THE EFFECTS OF CLUSTER SIZE ON PACKET INTER-ARRIVAL PATTERNS AND INTENSITY IN A DISTRIBUTED SYSTEM

Dennis Guster, St. Cloud State University, MN, USA, dcguster@stcloudstate.edu
Paul Safonov, St. Cloud State University, MN, USA, safonov@stcloudstate.edu
Renat Sultanov, St. Cloud State University, MN, USA, rasultanov@stcloudstate.edu
Mark Nordby, St. Cloud State University, MN, USA, noma0401@stcloudstate.edu

ABSTRACT

Although the design and implementation of parallel processing infrastructure is not considered a mainstream IS topic, a working knowledge of its capabilities can have an important influence on improving the design of on-line systems. In order to illustrate some of the advantages and limitations of using parallel processing in information systems two experiments were run to collect multiple node scaling information. Specifically, packet inter-arrival data was collected and analyzed for a message passing interface (MPI) and an HTTP problem that simulated packet movement in an enterprise level LAN test-bed. The parallelization method that we employed broke the problem into N subparts based in the number of processors used. Inter-processor communication was required whenever a processor needed to interact with another one to help solve the application. That communication was relayed via the MPI protocol using TCP packets with intensities in .0005 second range. Experimental trials were run on various platforms: from 2 to 12 two processor units for MPI, and 8 and 16 units for HTTP. In the MPI experiment, although the CPU time continued to drop as additional units were added, the elapsed time only dropped to the 4 unit level and then increased thereafter. In the HTTP experiments the two switch/server model provided only a slightly better performance than the one switch/server model.

Keywords: computer network performance, distributed systems, parallel processing infrastructure, MPI, HTTP.

INTRODUCTION

The concept of a cluster allows multiple hosts to work together to solve complex problems. This inexpensive multiprocessor environment has often been viewed as a poor man's supercomputer. However, because the performance of such a cluster is very dependent on the underlying network infrastructure understanding the network topology and bandwidth is critical to analyzing cluster performance.

Predicting packet inter-arrival patterns and the effect on network performance has been a challenging problem for many years [8] on a regular network, but the many to many communication pattern of clusters make this problem even more challenging. This can be explained in part by the failure of packet distribution to mimic the theoretically expected Poisson distribution. A number of studies confirm that the actual inter-arrival distribution of packets is not exponential as would be expected in the classical model [7, 8, 13, and 6].

It has been recognized that the low cost and availability of workstation based LANs as a parallel processing environment is a viable and inexpensive solution [14].
Historically, at 10Mbs there was a concern that the LAN environment had inadequate bandwidth to support inter-process communication [3]. It has been shown that workstation clusters if interconnected via a high speed LAN can be an effective parallel processing environment [2]. In recent years, the hardware available to support workstation based LANs has improved, but many feel that relying on hardware alone may not ensure adequate performance for the demanding applications of the future [1].

It is widely accepted that a response in a client server system must be completed in a couple of seconds or less if end-user confidence is to be maintained. In the client/server world the problem to be solved requires at least two and probably more computers. Further, the performance and interaction between/among those computers is very much influenced by network connectivity. This paper attempts to provide scaling data related to splitting complex problems, so that ultimately the end-user response times can remain in a reasonable range. If multiple computers must work in concert to solve a problem often the inter-processor communication can become a bottleneck,

thus network configurations and different protocols were tested as well.

Too often it is assumed that putting two processors on the job instead of one will double performance. Unfortunately, that is seldom the case and, hopefully, the data reported herein will help clarify that myth. Further, it is also important to understand that not every bit transferred on the network is a data bit. Network traffic contains bits that are the actual payload and headers containing management information. The management bits, which are in essence overhead, provide a means to address performance optimization on the network software level [11]. This method appears to offer an approach to study the efficiency of network connectivity in a distributed computer grid.

There are several components that can be studied in optimizing performance on the network protocol level, such as optimizing the buffers, minimizing management traffic, and scheduling applications. In many instances planning and implementation of such methods can be improved by studying historical network traffic data obtained on the system in question. Truong and Fahringer [12] are proponents of such methodology and suggest that more research is needed that starts with the analysis of experimental data. They further state that such research will require better planning in regard to how to capture, store and analyze such data.

EXPERIMENTAL SCENARIOS

This paper looked at two different data sets: one in a parallel processing environment using a message passing interface (MPI), and another in a web environment using multiple replicated web servers accessed by multiple web clients.

In the first scenario workload performance data was collected from a live network in which parallel molecular modeling software was used to generate the network workload. The MOLDY program [10] was used to generate the MPI requests. In this program the number of server machines can be programmed, and thereby the offered intensity can be varied. TCPdump was used to collect the performance data. The number of servers was varied so that performance scaling could be evaluated. The following data was reported for each test run: CPU time in seconds, elapsed time in seconds, number of packets in sample, mean inter-arrival time, mean throughput in bytes and packet intensity in packets per second.

