Power Consumption Optimization of MPI Programs on Multi-core Clusters

Yen-Jun Chen[1], Ching-Hsien Hsu[1], Kuan-Ching Li[2], Hsi-Ya Chang[3], and Shuen-Tai Wang[3]

[1] Department of Computer Science and Information Engineering
Chung Hua University, Hsinchu, Taiwan 300, R.O.C.
{Patrick,robert}@grid.chu.edu.tw
[2] Department of Computer Science and Information Engineering
Providence University, Taichung 43301, Taiwan
kuancli@pu.edu.tw
[3] National Center for High-Performance Computing
Hsinchu 30076, Taiwan
{jerry,stwang}@nchc.org.tw

Abstract. While the energy crisis and the environmental pollution become important global issues, the power consumption researching brings to computer sciences world. In this generation, high speed CPU structures include multi-core CPU have been provided to bring more computational cycles yet efficiently managing power the system needs. Cluster of SMPs and Multi-core CPUs are designed to bring more computational cycles in a sole computing platform, unavoidable extra energy consumption in loading jobs is incurred.

Data exchange among nodes is essential and needed during the execution of parallel applications in cluster environments. Popular networking technologies used are Fast Ethernet or Gigabit Ethernet, which are cheaper and much slower when compared to Infiniband or 10G Ethernet. Two questions on data exchange among nodes arise in multi-core CPU cluster environments. The former one is, if data are sent between two nodes, the network latency takes longer than system bus inside of a multi-core CPU, and thus, wait-for-sending data are blocked in cache. And the latter is, if a core keeps in waiting state, the unpredicted waiting time brings to cores higher load. These two situations consume extra power and no additional contribution for increasing overall speed. In this paper, we present a novel approach to tackle the congestion problem and taking into consideration energy in general network environments, by combining hardware power saving function, maintaining the transmission unchanged while saving more energy than any general and previous cases.

Keywords: Power Consumption, multi-core processor, cluster Computing, MPI.

1 Introduction

Reduction on power consumption of computer systems is a hot issue recently, since many CPUs and computer-related hardware has been produced and under operation

P. Mueller, J.-N. Cao, and C.-L. Wang (Eds.): Infoscale 2009, LNICST 18, pp. 108–120, 2009.

everywhere. As the number of single-core CPU has reached to physical limitation on current semi-conductor technology, the computing performance has met the bottleneck. Multi-core CPUs become a simple yet efficient solution to increase performance and speed since that concept SMP in a single chip, that is, making up a small cluster to be executed inside a host. Additionally, it reduces the amount of context switching while in single-core CPUs, increases straight forwardly the overall performance.

Figure 1 illustrates the architecture of Intel quad-core CPU, which looks like a combination of two dual-core CPUs. It has four individual execution engines, where each two cores share one set of L2 cache and system bus interface, and connect to the fixed system bus. The advantages of this architecture are twofold. The former one is that each core can fully utilize L2 cache as each core needs larger memory, while the latter is that each core accesses L2 cache through individual hub [7] simplifying system bus and cache memory structures. Intel CPU supports "SpeedStep" [3] frequency / voltage control, the technology that changes all cores' frequency at the same.

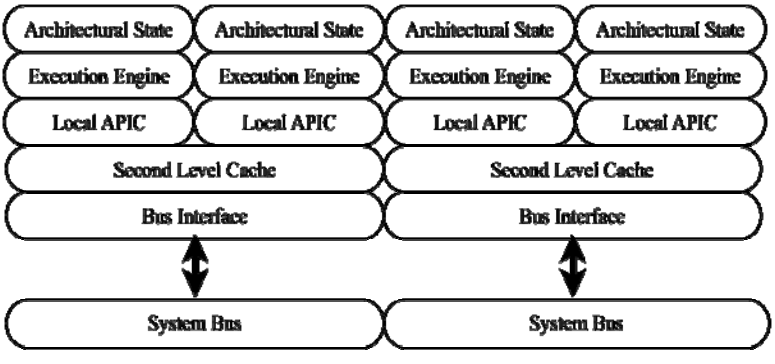

Fig. 1. Intel Quad-Core CPU system structure [11]

