

GEOGRAPHICALLY DISTRIBUTED COMPUTING:

ATM over the NASA ACTS Satellite⁰

PATRICK W. DOWD¹, FRANK A. PELLEGRINO,¹ TODD M. CARROZZI,¹ SARAGUR M. SRINIDHI²

¹ Dept. Electrical & Computer Engineering
State University of New York at Buffalo
Buffalo, NY 14260

² Sterling Software/NASA Lewis Research Center
21000 Brookpark Road
Cleveland, Ohio 44135

Abstract – This paper outlines some of the problems and the solutions developed to support geographically distributed computing via ATM. In particular, applications developed with the Parallel Virtual Machine (PVM) [1] message passing library, communicating via ATM at OC3c speeds (155 Mbps) through the NASA ACTS satellite are considered. A primary goal of this work is to assess the suitability of an ATM-based network to support interprocess communication and remote file I/O systems for distributed computing. This paper restricts itself to the behavior of PVM in a large propagation delay environment. Refer to [2–4] for additional detail and performance results.

1 Introduction

There is interest in developing a capability of supporting geographically distributed computing. This would allow more effective resource sharing and improved utilization of computing resources. However, the propagation delay between two computer systems can severely hamper the achievable performance. In this paper we consider this problem, define three possible solutions to it and then illustrate the results of the solutions as implemented.

We are primarily concerned with geographically distributed computing, and use the NASA ACTS satellite to investigate this problem. This poses a particularly extreme situation since the propagation delay is on the order of hundreds of milliseconds. The experimental scenario is illustrated in Figure 1, where the end points are a CRAY YMP supercomputer that is used for CFD computation, and an SGI Onyx which is used for visualization.

A major issue to be examined is the use of ATM-based local-area and wide-area networks in distributed computing. In particular, a primary goal of this work is to assess the suitability

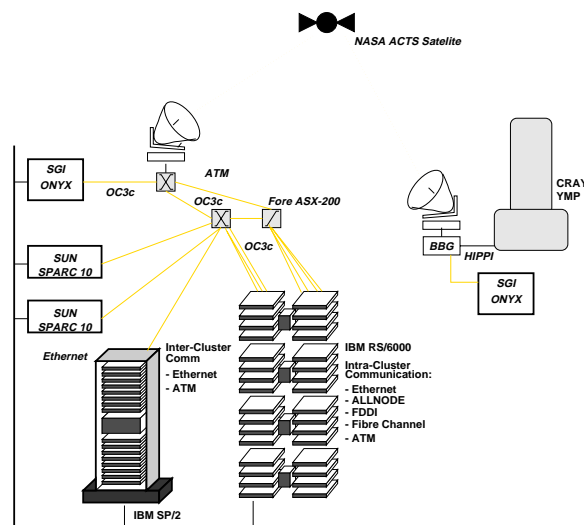


Figure 1: Experimental Scenario

of an ATM-based network to support the interprocess communication and remote file I/O system requirement of distributed computing.

ATM and cluster-based computing provide a possible solution to the need for supercomputing power without the economic implications. A very valuable resource can be created if only a fraction of the idle CPU cycles of workstations within an organization could be harnessed together. Many organizations already have a large quantity of high end workstations, which more than likely sit idle for a large amount of time. Cluster-computing is becoming increasingly important as evidenced by the number of large corporations who have recently de-commissioned their supercomputers to be replaced by clusters.

Organizations may also have supercomputers and workstations geographically dispersed, perhaps in different cities. ATM offers scalable performance and is attractive in that it enables a stronger degree of integration between high and low end processors. ATM, combined with support for locality-

⁰This work was supported by NASA through Grant NAG3-1548. The authors can be reached through {dowd,fap,carrozzi,saragur}@lace.lerc.nasa.gov and would like to thank Dave Brooks of NASA/Lewis for his assistance. Preprints of this paper, and other related works, can be obtained through WWW at <http://piranha.eng.buffalo.edu/>.

exploiting load balancing and process placement, offers the potential to support this objective. In this paper we will begin to examine some of the basic issues necessary for implementation as a large scale computing resource. This paper describes an effort to merge supercomputers and workstation clusters in an integrated environment, cooperating in both the local area and the wide area.

A geographically distributed application will be partitioned into localities based on communication patterns. There may be significant latency between localities, particularly when they are mapped to clusters of processors residing in different cities, so the applications need to be suitably partitionable. A goal of this paper is to examine the overhead incurred in supporting both levels and assess the improvements possible with ATM.

