# Configuring and Tuning a Distributed Computer System to Support Complex Molecular Simulation: Phase I Collecting Performance Metrics

Medina Sultanova
St. Cloud State University
St. Cloud, MN 56301
suma0801@stcloudstate.edu

Jake Soenneker
St. Cloud State University
St. Cloud, MN 56301
soja0704@stcloudstate.edu

Dennis Guster
St. Cloud State University
St. Cloud, MN 56301
dcguster@stcloudstate.edu

## Abstract

A more efficient method to collect data used in the analysis of molecular modeling parallel simulation was needed. The prior method required some manual analysis and the transferring of the TCPDUMP packet data from a specialized file to a package such as EXCEL that required a comma delimited file. This was a cumbersome and time consuming process so an automated operating level script file was written to stream line the process. To gain an understanding of the nature of the data and difficulty in analyzing it sample data from Guster, Sultanov, Nordby, & Slattery, 2007 is included as a model. That data includes the following fields: 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. It was apparent from the analysis to date that the script would in fact speed up the data collection phase of the research project. Further, the test runs of the script on the current cluster environment revealed that it is configured more effectively and with more powerful resources than the environment utilized in the 2007 study and therefore better results are expected once the actual simulation trials are actually run.

## Introduction

The sophistication of the algorithms available to conduct molecular simulation has made it possible to conduct numerous complex analyses. However, because of these complexities it is critical that the computing environment used be capable of producing results in a timely manner. This is especially true if that environment uses a loosely coupled architecture. Such configurations then are dependent on the network that binds them together. One method to address performance optimization on the network software level is to look at the overhead related to the data transfer logic (Riley, 1997). This approach has merit in studying the efficiency of the network connectivity in a distributed computer grid. There are several approaches that can be applied to optimizing performance on the network protocol level such as optimizing the buffers, minimizing management traffic and scheduling applications. In most cases the planning and implementation of such methods can be improved by studying historical network traffic data obtained on the system in question. Truong and Fahringer, 2003, 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.

Therefore, this paper will collect workload performance data from a live network in which parallel molecular modeling software will be used to generate the network workload. A software package such as MOLDY (Refson, 2000) will be 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* will be used to collect the performance data. The number of servers will be varied so that performance scaling can be evaluated. The following data will be 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 this manner multiple configurations can be tested and the most optimal utilized in solving the actual molecular modeling problem.

## Review of Metrics in Distributed Processing

Because distributed processing is dependent on a network infrastructure the performance of that infrastructure is critical to the effectiveness of any parallel simulation that tries to take advantage of a distributed architecture. While there are several middleware solutions that facilitate communication among the various computing nodes MPI (message passing interface) is one of the most popular. The literature recognizes that adequate performance on the network level is critical an offers guidelines such as the ns-3Project, 2011. The work of Subramoni, Lai, Sur and Panda, 2010, addresses this issue and delineates the importance of analyzing packet inter-arrival patterns when configuring/tuning a distributed cluster.

Predicting packet inter-arrival patterns and their effect on network performance has long been a challenging problem (Partridge, 1993). Computer systems represent huge and complex queuing problems. An analysis of packet inter-arrival rates reveals a failure of the packet distribution to mimic the theoretically expected Poisson distribution. There are a number of studies that confirm the actual inter-arrival distribution of packets is not exponential as would be expected in the classical model (Krzenski, 1998; Partridge, 1993; Vandolore, Babic & Jain, 1999; Guster, Robinson & Richardson 1999).

It should be noted that the data analyzed in these prior studies was taken from a general purpose network that served a wide variety of applications. The traffic was then made up of a large number of different protocols, each with their own workload dynamics. This variation within protocols has been studied by (Guster, Safonov and Sundheim, 2005) and they found the service used and its interaction with the protocol used tend to vary significantly. For example, while HTTP and SSH both utilize TCP on the transport layer, HTTP does a better job of maximizing payloads so it tends to generate fewer packets with larger payloads. In the case of this study the pattern of MPI traffic would be the primary concern as well as eliminating packet traffic from the other protocols unless absolutely needed.

