Core-idling on MPI Intra-node Communication Channels for Energy Efficiency

Keon-Woo Kim
Dept. of Comp. Sci. & Eng.
Konkuk University
Seoul, Korea
tmffka7@konkuk.ac.kr

Hyun-Wook Jin
Dept. of Comp. Sci. & Eng.
Konkuk University
Seoul, Korea
jinh@konkuk.ac.kr

Eun-Kyu Byun
Korea Institute of Science and Technology Information (KISTI)
Daejeon, Korea
ekbyun@kisti.re.kr

***Abstract*—Busy-waiting used to implement parallel programming models can provide low latency but at the expense of energy. There have been several studies to enhance the energy efficiency of MPI libraries and applications by lowering the CPU speed during busy-waiting or turning off CPU components. While most studies have focused on prediction-based schemes that determine to enter an energy-saving mode, a redesign of the MPI library for better supports for suspending and resuming processes over multiple communication channels has been paid less attention. In this study, we suggest a signaling-based core-idling approach for energy efficient MPI over multiple communication channels. Our preliminary implementation supports intra-node communication channels, such as shared memory and memory mapping. The performance measurement results show that our preliminary implementation can reduce the energy consumption of NPB up to 12% in the undersubscribed case and 91% in the oversubscribed case, respectively, on a NUMA-based system.**

***Index Terms*—energy efficiency, message passing interface, intra-node communication, core-idling**

I. Introduction

In high-performance computing systems, the runtime solely dedicates a CPU core to each parallel process to maximize the parallelism between processes. Accordingly, parallel programming libraries and parallel applications are optimized on the assumption that a parallel process occupies an entire CPU core without sharing it with other processes. Most Message Passing Interface (MPI) libraries, for instance, perform busy-waiting to check the completion of outstanding communications [1]. Using busy-waiting is acceptable and can provide low latency, as a CPU core runs only a single process. However, the longer the busy-waiting time, the higher the energy consumption. In small-scale systems, parallel processes are more likely to enter the communication phase at very similar time points, which results in a relatively short busy-waiting time. Such busy-waiting time, however, becomes unpredictable and even longer on a large-scale system mainly due to nonuniformity of network latency between processes, asynchronous semantics in communications, and load imbalance.

There have been several studies to enhance the energy efficiency of MPI libraries and applications. Using Dynamic Voltage and Frequency Scaling (DVFS) that provides different levels of voltage and frequency for operating processors (e.g., P-states) is a famous approach to enhance the energy efficiency [2]. Another approach is to promote core-idling (e.g., C-states) that turns off hardware components of idle cores [3]. In either approaches, to minimize the side effect of using energy-saving modes of CPUs, researchers have focused on prediction schemes that determine when and which energy-saving mode to enter. If the prediction can be done inside of the MPI library, it can gain much insight by utilizing internal information concealed beneath application programming interfaces. However, this requires a redesign of the MPI library, which has been paid less attention. The core-idling approach especially needs to suspend and resume MPI processes in the MPI library while supporting multiple communication channels. Although existing core-idling approaches considered a single communication channel, MPI libraries exploit different channels for inter and intra-node communications, respectively. In addition, intra-node communication is further optimized through multiple channels, such as shared memory and memory mapping [4]. Thus, it is challenging to support suspending on multiple communication channels without negatively impacting on performance and correctness. Some MPI libraries also allow processes to yield CPU resources to other processes running on the same core, but this is only useful when the processes are not only oversubscribed but also busy.

In this study, we aim to support the core-idling approach over multiple MPI communication channels. As a first step, we focus on the intra-node communications that utilize a shared memory channel for small messages and a memory mapping channel for large messages. We present a signaling-based design and show that its implementation can improve energy-efficiency on a NUMA-based system. The signaling-based design would be flexible enough to support the inter-node communication channel and compatible with a DVFS approach. We also believe that the proposed design can leverage existing prediction schemes used in DVFS and core-idling approaches.

II. Energy-efficient Intra-node Communication

MPI libraries usually provide different intra-node communication channels for different message sizes [4]. The shared

This research was supported by the Supercomputer Development Leading Program of the National Research Foundation of Korea (NRF) funded by the Korean government (Ministry of Science and ICT (MSIT)) (No. 2020M3H6A1084853).

memory channel moves messages from source to destination via a shared memory region. This channel is used for small messages based on the eager protocol. The memory mapping channel directly moves messages from source to destination without intermediate copies by means of kernel-level support. Though this channel can reduce the number of data copies, it has a high memory mapping overhead. Thus, the memory mapping channel is used for large messages based on the rendezvous protocol.

To support core-idling on MPI intra-node communication channels, we need to identify where to suspend and resume processes. In the case of the shared memory channel, a process should be suspended if a shared buffer to which a sending message is copied is not available or there is no received message in the shared memory region. The process is resumed when a shared buffer becomes available or a new message is arrived. In the case of the memory mapping channel, we need to take the control messages of the rendezvous protocol into account in addition to just being able to send or receive messages. For example, the sender that initiates the rendezvous protocol for the message transmission over the memory mapping channel needs to be suspended if the Clear-To-Send (CTS) message is not received. When a process is suspended and the core becomes idle, the operating system asks the core to enter an idle state determined by the governor.