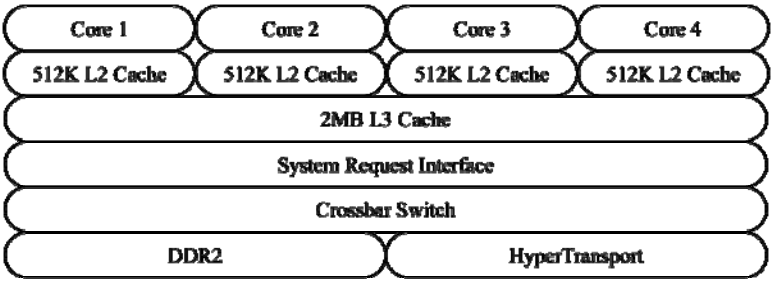

Fig. 2. AMD Quad-Core CPU system structure [12]

AMD quad-core CPU, as shown in Figure 2, has individual L2 cache in each core and share L3 cache, (a special design), and then integrated to DDR2 memory controller into CPU, helping to increase memory access speed. Each core has individual channel to access system bus, and L3 cache and peripheral chips from crossbar switch. AMD provides "PowerNow!" [4] technology to adjust each core's working frequency / voltage.

A cluster platform is built up by interconnecting a number of single-core CPU, and a message passing library, such as MPI is needed for data exchange among computing nodes in this distribution computing environment. In addition, high speed network as Infiniband is needed to interconnect the computing nodes. As multi-core CPUs are introduced and built in cluster environments, the architecture of this newly proposed cluster is as presented in Figure 3. The main advantages of data exchanges between cores inside of a CPU is much faster than passing by a network and South / North bridge chip.

Infiniband networking technology is a good and fast enough solution to connect all computing nodes of a cluster platform, but expensive. Gigabit Ethernet is cheaper solution and widely built in general network environment, though slower in transmission speed and definitely drop down data exchange performance. To send data to a core that is inside of a different host will be needed to consume extra energy when waiting for data.

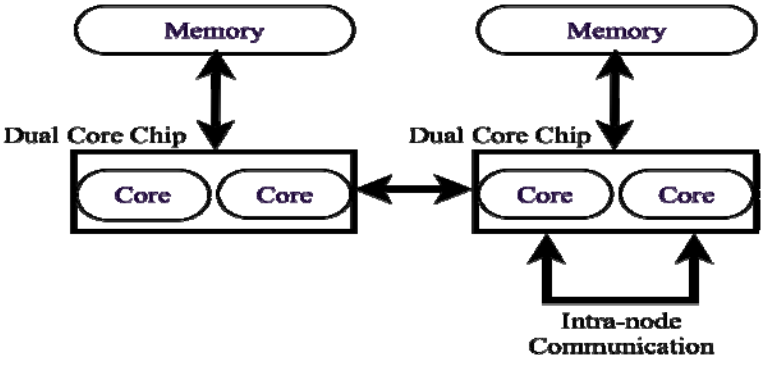

Fig. 3. Multi-core based cluster structure [13]

"SpeedStep" and "PowerNow!" technologies are good solutions to reduce power consumption, since they adjust CPU's frequency and voltage dynamically to save energy. The power consumption can be calculated by the function:

$$P=IV=V^2f=J/s. \tag{1}$$

where P is Watt, V is voltage, I is current, f is working frequency of CPU, J is joule and s is time in seconds. It means that lower voltage in the same current condition saves more energy. How and when to reduce voltage and frequency become an important issue, since one of main targets of clustering computing computers is to increase

the performance, while slowing down CPU's frequency is conflict with performance. Considering data latency of network, and CPU load in current CPU technologies, we would like to create a low energy cost cluster platform based on general network architecture, that keeps almost the same data transmission time though lower in energy consumption when CPU in full speed.