The objective of this paper is to determine the performance characteristics and associated overhead, particularly in the context of a high speed network in a large propagation-delay environment. PVM [1] is used in this paper as the message passing library, while other possible choices of message passing libraries include APPL [5], MPI [6, 7], Express [8]. Message passing library overhead, operating system issues such as buffer moves, and protocol overhead are important issues but are dwarfed by the potentially devastating performance impact of the large propagation delay. Refer to [9, 2] for additional detail. This paper quantifies the performance impact and develops ways to circumvent the limitations.

2 Experimental Setup

Figure 1 illustrates the basic scenario of the investigation. The objective is to support high-performance computing across the ACTS satellite, using a supercomputer on one side of the country to execute the application and another machine thousands of miles away for visualization in real-time.

A primary objective of this effort was to assess the impact of the propagation delay on the performance of distributed computing applications. A major impact to performance is the software structure that must be supported. In particular, PVM is used as the message passing library in this experiment, primarily since the main applications are written to PVM.

Figure 2 shows the software environment of the experiment. The application uses the PVM libraries which communicate through a transport-level protocol, through the operating system and then to the network device driver.

PVM can use both TCP and UDP as its transport level protocols. Figure 3 illustrates the relationship between the application processes and the PVM routing daemon (pvmd). Normally, TCP is used for communication within a host and the routing daemon while UDP is used for communication between the routing daemons located in different machines. An alternative technique is PvmRouteDirect which establishes a TCP connection directly between the communicating PVM

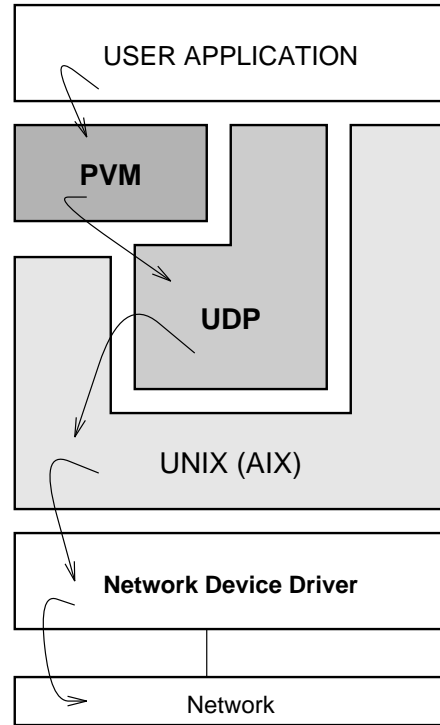


Figure 2: Experimental software environment.

processes and eliminates the intermediate step performed by the routing daemon.

The layers of software incur a significant latency due to the buffer moves and the context switches and other overhead.

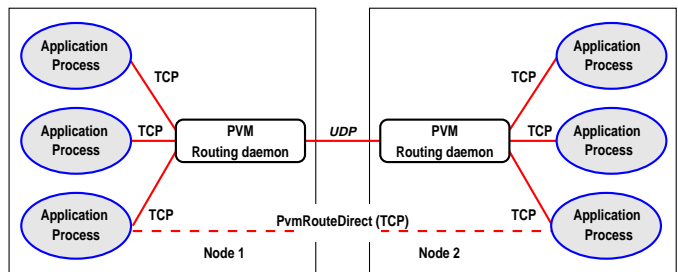


Figure 3: Relationship of transport-level protocol with application processes and PVM routing daemon.

3 The problem: protocol overhead and propagation delays

This section illustrates the major problems faced in accomplishing the objectives of this project. The primary concern

is software overhead caused by system calls and buffer moves between the layers, and propagation delay between remotely connected nodes.

Propagation delay is particularly severe in a satellite-based environment. Figures 4- 5 illustrate some of the performance results measured during this experiment.

3.1 Performance Metric

The graphs presented within this document use an aggregated latency, defined as follows:

1. Let X be the number of messages to be sent and P be the message size in bytes.
2. Let T denote the time required to send (and receive acknowledgment for) the X messages.
3. The aggregate latency is then defined to be $t = T/X$.

The graphs presented in this paper all use a repetition rate of $X = 100$, and P is varied.

