Accurate Analytical Models for Message Passing on Multi-core Clusters

Bibo Tu[1], Jianping Fan[2], Jianfeng Zhan[1], Xiaofang Zhao[1]

[1]*National Research Center of Intelligent Computing Systems*
Institute of Computing Technology
Chinese Academy of Sciences
Beijing 100190, China
{tbb, jfzhan}@ncic.ac.cn, zhaoxf@ict.ac.cn

[2]*Research Center of High Performance Computing*
Shenzhen Institute of Advanced Technology
Chinese Academy of Sciences
Shenzhen 518067, China
fan@ict.ac.cn

***Abstract*—Memory hierarchy on multi-core clusters has two-fold characteristics: vertical memory hierarchy and horizontal memory hierarchy. Vertical memory hierarchy has been modeled by previous work (e.g. memory logP, $\log_n P$, $\log_3 P$ etc.) to analyze middleware's effects on point-to-point communication with different message sizes and message strides; Horizontal memory hierarchy has become more prominent due to distinct performance among three levels of communication in a multi-core cluster: intra-CMP, inter-CMP and inter-node, which should adequately be considered. Derived from $\log_n P$ and $\log_3 P$ models, new analytical models $\mathrm{mlog}_n P$ and its reduction $2\log_{\{2,3\}} P$ are proposed to unitedly abstract memory hierarchy on multi-core clusters in vertical and horizontal levels. The results of performance evaluation show that it is indispensable to incorporate horizontal memory hierarchy into new models suitable for multi-core clusters, and $2\log_{\{2,3\}} P$ model can predict communication costs for message passing on multi-core clusters more accurately than $\log_3 P$ model.**

Keywords-analytical model; memory hierarchy; CMP; multi-core clusters; message passing

I. INTRODUCTION

As a matter of fact, multi-core clusters have been the most popular platforms in parallel computing. In the new Top500 supercomputer list published in November 2007, about 87.6% processors are multi-core processors and about 81.2% supercomputers have used cluster architecture [1]. Intuitively, multi-core processors can speedup application performance by dividing the workload to different cores. However, compared with SMP or NUMA clusters, applications on multi-core clusters have not gotten optimal performance and scalability. Accordingly, it is crucial to have an in-depth understanding on characteristics of multi-core clusters and their effects on application behaviors.

Initially, applications are likely to treat multi-core processors, also called Chip Multiprocessor (CMP), simply as conventional symmetric multiprocessors (SMPs). However, chip multiprocessors feature several interesting architectural attributes differently from SMPs. One such architectural feature is the design of multi-level cache hierarchies. The two broad strategies deployed in current multi-core processors exist in processors from Intel and AMD. Intel processors provide a shared L2 cache whereas AMD multi-core processors deploy HyperTransport links for quick data transfers. Also, these architectures employ efficient cache coherency protocols. However, irrespective of these different hierarchies, both the systems enable very fast sharing of data across the cores. Further, to scale the bandwidth available to the coherency traffic, the cores are either connected via multiple buses as in Intel or by 2-D mesh HyperTransport links. Apart from providing good data movement capabilities across the processing cores, the caching hierarchies are useful in reducing the pressure on the memory bandwidth [2].

Consequently, above multi-core specific characteristics complicate memory hierarchies on multi-core clusters. Understanding their effects will potentially benefit the performance optimization of not only application algorithms, but also message passing. For many scientific applications, the cost of message passing (e.g. MPI communication middleware) greatly affects overall execution time. Since MPI has still emerged as the primary programming paradigm for writing efficient parallel applications before the emergence of new programming paradigms suitable for multi-core/many-core platforms in the future, it is of practical significance to study on performance analysis and optimization of MPI communication middleware on multi-core clusters at present. This paper presents two-fold characteristics of memory hierarchy on multi-core clusters in detail: vertical memory hierarchy and horizontal memory hierarchy. Vertical memory hierarchy has been modeled by previous work (e.g. memory logP, $\log_n P$, $\log_3 P$ etc. [3, 4, 5]); Horizontal memory hierarchy has become more prominent due to distinct performance among three levels of communication in a multi-core cluster: intra-CMP, inter-CMP and inter-node, which should adequately be considered. Therefore, this paper proposes new analytical models $\mathrm{mlog}_n P$ and its reduction $2\log_{\{2,3\}} P$ to unitedly abstract memory hierarchy in vertical and horizontal levels so as to accurately predict and evaluate the performance of message passing on multi-core clusters, which will benefit our ongoing work on the multi-core aware optimization of MPI communication middleware.

