operating system.

To assess the performance of a number of commercially available parallel systems, we defined a set of *abstract* basic communication benchmarks (1.1 to 1.7), which are not specific for Atlas. We also added three *application* benchmarks, each consisting of a small set of processor nodes with or without a supervisor, and running a very simplified version of the currently proposed second-level trigger solutions [2].

All benchmarks are executed for a varying number of packet sizes (64, 256, 1024) and a varying number of processors where applicable (2, 4, 8, 16, 32). Packet sizes are restricted to the expected Atlas parameters, although implementors are free to scan the whole parameter space.

## 3 Benchmark Description

A more detailed definition of the benchmarks can be found at [3], as well as the full results in both tabular and graphical form. An example implementation in C is available. Default implementations for MPI [4] and shared memory are provided and the software has been adapted to several low-level libraries from different vendors.

*Ping-Pong (1.1)*

One node sends to another node and waits for the reply. We define two parameters: the overhead derived from dividing the round-trip time for 1 byte by two, and the effective bandwidth in the considered range of packet sizes. The values appear as 1.1$a$ and 1.1$b$ in the table of results.

*Two-Way (1.2)*

Both nodes are sending and receiving simultanously. We calculate the same values as before, now denoted as 1.2$a$ and 1.2$b$. Note that the overhead is smaller than the one for benchmark 1.1, since there we see twice the overhead on the send and receive side, while in this benchmark the setup times will partially overlap. The 1.2$c$ entry gives the ratio of 1.1$a$ and 1.2$a$.

*All-to-All (1.3)*

All nodes are sending to each other simultanously. Increasing the number of nodes in the system increases the overall throughput. What we extract is a parameter which describes the deviation from an system which would scale ideally:

$$T(kN) = \alpha \times k \times T(N)$$

$\alpha$ is the parameter we denote as 1.3. Results greater than 1.0 mean less perfect scaling behaviour. $\alpha$ is not necessarily constant for large systems, so the results also show the variation for the range of nodes we considered.

*Pairs (1.4)*

$N$ nodes send to $N$ receivers. The same calculations as for Ping-Pong and Two-Way are done. Again the overhead is smaller than for the previous benchmarks. The preparation of the next data packet can overlap with the actual sending of the current packet. The parameters are reported as 1.4$a$ and 1.4$b$.

*Outfarming (1.5), Broadcast (1.6), Funnel (1.7)*

For 1.5 one node sends packets to $N$ receivers, while 1.6 uses hardware broadcast if present. In 1.7 $N$ nodes send to a single receiver.

For all three benchmarks, we derive the same parameter as for the All-to-All measurements. The value of $\alpha$ is reported as 1.5, 1.6$a$ and 1.7. Results small than 1.0 show the amount of overlap in send or receives operations. Finally, we take the ratio of the timings for *Outfarming* and *Broadcast*, showing the performance of available broadcast hardware. This parameter is reported as 1.6$b$.

*Application Benchmarks*

The *push farm* (2.1) consists of a number of *sources* and *destinations* and a single *supervisor* which allocates processors in round-robin fashion. *Sources* get messages from the supervisor and then send their data to the *destination* node, which in turn reports back to the supervisor.

In the *pull farm* (2.2) the supervisor sends a single packet to the *destination* node, which then explicitly requests data from the *sources*.

The *pipeline* (2.3) essentially combines data streams like in the *Funnel* benchmark (1.7), but over several stages.

The application benchmarks can be considered to execute several of the basic operations described above.

It turns out that benchmark 2.1 (Push Farm) is mainly dominated by the *outfarming* step (which doesn't use broadcast in our case).

The *Pull Farm* does not have this restriction, since the supervisor only sends its message to a single node which then collects the data itself. The more receiver nodes are put into the system, the better the time per single event, until it reaches a saturation point, given by the speed of the network interface of the supervisor.