This metric was devised to give an indication of the amount of latency overlapping that takes place. Conceptually, the path between the satellite receivers can be viewed as a very long bit pipe where each 100 milliseconds of propagation delay can hold almost 2 MB of data at OC3 speeds. The overall performance is highly dependent on the transport protocol and the amount of packet concurrency that can be supported. For example, consider the case if the transport protocol support an stop-and-wait type of protocols where a packet is transmitted and then the transmitter waits for an acknowledgment from the receiver before it sends another packet. If the packet was 1024 bytes and the round trip propagation delay (RTT) is 540 ms, then the maximum throughput on an OC3 link would be about 15 kb/s or a utilization of 3 orders of magnitude less than optimal. This is particularly important in the large propagation delay environment of a satellite.

3.2 Performance Impact

Figure 4 illustrates the performance devastation that takes place in using PVM via ATM and the ACTS satellite. This graph plots average repetition latency (of the metric defined in Section 3) in milliseconds to the message size in bytes.

We see that there is essentially a two order magnitude increase in latency. The local link is a measurement of ATM between the CRAY YMP and SGI Onyx which are located with a physical distance on the order of 10's of meters.

The impact to throughput is illustrated in Figure 5 for this situation. The performance is severely impacted.

One cause of this performance degradation is the protocol implemented within PVM to provide reliability. UDP is connectionless, so guaranteed delivery is not provided. The

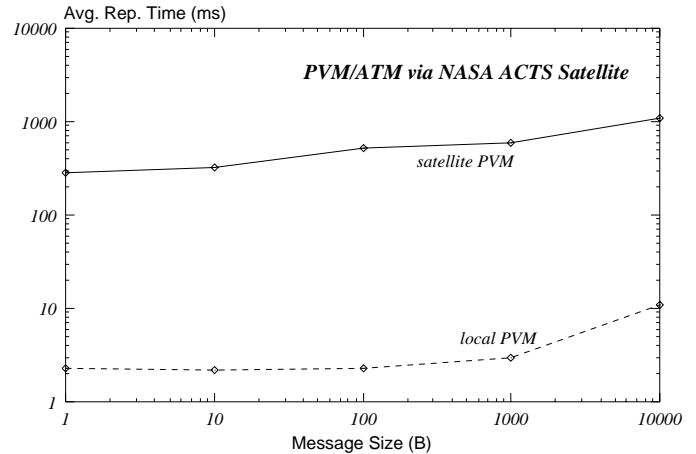


Figure 4: Latency (ms) versus message size (bytes) in a local ATM and satellite ATM environment.

reliability is provided by PVM. PVM essentially uses a stop-and-wait protocol, where a UDP datagram is sent, and further transmission by the source PVM application is deferred until an ACKNOWLEDGMENT (generated by the receiver PVM application) is received by the source PVM application. Therefore, one UDP datagram could be transmitted per round trip time (RTT). For example, if the RTT propagation delay is 540ms, then the throughput of that application would be limited to $P/0.540$ bytes/second.

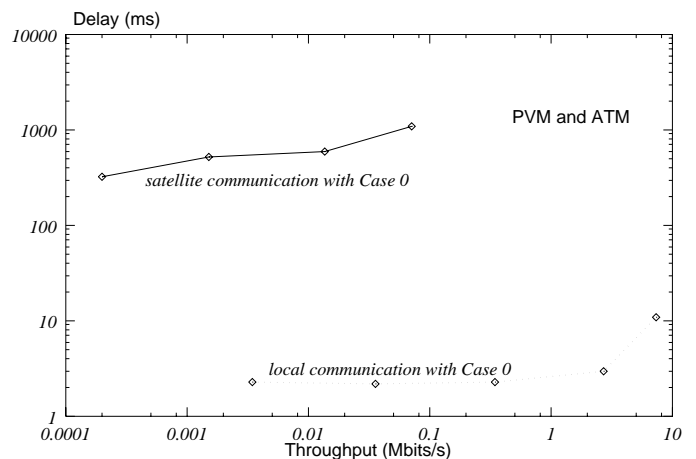


Figure 5: Latency (ms) versus throughput (Mbits/s) in a local ATM and satellite ATM environment.

4 Latency Overlap Strategy

This section describes below the plan for addressing the current limitation in PVM and TCP for the ACTS Satellite project.

The concern is that PVM uses TCP to communicate between processes within the same processor and uses UDP, via a communications daemon, to communicate between processors.

PVM has made UDP reliable by adding a stop-and-wait type of protocol to ensure packet delivery. In a satellite based environment, this would allow one packet per RTT period of 540ms. The result of this situation is discussed in the previous section.

The following sections define the three scenarios considered during this experiment and then illustrates the resulting performance variations.

4.1 Case 1

Two different modifications are considered to support this case. This case restricts the modification to PVM and does not try to improve TCP or UDP performance.

