Evaluating the Support of MTC Applications On Intel Xeon Phi Many-Core Accelerators

Poornima Nookala, Serapheim Dimitropoulos, Karl Stough, Ioan Raicu
Department of Computer Science, Illinois Institute of Technology, Chicago, Illinois
{pnookala,sdimitro,kstough,iraicu}@hawk.iit.edu

***Abstract* -As Many-Task Computing (MTC) is becoming common-place on clusters, grids, and supercomputers, research that aims to take advantage of the new advances in hardware for MTC workloads is becoming more relevant. A good example is the design of frameworks like GeMTC that incorporate general purpose GPU hardware to improve the concurrency of executing tasks. This work attempts to support MTC workloads on the Intel Xeon Phi accelerators. Our plan is to develop two frameworks that will achieve that goal. One based on OpenMP and the other one based on Intel's Symmetric Communication Interface (SCIF) provided for Many-Integrated Core (MIC) accelerators like the Xeon Phi. Both frameworks aim to provide the same interface as GeMTC, leveraging the integration efforts with the Swift parallel programming system. Our end-goal is to present how programming many-core computing processors can be made easier and more productive using OpenMP or SCIF, and enable the execution of MTC workloads hybrid accelerator-based systems.**

Keywords-Many-task computing; Accelerators; Intel Xeon Phi Coprocessor; Programming models; Execution models.

I. BACKGROUND INFORMATION

The Intel Xeon Phi is a hardware coprocessor from Intel. It is a PCI device with roughly 60 cores and 240 hardware threads. Its design makes it ideal for applications that are performance critical and need large levels of parallelism. Moreover, the fact that it implements x86 for its instruction set architecture, makes its integration with existing systems simpler than the integration of other accelerators like General Purpose GPUs (GPGPUs). GeMTC is a CUDA based framework which allows Many-Task Computing workloads to run efficiently on NVIDIA GPUs [3]. The novelty in the design of this framework is that different jobs that are running in parallel are also isolated from each other and therefore the utilization of the GPU is almost maximized. Unfortunately, the framework's implementation is very closely-tied to the architecture and the conventions of GPUs. Many-Task Computing (MTC) has been an emerging paradigm and area of research for some years now. Therefore, considering embedding the capabilities of the Xeon Phi in systems that support MTC workloads is considered a relatively new ground. This paper attempts to cover this ground.

II. ARCHITECTURE

Due to the foundations of Intel architecture, the coprocessor can be programmed in several different ways. For the OpenMP implementation, we used offloading approach for offloading computations from host to the Phi. For the SCIF part, we implemented the framework to run natively on the Phi while accepting jobs from clients running on the host CPU [6]. The major advantage of native execution coupled with SCIF over offloading is that the developer gets more control overall in the configuration and the architecture of their design in order to maximize performance. In addition, different MIC cards can communicate directly with each other basically making certain designs more efficient.

The OpenMP version of the framework is developed using a Producer-Consumer architecture which communicates using shared memory for IPC. The Consumer side hosts the framework which runs as multiple worker threads which use the shared memory space as a queue structure, continuously accepting new tasks from producer. Likewise, the producer acts as a client process which submits tasks to the queue. Asynchronous offloading is used to allow the framework to be non-blocking to continue accepting tasks while other tasks are running on the Phi. This approach was chosen to provide the same feature set as GeMTC while taking advantage of asynchronous offloading capabilities of OpenMP.

The SCIF implementation is a complete port of the GeMTC framework. The core architecture of GeMTC is actually completely rewritten in C from CUDA and abstracted out into a shared library. The library includes all the main functionality of GeMTC. The rest of the framework is modeled after a client-server architecture where clients send their tasks to the Phi from the host and a server, which runs natively on the Phi, accepts the jobs. After submitting the job, the clients can request the result and the server will deliver it to them when the task has finished processing. The whole procedure is non-blocking for the server who can handle multiple requests and submissions at the same time. The SCIF API is used for communications between the server and the clients.

III. EVALUATION

All of our experiments were run on the Midway High-Performance Computing Cluster at University of Chicago. Our testing host is an Intel Sandy Bridge with 32 cores at 2.6 Ghz and 32 GB of RAM. It has 2 Xeon Phis attached to it.

Both of them are from the 5100 series of Intel coprocessors and have 60 cores at 1.053 GHz each and 8 GB RAM.

A. *Synthetic Sleep Workloads*

Experiments were performed using various sleep length tasks. As seen below, preliminary results show that efficiency reaches higher 90s for task lengths at 1 msec when using 1 worker on host, 2 msec for 60 workers and 5 msec for 128 workers. This clearly shows that this framework using OpenMP performs better than GeMTC on Xeon Phi which reaches higher efficiency only at 5 ms. To reduce the overhead of multithreading, we took an approach of creating threads on the Phi before offloading tasks which reduced the overall execution time considerably [7]. Also, SCIF approach performs at higher 90s with sleep tasks of 1 msec.

