Research on Optimizing KNL Hybrid Memory Using a Job Scheduler

Seungwoo Rho, Geunchul Park, and Dukyun Nam

Abstract—Scientific problem, engineering, and data analysis involving very large-scale calculations requires efficient high performance computers and a batch scheduler system that can manage these computers. In this paper, we describe a technique for using Slurm, a widely employed batch scheduler for high performance computers, and Knights Landing, the cutting-edge manycore processor released by Intel. Also, we employed numa commands to allocate KNL hybrid memory, that is local memory (DDR) and the high bandwidth memory (MCDRAM) in Slurm. Finally, we performed experiments with our self-developed MPI benchmarks to show that there are performance improvements by running one bechmark on two nodes divisionally and assigning only a memory intensive benchmark to MCDRAM.

Research Keywords—Hybrid Memory, High Performance Computing, Knights Landing, Scheduler

1 INTRODUCTION

Owing to the limitations of integrated circuits, we departed from the multicore era wherein, several cores were placed on a single chip to improve computer computation performance, and we entered the manycore environment wherein tens or hundreds of CPUs are placed on a chip. [1] Major examples of systems equipped with manycore architectures include the Sunway TaihuLight, a supercomputer that was developed in China, as well as Intel's Xeon Phi series and the Tilera's tile CPU. In this kind of change in computer architecture, the overall performance of cores is increased; however, there have been obstacles to increasing computer performance due to ancillary problems such as limitations in memory bandwidth, low storage I/O speed, increases in power consumption, and changes in parallel programming environments. At the same time, to overcome these problems various attempts have been made.

In this kind of computing environment, memory bandwidth is one of the biggest bottlenecks in measuring the performance of very large computational science applications. To overcome this problem, Intel released the cutting-edge Xeon Phi processor based on the Knights Landing (KNL) [2] structure, which is equipped with both local memory (DDR) and high bandwidth memory (hbm) [3], the hbm was created based on multichannel DRAM (MCDRAM) and has more than five times the bandwidth of conventional memory.

One of the most economical and effective methods for improving the performance of this kind of manycore computing is to use a job scheduler, which helps manage and use hardware resources such as multiple cores and memory efficiently. In this paper, we chose Slurm [4], one of the job schedulers mainly used in the high performance computing environment, and we introduce techniques that can utilize hybrid memory with numa commands in Slurm. We also show that there are performance improvements by running one benchmark on two nodes divisionally and assigning only a memory intensive benchmark to MCDRAM.

- *Seungwoo Rho is with the Supercomputing R&D Center, KISTI, Yuseong-gu, Daejeon, South Korea. E-mail: seungwoo0926@kisti.re.kr*
- *Geunchul Park is with the Supercomputing R&D Center, KISTI, Yuseong-gu, Daejeon, South Korea. E-mail: gcpark@kisti.re.kr*
- *Dukyun Nam(corresponding author) with the Supercomputing R&D Center, KISTI, Yuseong-gu, Daejeon, South Korea. E-mail: dynam@kisti.re.kr*

2 RESEARCH BACKGROUND

Top 500 Project [8] announces the rankings of the world's top ultra high performance computers. We can make sure manycore systems in it. Some of the manycore systems with more than 60 cores use their own schedulers that they have made, but most use the PBS Pro or Slurm batch scheduler. PBS Pro is the commercial version of PBS, which

was developed in 1991 by NASA. Slurm is open source, easy to install, and used in about 60% of the Top 10 supercomputers. As such, in this paper we use Slurm in KNL.

3 Hybrid Memory Utilization

In this section, we introduce the method for setting up KNL and the Slurm batch scheduler as well as the method for using high bandwidth memory in Slurm. We also propose how to optimize hybrid memory using two MPI benchmarks developed in this study.

3.1 TestBed Configuration

We used the newest version at the time of the test, version 17.02.7. For the test bed, we used an S7200AP server board made with KNL 4 nodes. Each node has 68 cores, supports 272 threads, and has 96 GB of DDR4 memory and 16 GB of MCDRAM. Each node is connected to the others through an omnipath network connected by two 16-line PCIe cards and has a maximum transmission bandwidth of 200 Gbps. To integrate KNL in Slurm, two setting files (slurm.conf and knl_generic.conf) is needed. In this study, we set them to the commonly used quad cluster mode and the flat memory mode and performed testing.