The remainder of this paper is organized as follows. Some related works discussed in Section 2, and some challenges we found in our experiment is listed in Section 3. In Section 4, the proposed approach about reducing energy consumption is presented; while the testing environment and performance results in Section 5. Finally the conclusion and future works are discussed in Section 6.

2 Related Works

Based on the concept about reducing computing time, the job scheduling methodology as introduced in [8] was designed targeting for a faster complete data transmission; otherwise, adjust cache block size to find the fastest speed that transmits data using MPI between MPI nodes in situations as listed in [13] was studied, and similar implementation of the method using OpenMP was also observed in [14]. Another investigation focused on compiler that analyze program's semantics, and insert special hardware control command that automatically adjusts simulation board's working frequency and voltage, [10] research needs to be combined both hardware and software resources.

Base on a simulation board, researchers have designed routing path algorithm that tries to find a shortest path to transmit data in Networks-on-Chip [15], in order to reduce data transmission time between CPUs, as also to have opportunities to realistically port and implement it to a cluster environment.

Others, researches have applied Genetic Algorithms to make a dynamically and continuous improvement on power saving methodology [9]. Through a software based methodology, routing paths are modified, link speed and working voltage are monitored at the same time to reduce power consumption, while the voltage detection information required hardware support.

Modern Operating Systems as Linux and Windows provides hardware power saving function as introduced in [1] and [2], where they can drive "SpeedStep" [3] and "PowerNow!" [4] utilizing special driver to control CPU voltage and frequency. Of course hardware support is necessary, since depending on the CPU loading, CPU is automatically selected with lower frequency and voltage automatically.

3 Challenges of Power Saving in Multi-core CPU Cluster Platform

We have built a cluster platform that combines all technologies as listed above for experiment purposes. These advantages bring higher speed for data broadcasting, yet only between cores inside a CPU, a CPU core is maintained with high load means the CPU speed cannot be decreased. Analysis and reasoning on these situations are discussed next.

3.1 CPU Power Control Structure

The “SpeedStep” and “PowerNow!” were not show in Figure 1 and 2. The “Speed-Step” provides solely full CPU frequency and voltage adjustment. The design makes power control easier, though consumes extra energy. If only one core works with high load, power control mechanism cannot reduce other cores’ frequency / voltage, nor dropping down the performance of a busy core. Inefficient energy consumption brings temperature increasing, since low loading core generates the same heat as high load one, and brings the CPU’s temperature up at the same time.

AMD “PowerNow!” shows advantage in this issue, since we can reduce frequency when core works in lower loading without need to consider other cores’ situation, and heat reduction is also another benefit.

3.2 Network Bandwidth and Cache Structure

As shown in Figure 1, Intel’s CPU architecture shares L2 cache using individual hub, and it has two advantages and two problems:

A. Advantages

- **Flexible Cache Allocation**

Every core was allowed to use whole L2 cache from cache hub, the hub provides single memory access channel for each core, and simplifies internal cache access structure.

- **Decrease Cache Missing Rate**

When each core has massive cache request, cache memory decreases page swapping from main memory.

B. Problems

- **Cache Hub Congestion**

If huge amount of data request or sending commands happen suddenly, individual cache hub blocks data frames in cache memory or stops commands in queue. All cores and hub keep in busy state and thus consume extra energy.

- **Network Bandwidth Condition**

Lower network bandwidth makes previous situation more seriously in many nodes' cluster, since network speed cannot be as fast as internal CPU bus, if cross-node data frames appear, the delivering time is longer than intra-node data switch.

Compared with Intel, while data frame flood sends to CPU, AMD structure has no enough cache to save them, yet individual bus / memory access channel of each core provides isolated bandwidth, L2 cache built in core reduces data flow interference. Different CPU structure provides their advantages, and weakness appears while they are compared to each other.

