HIGH PERFORMANCE SCALABLE COMPUTING PERFORMANCE MODELING USING PTOLEMY

Eric K. Pauer
Sanders, a Lockheed Martin Company
Signal Processing Center
Nashua, NH 03061-0868
(603) 885-8358, Fax (603) 885-0631
pauer@sanders.com

ABSTRACT

Many of today's signal processing applications are becoming more and more computationally intensive, requiring an increasing number of processors to satisfy their needs. Along with the larger number of processors, the bandwidth needed to pass data among the processors has also increased dramatically. In many classes of applications, designing an architecture to satisfy the data passing requirements while maintaining cost and complexity at reasonable levels is very challenging. An answer to these classes of problems is the High Performance Scalable Computing (HPSC) architecture. It consists of clusters of processing nodes interconnected over a high performance network, implemented using the MYRINET[1] protocol. Application algorithms are partitioned and mapped onto the various processing nodes in the architecture for distributed execution. The goal of this simulation work is to provide a performance modeling capability to automate the evaluation of the performance of various architectural candidates with respect to network design choices. This capability allows the development team to quickly assess the impact of various design decisions. The results of the performance simulation help in tailoring an architecture and identifying a network topology and routing scheme that best satisfies the application's system requirements.

KEYWORDS

signal processing, distributed processing, performance modeling, discrete-event simulation, Ptolemy, MYRINET

1. HPSC AND MYRINET

The purpose of the HPSC architecture is to enable the implementation of computationally intensive algorithms with high data bandwidth requirements on a distributed multiprocessor system. The philosophy behind HPSC is to decouple the application software executing on the processing nodes from the system software which passes data and messages among the nodes. Data is passed between the nodes over the MYRINET network [2], as depicted in a conceptual diagram of the HPSC architecture (figure 1). A processing node may contain programmable logic, like an FPGA or ASIC, but often contains one or more programmable processors such as a DSP or RISC processor.

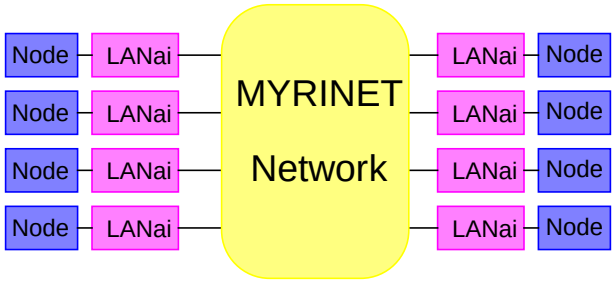

Figure 1: HPSC Architecture and the MYRINET Network

The HPSC architecture specifies how communication occurs between nodes, but does not specify what happens within a node. A programmable device, called a LANai, provides a common interface between all of the processors with a node and the network, via a device called a LANai.

The LANai is programmed to coordinate data transfers, between the node and all other nodes, over the network using the MYRINET protocol. The MYRINET network consists of a topology of multi-port MYRINET switches, interconnected together, with a LANai as an interface at each node. The LANai sends data to other nodes in the system using routing defined in a data synchronization table (DST) residing in the LANai memory. The DST contains a list of entries, each one specifying information about a block of data to be sent. This information includes the start address of the data within the node, the size of the data block, a list of port numbers for routing the data, and an index at the destination node.

When given the command to transmit, the LANai assembles data packets using the information in the DST, and begins sending the packets one after the other. Each data packet is given a header, which starts with the list of port numbers followed by the size and index. The LANai transmits the header and then body of the data packet containing the data block. The first MYRINET switch receives the data, and examines the first port number in the header. It uses this number to determine which of its ports will send the data. This port number is stripped from the packet, and the switch sends the rest of the packet out the specified port, assuming that port is not currently busy. This continues through a series of switches until the data packet finally arrives at the LANai of the destination node, at which time all port numbers in the header have been