The cost effectiveness and high availability of PC based LANs as a parallel processing environment has been recognized for many years (Vila-Sallent, Sole-Pareta, Jove, and Torres, 1997). At first there was a concern that the LAN environment had inadequate bandwidth to support inter-process communication (Fatoohi and Weeratunga, 1994). However, higher speed solutions such as infinti band have lessened the concern of providing adequate bandwidth (Subramoni, Lai, Sur and Panda, 2010). A number of studies have shown that if PC based clusters are interconnected via a high speed LAN they can provide an effective parallel processing environment (Luckenbach, Ruppelt and Schulz, 1994; Fahringer and Prodan, 2002; Mengjou, Hsiehn, Du, Thomas and MacDonald, 1994). Although the hardware available to support workstation based LANs continues to improve many feel that relying on hardware alone may not ensure adequate performance for the demanding applications of the future (Popescu, 1994; Courson, Mink, Marcais, and Traverse, 2000). This scenario is still true in later work as depicted by Takizawa, Endo and Matsuoka, 2008, who devised an inter-node communication algorithm, optimizes a network by appropriate scheduling of nodes according to application communication patterns. This logic is specifically adapted to utilizing relatively high EPS local switch bandwidth to forward messages to nodes with optical connections as a shortcut to maximize overall throughput.

Therefore, methods to address optimization on the network software level can still be effective. This logic is related back to the work of Riley, 1997 which suggested the need to evaluate the overhead in relation to the data transfer logic. This basic approach continues to be useful when studying the efficiency of the network connectivity in a distributed computer grid. There are certainly several approaches that can be applied when optimizing performance on the network protocol level such as optimizing the buffers, minimizing management traffic and scheduling applications. In most cases the planning and implementation of such methods can be improved by studying historical network traffic data obtained on the system in question. As stated earlier this was true in the work of Takizawa, Endo and Matsuoka, 2008 who included appropriate scheduling of nodes according to application communication patterns in their solution. Truong and Fahringer, 2003 are proponents of such methodology and suggest a continuing pattern of more research in the field that features the analysis of real experimental data. They further state that such research will require better planning in regard to how to capture, store and analyze such data. Further based on the work of Takizawa, Endo and Matsuoka, 2008 it is also important to link that work to the configuration of the physical network architecture utilized.

Therefore, to illustrate the characteristics of the data this paper reviews workload performance data from a live network in which parallel molecular modeling software was used to generate the network workload. The MOLDY program (Refson, 2000) was used to generate the MPI requests. This example was selected because the source code was readily available and the output dealt with three dimensional graphics an area that the authors are investigating to determine the possibility of using such parallel techniques in financial forecasting. In this program the number of server machines can be programmed and thereby the offered intensity can be varied. *Tcpdump* was be used to collect the performance data. Further, to speed up the analysis a LINUX script was devised to automate the data analysis. In future expected trials, the number of servers will be varied so that performance scaling can be evaluated. The following data are critical in evaluating the performance of the cluster environment:  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.

## CHARACTERISTICS OF THE PROBLEM SOLVED

To provide a sense of the applications that require this level of performance analysis a sample problem from Guster, Sultanov, Nordby, & Slattery, 2007 is described in brief. Specifically, this work used a computer simulation methodology to determine the characteristics of water using the first physical principals. The parallel algorithm used each processor to store a complete set of dynamic variable arrays for all molecules. The problem was selected to illustrate practical issues in parallel computing, and involves molecular dynamic simulations used to study the statistical mechanics of condensed phase matter composed of small molecules.

These simulations were carried out with water molecules using the 4-site model of Jorgensen. Only the oxygen site interacts via the Lennard-Jones potential, and the charge site is displaced 0.15 °A from the oxygen molecule. This simulation is particularly appropriate for testing the MPI metrics because the applied algorithm generates intensive communication among all parallel processors included in the system. It has proved a challenging problem and can generate excellent baseline data in regard to the performance of the MPI data across the network. The MOLDY code provides quite intensive interaction between computers during the simulations. It is therefore expected that investigation of this type of problem will add to the understanding of how different types of algorithms impact the performance of parallel applications. This intensity can be explained because the area in which the particles will collide is defined through a three dimensional matrix. To achieve parallelization that space is broken into N subparts based in the number of processors used. The inter-processor communication is required whenever a particle collision occurs and a particle moves from the space associated with one processor to the space area of another processor. Of course then, that movement needs to be relayed via the MPI protocol using TCP packets. The state of the movement among particles needs to be updated within the simulation every millisecond which results in a high intensity of packet exchange between/among processors.

