Performance Evaluation and Modeling of the Fujitsu AP3000 Message-Passing Libraries[⋆]

Juan Touriño and Ramón Doallo

Dep. of Electronics and Systems, University of A Coruña, Spain
{juan,doallo}@udc.es

Abstract. This paper evaluates, models and compares the performance of the message-passing libraries provided by the Fujitsu AP3000 multicomputer: MPI/AP, PVM/AP and Fujitsu APlib (versions 1.0). Our aim is to characterize the basic communication routines using general models. Several representative parameters and performance metrics help to compare the different primitives and to detect inefficient implementations.

1 Introduction

The performance of the communication primitives of a parallel computer does not only depend on the underlying hardware, but also on their implementation. Users do not know the quality of the message-passing implementations and they can find that the performance of their parallel applications makes worse in other machine or using other message-passing library.

We have focussed on low-level tests to study basic communication primitives on the Fujitsu AP3000 [2]. The AP3000 has UltraSparc-II processors connected via a high-speed communication network (AP-Net) in a two-dimensional torus topology. We have considered point-to-point communications, one-to-all (broadcast) and all-to-one (specifically, a reduction operation). Though more primitives could be analyzed, these ones are the basis for the design of more complex communication patterns in a parallel application.

2 Point-to-Point Communications

The Hockney's model [1] characterizes message latency T:

$$T(n) = \frac{n_{\frac{1}{2}} + n}{Bw_{as}} \tag{1}$$

where n is the message length, Bw_{as} is the asymptotic bandwidth, and $n_{\frac{1}{2}}$ is the half-peak length (n required to obtain $Bw_{as}/2$). Besides, $Bw_{as} = 1/t_b$ and $n_{\frac{1}{2}} = t_s/t_b$, being t_s the startup time and t_b the transfer time per data unit ($T(n) = t_s + t_b n$). The specific performance $\pi_s = 1/t_s$ characterizes short-message performance, while Bw_{as} shows long-message performance. Only blocking primitives are considered: *MPI_Send/Recv*, *pvm_psend/precv* and *l_asend/arecv* (APlib).

[⋆] This work was supported by the CICYT (Contract TIC96-1125-C03) and by the EU (1FD97-0118-C02). CESGA (Santiago de Compostela, Spain) provided the AP3000

Table 1. Point-to-point communication parameters and metrics

	MPI/AP	PVM/AP	APlib
t_s (μs)	69	53	46
t_b (μs)	0.0162	0.0162	0.0162
π_s (Kbytes/s)	14.15	18.43	21.23
Bw_{as} (Mbytes/s)	58.87	58.87	58.87
$n_{\frac{1}{2}}$ (bytes)	4260	3272	2840

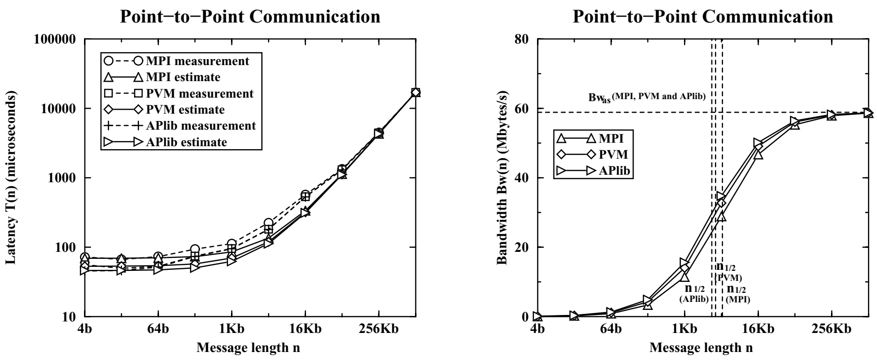

Fig. 1. Latency and bandwidth for point-to-point communications

Table 1 shows the estimated parameters for the three AP libraries. As can be observed, Bw_{as} is the same for the three libraries; therefore, their latencies tend to be similar as the message length increases. Regarding short messages, there are appreciable differences in the startup times. APlib has the lowest t_s and MPI the highest. Consequently, the best $n_{\frac{1}{2}}$ is achieved by APlib. The *pvm_send/recv* routines were also tested: we have estimated $t_s \approx 50\mu s$, but the transfer time increases excessively, $t_b = 0.0263\mu s$, which results in $Bw_{as} = 36.26$ Mbytes/s. It is due to message packing and unpacking operations. Figure 1 depicts in the left-hand graph a comparison between the estimated and measured latencies for the three libraries and different message sizes. The corresponding estimated bandwidth is presented in the right-hand graph.