Since the results depend on a number of input variables, like number of sources and destinations, they are not represented in the following table by a single number, but can be found at [3].

## 4 Results

The following table gives an overview of the results.

| | Cray T3E | IBM SP2 | Meiko CS2 | | Parsytec CC | DEC 8400 | SGI Challenge | SGI Origin |
|---|---|---|---|---|---|---|---|---|
| Nodes | 64 | 16 | 32 | | 16 | 8 | 16 | 40 |
| Comm | MPI | MPI | MPI | Channel | MPI | SHM | SHM | SHM |
| 1.1a | 15.7 | 74.4 | 128.5 | 20.3 | 217.0 | **3.7** | 12.9 | 7.1 |
| 1.1b | 20.3 | 6.5 | 5.2 | 10.4 | 3.4 | 39.6 | 22.1 | **41.8** |
| 1.2a | 13.0 | 44.5 | 68.5 | 17.2 | 177 | **3.5** | 10.0 | 6.4 |
| 1.2b | 23.6 | 10.4 | 7.2 | 14.9 | 4.3 | 37.4 | 25.4 | **47.1** |
| 1.2c | 1.21 | 1.67 | 1.88 | 1.18 | 1.23 | 1.06 | 1.29 | 1.11 |
| 1.3 | $\mathbf{1.10 \pm 0.10}$ | $1.21 \pm 0.08$ | $1.20 \pm 0.12$ | $1.22 \pm 0.17$ | $1.28 \pm 0.42$ | $1.58 \pm 0.08$ | $1.40 \pm 0.27$ | $1.66 \pm 0.40$ |
| 1.4a | 13.7 | 31.0 | 102.0 | 22.4 | 188 | 4.33 | 14.0 | **8.75** |
| 1.4b | 19.3 | 10.8 | 6.24 | 11.4 | 3.68 | 30.7 | 12.2 | **31.1** |
| 1.5 | $1.02 \pm 0.10$ | $0.98 \pm 0.07$ | $\mathbf{0.82 \pm 0.07}$ | $0.96 \pm 0.07$ | $1.00 \pm 0.03$ | $0.93 \pm 0.02$ | $0.95 \pm 0.12$ | $0.98 \pm 0.11$ |
| 1.6a | $0.58 \pm 0.07$ | $0.62 \pm 0.02$ | $0.67 \pm 0.07$ | $\mathbf{0.50 \pm 0.02}$ | $0.67 \pm 0.08$ | $0.81 \pm 0.18$ | $0.78 \pm 0.20$ | $0.78 \pm 0.16$ |
| 1.6b | 1.78 | 1.56 | 1.23 | **1.88** | 1.50 | 1.15 | 1.23 | 1.25 |
| 1.7 | $\mathbf{0.69 \pm 0.15}$ | - | 1.0 | $0.95 \pm 0.12$ | $1.02 \pm 0.07$ | $1.02 \pm 0.07$ | $1.25 \pm 0.37$ | $0.97 \pm 0.04$ |

1.1$a$, 1.2$a$, 1.4$a$ in $\mu s$,1.1$b$, 1.2$b$, 1.4$b$ in Mbyte/s, all others are dimensionless.

## 5 Conclusions

Several of the systems show communication overheads which are below 10 $\mu s$ for small packets, corresponding to 100 kHz. Results for low overhead are best for shared memory machines. So far pure SMP systems were restricted to a few tens of processors, but the recent move to NUMA (non-uniform memory access) architectures shows that shared memory systems can scale to more than a hundred CPUs. The best results for MPP systems like the Cray T3E differ only by about a factor of two for the basic overhead.

## References


[1] ATLAS Technical Proposal, CERN/LHCC/94-43, 1994.

[2] Atlas Level-2 Trigger Groups, Atlas Second-Level Trigger Options, submitted to CHEP'97.

[3] `http://www.cern.ch/RD11/combench.html`

[4] A Message Passing Interface Standard, Version 1.1, 1995.