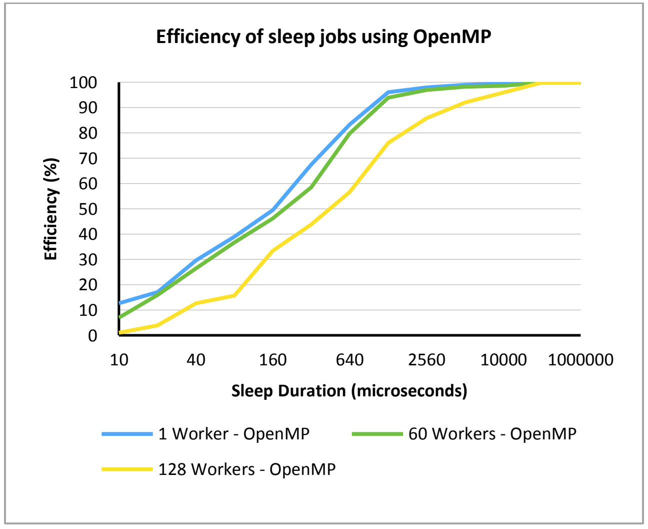

Figure 1. Efficiency of sleep jobs (usec) on Xeon Phi using OpenMP measured by varying number of worker threads and tasks on Xeon Phi.

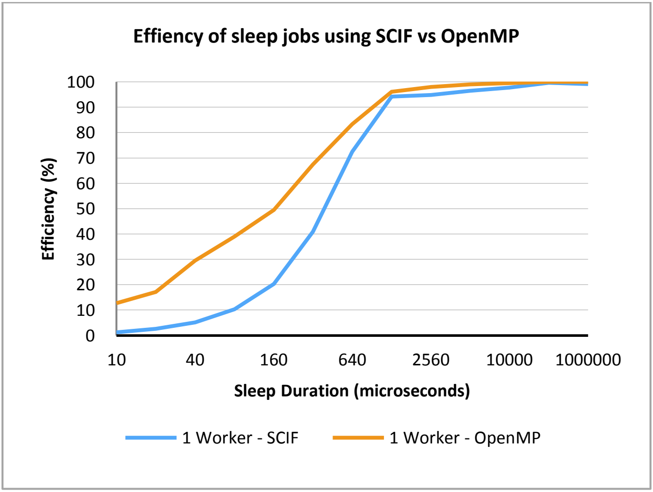

Figure 2. Efficiency of sleep jobs using OpenMP and SCIF (a comparison).

B. *Matrix Multiplication Results*

In order to assess the real-world performance of Xeon Phi, the team performed tests using naïve matrix multiplication. The overhead of OpenMP data transfer only became negligible when matrix sizes of 64x64 were tested. Before that point, the task completion time remained fairly constant. At larger matrix sizes, the time taken increases linearly with the amount of work performed. The team also analyzed the performance gain from an increase in the number of threads. It was found that while single-threaded tasks scaled fairly linearly with the workload, many-threaded tasks didn't achieve optimal scalability until much larger matrices were tested. Data transfer offload overhead seems to be high when offloading large amounts of data and techniques for reducing data transfer overhead and reusing the allocated memory are being investigated. Our SCIF implementation does not support bulk I/O for testing with large matrixes. Enabling bulk I/O for SCIF implementation using RMA API is part of future work.

IV. CONCLUSION AND FUTURE WORK

To enable running MTC workloads on Xeon Phi, we designed a framework that not only sends and executes tasks on Xeon Phi but also ensures that these tasks are isolated from each other and can run in parallel. Our work is built upon the existing functionality of GeMTC and in the future would allow for an identical interface which could be dropped into Swift/T. We implemented both OpenMP as well as SCIF-based frameworks and were able to run MTC workloads on Xeon Phi. Our preliminary evaluation data are encouraging and should provide enough motivation for future. Our future work includes evaluating the design further and enabling Swift/T integration.

REFERENCES

[1] Poornima Nookala, Karl Stough, ”GeMTC-OpenMP Source Code Repository”, https://github.com/pnookala/MIC OpenMP GeMTC.

[2] Serapheim Dimitropoulos, ”GeMTC-SCIF Source Code Repository”, https://github.com/sdimitro/scif-modules/tree/master/scif-sc.

[3] S. Krieder, J. Wozniak, T. Armstrong, M. Wilde, D. Katz, B. Grimmer, I. Foster and I. Raicu, ”Design and Evaluation of the GeMTC Framework for GPU-enabled Many-Task Computing”, ACM HPDC, 2014.

[4] J. Johnson, S. Krieder, B. Grimmer, J. Wozniak, M. Wilde and I. Raicu, ”Understanding the Costs of Many-Task Computing Workloads on Intel Xeon Phi Coprocessors”, GCASR, 2013

[5] E. Wang, Q. Zhang, B. Shen, G. Zhang, X. Lu, Q. Wu and Y. Wang, ”High-Performance Computing on the Intel Xeon Phi: How to Fully Exploit MIC Architectures”, Springer, 2014, pp. 3-30.

[6] Intel, ”Intel Many Integrated Core Symmetric Communications Interface (SCIF) User Guide”, 2012.

[7] High Performance Parallelism Pearls Volume One: Multicore and Many-core Programming Approaches, 1st Edition – James Reinders (Author),James Jeffers (Author).