An MPI Tool to Measure Application Sensitivity to Variation in Communication Parameters

Edgar A. León[1], Arthur B. Maccabe[1], and Ron Brightwell[2]

[1] Computer Science Department, The University of New Mexico, Albuquerque, NM 87131-1386, {leon,maccabe}@cs.unm.edu
[2] Scalable Computing Systems, Sandia National Laboratories, Albuquerque, NM 87185-1110, bright@cs.sandia.gov

Abstract. This work describes an apparatus which can be used to vary communication performance parameters for MPI applications, and provides a tool to analyze the impact of communication performance on parallel applications. Our tool is based on Myrinet (along with GM). We use an extension of the LogP model to allow greater flexibility in determining the parameter(s) to which parallel applications may be sensitive. We show that individual communication parameters can be independently controlled within a small percentage error. We also present the results of using our tool on a suite of parallel benchmarks.

1 Introduction

Parallel architectures are driven by the needs of applications. Message-passing developers and parallel architecture designers often face optimization decisions that present trade-offs which affect the end performance of applications. A better understanding of the communication requirements of these applications is needed. To this end, we have created an apparatus which allows the alteration of communication parameters to measure their effects on application performance. This apparatus identifies the communication parameters to which an application is sensitive. Our apparatus is based on Myrinet along with the GM message-passing system. The communication parameters used are based on the LogP model [1] and have been instrumented into GM so we can vary communication performance. MPI applications can be analyzed through a port of MPICH on top of GM.

The remainder of this paper is organized as follows. Section 2 provides a brief overview of GM and the communication parameters used in this work. Section 3 describes the instrumentation and validation of our tool. Section 4 provides results of applying our tool to a collection of parallel benchmarks. Section 5 concludes with a discussion of related work and conclusions.

2 Background

GM is a low-level message-passing system created by Myricom as a communication layer for its Myrinet network. GM is comprised of a kernel driver, a user library, and

a program that runs on the Myrinet network interface, called MCP (Myrinet Control Program). GM provides reliable, in order delivery of messages.

To characterize communication performance of a parallel system, we use an extension of the LogP model [1]. This model is characterized by four parameters: (L) latency, the time to transmit a small message from processor to processor; (o) overhead, the time the host processor is involved in message transmissions and receptions and cannot do any other work; (g) gap, the time between consecutive message transmissions and receptions (the reciprocal of g corresponds to the available per-node bandwidth); and (P) the number of nodes in the system. Considering that the send and receive operations are often not symmetric, the overhead and gap have been further separated into four parameters: o_s and o_r, the send and receive overhead; and g_s and g_r, the send and receive gap. This further separation provides a more detailed characterization of which communication parameters have a significant impact on applications. We also consider the gap per byte (Gap) to distinguish between short and long messages.

3 Instrumentation and Validation

To vary communication performance, we have modified GM to allow the alteration of communication parameters [2]. To empirically validate and calibrate our tool, we varied each communication parameter over a fixed-range of values while leaving the remaining parameters unmodified. Using a micro-benchmark [3], we measure the value of the parameters and verify that the average error difference (*Err*) between the desired value of the varied parameter and the measured value is small. We also verify that the remaining parameters remain fairly constant with low standard deviation (*Std*).

The results presented here were gathered using single-processor nodes on a Myrinet network. Each node has a 400 MHz Intel Pentium II processor with 512 MB of main memory and a Myrinet LANai 7.2 network interface card with 2 MB of memory. GM version 1.2.3 and a Linux 2.2.14 kernel were used.

Table 1 shows the results of varying o_s and g_s for 8 byte messages. The *Measured values* columns represent the values of the communication parameters measured using the micro-benchmark. As expected, the added overhead in Table 1.A is not independent of the gap (g). To keep the latency (L_1) constant when varying the gap (Table 1.B), we use only one ping-pong trial in the computation of the round-trip-time (RTT_1), as opposed to one-hundred as in Table 1.A (RTT). Since just one trial is considered, the value of the latency, L_1, is greater than L due to *warm-up issues*. The remaining communication parameters also show low error difference and low standard deviation on unmodified parameters [2].

4 Results

To test our tool on real applications, we measure the effects of varying the communication parameters on a collection of parallel benchmarks. This effect is quantified by measuring the running time slowdown of applications due to changing communication parameters. The variation in communication parameters is done by increasing each

Table 1. Varying o_s and g_s for 8 byte messages. All units are given in μs.

	Table 1.A		Measured values				
	%Err	Goal	o_s	o_r	g	RTT	L
		0.00	1.34	4.33	23.92	48.82	18.73
	0.14	21.34	21.31	3.82	31.22	88.72	19.21
	3.39	41.34	42.74	3.07	51.29	128.48	18.41
	0.42	61.34	61.60	4.09	71.21	169.05	18.83
	0.00	81.34	81.34	4.34	91.26	212.04	20.32
	0.01	101.34	101.35	4.18	111.24	252.01	20.46
	0.13	121.34	121.18	4.63	131.14	290.76	19.55
Avg	0.61			4.16			19.50
Std	1.05			0.39			0.82

Table 1.B	Measured values						
%Err	Goal	g	o_s	o_r	RTT_1	L_1
	0	23.91	1.36	4.33	125.07	56.84
	20	27.70	1.35	4.33	125.06	56.84
1.95	40	40.78	1.38	4.28	125.06	56.86
0.52	60	60.31	1.52	4.15	124.06	56.35
0.23	80	80.18	1.39	4.27	125.05	56.85
0.29	100	100.29	1.41	4.27	125.06	56.85
0.15	120	120.18	1.37	4.37	125.06	56.78
1.52			1.40	4.27	124.98	56.80
3.31			0.05	0.06	0.63	0.31

