Understanding the Costs of Many-Task Computing Workloads on Intel Xeon Phi Coprocessors

Jeffrey Johnson*, Scott J. Krieder*, Benjamin Grimmer*
Justin M. Wozniak†, Michael Wilde†‡, Ioan Raicu*†
*Department of Computer Science, Illinois Institute of Technology
†MCS Division, Argonne National Laboratory
‡Computation Institute, University of Chicago

***Abstract*—Many-Task Computing (MTC) aims to bridge the gap between HPC and HTC. MTC emphasizes running many computational tasks over a short period of time, where tasks can be either dependent or independent of one another. MTC has been well supported on Clouds, Grids, and Supercomputers on traditional computing architectures, but the abundance of hybrid large-scale systems using accelerators has motivated us to explore the support of MTC on the new Intel Xeon Phi accelerators. The Xeon Phi is a PCI-Express based expansion card comprised of 60 cores supporting 240 hardware threads to produce up to 1 teraflop of double- precision performance in a single accelerator. These cards are already being integrated into super-computing clusters such as Stampede, which hosts over 6,400 Xeon Phi Accelerators totaling in over 7 petaflops of double- precision performance. This work provides an in depth understanding of MTC on the Intel Xeon Phi and presents our preliminary results of running several different workloads on pre-production Intel Xeon Phi hardware. By utilizing Intel's provided SCIF protocol for communicating across the PCI-Express bus we have achieved over 90% efficiency near or outperforming OpenMP offloading tasks over 300 uS with our batch framework. This performance opens the opportunity for the development of a framework for executing heterogeneous tasks on the Xeon Phi alongside other potential accelerators including graphics cards for MTC applications. Our framework will provide fine granularity for executing MTC applications across large scale compute clusters. It will be integrated with our existing graphics card framework, GeMTC, to provide transparent access to GPUs, Xeon Phis, and future generations of accelerators to help bridge the gap into Exascale computing**

***Keywords*-MIMD, MTC, Accelerator, Intel Xeon Phi, Coprocessor**

I. Introduction

In this work we provide preliminary results evaluating MTC workloads running on Intel Xeon Phi Coprocessors. The Intel Xeon Phi Coprocessor is physically similar to other hardware accelerators such as GPGPUs but contains many significant underlying differences.

The High Performance Computing (HPC) community is seeing a large adaptation of general-purpose accelerator cards. In the past years, graphics cards have become a significant part of newly built computing clusters providing unprecedented parallelism with a low power footprint. Accelerators often require the use of reworking a program in order to use or fully utilize the device and as a result increase development time.

The pre-production Xeon Phi is a 61-core accelerator featuring 8 GB of GDDR5 connected to the host via a PCI Express bus. The Phi runs an instance of the Linux operating system, which occupies a single core of the accelerator to provide the developer with a familiar programming interface. As of the pre- production Xeon Phi, it is possible to use OpenMP, POSIX threads, OpenCL, Intel Math Kernel Library, MPI, or other popular libraries to develop and offload applications to the accelerator. Intel also provides a socket protocol, SCIF, to aid in data transfer between the host and the Phi to transfer data across the PCI Express bus. Intel also provides the application micnativeloadex to launch an application compiled for the Phi on the accelerator. This application ensures the correct libraries and program code are copied and executed on the accelerator.

II. Architecture

In order to provide a seamless and easy to use interface, our framework sets out to provides identical functionality to GeMTC. [1] GeMTC has three types off operations: Push/Poll for sending and receiving jobs, Malloc/Free for preparing device memory, and a memory copy operation to copy data to or from the accelerator. [2] By providing this interface we can easily tie in with Swift/T [3] to open up our solution to multi-node configurations.

A. SCIF Implementation

Our SCIF framework employs a client server architecture, which communicates via the Symmetric Communications Interface (SCIF). This interface abstracts communication across the PCI- Express bus to a UNIX socket semantics. The server application uses a single process launched on the Intel Xeon Phi, which then launches a single processing thread per hardware core to handle incoming work from the host. The client starts from a single process and launches no more threads than the server. Each server thread listens via SCIF on a discrete port, which is used to accept and return work to the client. Each client thread produces a batch of work to the accelerator, which can contain a heterogeneous list of tasks with varying lengths and data payload sizes. The client blocks until the entire batch is completed. This architecture allows the overhead of launching threads as well as offloading the code to be executed to be ignored and performance becomes only dependent on data transfer rate and processing rate of the processor. This framework does not provide the feature set of GeMTC due to the complexity of data transfer between the device and host over SCIF. Each party in the communication is forced to manually parse out received streams of data which is a highly error prone operation when dealing with complex lists of tasks.

