© 2014 IJIRT | Volume 1 Issue 4 | ISSN : 2349-6002

UNDERSTANDING THE IMPACT OF MULTI-CORE ARCHITECTURE IN CLUSTER COMPUTING: A CASE STUDY WITH INTEL DUAL-CORE SYSTEM

Sweety Sen, Sonali Samanta
B.Tech, Information Technology,
Dronacharya College of Engineering, Gurgaon , India

***Abstract*- Multi-core processor is a growing industry trend as single core processors rapidly reach the physical limits of possible complexity and speed. In the new Top500 supercomputer list, more than 20% processors belong to multi-core processor family. However, without an in-depth study on application behaviors and trends on multi-core cluster, we might not be able to understand the characteristics of multicore cluster in a comprehensive manner and hence not be able to get optimal performance. In this paper, we take on the challenges and design a set of experiments to study the impact of multi-core architecture on cluster computing. We choose to use one of the most advanced multi-core servers, Intel Bensley system with Woodcrest processors, as our evaluation platform, and use popular benchmarks including HPL, NAMD, and NAS as the applications to study. From our message distribution experiments, we find that on an average about 50% messages are transferred through intra-node communication, which is much higher than intuition. This trend indicates that optimizing intra-node communication is as important as optimizing inter-node communication in a multi-core cluster. We also observe that cache and memory contention may be a potential bottleneck in multi-core cluster, and communication middleware and applications should be multi-core aware to alleviate this problem. We demonstrate that multi-core aware algorithm, e.g. data tiling, improves benchmark execution time by up to 70%. We also compare the scalability of multicore cluster with that of single-core cluster and find that the scalability of multi-core cluster is promising.**

I. INTRODUCTION

The pace people pursuing computing power has never slowed down. Moore's Law has been proven to be true over the passage of time - the performance of microchips has been increasing at an exponential rate, doubling every two years.In 1978, a commercial fight between New York and Paris cost around $900 and took seven hours. If the principles of Moore's Law had been applied to the airline industry the way they have to the semiconductor industry since 1978, that fight would now cost about a penny and take less than one second.(a statement from Intel) However,it becomes more difficult to speedup processors nowadays by increasing frequency. One major barrier is the overheat problem, which high-frequency CPU must deal with carefully. The other issue is power consumption. These concerns make it less cost-to-performance effective to increase processor clock rate. Therefore, computer architects have designed multi-core processor, which means to place two or more processing cores on the same chip Multi-coreprocessors speedup application performance by dividing the workload to different cores. It is also referred to as Chip Multiprocessor (CMP).

On the other hand, cluster has been one of the most popular models in parallel computing for decades. The emergence of multi-core architecture will bring clusters into a multi-core era. As a matter of fact, multi-core processors have already been widely deployed in parallel computing. In the new Top500 supercomputer list published in November 2006, more than 20% processors are multi-core processors from Intel and AMD . In order to get optimal performance, it is crucial to have in-depth understanding on application behaviors and trends on multi-core cluster. It is also very important to identify potential bottleneck in multi-core cluster through evaluation, and explore possible solutions. However, since multi-core is a relatively new technology, few research has been done in the literature. In this paper, we take on the challenges and design a set of experiments to study the impact of multi-core architecture on cluster computing. The purpose is to give bothapplication and communication middleware developers insights on how to improve overall performance on multi-core clusters. We aim to answer the following questions:

- What are the application communication characteristics in multi-core cluster?
- What are the potential bottlenecksin multi-core cluster and how to possibly avoid them?
- Can multi-core cluster scale well?

We choose to use one of the most advanced servers, Intel Bensley system with dual-core

Woodcrest processor, as a case study platform. The benchmarks used include HPL, NAMD, and NAS parallel benchmarks.From our message distribution experiments, we find that on an average about 50% of messages are transferred through intranode communication, which is much higher than intuition. This trend indicates that optimizing intra-node communication is as important as optimizing inter-node communication in a multi-core cluster. An interesting observation from our bottleneck identification experiments is that cache and memory contention may be a potential bottleneck in multi-core cluster, and communication middleware and applications should be written in a multi-core aware manner to alleviate this problem. We demonstrate that data tiling, a data locality optimization technique improves benchmark execution time by up to 70%. We also compare the scalability of multi-core cluster with that of single-core cluster and find that the scalability of multi-core cluster is promising The rest of the paper is organized as follows: In Section 2 we introduce the background knowledge of multi-core architecture. In Section 3 we describe the methodology of our evaluation. Setup of the evaluation system is described in Section 4 and the evaluation results and analysis are presented in Section 5. Related work is discussed in Section 6. And finally we conclude and point out future work directions in Section.