Our approach has been implemented in MVAPICH2 v2.3.1. We implement our approach in such a way that the framework invokes a user-defined energy-saving policy when the process is about to enter the busy-waiting mode. In this manner, we separate mechanism and policy and make it easy to change the energy-saving policy. We believe that this design can leverage other energy-saving policies, though our current policy is to simply force processes in the busy-waiting mode to suspend.

Suspending and resuming a process can be implemented in several ways. We can use assembly instructions, such as `mwait` on Intel processors, that have the core enter an energy-saving mode and wait for an event. We can also use CPU-independent system calls, such as timers, semaphores and signals. Since timers are suitable only for coarse-grained controls and semaphores are deadlock-prone, we use signaling in our preliminary implementation. Thus, our implementation is CPU-independent and easy to extend the implementation later to support the inter-node communication channel.

III. Performance Evaluation

We measured the execution time of NAS Parallel Benchmarks (NPB) and their energy consumption on a NUMA-based computing node that equipped with two Intel Xeon Ivy Bridge 2.8 GHz processors (Deca-core $\times$ 2). We measured the package-level energy consumption by using Running Average Power Limit (RAPL) [5]. We considered an undersubscribed case (the number of processes is smaller than that of cores) and an oversubscribed case (the number of processes is larger than that of cores).

Fig. 1 compares the execution time and energy consumption of NPB MG, CG, IS, and FT Class C. The value 100% on y-axis represents the performance of unmodified MVAPICH2 v2.3.1. In the undersubscribed case (Fig. 1(a)), our core-idling could save energy consumption of NPB up to 12% with a marginal (less than 2%) impact on execution time. In the oversubscribed case (Fig. 1(b)), both energy consumption and execution time were improved up to 91% in CG. This large improvement is mainly because core-idling forces processes that wait for a message with busy-waiting to suspend and significantly mitigates needless competition for CPU resources.

We also measured how long the cores stayed in the deepest energy-saving mode by using the `cpupower` command. Fig. 2 shows the ratio of the average per-core time spent in the idle state (C6) to the total execution time. This idle time increased significantly with core-idling, which was the main reason for observing a higher energy saving in Fig. 1.

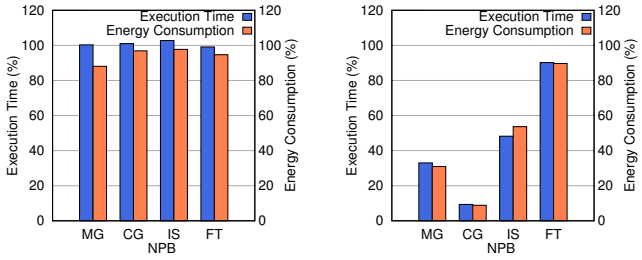

Fig. 1. Comparisons of execution time and energy consumption: (a) 16 processes (undersubscribed case); (b) 32 processes (oversubscribed case).

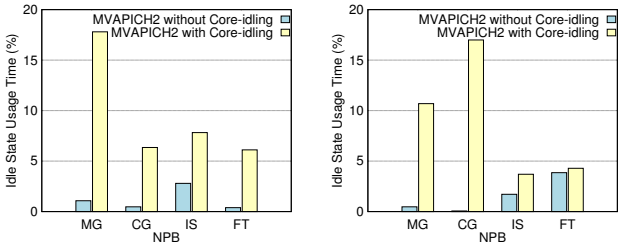

Fig. 2. Average time spent in the deepest energy-saving mode: (a) 16 processes (undersubscribed case); (b) 32 processes (oversubscribed case).

IV. Conclusions and Future Work

In this paper, we suggested signaling-based core-idling for energy efficient MPI over multiple intra-node communication channels. We identified when a process should be suspended and resumed on the shared memory and memory mapping channels. The performance measurement results showed that our implementation could reduce the energy consumption by 12% in the undersubscribed case with NPB MG and 91% in the oversubscribed case with NPB CG, respectively. It was also to be noted that the execution time had a negligible impact in the undersubscribed case, while a significant improvement was observed in the oversubscribed case.

As future work, we plan to measure the energy efficiency of signaling-based core-idling on various CPU architectures. In addition, we will expand our implementation to support the inter-node communication channel.

REFERENCES

[1] L. Piga, I. Paul, and W. Huang, "Performance boosting opportunities under communication imbalance in power-constrained HPC clusters," in Proceedings of the 45th International Conference on Parallel Processing (ICPP '16), pp. 31–40, 2016.

[2] D. Cesarini, A. Bartolini, A. Borghesi, C. Cavazzoni, M. Luisier, and L. Benini, "Countdown Slack: A run-time library to reduce energy footprint in large-scale MPI applications," IEEE Transactions on Parallel & Distributed Systems, vol. 31, no. 11, pp. 2696-2709, November 2020.

[3] A. Venkatesh, A. Vishnu, K. Hamidouche, N. Tallent, D. K. Panda, D. Kerbyson, and A. Hoisie, "A case for application-oblivious energy-efficient MPI runtime," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '15), pp. 1–12, 2015.

[4] L. Chai, P. Lai, H.-W. Jin, and D. K. Panda, "Designing an efficient kernel-level and user-level hybrid approach for MPI intra-node communication on multi-core systems," in Proceedings of the 37th International Conference on Parallel Processing (ICPP '08), pp. 222-229, 2008.

[5] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "RAPL: Memory power estimation and capping," in Proceedings of the ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED '10), pp. 189-194, 2010.