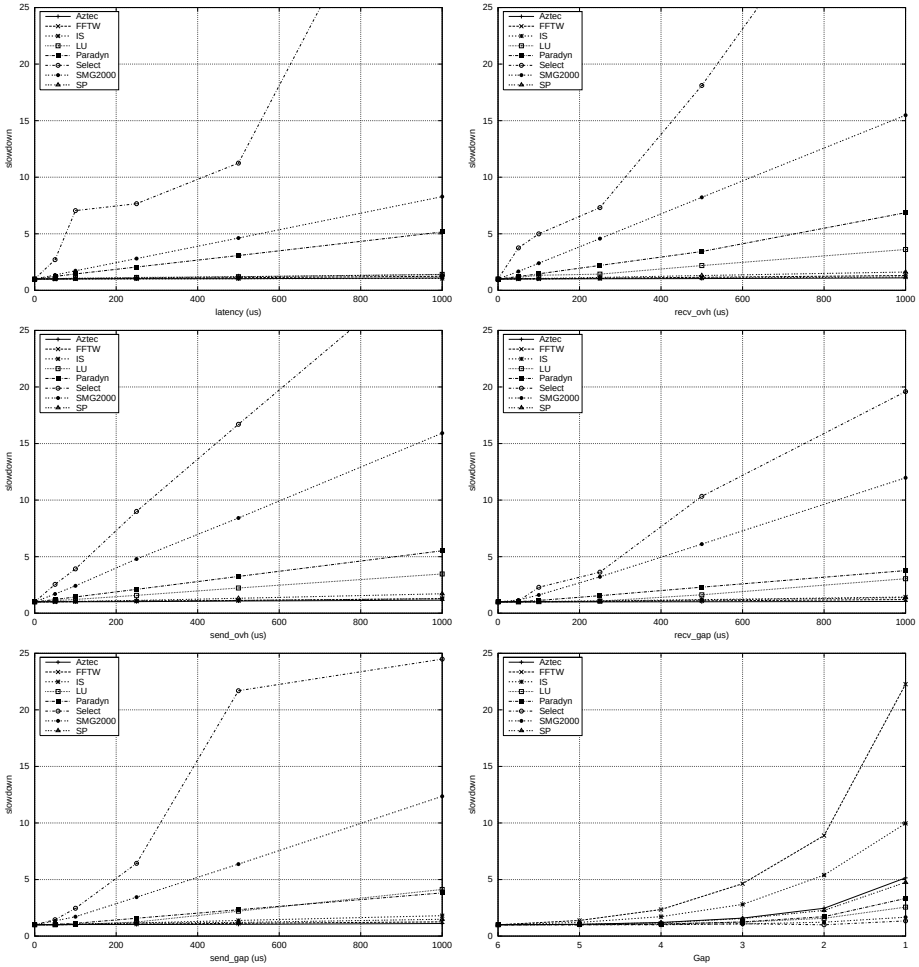

Fig. 1. Sensitivity to communication parameters. These set of fi gures plot the run time slowdown of applications vs. the communication parameters. The Gap is given by i, where 2^i represents the available bandwidth of the system.

parameter independently of the others. The results were gathered using 16 computational nodes with MPICH-GM version 1.2.1..7b. Our collection of benchmarks consists of eight applications: *FFTW* (Fast Fourier Transform) and *Select* (Selection problem); *Aztec*, *ParaDyn* and *SMG2000* from the ASCI Purple set of benchmarks; and *IS*, *LU* and *SP* from NPB2. These applications represent a variety of (a) degree of exploitable concurrency, (b) computation to communication ratio and (c) frequency and granularity of inter-processor communications.

Figure 1 shows the graphs of application sensitivity to variation in communication parameters. We found that the applications used in this work have no significant difference between sensitivity to send parameters (overhead and gap) and receive parameters due to the symmetry of their communication patterns. Also, overhead has a more significant impact on applications than the other communication parameters. With the exception of Select, applications are fairly insensitive to latency. Bandwidth or Gap per byte has been identified as a significant parameter to application performance due to the current trend of programmers to cluster small messages together into one message, reducing a significant amount of communication overhead and taking advantage of the increasing communication bandwidth.

5 Related Work and Conclusions

Martin and others studied the impact of LogGP communication parameters on parallel applications [4]. The applications used were written in Split-C on top of Generic Active Messages. They found the host overhead to be critical to application performance. Our work, in contrast, uses GM as the communication layer and focuses on applications written in MPI. Although the applications used in our work differ from those used by Martin, the results are similar. Applications were fairly insensitive to latency and sensitive to overhead.

We have created an apparatus to vary communication performance, based on the LogP model, on high-performance computational clusters. This tool is useful to designers of parallel architectures and message-passing developers to answer the question: What do applications really need? It can also be useful to parallel application developers and users to identify sources of performance degradation of an application on parallel architectures.

References

1. Culler, D., Karp, R., Patterson, D., Sahay, A., Schauser, K.E., Santos, E., Subramonian, R., von Eicken, T.: LogP: Towards a realistic model of parallel computation. In: Proceedings of the 4th ACM SIGPLAN PPoPP, San Diego, CA (1993) 262–273
2. León, E.A., Maccabe, A.B., Brightwell, R.: Instrumenting LogP parameters in GM: Implementation and validation. In: Workshop on HSLN, Tampa, FL (2002)
3. Culler, D.E., Liu, L.T., Martin, R.P., Yoshikawa, C.O.: Assessing fast network interfaces. IEEE Micro **16** (1996) 35–43
4. Martin, R.P., Vahdat, A.M., Culler, D.E., Anderson, T.E.: Effects of communication latency, overhead, and bandwidth in a cluster architecture. In: Proceedings of the 24th ISCA, Denver, CO (1997) 85–97