The rest of the paper is organized as follows: In Section II we introduce two-fold characteristics on memory hierarchy on multi-core clusters: vertical memory hierarchy and horizontal memory hierarchy. In Section III we describe new analytical models suitable for multi-core clusters. Performance evaluation is given in Section IV. Related work is discussed in Section V. Finally we conclude and point out future work directions in Section VI.

1066-6192/09 $25.00 © 2009 IEEE
DOI 10.1109/PDP.2009.18

II. MEMORY HIERARCHY ON MULTI-CORE CLUSTERS

The ever increasing speed gap between CPUs and memory systems and current multi-core technology complicate memory hierarchy. Memory hierarchy on multi-core clusters takes on two-fold characteristics in the vertical and horizontal levels as shown in Figure 1 as an example of Intel Dual-Core Xeon.

A. Vertical Memory Hierarchy

For message passing in a cluster system, distributed point-to-point communication requires moving data from the source process' local memory to the target process' local memory. Sends and receives are explicit communications accomplished using implicit communication mechanisms provided in middleware and operating system. Communication cost consists of the sum of memory and network communication times. Memory communication is the transmission of data to/from user buffer, kernel buffer, from/to the local network buffer. Network communication is data movement between source and target network buffers. For example, an MPI_Send() of a strided message describes a point-to-point transfer explicitly. To ensure that packed data is actually sent across the network, MPI middleware performs a series of implicit communications to complete the transfer (i.e., packing strided data at the source and unpacking data by stride at the target). Some transmissions occur in user space, others via the operating system in kernel space [4]. The message path of above point-to-point communication shows memory hierarchy in a vertical level as shown in Figure 1.

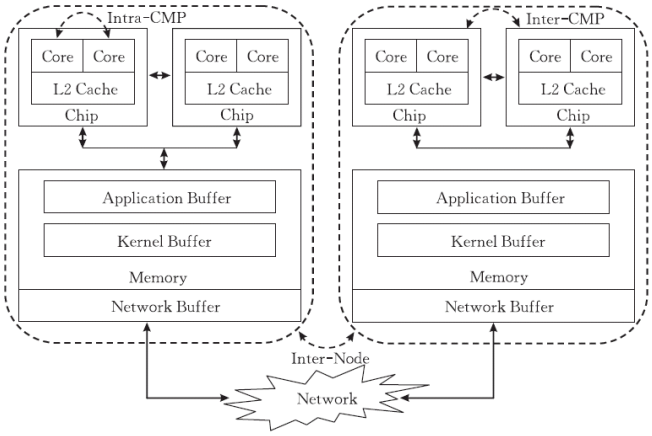

Fig.1. Memory hierarchy on multi-core clusters

Improvements in memory speed will continue to lag behind improvements in processor and network interconnect technologies. Given current technological trends, memory communication cost (e.g. MPI middleware cost) will increase in proportion to overall communication cost. So it is indispensable to incorporate middleware costs into analytical models for message passing.

Log_nP model and its reduction log_3P model can accurately analyze communication performance with memory hierarchy in the vertical level. Log_nP model is formalized as follows using five parameters [4, 5]:

l: the effective latency, defined as the length of time the processor is engaged in the transmission or reception of a strided or non-contiguous message over and above the cost of a non-strided or contiguous transfer. The system-dependent *l* cost is a function of the message data size (*s*) under a variable stride or distribution (*d*). We denote this function as *f(s,d)=l*, where variable *s* corresponds to a series of discrete message sizes in bytes, variable *d* corresponds to a series of discrete distributions or stride distances in bytes between array elements, and function *f* is the additional time for transmission in microseconds over and above the non-strided or contiguous cost for variable message data size *s* and distribution or stride *d*.

o: the overhead, defined as the length of time the processor is engaged in the transmission or reception of a non-strided or contiguous message. The system-dependent *o* cost is a function of the message data size (*s*) under a fixed unit stride or distribution (*d=1* array element). We denote this function as *f(s,d)=f(s,1)=o*, where variable *s* corresponds to a series of discrete message sizes in bytes, variable *d=1* array element corresponds to the unit stride or distribution distance in bytes between adjacent array elements, and function *f* is the time for transmission in microseconds for variable message data size *s* and distribution or stride *d=1* element. This average, unavoidable overhead typically represents the best case for data transfer on a target system. This cost is bounded below by the data size divided by the hardware bandwidth.

g: the gap, defined as the minimum time interval between consecutive message receptions at a processor. Without resource contention, assuming this parameter has no impact on communication cost, effectively using *o = g*.

n: the number of implicit transfers along the data transfer path between two endpoints. Endpoints can be as simple as two distinct local memory arrays or as complex as a remote transfer between source and target memories across a network.