stripped and used. On the receiving side, the LANai also has a receive DST to specify the address and size of each data block that it receives. The index sent along with each data packet tells the receiving LANai which entry in this DST to use. Once all data packets are received, the LANai then transfers the data to the appropriate memory locations. MYRINET is a high performance interconnect capability, with a network data rate of 160 Megabytes/ second.When a port, that is currently busy transmitting, receives a second data packet also needing to use this same port, contention occurs. The switch responds by queuing the second request, and sending a Stop control packet back through the switched network to the originating LANai. This LANai and a port on each of the intervening switches, used to route the packet up to the switch where the contention occurred, are now idle but remain blocked. Once the port becomes free, the switch sends a Go control packet to the originating LANai and the transmission of the packet continues. Multiple requests for the same port are queued and served in the order received. Control packets are small and are not blocked like data packets, as they can be interleaved, if necessary, over busy or blocked ports. Because of the chain reaction effect of port contention in a switch, these conditions should be minimized as they will degrade the effective bandwidth of the network.

2. GOAL OF PERFORMANCE MODELING

One benefit of a performance modeling capability is that it provides a means of evaluating candidate network topologies without having to build them. As the number of nodes and switches grow, it becomes more and more difficult to estimate or predict performance. Questions like how many switches, what type of switches, how they should be interconnected, and what the DST should look like become more difficult to answer, and an automated means of evaluating options becomes necessary. A second benefit is that the DST tables developed in the simulation can be used directly to program the LANai's to implement the desired connectivity when the target hardware is available. The type of problems that are targeted by this type of simulation are applications that are synchronous in nature and which have a data flow that is predictable and does not depend on the content of the data. Typical applications of HPSC include synthetic aperture radar (SAR), infrared search and track (IRST) systems, and space-time adaptive processing (STAP) based problems. Performance modeling is useful when the customer, due to cost or environmental constraints (like power consumption, weight, or size), cannot afford the hardware to provide a large margin in bandwidth and processing capability. Thus, performance modeling can be used to predict whether system requirements will be met, and it is very useful in optimizing the size of the architecture required to provide sufficient capability without excess hardware.

3. PTOLEMY

Ptolemy [3] is a software environment developed at the University of California at Berkeley that supports heterogeneous system simulation and design using several different models of computation, each implemented in a separate domain. The class of application problems addressed by HPSC falls into Ptolemy's synchronous dataflow (SDF) domain, where the flow of data is predictable and does not change. Ptolemy's Discrete Event (DE) domain is employed the engine for this performance modeling capability [4]. The DE domain is a discrete-event simulator, which uses a model of computation in which tokens with time stamps, called particles, representing events are passed among the simulation building blocks, called stars. Each star has one or more input and/or output ports that are used to pass the particles to other stars. In this event-driven model of computation, a chronologically sorted list of events is maintained, oldest first. The simulator's schedule examines the oldest event in the list, and in general, executes the star where that particle resides. Ptolemy has been developed in C++ using an object-oriented software architecture to facilitate modularity and extensibility. All of the source code for Ptolemy is freely available, which further facilitates adding extensions to the tool. These attributes of Ptolemy enabled the development of the extensions to support the HPSC performance modeling needs.

4. EXTENDING PTOLEMY'S DE DOMAIN

Extensions to the DE domain, in the form of new stars and particles, were created to support our specific MYRINET performance modeling needs. Several new stars and particles were developed to model the HPSC architecture and MYRINET protocol. This effort leveraged off of similar performance modeling work started here at Sanders under the RASSP program [5], in addition to the work done by the Ptolemy project.

The key components in the HPSC architecture, implemented as stars, include the data sources (*SourceNode*), processing nodes (*Node*), LANai interfaces (*LANai*), and MYRINET switches (*4/8/16-port Switches*). The stars can be considered virtual prototypes of these components, as they implement behavioral models at the appropriate level of abstraction. Each type of star has a group of settable states which allow the behavior of the model to be adjusted or fine-tuned, as appropriate. The stars closely model the actual behavior of the components of the HPSC architecture and MYRINET protocol. However, approximations to simplify the behaviors were made when the impact of the simplification was minimal and justified by a significant savings of simulation time and complexity.

4.1. NEW STARS

The SourceNode star creates blocks of data at a periodic rate, and can represent a sensor or a source of data from another subsystem, which is outside the scope of the simulation. The size of the data blocks and their frequency are settable using state variables. Typically, the SourceNode represents a separate node in the architecture, and hence is connected to its own LANai.