II. Multi-Core Cluster

Multi-core means to integrate two or more complete computational cores within a single chip . The motivation of the development of multi-core processors is the fact that scaling up processor speed results in dramatic rise in power consumption and heat generation. In addition, it becomes more difcult to increase processor speed nowadays that even a little increase in performance will be costly. Realizing these factors, computer architects have proposed multi-core processors that speed up application performance by dividing the workload among multiple processing cores instead of using one super fastsingle processor. Multi-core processor is also referred to as Chip Multiprocessor (CMP). Since a processing core can be viewed as an independent processor, in this paper we use processor and core interchangeably.

Most processor venders have multi-core products, e.g. Intel Quad- and Dual-Core Xeon, AMD Quad- and DualCore Opteron, Sun Microsystems UltraSPARC T1 (8cores), IBM Cell, etc. There are various alternatives in designing cache hierarchy organization and memory access model. Figure 1 illustrates two typical multi-core system designs. The left box shows a NUMA [1] based dual-core system in which each core has its own L2 cache. Two cores on the same chip share the memory controller and local memory. Processors can also access remote memory, although local memory access is much faster. The right box shows a bus based dual-core system, in which two cores on the same chip share the same L2 cache and memory controller, and all the cores access the main memory through a shared bus.

Due to its greater computing power and cost-to performance effectiveness, multi-core processor has been deployed in cluster computing. In a multi-core cluster, there are three levels of 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. And the communication between two processors on different nodes is referred to as inter-node communication.

Multi-core cluster imposes new challenges in software design, both on middleware level and application level. How to design multi-core aware parallel programs and communication middleware to get optimal performance is a hot topic.

III. Design Of Experiments For Evaluating Multicore Clusters

In this section we describe the evaluation methodology and explain the design and rational of each experiment.

A. Programming Model and Benchmarks

We choose to use MPI [4] as the programming model because it is the de facto standard used in cluster computing. The MPI library used is MVAPICH2 [5], which is a high performance MPI-2 implementation over InfiniBand [2]. In MVAPICH2, intra-node communication, including both intra-CMP and inter-CMP, is achieved by user level memory copy.

We evaluate both microbenchmarksand application level benchmarks to get a comprehensive understanding on the system. Microbenchmarks include latency and bandwidth tests. And application level benchmarks include HPL from HPCC benchmark suite [16], NAMD [21] apoa1 data set, and NAS parallel benchmarks [12].

B. Design of Experiments

We have designed to carry out four sets of experiments for our study: latency and bandwidth, message distribution, potential bottleneck identification, and scalability tests. We describe them in detail below.

- Latency and Bandwidth: These are standard ping-pong latency and bandwidth tests to characterize the three levels of communication in multi-core cluster: intraCMP, inter-CMP, and inter-node communication.
- Message Distribution: We define message distribution as a two dimensional metric. One dimension is with respect to the communication channel, i.e. the percentage of traffic going through intra-CMP, inter-CMP, and inter-node respectively. The other dimension is in terms of message size. This experiment is very important because understanding message distribution facilitates communication middleware developers, e.g. MPI implementors, to optimize critical communication channels and message size range for applications. The message distribution is measured in terms of both number of messages and data volume.
- Potential Bottleneck Identification: In this experiment, we run application level benchmarks on different configurations, e.g. four processes on the same node, four processes on two different nodes, and four processes on four different nodes. We want to discover the potential bottlenecksin multi-core cluster if any, and explore approaches to alleviate or eliminate the bottlenecks. This will give insights to application writers how to optimize algorithms and/or data distribution for multicore cluster. We also design an example to demonstrate the effect of multi-core aware algorithm.
- Scalability Tests: This set of experiments is carried out to study the scalability of multi-core cluster.

C. Processor Affinity

In all our experiments, we use sched affinity system call to ensure the binding of process with processor. The effect of processor affinity is two-fold. First, it eases our analysis,
because we know exactly the mapping of processes with processors. And second, it makes application performance more stable, because process migration requires cache invalidation and may degrade performance.