P: the number of processes or processors.

In a word, vertical memory hierarchy behaves as follows: parameter *n* denotes the number of implicit communication along the message path; the memory communication cost varies with message size (*s*) and message distribution (*d*, strides). Log_nP denotes the function *f(s,d)* to express the effective latency (*l*) and overhead (*o*) in message transmission, i.e. *l=f(s,d)* and *o=f(s,1)*. So log_nP estimates point-to-point communication cost as:

$$T = \sum_{i=0}^{n-1} \{o_i + l_i\} = \sum_{i=0}^{n-1} \{f(s,1)_i + f(s,d)_i\} \qquad (1)$$

For simplicity, log_3P model assumes n=3 points of implicit communication. Equation (1) reduces to:

$$T = \sum_{i=0}^{2} \{f(s,1)_i + f(s,d)_i\}$$
$$= \{f(s,1)_0 + f(s,d)_0\} + \{f(s,1)_1 + f(s,d)_1\} + \{f(s,1)_2 + f(s,d)_2\}$$

i.e. $T = \{o_0 + l_0\} + \{o_1 + l_1\} + \{o_2 + l_2\}$ (2)

Equation (2) describes three implicit communication points along the message path of distributed point-to-point communication as follows: 0) Middleware communication from user buffer to the network interface buffer; this includes the effects of hierarchical memory. 1) Communication across the network. 2) Middleware communication from the network interface buffer to user buffer; this includes the effects of hierarchical memory. Since packets are contiguous and fixed size for point $i=1$, l_1 is assumed to be zero. More precisely and simply, the costs at point $i=0$ and $i=2$ are regarded as equal. Assuming:

middleware overhead=o_{mw}=$\{o_0+o_2\}$,
middleware latency =l_{mw}=$\{l_0+l_2\}$,
network overhead=o_{net}= o_1.

Thus, the $\log_3$P model can be expressed semantically as:

$$T = \{o_0 + o_2\} + \{l_0 + l_2\} + \{o_1 + l_1\} = o_{mw} + l_{mw} + o_{net} \quad (3)$$

In a word, $\log_n$P and $\log_3$P models disclose the intrinsic characteristic of distributed point-to-point communication: software-parameterized memory communication (middleware costs) and hardware-parameterized network communication, i.e. memory hierarchy in the vertical level. So they are very suitable for conventional cluster computing platforms, including SMP and NUMA clusters.

B. Horizontal Memory Hierarchy

Apart from vertical memory hierarchy, multi-core clusters take on prominent characteristic of horizontal memory hierarchy. Horizontal memory hierarchy is mainly brought by distinct performance among three levels of communication in a multi-core cluster: intra-CMP, inter-CMP and inter-node communication as shown in Figure. 1. The communication between two processors on the same chip is referred to as intra-CMP communication in this paper. The communication across chips but within a node is referred to as inter-CMP communication. Intra-CMP and inter-CMP communications in the same node are also referred to as intra-node communication. And the communication between two processors on different nodes is referred to as inter-node communication.

Due to multi-level cache hierarchies, CMPs offer unique capabilities that are fundamentally different from SMPs [6]:

- The inter-core communication performance (bandwidth and latency) on a CMP can be many times better than is typical for a SMP.
- Cache hierarchy organization (Intel processors provide a shared L2 cache whereas AMD multi-cores deploy HyperTransport links for quick data transfers) on multi-core processors makes inter-core memory access far faster than is typical for a SMP, only with shared main memory.

So it does, horizontal memory hierarchy on multi-core clusters becomes more prominent and brings the following features differently from conventional SMP clusters, which can also be seen from recent studies [7, 8]:

Prominent intra-node communication performance: For multi-core clusters, the performance (bandwidth and latency) of intra-node communication is many times better than inter-node, and intra-CMP has the best performance because data can be shared through L2 cache as the example of Intel Quad-core Xeon. However, for SMP clusters, the performance of intra-node communication only with shared main memory can't absolutely excel that of inter-node communication, because of memory copy cost of intra-node communication and some zero-copy and high performance schemes of inter-node communication, e.g. InfiniBand and RDMA-enabled interconnects[9].