In the second scenario workload performance data was collected from a live network in which a web workload generator was used to create the network workload. The SEIGE program [4] was used to generate the WWW requests. In this program the number of server/client machines can be programmed and thereby the offered intensity can also be varied. Similarly to the first scenario, TCPdump was used to collect the performance data, and the number of servers was varied so that performance scaling could be evaluated. The following data was reported for each test run: number of packets in sample, mean inter-arrival time, mean throughput in bytes and packet intensity in packets per second.

NETWORK CONFIGURATIONS

The basic configuration of network follows. Two to sixteen Intel based hosts were used to run the experiment. Each host was configured with two 550 MHz CPUs that utilized symmetric multi-processing as supported by the Linux operating system. Each unit ran its part of the MOLDY or SEIGE program as required and housed the TCPdump program which collected packet traffic. Each unit was connected to a Force 10, E300 switch (a high end Enterprise level switch) at 100Mbs.

DATA COLLECTION STRATEGY

The parallel test-bed was programmed using either the MOLDY or SEIGE software. TCPdump was run on each unit and therefore traffic for each unit was collected locally. These files were later linked together by combining the files from each unit and then sorting by time stamp. The time on each unit was synchronized via a time server. Once combined the file could be analyzed and the inter-arrival times and throughput statistics were available from two to sixteen units. Two different sets of trials were run; one for the MPI problem the other for the WWW problem.

The first step was to obtain performance data on the live network for a range of different number of processors. Thus, each problem was solved by varying the number of units. In the MPI data the number of units was varied from 2 to 12. For the WWW traffic the number of servers was fixed at one per switch and the number of clients per switch was varied between 8 and 16 units. The results are listed in Tables 1 and 2.

ANALYSIS

The two tables provide strikingly different results, except for that in both cases the network load was quite intense. For the MPI data the CPU time required to solve the problems decreases as additional units are added. However, there is not a linear decrease. In fact, the decrease is only about three times when varying from 2 to 12 units. Elapsed time also exhibited similar traits within the two samples. In both cases it was reduced when four units were used, but increased steadily as additional units were added. Interestingly, packets in the sample provide a clue as to why the elapsed time increases when more than four units are utilized. When the number of units is increased from 4 to 6 the number of required packets about doubles, which results in significant communication overhead. Mean packet inter-arrival times are quite intense and continue to shorten as units are added. Throughput and packet intensity also exhibited for the most part an increasing intensity as units were added. In the WWW data, the average packet size was rather stabile which makes sense if one understands the profile of the HTTP protocol. Inter-arrival time was more intense for the 16 client trials than the 8 client trials. Distributing the clients across two switches increased the intensity slightly. Throughput and packet intensity were both larger in the 16 client trials and highest per capita in the two switch trials.

Table 1. MPI DATA.
Timings and Means for Packet Arrival, Throughput, and Intensity
10,000 Iterations

	2 concurrent hosts	**4 concurrent hosts**	**6 concurrent hosts**	**8 concurrent hosts**	**10 concurrent hosts**	**12 concurrent hosts**
CPU Time	141.84	91.06	67.06	59.58	50.70	50.22
Elapsed Time	245.99	200.4	227.58	263.91	259.14	264.85
Packets in Sample	201,361	191,377	470,706	469,598	602,578	603,034
Inter-arrival time	.00122	.00104	.000483	.000562	.000430	.000439
Throughput	575,668	707,134	1,265,710	1,090,568	1,396,049	1,337,841
Packet Intensity	817.64	960.42	2,069.78	1,771.63	2,327.71	2,231.22

Table 2. WWW Data.
Means for Packet Arrival, Throughput, and Intensity

	8 clients 1 switch	**16 clients 1 switch**	**8 clients 2 switches**	**16 clients 2 switches**
Average packet size	1,022	1,028	1,029	1,031
Packets in Sample	100,000	100,000	100,000	100,000
Inter-arrival time	.000439	.000351	.000410	.000300
Throughput	1,350,370	1,576,571	1,461,096	1,788,513
Packet Intensity	1,320	1,532	1,770	1,733

DISCUSSION

Within the MPI data in terms of providing an advantage in solving the problems in less time the method used fails when more than four units are utilized. While the reduction in the CPU time is encouraging as more units are added, the increase in network traffic offsets this advantage. Therefore, for the algorithm to be effective it needs to be reevaluated in terms of the massive amount of inter-processor communication utilized. However, if the problem was run on cluster that was not dedicated only to solving this problem, but had several parallel problems running at the same time, the saving observed in CPU time herein would be attractive.

Within the WWW data it appears that the test bed handled the traffic of up to 8 clients well. Distributing the data across two switches and hence two replicated servers only provides a slight increase in performance. It would be interesting to increase the workload well beyond the 16 clients to see how the test-bed might scale under increasing loads.