IV. Evaluation Platforms

Our evaluation system consists of 4 Intel Bensley systems connected by InfiniBand. Each node is equipped with two sets of dual-core 2.6GHz Woodcrest processor, i.e. 4 processors per node. Two processors on the same chip share a 4MB L2 cache. The overall architecture is similar to that shown in the right box in Figure 1. However, Bensley system has added more dedicated memory bandwidth per processor by doubling up on memory buses, with one bus dedicated to each of Bensley's two CPU chips. The InfiniBand HCA is Mellanox MT25208 DDR and the operating system is Linux 2.6.

To compare scalability, we also used a single-core Intel cluster connected by InniBand. Each node is equipped with dual Intel Xeon 3.6GHz processor and each processor has a 2MB L2 cache.

V. Evaluation Results

In this section we present the experimental results and analyze them in depth. We use the format pxq to represent a configuration. Here p is the number of nodes, and q is the number of processors per node.

A. Latency and Bandwidth

Figure 2 shows the basic latency and bandwidth of the three levels of communication in a multi-core cluster. The numbers are taken at the MPI level. The small message latency is 0.42us, 0.89us, and 2.83us for intra-CMP, interCMP, and inter-node communication respectively. The corresponding peak bandwidth is 6684MB/s, 1258MB/s, and 1532MB/s.

From Figure 2 we can see that intra-CMP performance is far better than inter-CMP and inter-node performance, especially for small and medium messages. This is because
in Intel Bensley system two cores on the same chip share the same L2 cache. Therefore, the communication just involves two cache operations if the communication buffers
are in the cache. From the figure we can also see that for large messages, inter-CMP performance is not as good as inter-node performance, although memory performance is
supposed to be better than network performance. This is because the intra-node communication is achieved through a shared buffer, where two memory copies are involved. On the other hand, the inter-node communication uses the Remote Direct Memory Access (RDMA) operation provided by InfiniBand and rendezvous protocol [20], which forms a
zero-copy and high performance scheme. This also explains why for large messages (when the buffers are out of cache) intra-CMP and inter-node perform comparably.

This set of results indicate that to optimize MPI intranode communication performance, one way is to have better L2 cache utilization to keep communication buffers in the L2 cache as much as possible, and the other way is to reduce the number of memory copies. We have proposed a preliminary enhanced MPI intra-node communication design in our previous work.

B. Message Distribution

As mentioned in Section 3.2, this set of experiments is designed to get more insights with respect to the usage pattern of the communication channels, as well as the message size distribution. Figures 3 and 4 show the profiing results for NAMD and HPL respectively. The results for NAS benchmarks are listed in Table 1. The experiments are carried out on a 4x4 configuration and the numbers are the average of all the processes.

Figures 3 and 4 are interpreted as the following. Suppose there are n messages transferred during the application run, in which m messages are in the range(a,b]. Also suppose in these m messages, m1 are transferred through intra-CMP, m2 through inter-CMP, and m3 through inter-node. Then:

- Bar Intra-CMP(a, b] = m1/m
- Bar Inter-CMP(a, b] = m2/m
- Bar Inter-node(a, b] = m3/m
- Point Overall(a, b] = m/n

From Figure 3 we have observed that most of the messages in NAMD are of size 4KB to 64KB. Messages in this range take more than 90% of the total number of messages

and byte volume. Optimizing medium message communication is important to NAMD performance. In the 4KB to 64KB message range, about 10% messages are transferred

through intra-CMP, 30% are transferred through inter-CMP, and 60% are transferred through inter-node. This is interesting and kind of surprising. Intuitively, in a cluster environment intra-node communication is much less than internode communication, because a process has much more inter-node peers than intra-node peers. E.g. in our testbed, a process has 1 intra-CMP peer, 2 inter-CMP peers, and 15 inter-node peers. If a process has the same chance to communicate with every other process, then theoretically:

- Intra-CMP = 6.7%
- Inter-CMP = 13.3%
- Inter-node = 80%

If we call this distribution even distribution, then we see that intra-node communication in NAMD is well above that in even distribution, for almost all the message sizes. Optimizing intra-node communication is as important as optimizing inter-node communication to NAMD.

From Figure 4 we observe that most messages are small messages in HPL, from 256 bytes to 4KB. However, with respect to data volume messages larger than 256KB take

more percentage. We also find that almost all the messages are transferred through intra-node in our experiment. However, this is a special case. In HPL, a process only talks to processes on the same row or column with itself. In our 4x4 configuration, a process and its row or column peers are always mapped to the same node, therefore, almost all the communication take place within a node. We have also conducted the same experiment on a 16x4 configuration for HPL. The results show that 15% messages are transferred through intra-CMP, 42% through inter-CMP, and 43% through inter-node. Although the trend is not as

extreme as in the 4x4 case, we can still see that intra-node communication in HPL is well above that in even distribution.