The Node star is used to represent a processing node in the HPSC architecture. It models the computational load on the node at a fairly high level of abstraction, treating the processing taking place on the node as a single measurable task. Its state variables include input and output data block sizes, a clock rate, and a number of clock cycles needed to complete the processing on that node. Using hierarchical modeling, a more detailed behavior model on the nodes, is possible. Just as with the SourceNode star, the Node star connects to a LANai star.

The LANai and Switch stars are responsible for modeling the behavior of the MYRINET network. The LANai has states such as: clock rate, various latencies, local/node data rate, network data rate, packet header sizes, and transmit and receive DST information. The DST information includes the input packet sizes, output packet sizes, port numbers used for the routing the packets, and destination packet indices. The LANai has independent transmit and receive sides, and therefore can simultaneously transmit and receive data from the MYRINET network. A state model was developed to represent the behavior of the transmit side of the LANai, and consists of the following states: free, transmitting, and blocked. The receive side of the LANai does not use a state model, as potentially conflicting events are properly handled by the Switch star to which it is connected.

The Switch star captures and simulates the rest of the behavior of the MYRINET protocol. MYRINET switches come in a variety of sizes, most commonly four, eight, or sixteen ports. As with the LANai, each port in the switch has an independent transmit and receive ports, with the behavior of the transmit side being much more complex. Each transmit port within a Switch star obeys a state model consisting of the following states: free, transmitting, blocked, and waiting. The transmit port model handles potentially conflicting requests, like simultaneous or overlapping data packets. The state model for the switch transmit port greatly simplifies the modeling of switch receive ports as well as the LANai receive side, as previously mentioned.

A switch, implementing data transfer via the MYRINET protocol, transmits an incoming data packet after a short latency, after decoding the header and long before the entire packet is received. This behavior means that a single data packet will typically be in the process of being received and retransmitted by several switches and LANai's simultaneously. The modeling environment must relinquish execution in a star before the packet is completely processed in order to execute the other stars, in order to properly model the flow of data packets. In addition, the environment must also revisit the proper stars at the correct simulated future time to model the completion of the packet transmission, and to begin processing any requests that were queued in the interim. To accomplish this, the LANai and Switch stars each have an internal particle feedback queue and some counter variables. The time stamped particles placed in this feedback queue cause the stars to be revisited at future times, and the counters help in determining if the packet transmissions are completed or delayed.This approach works well with the state models to accurately simulate this distributed packet behavior.

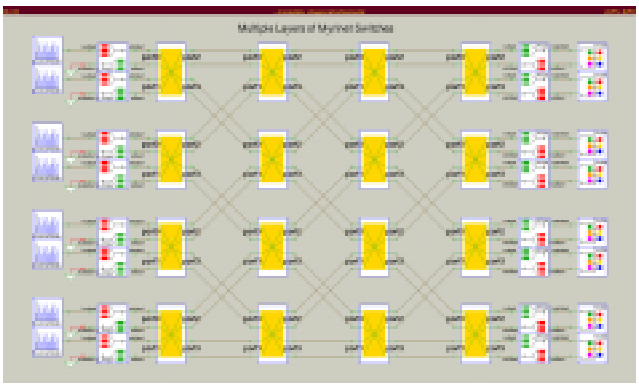

Figure 2: HPSC Architecture Example

4.2. NEW PARTICLES

As mentioned in section 4, several new particles were developed for this performance modeling capability. A generic Packet particle, derived from Ptolemy's general purpose Message particle class, was created. It provides a representation of the basic structure of a MYRINET packet and serves as a pure virtual base class for the DataPacket and ControlPacket particles classes. The DataPacket particle provides a representation of a typical MYRINET data packet, and allows a variable body size. The ControlPacket particle is currently used to represent both Stop and Go control packets. The new NodeDataBlock particle class, derived from the Message class, represents the passing of blocks of data between the LANai and the processing node. Lastly, an extension to the manner in which the Ptolemy kernel handles feedback particles was made to allow them to take on assigned integer values. This was especially useful in the Switch star where the particles could be given values corresponding to the port to which they were associated.