3.3 MPI Environment Support

In a general situation, each computing node executed under a given core / host randomly indicated by cluster software, signifies that programmer cannot obtain additional core loading from node's code section. Following our purpose, finding system information about thread / node location works, but it is a hard method since the program would spend large amount of time in device I/O, includes open system state file, analysis information and obtaining node's location. Another alternative method is easier, where we make cluster platform that fixes node location in indicated core or host, and the function helps to get core loading from node's code. OpenMPI is selected for this issue.

4 The Proposed Approach

Upon with CPU specification, CPU power control interface and network structure, we provide a data broadcasting strategy that combines data flow limitation and core frequency controlling as shown below.

4.1 Drop Down Data Transmission Speed

It is not a good method to keep performance. In fact, we add 1μs delay between two packets, in a real environment, and the total transmission time is added as:

$$(N - 1) \times T \tag{2}$$

where N is total number of nodes and T is delay time between packets. We found that the total time has just been added less than one to five seconds in average, when is transmitted 100K data frames across two hosts that are connected via Gigabit Ethernet. Additionally, the advantage is that the loading of a central node that sends data to other nodes is decreased by almost 50%, although the method solves problems as cache hub and CPU internal bus congestion. On the other hand, data receiving core load is decreased by 15% in average when we added 10μs delay in these nodes, yet total transmission time is increased by less than 0.5s.

4.2 Data Broadcasting According to Core Loading

Following the previous result, we provide a Loading-Aware Broadcasting method (LAB). Based on the "PowerNow!" hardware structure, and keeping the same load on all cores is necessary for efficient energy consumption, thus sending data from central node to lowest loading node makes sense. If the load can be reduced on a core, then reducing CPU frequency is permitted for saving energy.

Still in LAB algorithm, as indicated in Figure 4, data frames are sent sequentially from Host 1-Core 0 to other cores. This method is often used to distribute wait-for-calculate data blocks in complex math parallel calculations. MPI provides broadcast command to distribute data block and reduce command to receive result. In order to changing data frame transmission path dynamically, we use point-to-point command to switch data, since this type of command can indicate sending and receiving node.

4.3 Slow Down Lower Loading CPU / Core

Although the challenge presented in subsection 3.1 exists, as for power saving issue, we use AMD system and "PowerNow!" to slow down lowering loading core frequency. The given CPU supports 2 steps frequency, and therefore they work in different voltage and current. Thus we focus on frequency adjustment, and calculating power consumption of each core as below:

$$P = V_{max} \times I_{max} \times T \tag{3}$$

where V_{max} and I_{max} are found from AMD CPU technology specification [6], and T is program execution time. Since "Time" joins the function, the unit of P is Joule.

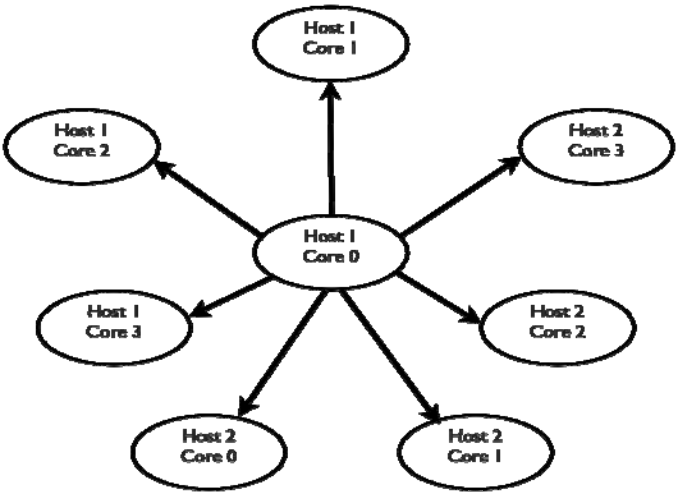

Fig. 4. LAB Algorithm structure diagram

The data distribution algorithm is given as below.