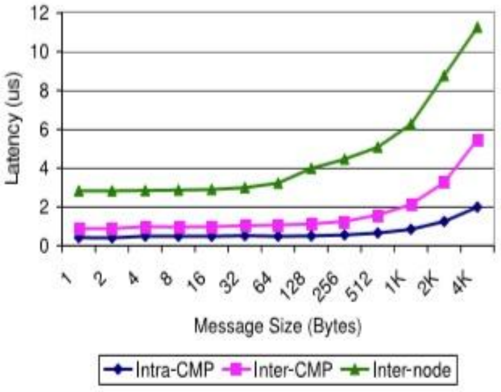
Table 1 presents the total message distribution in NAS benchmarks, in terms of communication channel. Again, we see that the amount of intra-node (intra-CMP and interCMP) communication is much larger than that in even distribution for most benchmarks. On an average, about 50% messages going through intra-node communication. This

trend is not random. It is because most applications havecertain communication patterns, e.g. row or column based communication, ring based communication, etc. which increase the intra-node communication chance. Therefore, even in a large multi-core cluster, optimizing intra-node communication is critical to the overall application performance.

C. Potential Cache and Memory Contention

In this experiment, we run all the benchmarks on 1x4,2x2, and 4x1 configurations respectively, to examine the potential bottleneck in the system. As mentioned in the beginning of Section 5, we use the format pxq to represent a configuration, in which p is the number of nodes, and q is the number of processors per node. The results are shown

in Figure 5. The execution time is normalized to that on 4x1 configuration.



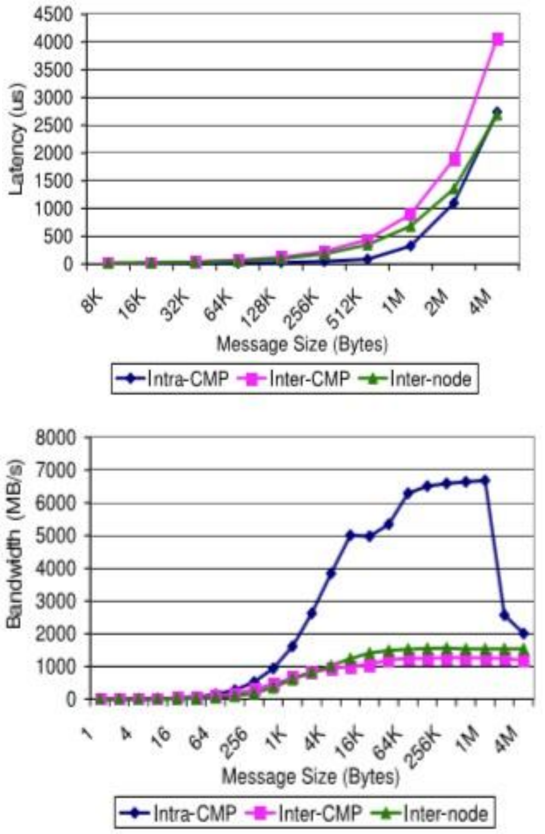


Figure 1. Latency and Bandwidth in Multi-core Cluster

D. Benefits of Data Tiling

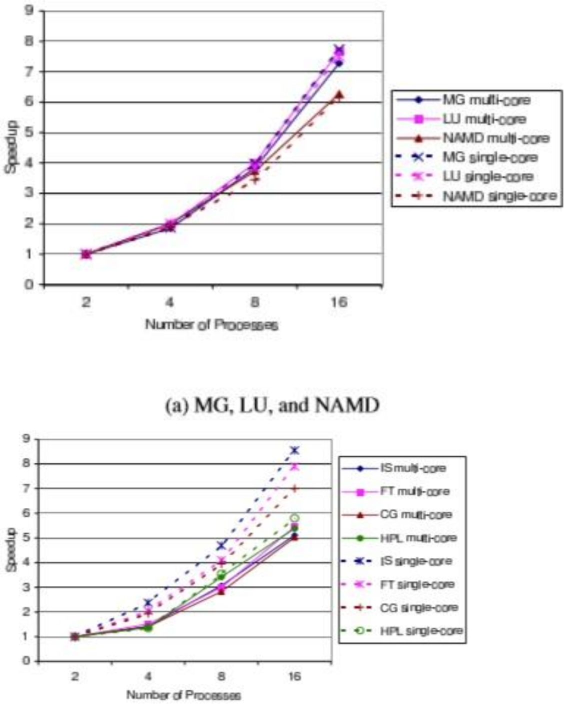
To study the benefits of data tiling on multi-core cluster, we design a micro benchmark, which does computation and communication in a ring-based manner. Each process has a piece of data (64MB) to be processed for a number of iterations. During execution, each process computes on its own data, sends them to its right neighbor and receives data from its left neighbor, and then starts another iteration of computation. In the original scheme, the data processed in the original chunk size (64MB) while in the data tiling scheme, the data are divided into smaller chunks in the size of 256KB, which can easily fit in L2 cache.