- (a) Using the 'nopax' field within the pvmd daemon, it may be possible to increase the window size and use a go-back-n strategy rather than the current stop-and-wait approach.

This is not supported or encouraged by the PVM developers since it requires a modification to the pvmd daemon.

- (b) Another aspect of this solution is a direct modification of PVM. There are three major timers that needed to be adjusted to deal with the widely varying propagation delay. We found, through testing, that PVM was timing-out early and transmitted as many as 14 packets for each successfully transmitted and acknowledged packet.

4.2 Case 2

The next case exploits PvmRouteDirect which provides TCP connections between communicating processes. PvmRouteDirect is not used very often since it is unstable and has buffer race conditions that frequently causes a process to lock up.

This approach will also require extensive testing. We would need to do three things to the TCP versions located on the CRAY and SGI machines:

- ▷ increase the maximum window size through the window scale capability of TCP
- ▷ disable delayed ACKs
- ▷ disable cold-start so the system does not self-modulate

However, in this case it may be suitable since it is a direct point-to-point connection.

4.3 Case 3

This case would require us to make major modifications to the PVM communication daemon, replacing the UDP connections with TCP sockets. We would also need to include the TCP changes described in Case 2.

This modified version of PVM would be of only limited interest since it will require a large number of open sockets and will not offer any significant latency improvement over UDP since all external communication will still be routed via the intermediary routing daemon. However, the number of open sockets is not a problem for the ACTS experiment since it will be a point-to-point link and so we do not expect a large number of connections.

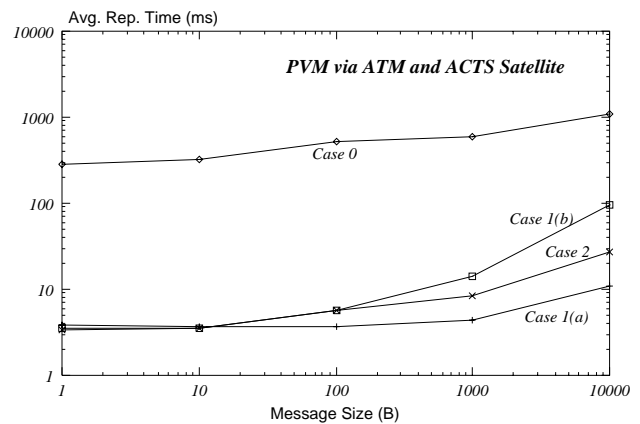


Figure 6: Delay (ms) versus message size (bytes), includes the baseline with the three cases.

5 Performance Results

This section describes the improvements to performance resulting from the change and modifications described in the previous section.

Figure 6 plots the delay metric (ms), defined in Section 3, with the message size (bytes). This graph includes the baseline, using PVM in a standard way: PVM Application(TCP)→PVMD(UDP)→PVMD(TCP)→PVM Application. and also includes the performance with the first 3 cases. Case 3 is not listed since it did not perform as well as Case 2 in all cases.

Figure 6 indicates a “reduction” in latency of 2 orders of magnitude. Actually, this is not true since the RTT has not decreased but the amount of overlap has increased. In particular, this case enables the entire 100 packets to be transmitted without waiting for the acknowledgment.

Figure 7 plots delay (ms) versus throughput (Mbits/s), includes the baseline with the three cases. This graph shows how the usable throughput is significantly increased and the “aggre-

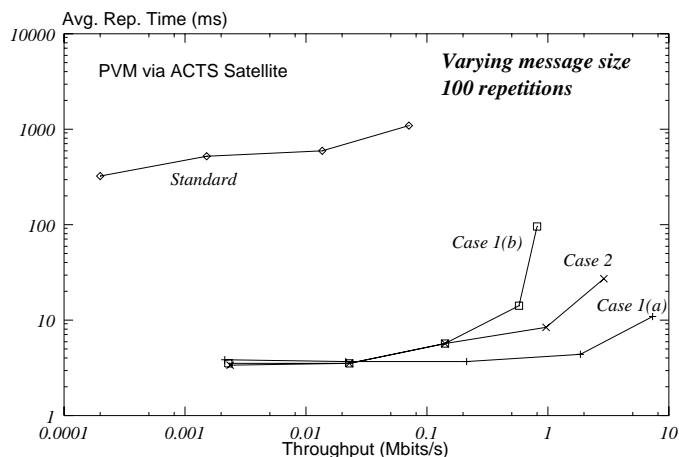


Figure 7: Delay (ms) versus throughput (Mbits/s), includes the baseline with the three cases.