3 One-to-All Communications: Broadcast

We have applied the model proposed by Xu and Hwang in [3] to characterize one-to-all communications and, more specifically, the broadcast communication:

$$T(n,p) = t_s(p) + \frac{n}{Bw_{as}(p)} \tag{2}$$

where p is the number of processors; $Bw_{as}(p)$ can be also expressed as $1/t_b(p)$ and we can similarly define $n_{\frac{1}{2}}(p) = t_s(p)/t_b(p)$ and $\pi_s(p) = 1/t_s(p)$.

An additional metric is the aggregated (ag.) asymptotic bandwidth Bw_{as}^{ag}, the ratio of the total number of bytes transferred in the collective operation

Table 2. Broadcast parameters and metrics (k$_1$=$10^6/2^{10}$, k$_2$=$10^6/2^{20}$)

	MPI/AP	PVM/AP	APlib
$t_s(p)$ (μs)	69log$_2$p	22p	46+25(p-2)
$t_b(p)$ (μs)	0.0162log$_2$p	0.0110p	0.0162+0.0110(p-2)
$\pi_s(p)$ (Kbytes/s)	k$_1$/(69 log$_2$p)	k$_1$/(22p)	k$_1$/(25p-4)
$\pi_s^{ag}(p)$ (Kbytes/s)	k$_1$(p-1)/(69 log$_2$p)	k$_1$(p-1)/(22p)	k$_1$(p-1)/(25p-4)
π_s^{pag} (Kbytes/s)	43.43	40.69	36.29
$Bw_{as}(p)$ (Mbytes/s)	k$_2$61.73/log$_2$p	k$_2$90.91/p	k$_2$/(0.0110p-0.0058)
$Bw_{as}^{ag}(p)$ (Mbytes/s)	k$_2$61.73(p-1)/log$_2$p	k$_2$90.91(p-1)/p	k$_2$(p-1)/(0.0110p-0.0058)
Bw_{as}^{pag} (Mbytes/s)	180.63	79.47	83.13
$n_{\frac{1}{2}}(p) = n_{\frac{1}{2}}^{ag}(p)$(bytes)	4260	2000	(25p-4)/(0.0110p-0.0058)
$n_{\frac{1}{2}}^{mag}$(bytes)	4260	2000	2346

and the time required to perform the operation, as $n \to \infty$. For a broadcast, $Bw_{as}^{ag}(p) = (p-1)Bw_{as}(p)$. Similarly, the ag. specific performance $\pi_s^{ag}(p) = (p-1)\pi_s(p)$ shows the performance of a broadcast for short messages. The ag. half-peak performance $n_{\frac{1}{2}}^{ag}(p)$ can be also defined as n that achieves $Bw_{as}^{ag}(p)/2$ ($n_{\frac{1}{2}}^{ag}(p) = n_{\frac{1}{2}}(p)$). All these measures depend on p. It would be interesting to define peak performance measures to have a global estimate of the behaviour of collective communications. Therefore, we propose the following metrics: the peak ag. bandwidth $Bw_{as}^{pag} = \max_{2 \le p \le p^{max}} Bw_{as}^{ag}(p)$, the peak ag. specific performance $\pi_s^{pag} = \max_{2 \le p \le p^{max}} \pi_s^{ag}(p)$ and the minimum (ag.) half-peak length $n_{\frac{1}{2}}^{mag} = \min_{2 \le p \le p^{max}} n_{\frac{1}{2}}^{ag}(p)$, being p^{max} the maximum p available for users ($p^{max} = 12$ in our machine).