## NETWORK CONFIGURATION

The basic configuration of network follows. Two to twelve Intel virtual hosts are proposed to run the experiment. Each host will be configured with two 3.2 GHz CPUs that utilize symmetric multi-processing as supported by the Linux operating system. Each unit will run its part of the MOLDY program as required. The master server that will run the primary MPI process, will house the *tcpdump* program to collect packet traffic. Each unit is connected via a virtual bridge at 1000Mbps. The Linux flavor that will be used is CentOS 6, compiling with *gcc*.

## DATA COLLECTION STRATEGY

The goal is to be prepared to collect and format data quickly under a number of "what if" scenarios. To illustrate that process data is being displayed below from Guster, Sultanov, Nordby, & Slattery, 2007. This data depicted in Table 1 provides both the CPU and network level metrics required to analyze the effectiveness of parallel molecular modeling simulations. The parallel test-bed was programmed using the MOLDY software. *Tcpdump* was run on the master server and therefore traffic for each unit was collected at a central location. This eliminates the need to link decentralized files together from each host at a later time and then sorting by time stamp. The time on each host was synchronized via a time server to ensure continuity. This improvement eliminated a very cumbersome process and illustrated that the prime purpose of the script described in this paper was to automate this procedure. In the sample data below multiple trials were run, but just the one in which the defined the number of steps to reach convergence was at 10,000 is reported herein.

| | 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 | 201361 | 191377 | 470706 | 469598 | 602578 | 603034 |
| Inter-arrival time | .00122 | .00104 | .000483 | .000562 | .000430 | .000439 |
| Throughput | 575668 | 707134 | 1265710 | 1090568 | 1396049 | 1337841 |
| Packet Intensity | 817.64 | 960.42 | 2069.78 | 1771.63 | 2327.71 | 2231.22 |

Table 1: Timings and Means for Packet Arrival, Throughput, and Intensity
10,000 Iterations

## ANALYSIS

The table above provides some interesting results in regard to the wait time. In other words there are some large disparities between the CPU and the elapsed times. Further, while the CPU times scale pretty well to the 10 host level the elapsed times actually start increasing at the 6 host level which can be attributed to the network overhead if one looks at the corresponding increases in network intensity. More specifically, 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 from 2 units to 12 units is only about three times. 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. The output below illustrates how easy it will be to extract the same information from a TCP packet capture file. All that is needed is the host to analyze the round trip times, the number of times to average those round trip tests, the file name, and whether or not the inter-arrival times are desired.

```
######################
## DUMP STATISTICS ##
######################

Name of the TCP packet capture file to analyze:
output
Calculate Inter-arrival? (Expensive!) [y/n]
```

```
y
Host to analyze Round Trip Time (RTT):
localhost
Number of times to analyze RTT:
10
Please wait...

 100%
Pinging...

Number of packets: 1954
Data exchanged: 1291692 (bytes)
Time difference: 16.113371 (seconds)
Packets per second: 121.26574879955286823595
Average Inter-arrival: .00824635158648925281 (seconds)
Average packet length: 661.05015353121801432958 (bytes)
Average window size: 1140.29119754350051177072 (bytes)
Average Round Trip Time: .00004700000000000000 (seconds) to localhost
Average Max Bandwidth [Throughput]: 194092118.73080859774820765957 (bits/sec)
Average Transmission Rate: 641301.93489618032129962128 (bits/sec)
Traffic Intensity: .12499999999999999999
```

In looking at this sample run there are a number of encouraging statistics in regard to supporting an effective parallel computing environment for molecular modeling simulation. This test was undertaken in a virtual environment whereby the resources are multiplexed across multiple zones. Even so the performance characteristics in regard to though put compare favorably with the dedicated host cluster used in the 2007 study. As the current test-bed is tuned one can certainly expect superior performance from the new virtualized environment.

## DISCUSSION

In terms of providing an advantage in solving the problems in less time the 2007 data set fails when more than four units are utilized. While the reduction in the CPU time is encouraging as more host were added, the increase in network traffic offsets this advantage. Therefore, the computing environment needed to be enhanced for the algorithm to be effective given the massive amount of inter-processor communication utilized.

To provide the desired improvement several factors were considered besides redesigning the distribution algorithm. First, it was important to secure a network speedup to at least 1Gbs. It was expected that the added speed would also help reduce the loss in packet payload efficiency as hosts are added. In both data sets packet average is about 650 bytes. If this could be increased higher transmission efficiency could be achieved. This moderate size 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 which is based on UDP which is connectionless. (Guster, Al-Hammah, Safonov and Bachman, 2003) found that PVM could greatly reduce the communications overhead when compared to MPI. Also, perhaps multi-core hosts 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 in 2007 data 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. Last, the proposed script goes a long way to improving the data collection process and will especially be useful in simulations that require many trail in which the number of processors and iterations are varied.