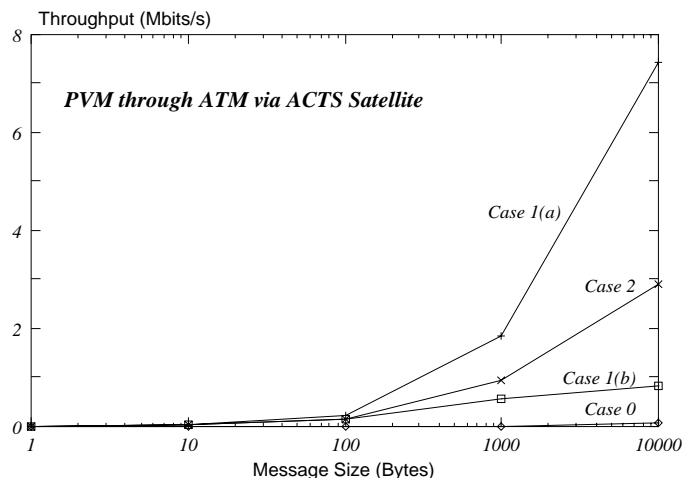


Figure 8: Throughput (Mb/s) versus message size (bytes), includes the baseline with the three cases.

gate" latency is reduced. This illustrates the significant amount of overlapping that is taking place, enabling the propagation delay to be de-emphasized.

Figure 8 plots the throughput as it varies with message size. This graph illustrates that a significant fraction of the capacity that was wasted due to protocol inefficiencies is recovered.

6 Conclusions

This paper has reported on the performance improvements achieved through a series of developments to both PVM and its associated transport-level protocol. The main problems with PVM was the stop-and-wait ARQ type protocol that was used to ensure reliability. A second issue was with protocol timers, which actually had a surprisingly large performance impact and caused PVM to repeat many transmissions per successful attempt. We have reduced some of the performance degradation of PVM, but have not completely eliminated it since there are buffer copies between the layers. PVM was not written for a high-speed environment and so the authors did not focus on minimizing the latency. The main concern with the transport protocol was the small receive buffers, delayed acknowledgments and slow start. These problems can be alleviated to reduce the performance degradation somewhat through an increase in the receive buffers, but the problem is not solved. This problem is just postponed, and will soon return as data rates continue to increase.

References

[1] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam, "PVM 3.1 Users Guide and Reference Manual,"

Tech. Rep. ORNL/TM-12187, Oak Ridge National Laboratory, May 1993.

- [2] P. Dowd, E. Blade, S. Srinidhi, and R. Claus, "Issues in atm support of high performance geographically distributed computing," in *Proc. IPPS'95 Workshop on High Speed Networks*, (Santa Barbara, CA), pp. 352–358, April 1995.
- [3] P. Dowd, E. Blade, S. Srinidhi, and R. Claus, "Affordable high performance computing: Evaluation of cluster-based computing via ATM," in *Proc. 7th IEEE LAN/MAN Workshop*, (Marathon, FL), pp. 352–358, March 1995.
- [4] E. Blade, R. Claus, P. Dowd, and S. Srinidhi, "An experimental assessment of network impact on cluster-based computing," in *Proc. 1994 Gigabit Network Workshop (GBN'94)*, (Toronto, Ontario Canada), June 1994.
- [5] A. Quealy, G. L. Cole, and R. A. Blech, "Portable programming on parallel/networked computers using the application portable parallel library," Tech. Rep. 106238, NASA Technical Memorandum, July 1993.
- [6] Message Passing Interface Forum, University of Tennessee, *MPI: A Message-Passing Interface Standard*, 1.0 ed., May 1994.
- [7] J. Dongarra, R. Hempel, A. Hey, and D. Walker, "A draft standard for message passing in a distributed memory environment," in *Proceedings of the Fifth ECMWF Workshop on the Use of Parallel Processors in Meteorology*, (Reading (UK)), pp. 465–81, World Scientific (Singapore), Nov 1993.
- [8] J. Flower and A. Kolawa, "Express is not just a message passing system: current and future directions in express," *Parallel Computing*, vol. 20, pp. 597–614, April 1994.
- [9] M. Lin, J. Hsieh, D. Du, J. Thomas, and J. MacDonald, "Distributed network computing over local atm networks," Tech. Rep. TR-94-17, Computer Science Department, University of Minnesota, Minneapolis, Minnesota, 1994.