5. MYRINET MODELING EXAMPLES

An example HPSC architecture model is shown in figure 2. There are eight SourceNodes and eight Nodes, each paired with a LANai, connected by grid-like topology of sixteen 4-port switches. This application requires each SourceNode (on the left) to send a data packet to each Node (on the right), meaning each SourceNode will send eight data packets for a total of 64 packets across the MYRINET network per iteration. This example is representative of a typical subsystem in a distributed space-time adaptive application.

This model was simulated with the performance modeling extensions described in section 4. Results were written to a log file as the simulation progressed. A gantt tool was developed in order to more easily view and interpret the

Figure 3: HPSC Performance Modeling Results

results of the simulation. Figure 3 shows the display of the gantt tool for this simulation. The gantt tool displays the activity on each star in separate rows as a function of time (along the x-axis). A row is provided for each SourceNode generation of data, each LANai transmit activity, each LANai receive activity, the transmit activity for each port in every switch, the transmit queue for each port in each switch, and the processing on the Node. Rows are not displayed when there is no activity, and also, the displaying of rows may be disabled, as was done in this example for the SourceNodes. The various activities are color-coded to facilitate viewing, but have been shown in gray scale here. Yellow denotes a start up latency, blue indicates normal transmission or reception of data, and green indicates processing of data by the node. Problems are shown in orange and red: orange indicates that one or more data blocks have currently originated in the switch port and have caused queuing of requests; red is used where switch ports or LANai's are idle due to blockages that occurred somewhere in the current route path.

This particular simulation takes approximately twenty seconds to complete on a Sun SPARCstation 10. A simulation of this size requires more activity rows than can be shown at once, so a scroll capability allows the user to view the others. The switches have been conveniently named so that their names include their relative XY position within the topology of figure 2. In this example, a number of switches experience serious contention problems. Clearly, the performance simulation is useful for determining where such potential contention problems exist, and for examining potential solutions. Other architectural options to look at here include providing connectivity from the bottom row of switches to the top row so that packets have more routing possibilities, or considering the use of 8-port switches. In most cases, such as this one, the designer wants to make sure that the latency introduced by the network is short enough so that it does not adversely affect the system throughput requirements, without having a system with an over-abundance of hardware.

6. ROLE OF HIERARCHY IN PERFORMANCE MODELING WITH PTOLEMY

To help in managing the simulation of large systems, Ptolemy has a built-in hierarchical capability. Groups of connected stars can be captured into a single entity called a galaxy, and can be treated as a reusable component. This is useful in defining systems where there are regular patterns in the topology. The hierarchy is also useful for modeling the logical capability of physical boards that have already been developed and built, and then using these board-level models as fundamental building blocks in designing and developing a system solution for an application. As previously mentioned, hierarchy can be useful in refining the performance simulation, especially with respect to the processing nodes. More detailed models of the processing nodes in terms of processors, buses, and memories can be developed, captured, and integrated into this same modeling capability. Thus, this hierarchical modeling capability allows different levels of abstraction, to be used where appropriate.

7. CONCLUSION

A performance modeling capability to model HPSC architectures and MYRINET using Ptolemy has been developed. These extensions take the form of new stars and particles which implement the behavior of the MYRINET protocol in the Ptolemy Discrete Event domain. By using these extensions, the designer may explore many implementation alternatives in terms of network topology and routing. A gantt tool has been developed to facilitate the viewing and interpretation of the performance results. The hierarchical capabilities of Ptolemy may be utilized to aid in building larger system models or in refining the behavior of the processing nodes. In addition, this performance modeling approach with Ptolemy is quite extensible to other types of architectures, network protocols and strategies, and can support a variety of system modeling needs. In summary, this capability enhances the ability of designers to explore many options in order to find the HPSC architecture that best satisfies their system requirements.

REFERENCES

1. MYRINET is a trade name of Myricom, Inc.
2. Myricom, Inc., "Myrinet, A Brief, Technical Overview," Arcadia, California, 1995.
3. E. A. Lee, et al., "An Overview of the Ptolemy Project," Department of EECS, University of California, Berkeley, 1994.
4. E. A. Lee, et al., *The Almagest*, Volumes 1-4 (Berkeley, California: Regents of the University of California, 1996).
5. E. K. Pauer and J. B. Prime, "An Architectural Trade Capability Using the Ptolemy Kernel," Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 1996.