Equal chance between intra-node and inter-node communications: For SMP clusters, the most of message distribution is distributed point-to-point communication (inter-node communication). However, on average about 50% messages in a multi-core cluster are transferred through the intra-node communication [7]. With the ever-increased number of computing cores in a CMP, especially for future many-core processors, the chance of intra-node communication will be more dominant.

III. New Models Suitable for Multi-core Clusters

Vertical memory hierarchy on multi-core clusters implies overlooked memory communication costs of point-to-point message passing varied with different message sizes and message strides, while horizontal memory hierarchy implies distinct performance of point-to-point message passing among different communication levels on multi-core clusters. This paper proposes new models $m\log_n$P and its reduction $2\log_{\{2,3\}}$P to abstract memory hierarchy in vertical and horizontal levels to accurately analyze communication performance on multi-core clusters, derived from $\log_n$P and $\log_3$P models only focusing on vertical memory hierarchy.

A. $mlog_nP$ Model

Differently from $\log_n$P model, $m\log_n$P model adds a new parameter 'm' to define different communication levels, while parameter 'n' denotes the number of implicit transfers along the message path in different communication levels. $m\log_n$P model is formalized as follows using six parameters:

m: the number of different communication levels. Typically for three communication levels: intra-CMP, inter-CMP and inter-node. Non-uniform memory hierarchy and large-scale cascading network may bring more communication levels.

l: the effective latency, defined as the length of time the processor is engaged in the transmission or reception of a strided or non-contiguous message over and above the cost of a non-strided or contiguous transfer.

o: the overhead, defined as the length of time the processor is engaged in the transmission or reception of a non-strided or contiguous message.

g: the gap, defined as the minimum time interval between consecutive message receptions at a processor. Without resource contention, assuming this parameter has no impact on communication cost, effectively using $o = g$.

n: the number of implicit transfers along the message path in different communication levels.

P: the number of processes or processors.

B. $2log_{\{2,3\}}P$ Model

For simplicity and practical use, $2\log_{\{2,3\}}P$ model is proposed to simplify $m\log_nP$ model. In $2\log_{\{2,3\}}P$ model, we assume two communication levels ($m=2$): intra-node and inter-node. We ignore the inter-CMP communication, because of awkward performance and poor scalability of CMP interconnection. The intra-node communication has $n=2$ points of implicit communication without network transfer, while inter-node communication has $n=3$ points of implicit communication. Because of distinct memory communication costs (middleware costs) in different communication levels, the point-to-point communication cost in $2\log_{\{2,3\}}P$ model will be estimated over again.

For $m=2$ and $n=2$ or 3, Equation (3) is reformulated as

$$T=\begin{cases} o_{mw}+l_{mw}(o_{net}=0,n=2) \\ \text{for intra-node communication} \\ o'_{mw}+l'_{mw}+o'_{net}(n=3) \\ \text{for inter-node communication} \end{cases} \quad (4)$$

We use the function $f(s,d)$ to rewrite Equation (4) as

$$T=\begin{cases} 2f(s,1)_0+2f(s,d)_0 \\ \text{for intra-node communication} \\ 2f'(s,1)_0+2f'(s,d)_0+f'(s,1)_1 \\ \text{for inter-node communication} \end{cases} \quad (5)$$

For the same message size and stride, the middleware communication costs ($f(s,1)_0$ and $f'(s,1)_0$, $f(s,d)_0$ and $f'(s,d)_0$) between intra-node and inter-node communications are distinct. Differently from $\log_3P$ model expressed by Equation (3), $2\log_{\{2,3\}}P$ model expressed by Equation (4) and (5) discloses not only vertical memory hierarchy, but also horizontal memory hierarchy on multi-core clusters.

IV. Performance Evaluation

Since the importance of memory communication as to vertical memory hierarchy in point-to-point communication has been verified in previous work [4, 5], our objective will verify that the effects of horizontal memory hierarchy on point-to-point communication on multi-core clusters should adequately be considered. Furthermore, we will verify $2\log_{\{2,3\}}P$ model is more suitable for multi-core clusters than $\log_3P$ model.