Loading-Aware Broadcasting (LAB)Algorithm

```
generating wait-for-send data frame
if (node 0)
{
  //detect nodes' loading from system information and
    save in TargetNode
  OpenCPUState;
  CalculateCPULoading;
  //sort TargetNode from low to high
  CPULoadingSorting;
  //send data follow sorting result
  while(!DataSendingFinish)
```

```
  {
    for(i=1; i<NodeNumber; i++)
      SendData(TargetNode[i]);
  }
  //send finish message to receiving nodes
  for(i=1; i<NodeNumber; i++)
    SendData(i);
}
if (other nodes)
{
  //receive data from node 0
  ReceiveData(0);
  usleep();
}
```

5 Performance Evaluation

In this section, experimental results on proposed power-saving strategy are presented. The cluster platform includes two computing nodes and connected via Gigabit Ethernet, and each node is installed with Ubuntu Linux 8.10 / kernel 2.6.27-9, Open-MPI message passing library is selected for thread execution affinity function, the hardware specification is listed as next:

Table 1. Host specification

CPU	AMD Phenom X4 9650 Quad-Core 2.3GHz
Layer 1 Cache	64K Instruction Cache and 64K Data Cache Per Core
Layer 2 Cache	512K Per Core
Layer 3 Cache	Share 2M for 4 Cores
Main Memory	DDR2-800 4GB

Three different sizes of data frames are transmitted between nodes: 1 byte, 1460 bytes and 8000 bytes. 1 byte frame is not only the smallest one in MPI data frame, but also in network, for complete data transmission in shortest time, source node generates huge amount of 1 byte frame, these packets congest CPU internal bus and network. 1518 bytes frame is the largest one in network, but considering that network header should be inserted into network packet, we select 1460 bytes frame for testing, and then, this size of packet brings largest amount of data in a single packet, and trigger fewest interrupt to CPU. Finally 8000 bytes frame is set for large data frame testing, since it needs to be separated to several other packets by network driver for transmission, and thus need the longest time for data transmission. While

the experiment is executed, we send 100K data frames between two nodes, and calculate the power consumption.

Each figure that follows next has four blocks. The first one is executed in Performance Mode (PM, CPU works in 2.3GHz), the second one is PowerSave Mode (PS, 1.15GHz), the third one is OnDemand Mode (OD, slows down frequency while CPU loading lower than 80%), and last one is LAB algorithm. Each block has four delay time configurations, the first one contains no delay between each data frame, the second delays 5μs, the third one delays 10μs, and last one delay 20μs. Still in figures that follows next, TD stands for Transmission Delay, Transmission Time as TT, and PC for Power Consumption.

The "Rank Number" in each figure means the number of nodes / cores join data broadcasting. For example, rank 2 means rank 0 broadcasts data to rank 1, and rank 4 means rank 0 broad-casts data to rank 1, 2, and 3. Since each host has four cores, the rank number 2~4 are internal node data transmission, and rank 5~8 are cross node data transmission. Although only 4 cores join work in rank number 2~4 other cores consume energy at the same time, and we still need to add the energy consumed.

Figure 5 shows the TT for one byte frame, and Figure 6 the PC. Comparing PM, PS and OD mode, we find that TD increases the TT over 3 seconds in rank 2~4 in every frequency level, but increases less than 1 second in 5~8. Figure 6 displayed one byte frame PC. Clearly, the PS mode spends the longest time to transmit data, though consumes the lowest energy. OD mode has none remarkable performance in power saving in rank 7~8, but it uses average 100J less than PM mode in rank 2~6, and keeps TT increasing less than 0.4s in cross-node situation. LAB algorithm displays advantage in no delay situation, less than 1s TT increasing yet consumes almost the same energy in rank 7~8. In other situations, LAB spends maximum 4s longer than OD mode, and saves 250J.