## REFERENCES

Courson, M., Mink, A., Marcais, M. and Traverse, B., *An Automated Benchmarking Toolset*, In HPCN Europe, p497-506, 2000.

Fahringer, T. and Prodan, R., *ZENTURIO: An Experiment Management System for Cluster and Grid Computing*, 4th Intl. APART Workshop, Euro-Par, 2002.

Fatoohi, R. and Weeratunga, S., *Performance Evaluation of Three Distributed Computing Environments for Scientific Applications*, In Proceedings of Supercomputing '94, p400-409, 1994.

Guster, D., Al-Hamamah, A., Safonov, P. and Bachman, E., *Computing and Network Performance of A Distributed Parallel Processing Environment Using the MPI and PVM Communication Methods*, The Journal of Computing Sciences in Colleges, Vol. XVIII (4), p248-253, 2003.

Guster, D., Robinson, D. and Richardson, M., *The Application of the Power Law Process in Modeling the Inter-Arrival Times of Packets in a Computer Network*, Proceedings of the Midwest Decision Sciences Institute, Springfield, IL, April 22-24, 1999.

Guster, D., Safonov, P. and Sundheim, R., *Analysis of End-user Services and Their Potential Load on the Network*, Journal of the Academy of Business and Economics. Vol. V (3), 2005.

Guster, D. C., Sultanov, R., Nordby, M., & Slattery, T. *Managing a Computer Cluster: Tradeoffs and Scalability*. Presentation at the International Academy of Business and Economics, Las Vegas, NV. October, 2007.

Krzenski, K. *The Effect of Varying the Packet Interarrival Distribution in the Simulation of Ethernet Computer Networks*, Unpublished graduate research project, St. Cloud State University, 1998.

Luckenbach, T., Ruppelt, R. and Schulz, F., *Performance Experiments within Local ATM Networks*, GMD-FOKUS, Berlin, Germany, 1994.

Mengjou, L., Hsiehn J., Du, H., C., Thomas, J., P. and MacDonald, J., *Distributed Network Computing over Local ATM Networks*, Computer Science Department University of Minnesota, Computer Science Department Technical Report, p94-17, 1994.

Ns-3Project. http://www.nsnam.org/docs/models/html/distributed.html, 2011.

Partridge, C. *The End of Simple Traffic Models*, Editor's Note, IEEE Network, Vol. VII (5), 1993.

Popescu, A. *A Parallel Approach to Integrated Multi-Gbit/s Communication over Multiwavelength Optical Networks*, Ph.D. dissertation, Royal Institute of Technology, Stockholm, 1994.

Refson, K., *Moldy: a portable molecular dynamics simulation program for serial and parallel computers*, Computer Physics Communications, Vol. CXXVI (310), 2000.

Riley, G. et al., *Performance Improvement Through Overhead Analysis: a case study in molecular dynamics,* Proc. 11th ACM Intl. Conf. Supercomputing, p36-43, 1997.

Subramoni, H., Lai, P., Sur, S. and Panda, D. *Improving Application Performance and Predictability using Multiple Virtual Lanes in Modern Multi-Core InfiniBand Clusters*, 39th International Conference, San Diego, CA, Sept 2010.

Takizawa, S. Endo, T. and Matsuoka, S. *Locality aware MPI communication on a commodity opto-electronic hybrid network*. Proceeding of the IEEE International Symposium on Parallel and Distributed Processing, April 2008.

Truong, H. and Fahringer, T., *On Utilizing Experiment Data Repository for Performance Analysis of Parallel Applications.* In 9th International Europar Conference (EuroPar 03), LNCS, Klagenfurt, Austria, Springer-Verlag, 2003.

Vandolore, B., Babic, G. and Jain, R., *Analysis and Modeling of Traffic in Modern Data Communications Networks*. A paper submitted to the Applied Telecommunication Symposium, 1999.

Vila-Sallent, J., Sole-Pareta, J., Jove, T. and Torres, J., *Potential Performance of Distributed Computing Systems over ATM Networks*, INFOCOM '97, 1996.