Our evaluation system consists of eight nodes connected by Gigabit Ethernet. Each node is equipped with two sets of quad-core 2.0GHz Intel Xeon processor, i.e. eight computing cores per node. Four cores on the same chip share 8M L2 cache. The operating system is Linux 2.6, MPI version is mpich2-1.0.6p1. Without loss of generality, we only select strided message to measure performance, because strided message must be packed into a contiguous buffer (like non-strided message), and then sent across memory or network to its destination. In the following tests, message size is classified as 1Kbyte, 4Kbyte and 16Kbyte and message stride is classified as 8byte, 64byte and 512byte. Messages greater than 16Kbyte are not considered, because handshakes or acknowledgements are required for long messages. A strided message is expressed as "*K*S", as shown in the x-axis in the following figures. For example, "4K8S" means that message size is 4Kbyte and message stride is 8byte.

A. Middleware costs of Point-to-Point Communication

We use blocking MPI_Send and MPI_Recv calls to get roundtrip times (RTT) so as to measure memory communication costs, which is suggested by related papers [4, 10, 11]. Firstly, we use PRTT method [11] to get the network overhead (o'_{net}) of inter-node communication. Subsequently, we begin by obtaining the roundtrip times of contiguous and non-contiguous message transfers for intra-node communication and inter-node communication. According to Equation (4) and (5), there are the following equations to get individual middleware and network costs:

$RTT(s)=2o_{mw}$ (contiguous transfers for intra-node point-to-point communication) and

$RTT(s,d)=2(o_{mw}+l_{mw})$ (non-contiguous transfers for intra-node point-to-point communication);

$RTT'(s)=2(o'_{mw}+o'_{net})$ (contiguous transfers for inter-node point-to-point communication) and

$RTT'(s,d)=2(o'_{mw}+l'_{mw}+o'_{net})$ (non-contiguous transfers for inter-node point-to-point communication).

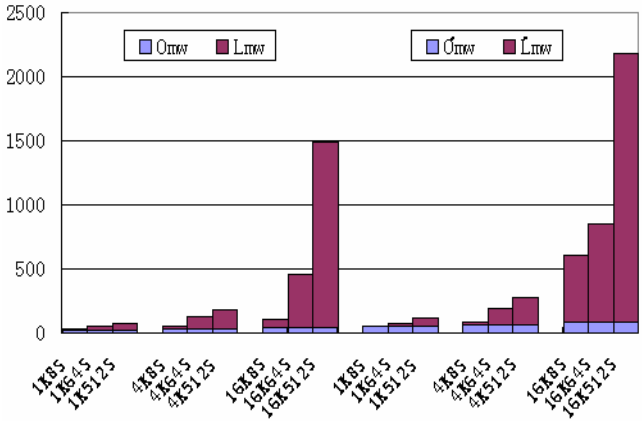

Fig. 2. Middleware costs of point-to-point communication on a multi-core cluster. The x-axis is message size and stride and y-axis is time in microseconds

Figure 2 shows middleware costs of point-to-point communication on our evaluation system. The left portion is the intra-node communication; the right one is the inter-node communication. From the figure, we can see middleware overhead (o_{mw} or o'_{mw}) and middleware latency (l_{mw} or l'_{mw}) increase with message sizes and message strides, and middleware latency dominates memory communication costs with increased message sizes and message strides. For the same message, middleware overhead of intra-node communication is on average 49.2% less than that of inter-node communication; The difference of middleware

latencies between intra-node and inter-node communication is small when message size and stride are small, but the largest difference of middleware latencies is approaching 87.9% with increased message sizes and strides; The whole middleware cost (middleware overhead plus latency) of intra-node communication is on average 41% less than that of inter-node communication. Thus it can be seen that the difference of middleware costs between intra-node and inter-node communications on the multi-core cluster is prominent, and it is indispensable to incorporate horizontal memory hierarchy into analytical models.

B. Cost prediction of Collective Communication

In this section, we use $2\log_{\{2,3\}}P$ model to analyze common collective communication algorithms, linear broadcast algorithm and binomial tree broadcast algorithm, compared with $\log_3 P$ model.

The linear broadcast algorithm is based on point-to-point communication and used to broadcast the message from the first process in a group to all other processes, in which *(P-1)* individual consecutive MPI_Sends are used at the source/root process to transfer data to each remaining process, where P is the number of processes. In all our experiments, we use sched_affinity system call to ensure the binding of process with processor.