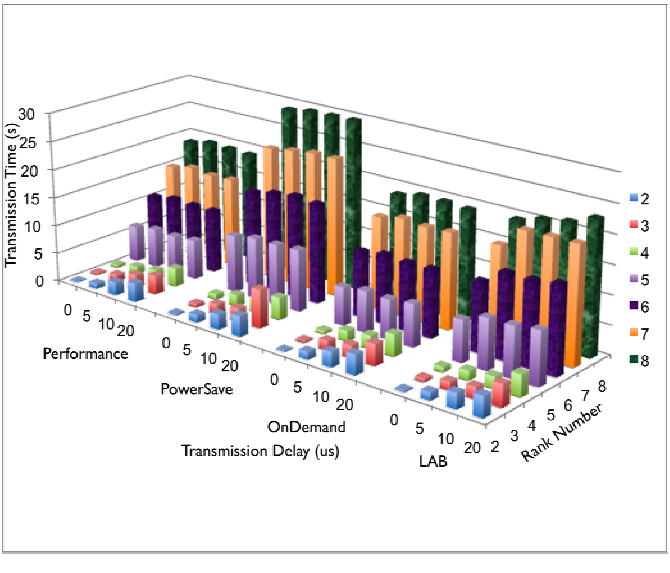

Fig. 5. Time Effect of TD on TT (Frame = 1 Byte)

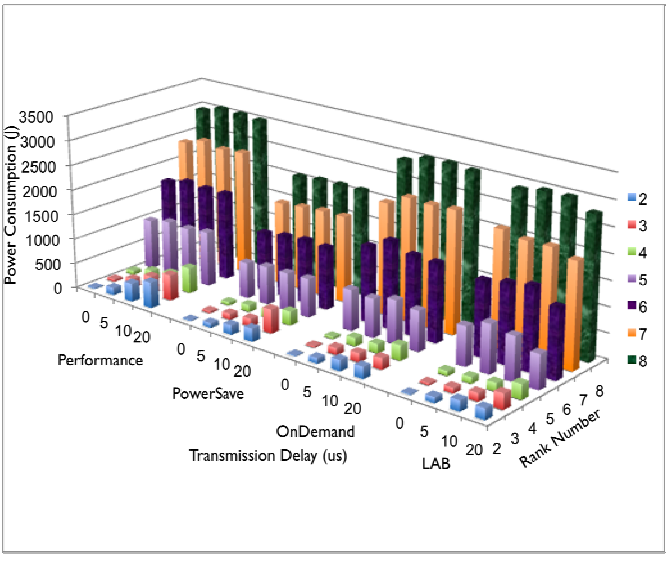

Fig. 6. Power Effect of TD on PC (Frame = 1 Byte)

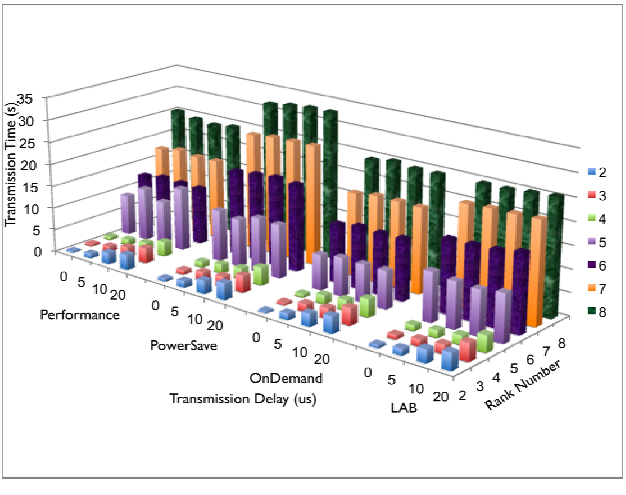

Fig. 7. Time Effect of TD on TT (Frame = 1460 Byte)

Figure 7 shows 1460 bytes frame TT. By comparing PM mode and OD mode, the completed time is longer than 1 byte frame in all situations. In Figure 8, OD mode uses in average over 200J less than PM mode. Our LAB algorithm made uses of 24~25s to complete data transmission as OD mode, yet consumes less than OD mode 200~600J in 8 ranks. In other situations, LAB keeps nearly the same performance, spending 4s longer than OD mode and consuming 200~400J less than OD mode.