In the MPI data it appears that there are several other things to consider besides a redesign of the distribution algorithm. A network speedup to 1Gbs would have limited effect because the average throughput observed did not exceed 1.4 MBs. Also of note is the loss in packet payload efficiency as units are added. The packet average is about 700 bytes in the two and four unit tests, but it drops off to about 600 bytes in the 12 unit experiment. This may in part be explained by the overhead of setting up and maintaining the additional TCP connections used by MPI. There may be some promise in adapting the software to use PVM since it is based on UDP which is connectionless. Guster, Al-Hammah, Safonov, and Bachman [4] found that PVM could greatly reduce the communications overhead when compared to MPI. Also, perhaps a supercomputer would help in this regard because the processors would all be in the same box and connected via a high speed bus. However, an analysis of the number of management packets (such as TCP syn) in the data revealed that they typically accounted for only .05% of the total packets which may negate the potential of PVM. A further analysis of the packet sizes revealed that there were often a large number of small packets. In fact the number of packets less than 100 bytes averaged (payload less than 40 bytes) around 35% across the data and in some cases exceeded 45% within a single trial. These values help explain why the transfer rates observed were well below Ethernet's maximum of 1514 bytes.

Although a reduction in elapsed time was obtained up to the four unit level the scaling beyond that point was counter productive. Even though the decrease in CPU time is encouraging it is the elapsed time that counts on the end-user level. For this particular problem the effectiveness of using CPU distributed across a LAN is negated by the communication overhead. It appears this problem is quite complex and is a function of the time any CPU has to wait for a piece of the "puzzle" from another CPU. Speeding up the network may have some slight effect, but the transfer rates never exceed 1.5MBs (12Mbs) on a 100Mbs network. Therefore, more research is needed that addresses alternates to MPI (such as PVM and

Open MP) and reevaluates the algorithm used to distribute the workload among the CPUs.

Although the MPI and WWW data come from vastly different applications it is interesting to note that the throughput at the 8 unit (or client) level is similar. This level could be viewed as the concatenated volume that might be expected from 8 of the type of computers used in this experiment.

REFERENCES

1. Courson, M., Mink, A., Marcais, M. & B. Traverse. (2000) "An automated benchmarking toolset", *HPCN Europe*, pp. 497-506.
2. Fahringer, T. & R. Prodan. (2002) "ZENTURIO: An Experiment Management System for Cluster and Grid Computing", *4th International APART Workshop*, Euro-Par.
3. Fatoohi, R. & S. Weeratunga. (1994). "Performance Evaluation of Three Dis-tributed Computing Environments for Scientific Applications", *Proceedings of Supercomputing'94*, pp. 400-409.
   Fulmer, J. (2006), "SEIGE: An HTTP Regression Tester and Benchmark Utility", http://www.joedog.org/Seige/Manual.
4. Guster, D., Al-Hamamah, A., Safonov, P. & E. Bachman. (2003), "Computing and Network Performance of A Distributed Parallel Processing Environment Using the MPI and PVM Communication Methods"**,** *The Journal of Computing Sciences in Colleges*, 18(4), pp 248-253.
5. Guster, D., Robinson, D. & M. Richardson. (1999), "The Application of the Power Law Process in Modeling the Inter-Arrival Times of Packets in a Computer Network", *Midwest Decision Sciences Institute*, Springfield, IL, April 22-24.
6. Krzenski, K. (1998), "The Effect of Varying the Packet Inter-arrival Distribution in the Simulation of Ethernet Computer Networks", unpublished graduate research project, St. Cloud State University, MN.
7. Partridge, C. (1993), "The End of Simple Traffic Models (Editor's Note)", *IEEE Network*, 7(5).
8. Popescu, A. (1994), "A Parallel Approach to Integrated Multi-Gbit/s Communication over Multiwavelength Optical Networks", Ph.D. dissertation, Royal Institute of Technology, Stockholm.
9. Refson, K. (2000), "Moldy: a portable molecular dynamics simulation program for serial and parallel computers", *Computer Physics Communications*, 126 (310).
10. Riley, G. et al. (1997), "Performance Improvement through Overhead Analy-sis: a case study in molecular dynamics", *Proceedings of 11th ACM International Conference on Supercomputing*, 36-43.
11. Truong, H. & T. Fahringer. (2003), "On Utilizing Experiment Data Repository for Performance Analysis of Parallel Applications", *9th International EuroPar Conference (EuroPar'03)*, LNCS, Klagenfurt, Austria, Springer-Verlag.
12. Vandolore, B., Babic, G. & Jain, R. (1999), "Analysis and Modeling of Traffic in Modern Data Communications Networks". A paper submitted to the Applied Telecommunication Symposi-um, 1999.
13. Vila-Sallent, J., Sole-Pareta, J., Jove, T. & J. Torres. (1996), "Potential Performance of Distributed Computing Systems over ATM Networks", *INFOCOM'97*.