Using $\log_3 P$, the cost of the linear broadcast algorithm is $P*(o'_{mw}/2+l'_{mw}/2)+o'_{net}$, where $P*(o'_{mw}/2+l'_{mw}/2)$ is the middleware overhead and middleware latency occurring at the source process for sending data to other *(P-1)* processes, and o'_{net} is the network overhead.

Using $2\log_{\{2,3\}}P$ model, the cost of the linear broadcast algorithm is divided into intra-node and inter-node, expressed as $(P/n-1)*(o_{mw}/2+l_{mw}/2) +(P-(P/n-1)-1)*(o'_{mw}/2+l'_{mw}/2)+ o'_{net}$, *n is the number of nodes,* assuming all intra-node communication are started at first.

Figure 3 shows costs predicted by $\log_3 P$ and $2\log_{\{2,3\}}P$ models and measured cost for the linear broadcast algorithm, respectively. The size of the linear broadcast algorithm in Figure 3 is 64 processes (8 processes per node).

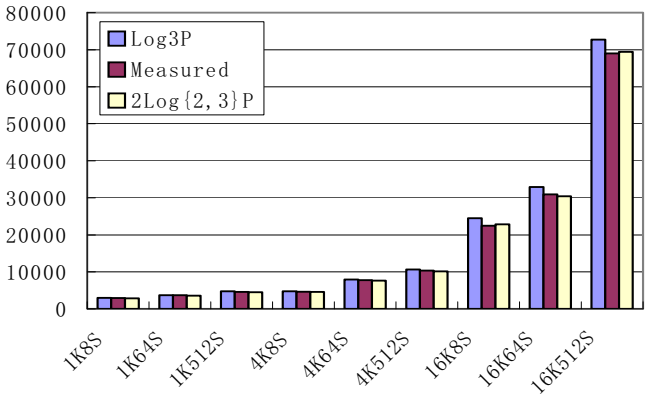

Fig. 3. Cost prediction of the linear broadcast (8*8=64 processes). The x-axis is message size and stride and y-axis is time in microseconds.

The binomial tree broadcast algorithm is commonly used in MPICH. In the first step, the root sends data to process (root + P/2). This process and the root then act as new roots within their own subtrees and recursively continue this algorithm (P denotes the number of processes). This communication takes a total of $\log_2 P$ steps (the height of the tree).

Therefore, the times taken by the binomial tree broadcast algorithm are $(o'_{mw}+l'_{mw}+o'_{net})*\log_2 P$ (Using $\log_3 P$) and $(o_{mw}+l_{mw})*n+ (o'_{mw}+l'_{mw}+o'_{net})*(\log_2 P-n)$ (Using $2\log_{\{2,3\}}P$, n denotes the number of intra-node transfers along the longest distance to a leaf node in $\log_2 P$ steps and its value is determined by the location of the root).

Figure 4 shows costs predicted by $\log_3 P$ and $2\log_{\{2,3\}}P$ models and measured cost for the binomial tree broadcast algorithm, respectively. The size of the binomial tree broadcast algorithm in Figure 4 is 64 processes (8 processes per node).

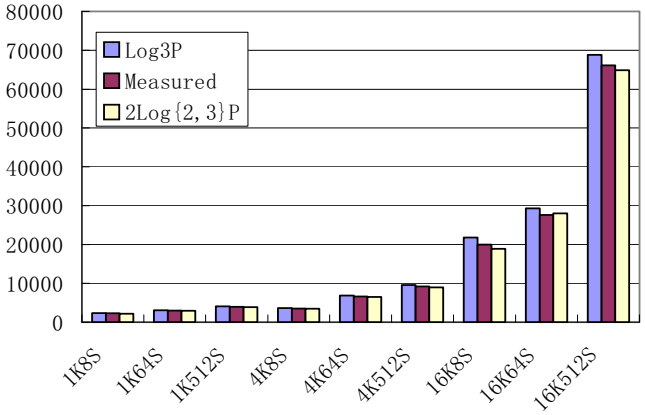

Fig. 4. Cost prediction of the binomial tree broadcast (8*8=64 processes). The x-axis is message size and stride and y-axis is time in microseconds.

From Figure 3 we can see that $2\log_{\{2,3\}}P$ prediction is more accurate than $\log_3 P$ for all measured message sizes and message strides. For the data points measured, maximum relative error of $\log_3 P$ is 15.32% and average relative error is 5.64%; while the maximum relative error of $2\log_{\{2,3\}}P$ is 3.61% and average relative error is 1.59%.