In the tiling case, since the intra-node communications using CPU-based memory copy, the data are actually preloaded into L2 cache during the communication. In addition, we observe that in the cases where 2 processes running on 2 cores on the same chip, since most communication happens in L2cache in data tiling case, the improvement is most significant, around 70% percent. The improvement in the case where 4 processes running on 4 cores on the same node, 8 processes running on 2 nodes, and 16 processes running on 4 nodes is 60%, 50%, and 50% respectively. The improvements are not as large as that in the 2 process case because the communication of inter-CMP and inter-node is not as efficient as the intra-CMP for 256KB message size.

E. Scalability

Scalability is always an important angle to look at when evaluating clusters. Although our test bed only contains 4 nodes, we want to do an initial study on multi-core cluster scalability. We also compare the scalability of multi- core cluster with that of single-core cluster. The results are shown in Figure 2. It is to be noted that the performance is normalized to that on2 processes, so 8 is the ideal speedup forthe16processcase.

It can be seen from Figure 2(a) that some applications show almost ideal speedup on multi-core cluster, e.g. LU and MG. Compared with single-core cluster scalability, we find that for applications that show cache or memory contention, such as IS, FT, and CG, the scalability on single-core cluster is better than that on multi-core cluster. For other applications such as MG, LU and NAMD, multi-core cluster shows the same scalability as single-core cluster. As an initial study we find that multi-core cluster is promising in scalability.



(a) MG, LU, and NAMD

(b) IS, FT, CG, and HPL

Figure 2. Application Scalability

VI. CONCLUSIONS

In this paper we have done a comprehensive performance evaluation, profiling, and analysis on multi-core cluster, using both micro benchmarks and application level benchmarks. We have several interesting observations from the experimental results that give insights to both application and communication middleware developers. From micro benchmark results, we see that there are three levels of communication in a multi-core cluster with different performances: intra-CMP, inter-CMP, and inter-node communication. Intra-CMP has the best performance because data can be shared through L2 cache. Large message performance of inter-CMP is not as good as inter-node because of memory copy cost. With respect to applications, the first observation is that counter-intuitively, much more intra-node communication takes place in applications than that in even distribution, which indicates that optimizing intra-node communication is as important as optimizing inter-node communication in a multi-core cluster. Another observation is that when all the cores are activated for execution, cache and memory contention may prevent the multi-core system from achieving best performance, because two cores on the same chip share the same L2 cache and memory controller. This indicates that communication middleware and applications should be written in a multi core aware manner to get optimal performance. We have demonstrated an example on application optimization technique which improves bench- mark performance by up to 70%. Compared with single- core cluster, multi-core cluster does not scale well for applications that show cache/memory contention. However, for other applications multi-core cluster has the same scalability as single-core cluster.

REFERENCES

[1] http://lse.sourceforge.net/numa/faq/.

[2] MPI over InfiniBand Project. http://nowlab.cse.ohio- state.edu/projects/mpi-iba/.

[3] D.H.Baileyetal. TheNASparallelbenchmarks. volume 5, pages 63–73, Fall1991.

[4] Innovative Computing Laboratory (ICL). HPC Challenge Benchmark. http://icl.cs.utk.edu/hpcc/.

[5] M. Koop, W. Huang, A. Vishnu, and D. K. Panda. Memory Scalability Evaluation of the Next-Generation Intel Bensley Platformwith InfiniBand. In Hot Interconnect, 2006.

[6] J. C. Phillips, G. Zheng, S. Kumar, and L. V. Kale. NAMD: Biomolecular Simulation on Thousands of Processors. In SuperComputing, 2002.