The routines considered in the comparison are: *MPI_Bcast*, *pvm_mcast* and *cbroad* (APlib). The fitting of the components of Eq. 2 and the additional performance metrics are shown in Table 2. Note that, for p=2, in MPI and APlib the numerical values of the model's parameters are the same as the ones of the point-to-point model. Latency is $O(log_2 p)$ in MPI, which reveals that the broadcast in MPI is implemented using a binomial tree-structured approach. In PVM, latency is $O(p)$; it seems that *pvm_mcast* is implemented as a sequence of sends all originating from the root processor. The APlib broadcast is also $O(p)$. Therefore, MPI performance is the best, and it is better as p increases. The results for PVM and APlib are very similar, although $t_b(p)$ is slightly better for the APlib broadcast and the startup time is a bit lower in PVM. Regarding $n_{\frac{1}{2}}(p)$, in MPI and PVM is a constant and in APlib is almost constant, because the complexities of $t_s(p)$ and $t_b(p)$ are the same within each message-passing library.

Figure 2 shows some experimental results for the broadcast routines, by fixing p=8 and n=64 Kb, respectively. The second graph shows that the model is very accurate for PVM and APlib. This graph also reveals that in MPI, for n=64 Kb, the startup time of the model should be a bit higher, although the fitting is acceptable. Figure 3 depicts $\pi_s^{ag}(p)$ and $Bw_{as}^{ag}(p)$. It can be observed that $\pi_s^{ag}(p)$ for MPI is lower than for PVM and APlib because the startup in MPI is

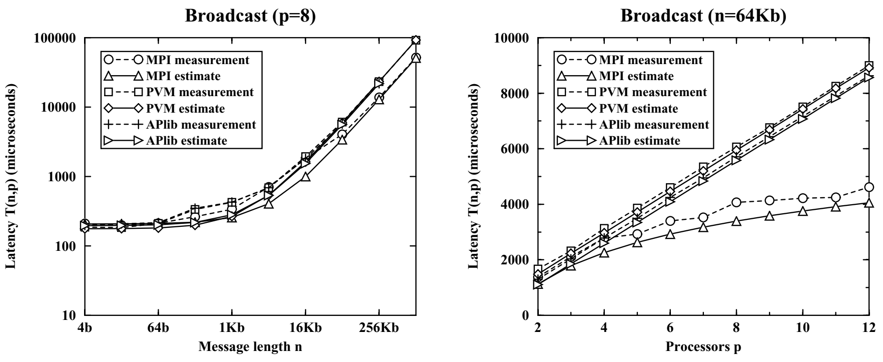

Fig. 2. Latency for broadcast operations: a) p=8, b) n=64 Kb

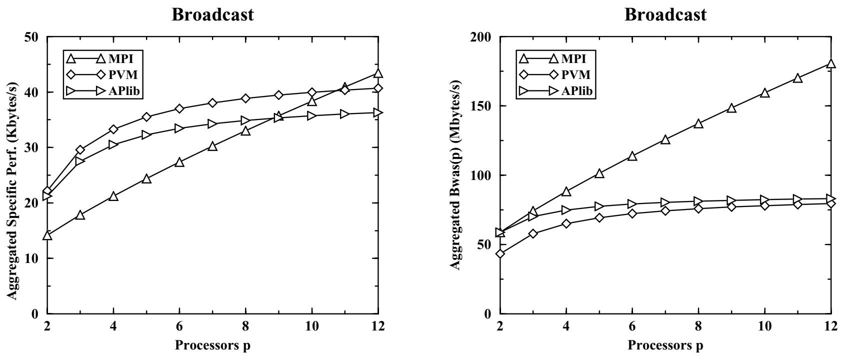

Fig. 3. Broadcast aggregated metrics: a) $\pi_s^{ag}(p)$, b) $Bw_{as}^{ag}(p)$

high for a small number of processors. But, from p=10, the $O(log_2 p)$ complexity of t_s leads to a better $\pi_s^{ag}(p)$. This improvement would be more pronounced for a greater number of processors. In the second graph, $Bw_{as}^{ag}(p)$ of MPI is clearly the best due to the log complexity of the broadcast. In PVM and APlib $Bw_{as}^{ag}(p)$ is poor and it tends to be constant as p increases. Consult the corresponding peak values of these functions (π_s^{pag} and Bw_{as}^{pag}) in Table 2.