From Figure 4 we can see that for small message sizes and small strides, $\log_3 P$ prediction is accurate, but for large message sizes and larger strides, $2\log_{\{2,3\}}P$ prediction is more accurate than $\log_3 P$. For the data points measured, maximum relative error of $\log_3 P$ is 10.53% and average relative error is 4.68%; while the maximum relative error of $2\log_{\{2,3\}}P$ is 4.36% and average relative error is 2.03%.

Otherwise, from Figure 3 and 4 we can see that $\log_3 P$ prediction consistently over-estimates the communication time, because it overlooks high-efficient intra-node communication. With the increased message sizes and message strides, the error of $\log_3 P$ prediction increases; while $2\log_{\{2,3\}}P$ prediction does not follow this trend. In a word, the results in Figure 3 and 4 show that $2\log_{\{2,3\}}P$

model is more suitable for communication evaluation on multi-core clusters than $\log_3$P model.

V. Related Work

In recent years, many models have been proposed to predict and evaluate distributed communication performance. Previous work modeling either interconnect performance or memory hierarchy performance may be classified as hardware-parameterized models and software-parameterized modes.

Hardware-parameterized models ignore the increasing effects of middleware on communication cost. LogP model [12] approximate memory communication in parallel systems with a fixed overhead parameter (o), the reciprocal of the bandwidth between application and network buffers. LogGP [13] is an extension of the LogP model that additionally allows for large messages by introducing the gap per byte parameter, G and ignores the effects of data distribution. There are also other varieties of LogP model, such as PLogP [10, 14], LoPC [15], LoGPC [16], LoGPS [17] and so on, they have more or less ignored important effects of memory communication.

With the ever increasing speed gap between the CPU and memory systems, incorporating memory hierarchy into computational models has become unavoidable. The Hierarchical Memory Model (HMM) [18] applies the characteristics of memory hierarchies to network communication, but ignores the network attributes common to parallel and distributed systems. Memory logP model [3] uses parameters to describe the cost effects of message size and distribution when message passes through memory. Inspired by the memory logP model, $\log_n$P model and its reduction $\log_3$P model [4, 5] combine hierarchical memory performance with estimates of network communication cost, and accurately predict the performance of distributed point-to-point communication.

With the emergency of multi-core processors, memory hierarchy on multi-core clusters becomes more complicated. Apart from memory hierarchy in the vertical level, memory hierarchy in the horizontal level becomes more prominent. Above software-parameterized models, such as memory logP, $\log_n$P and $\log_3$P, only focus on vertical memory hierarchy. $\mathrm{mlog_n P}$ and $2\log_{\{2,3\}}$P models provide a general abstract for vertical and horizontal memory hierarchies so as to accurately predict the performance of message passing on multi-core clusters.

VI. Conclusions and Future Work

Differently from conventional SMP clusters, horizontal memory hierarchy on multi-core clusters behaves more prominent, interweaved with vertical memory hierarchy. Vertical memory hierarchy implies overlooked memory communication costs of point-to-point message passing varied with different message sizes and message strides, whereas horizontal memory hierarchy implies distinct performance of point-to-point message passing among different communication levels on multi-core clusters. Derived from $\log_n$P and $\log_3$P models, new analytical models $\mathrm{mlog_n P}$ and $2\log_{\{2,3\}}$P unitedly abstract vertical and horizontal memory hierarchies for message passing on multi-core clusters. Some results of performance evaluation show that it is indispensable to incorporate horizontal memory hierarchy into new analytical models, and $2\log_{\{2,3\}}$P model can predict communication costs for message passing on multi-core clusters more accurately than $\log_3$P model.

Our models are general and applicable to current multi-core clusters. Though model analysis and evaluation in this paper take the example of Intel system with shared L2 cache, the same is suitable for AMD multi-core platforms, because of the same characteristics of memory hierarchy. Moreover, it is our current work to use $\mathrm{mlog_n P}$ and $2\log_{\{2,3\}}$P models to analyze and find optimized solutions for multi-core aware MPI collective operations, which is the key to obtaining good performance speed-ups for many parallel applications since MPI programming has still been the mainstream in parallel computing before the emergence of new programming paradigms suitable for multi-core platforms in the future. Otherwise, multi-core processors also complicate hierarchical parallelization, so our work may require extension (e.g. OpenMP, multi-threaded programming model [19]) in the future.

Acknowledgement

This work is supported by the National High Technology Research and Development Program of China (Grant No. 2006AA01A102).