3.2 Hybrid Memory Optimization

Slurm supports GPU and MIC plug-in modules, but only hbm plug-in modules are not officially supported. As such, we had to develop our own hbm plug-in module to fully use MCDRAM. However, in this study, rather than developing an hbm plug-in module, we used numactl, which is a command for accessing local memory. To use numactl, the insertion position is very important. That is, we must insert the numactl command before the actual execution command. For example, in case of using MCDRAM, we can write it like this: “srun mpirun –n 128 numactl –p 1 . /memory_benchmark”.

We performed two experiments with two MPI benchmarks, which is the self-developed benchmark [6]. We also studied the performance differences in single-node experiments and two-node division experiments to investigate the possibility of optimizing limited high bandwidth memory using Slurm. In the single-node experiment, which had one CPU task and one memory task, one task was performed on each node, while in the two-node division experiment, one task was split into two tasks and executed concurrently, and the performance differences between them were compared. We ran the experiments on common DDR memory, MCDRAM, and MCDRAM for only a memory intensive benchmark. As a result, the two-node division experiment was better than single node experiment. In particular, when the memory intensive benchmark was assigned to the MCDRAM, there was a 7.8% performance improvement over when both CPU and memory-intensive benchmark were assigned to MCDRAM. The reason for the performance improvement was that the memory-intensive benchmark can use twice MCDRAM in the two-node division experiment.

4 Conclusion

In this paper, we have described a method for employing Slurm, a widely used job scheduler, in KNL, a cutting-edge manycore system from Intel. Specifically, we have presented a method for employing the high bandwidth memory of Knights Landing in Slurm, and we have shown the 7.8% performance improvement on the two-node division experiment when only the memory intensive benchmark was assigned to the MCDRAM.

Acknowledgment

This work is supported by the KISTI (Grant No. K-17-L01-C01) and the IITP (Grant No. R0190-15-2012).

References

[1] A. Singh, M. Shafique, A. Kumar, J. Henkel, “Mapping on multi/many-core systems: Survey of current and emerging trends,” IEEE DAC, 2013.

[2] A. Sodani, R. Gramunt, J. Corbal, H.-S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, Y.-C. Liu, “Knights landing: Second-generation intel xeon phi product,” IEEE Micro, vol. 36, no. 2, pp. 34-46, 2016.

[3] An Intro to MCDRAM (High Bandwith Memory) on Knights Landing(KNL): https://software.intel.com/en-us/blogs/2016/01/20/an-intro-to-mcdram-high-bandwidth-memory-on-knights-landing

[4] Yoo, A.B., M.A. Jette, and M. Grondona, “SLURM: Simple Linux Utility for Resource Management, in Job Scheduling Strategies for Parallel Processing,” L. Rudolph and U. Schwiegelshohn, Editors. SpringerVerlag. p. 44-60, 2003.

[5] J. Dongarra, H. Meuer, and E. Strohmaier. Top 500 Supercomputer Sites. Technical report, University of Tennessee, Knoxville, TN, USA, 1999.

[6] S.W. Rho, S.Y. Kim, D.Y. Nam, G.C Park, J.S. Kim, “Enhancing the Performance of Multiple Parallel Applications using Heterogeneous Memory on the Intel‘s Next-Generation Many-core Processor,”

Journal of KIISE, vol 44, no. 9, pp. 878-886, Sep 2017.

Seungwoo Rho He is a researcher in National Institute of Supercomputing and Networking (NISN) at KISTI (Korea institute of Science and Technology Information). He received his B.S. and M.S. degree in electrical and computer engineering from University of Seoul in 2009 and 2011, respectively. His main research interests include supercomputer, distributed computing, parallel computing, deep learning, and block chain.

Geunchul Park He received his B.S. & M.S. in Computer Science and Engineering from Chung-Ang University in Korea. He is currently a researcher at supercomputer SW research lab in the National Institute of Supercomputing and Networking at Korea Institute of Science and Technology Information (KISTI). His research interests are in high performance and distributed computing, parallel program optimization, system software in HPC, etc.

Dukyun Nam He received his B.S. in computer science and engineering from Pohang University of Science and Technology (POSTECH), Korea, in 1999, M.S. and Ph.D. in information and communication engineering from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 2001 and 2006, respectively. He is currently a head of supercomputer SW research lab in the National Institute of Supercomputing and Networking at Korea Institute of Science and Technology Information (KISTI). His research interests are in high performance and distributed computing, low power computing, system software in HPC, etc.