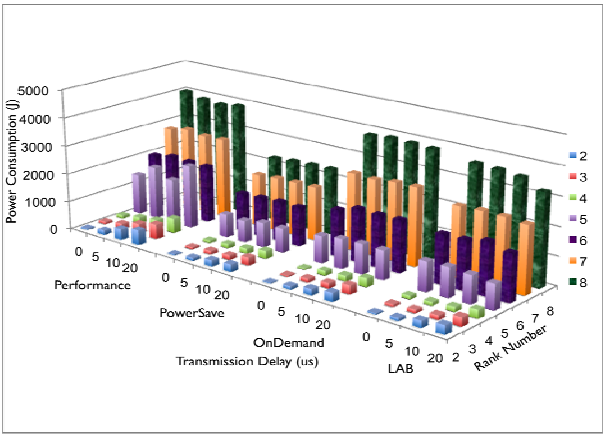

Fig. 8. Power Effect of TD on PC (Frame = 1460 Byte)

Although 8000 byte frame is the longest one, PS mode TT keeps 6s longer than other frames' size, as in Figure 9. Comparing OD and PM Mode, OD mode spends less than 1s longer than PM Mode, yet saves 200~400J in other cases. Comparing LAB algorithm and OD mode, LAB algorithm still keeps its advantages in the longest frame size, spends almost the same TT in 8 ranks and average 2~3s longer in other cross-node situations, consuming 100~ 400J less than OD mode.

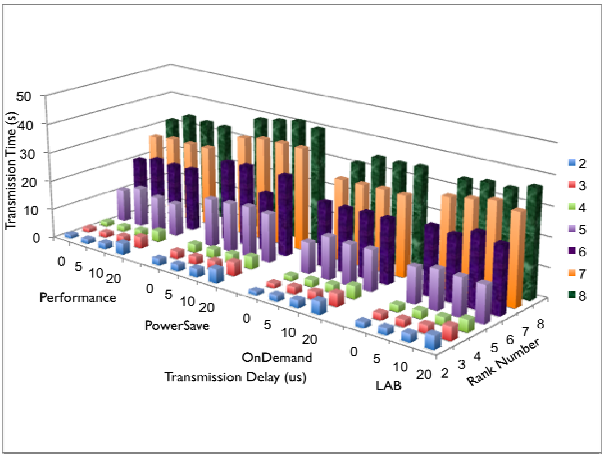

Fig. 9. Time Effect of TD on TT (Frame = 8000 Byte)

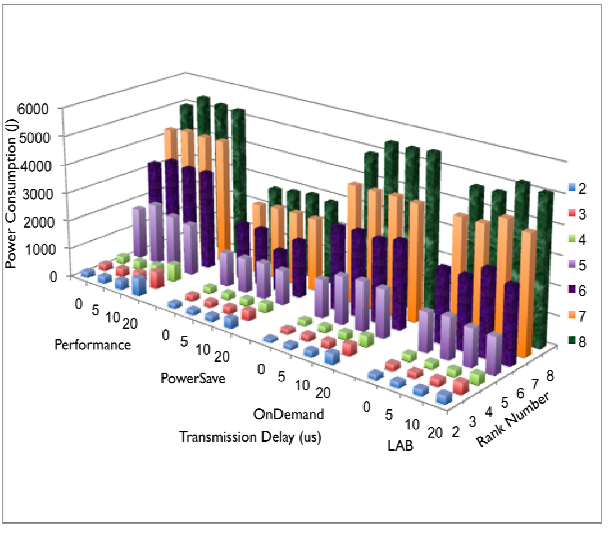

Fig. 10. Power Effect of TD on PC (Frame = 8000 Byte)