References

[1] TOP500 Team, TOP500 Report for November 2007, http://www.top500.org.

[2] Amith R. Mamidala, Rahul Kumar, Debraj De, D. K. Panda, MPI Collectives on Modern Multicore Clusters: Performance Optimizations and Communication Characteristics, 8th IEEE International Conference on Cluster Computing and the Grid (CCGRID '08), 2008.

[3] K. W. Cameron and X.-H. Sun, "Quantifying Locality Effect in Data Access Delay: Memory logP," in proceedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS 2003), Nice, France, 2003.

[4] Kirk W. Cameron, Rong Ge, Predicting and Evaluating Distributed Communication Performance, Proceedings of the 2004 ACM/IEEE Supercomputing Conference, 2004.

[5] Kirk W. Cameron, Rong Ge, Xian-He Sun. $\log_n$P and $\log_3$P: Accurate Analytical Models of Point-to-Point Communication in Distributed Systems. IEEE TRANSACTIONS ON COMPUTERS, MARCH 2007: VOL. 56, NO. 3: 314-327.

[6] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro etc. The Landscape of Parallel Computing Research: A View from Berkeley. Electrical Engineering and Computer Sciences, University of California at Berkeley. Technical Report No: UCB/EECS-2006-183, 2006, 12.

[7] Lei Chai, Qi Gao, Dhabaleswar K. Panda. Understanding the Impact of Multi-Core Architecture in Cluster Computing: A Case Study with Intel Dual-Core System. Seventh IEEE International

Symposium on Cluster Computing and the Grid (CCGrid'07), pp. 471-478, 2007.

[8] Sadaf R. Alam, Richard F. Barrett, Jeffery A. Kuehn, Philip C. Roth, Jeffrey S. Vetter, Characterization of Scientific Workloads on Systems with Multi-Core Processors, In International Symposium on Workload Characterization, 2006.

[9] J. Liu, J. Wu, and D. K. Panda. High Performance RDMA-Based MPI Implementation over InfiniBand. Int'l Journal of Parallel Programming, 2004.

[10] Thilo Kielmann, Henri E. Bal. Fast Measurement of LogP Parameters for Message Passing Platforms. Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing, 2000: 1176-1183.

[11] Torsten Hoefler, Andre Lichei, and Wolfgang Rehm, Low-Overhead LogGP Parameter Assessment for Modern Interconnection Networks, in proceedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS'2007).

[12] D.E. Culler, R. Karp, D.A. Patterson, A. Sahay, E. Santos, K. Schauser, R. Subramonian, and T. von Eicken, "LogP: A Practical Model of Parallel Computation," Comm. ACM, vol. 39, pp. 78-85, 1996.

[13] A. Alexandrov, M. F. Ionescu, K. Schauser, and C. Scheiman, "LogGP: Incorporating Long Messages into the LogP model," in proceedings of Seventh Annual Symposium on Parallel Algorithms and Architecture, pp. 95-105, Santa Barbara, CA, 1995.

[14] Jelena Pjesivac-Grbovic, Thara Angskun, George Bosilca, Graham E. Fagg, Edgar Gabriel, Jack J. Dongarra, Performance Analysis of MPI Collective Operations, Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), 2005

[15] M. I. Frank, A. Agarwal, and M. K. Vernon, "LoPC: Modeling Contention in Parallel Algorithms," in proceedings of Sixth Symposium on Principles and Practice of Parallel Programming, pp. 276-87, Las Vegas, NV, 1997.

[16] C. A. Moritz and M. I. Frank, "LoGPC: Modeling Network Contention in Message-Passing Programs," in proceedings of SIGMETRICS'98, pp. 254-63, Madison, WI, 1998.

[17] F. Ino, N. Fujimoto, and K. Hagihara, "LogGPS: A Parallel Computational Model for Synchronization Analysis," in proceedings of PPoPP'01, pp. 133-42, Snowbird, Utah, 2001.

[18] B. Alpern, L. Carter, E. Feig, and T. Selker, "The Uniform Memory Hierarchy Model of Computation," Algorithmica, 12 (2/3), pp. 72-109, 1994.

[19] Matthew Curtis-Maury, Xiaoning Ding, Christos D. Antonopoulos and Dimitrios S. Nikolopoulos, An Evaluation of OpenMP on Current and Emerging Multithreaded/Multicore Processors. In IWOMP, 2005.