4 All-to-One Communications: Reduction

The model of Eq. 2 hides a parameter in collective primitives that involve computations (e.g., a reduction): t_c, the cost per byte of the performed computation. We propose an extension of the model, valid for all collective communications:

$$T(n,p) = t_s(p) + \frac{n}{Bw_{as}(p)} + t_c(p)n \tag{3}$$

Clearly, for a broadcast $t_c(p) = 0$. The metrics defined in Section 3 can be also applied here, and we propose a new metric: the ratio transfer time-computation time $r_{cc}(p) = t_b(p)/t_c(p)$, which provides a view of the weight of the computation factor as opposed to the communication factor in the total latency.

We modeled the sum reduction of doubles in MPI (*MPI_Reduce*) and PVM (*pvm_reduce*). The APlib reduction was not modeled because it only works on single numbers and stores the result in all the processors involved in the reduction; therefore, it is not comparable to MPI and PVM.

We have found that the PVM reduction (and, in general, the group management routines) are poorly implemented. The reduction routine is not robust: it does not work for $p > 6$. Besides, latencies are dominated by very high startup times: for p=2, $t_s \approx 5.55ms$ and it seems to be $O(p)$; for p=3 and n=64Kb, t_s represents $\approx$ 80% of latency.

Regarding MPI reduction, we have obtained the following results: $t_s(p) = 90log_2p - 15$, $Bw_{as}(p) = 1/(0.0171log_2p + 0.0037)$ and $t_c(p) = 0.0051log_2p - 0.0037$. As expected, MPI reduction is $O(log_2p)$. Additional performance metrics are: $\pi_s^{pag} = 34.92$ Kbytes/s, $Bw_{as}^{pag} = 161.38$ Mbytes/s and $r_{cc}(p) = (0.0171log_2p + 0.0037)/(0.0051log_2p - 0.0037)$. Although $r_{cc}(p)$ varies from 14.86 (for p=2) to 4.46 (for p=12), it tends to be a constant since p=4 ($t_b(p) \approx 5t_c(p)$).

5 Conclusions

The models and metrics used in the previous sections help us to identify design faults in the communication routines and, furthermore, to estimate the performance of parallel programs. Machine vendors should provide the parameters of these models (or, at least, complexities in the case of collective communications) for basic communication routines.

Regarding the AP3000 message-passing libraries, we can conclude that the PVM/AP library (specially, the group routines) is a naive implementation which should be greatly improved. The APlib routines are not robust for long messages (the machine crashes for 1 Mbyte messages), the broadcast implementation is inefficient and the reduction routines only work on single numbers. Besides, APlib is a proprietary library with a small set of primitives compared to MPI.

Currently, MPI/AP is the best choice to program the AP3000. Nevertheless, the performance of the AP3000 hardware is not fully exploited by the MPI/AP library. Message latencies could be reduced by re-designing the low-level communication mechanisms. Hardware improvements, such as the SBus design (the I/O bus which connects the processor and the message controller), could also help to reach this aim.

References

[1] Hockney, R.W.: The Communication Challenge for MPP: Intel Paragon and Meiko CS-2, Parallel Computing **20**(3) (1994) 389–398

[2] Ishihata, H., Takahashi, M., Sato, H.: Hardware of AP3000 Scalar Parallel Server, Fujitsu Sci. Tech. J. **33**(1) (1997) 24–30

[3] Xu, Z., Hwang, K.: Modeling Communication Overhead: MPI and MPL Performance on the IBM SP2, IEEE Parallel & Distributed Technology **4**(1) (1996) 9–23