6 Conclusion and Future Works

In this proposed research, LAB algorithm keeps in average 4s TT increasing, yet saves 200~600J that compares with OD mode in cross-node situation. Limited by only 2 steps experimental cases of CPU frequencies (2.3GHz and 1.15GHz), we cannot keep CPU loading in a smooth curve. In desktop and server CPU, they do not keep in high loading work longer time, while they complete a concurrent job and next one does not be started. Power saving technology helps to decrease host energy consumption, and decreasing energy cost and carbon dioxide emissions can be reduced.

There are many directions to continue this investigation, to develop methods to save energy. If hardware and software provides functions about voltage or speed control, motherboard or any other type of peripheral device, then a hardware driver, power-aware job scheduling and data distribution algorithms can be combined and implemented, targeting in the construction of a low energy cost cluster computing platform in future.

References

1. Power Management Guide, `http://www.gentoo.com/doc/en/power-management-guide.xml`
2. Enabling CPU Frequency Scaling, `http://ubuntu.wordpress.com/2005/11/04/enabling-cpu-frequency-scaling/`
3. Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor, `ftp://download.intel.com/design/network/papers/30117401.pdf`

4. AMD PowerNow! Technology Platform Design Guide for Embedded Processors, http://www.amd.com/epd/processors/6.32bitproc/8.amdk6fami/x24267/24267a.pdf
5. AMD / Intel CPU voltage control driver down load, http://www.linux-phc.org/viewtopic.php?f=13&t=2
6. AMD Family 10h Desktop Processor Power and Thermal Data Sheet, http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/GH_43375_10h_DT_PTDS_PUB_3.14.pdf
7. AMD Opteron Processor with Direct Connect Architecture, http://enterprise.amd.com/downloads/4P_Power_PID_41498.pdf
8. Lan, C.-Y., Hsu, C.-H., Chen, S.-C.: Scheduling Contention-Free Irregular Redistributions in Parallelizing Compilers. The Journal of Supercomputing 40(3), 229–247 (2007)
9. Shin, D., Kim, J.: Power-Aware Communication Optimization for Networks-on-Chips with Voltage Scalable Links. In: Proceeding of the International Conference on Hardware/Software Code sign and System Synthesis, pp. 170–175 (2004)
10. Chen, G., Li, F., Kandemir, M.: Reducing Energy Consumption of On-Chip Networks Through a Hybrid Compiler-Runtime Approach. In: 16th International Conference on Parallel Architecture and Compilation Techniques (PACT 2007), pp. 163–174 (2007)
11. Intel 64 And IA-32 Architectures Software Developers Manual, vol.1, http://download.intel.com/design/processor/manuals/253665.pdf
12. Key Architectural Features of AMD Phenom X4 Quad-Core Processors, http://www.amd.com/us-en/Processors/ProductInformation/0,30_118_15331_15332%5E15334,00.html
13. Chia, L., Hartono, A., Panda, D.K.: Designing High Performance and Scalable MPI Internode Communication Support for Clusters. In: 2006 IEEE International Conference on Cluster Computing, September 25-28, pp. 1–10 (2006)
14. Noronha, R., Panda, D.K.: Improving Scalability of OpenMP Applications on Multi-core Systems Using Large Page Support. In: 2007 IEEE International Parallel and Distributed Processing Symposium, March 26-30, pp. 1–8 (2007)
15. Ogras, U.Y., Marculescu, R., Lee, H.G., Chang, N.E.: Communication Architecture Optimization: Making the Shortest Path Shorter in Regular Networks-on-Chip. In: 2006 Proceedings of the conference on Design, Automation and Test in Europe, Munich, Germany, March 2006, vol. 1, pp. 712–717 (2006)