III. Preliminary Results

In this section we present preliminary results for running several synthetic benchmarks on pre-production Intel Xeon Phi hardware.

A. Synthetic Sleep Workloads

To compare the overhead of our architecture with OpenMP three sleep programs are analyzed. The first uses our SCIF framework to launch a batch of sleep jobs to the Xeon Phi and block until completion then separately tests launching individual jobs to the device and waiting for the result before pushing the next job. The second uses OpenMP to offload a single block, which calls a sleep job 128 times by repetitively placing the function call in the block. This method was chosen to more accurately mimic the loop-unrolling style of our SCIF framework. The final program uses OpenMP to offload a single sleep task and is repeatedly called. Figure 1demonstrates the results of comparing all four different configurations in varying sleep lengths. By testing varying lengths of sleeps, we find that jobs over 320 uS benefit from the SCIF framework when sent in this length of a batch. At this point the performance is slightly above OpenMP but can enjoy the potential benefits of the framework.

IV. Related Work

The Xeon Phi is a very new technology and research is just beginning to show the usefulness of the technology. IRWTH Aachen University and Intel have collaborated on a paper outlining the initial performance of the preproduction Xeon Phi demonstrating the power of its dense processing using OpenMP related to a traditional 128-core system. It was shown that a single Phi had a lower overall performance but the power per core and power efficiency significantly outweighs the latter solution towards many-core. Intel also advocates the offloading capabilities of its compiler and OpenMP with use of the Phi showing that many OpenMP programs can be launched on the Phi without code change.

V. Conclusions and Future Work

It has been demonstrated that is it possible to achieve minimum overhead with the Xeon Phi by directly communicating between the host and accelerator via SCIF across the PCI Express bus. Using our proposed framework, it will be possible to share the resources of a Xeon Phi across multiple processes and users in a large scale computing environment while maintaining high performance through the use of specialized microkernels. Enabling the Xeon Phi to run heterogeneous workloads can enable the device to be used in a Many-Task Computing [4] environment where resources are shared between users and latency is important. The findings of this paper shows no significant performance tradeoff to the framework opening the way for a promising use of the Xeon Phi in new environments. In addition we plan to evaluate additional scientific applications including Molecular Dynamics applications, Protein Simulators [5] and many more. Future work will also enable the Swift Parallel Scripting Language to make use of Xeon Phi hardware accelerators. [6]

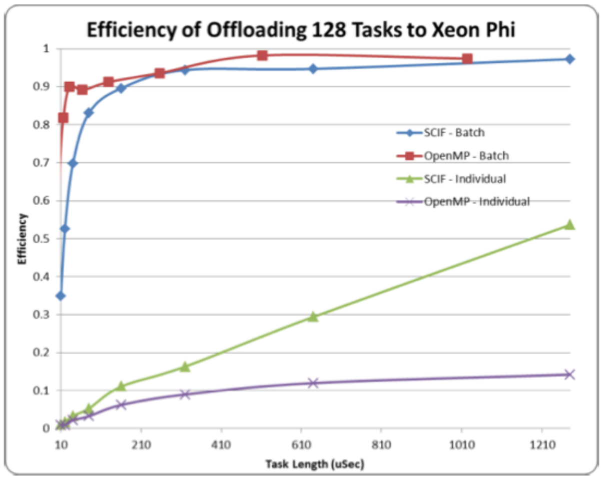

Fig. 1. Efficiency of Offloading 128 Tasks to Xeon Phi Comparison between OpenMP and SCIF with individual offloads and batch offloads.

References

[1] S. J. Krieder and I. Raicu, “Towards the support for many-task computing on many-core computing platforms,” Doctoral Showcase, IEEE/ACM Supercomputing/SC, 2012.

[2] B. Grimmer, S. J. Krieder, and I. Raicu, “Enabling dynamic memory management support for mtc on nvidia gpus,” EuroSys Poster Session, 2013.

[3] J. M. Wozniak, T. G. Armstrong, M. Wilde, D. S. Katz, E. Lusk, and I. T. Foster, “Swift/t: Large-scale application composition via distributedmemory data flow processing,” in *Proc. CCGrid*, vol. 13.

[4] I. Raicu, Z. Zhang, M. Wilde, I. Foster, P. Beckman, K. Iskra, and B. Clifford, “Toward loosely coupled programming on petascale systems,” in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, p. 22.

[5] A. N. Adhikari, J. Peng, M. Wilde, J. Xu, K. F. Freed, and T. R. Sosnick, “Modeling large regions in proteins: Applications to loops, termini, and folding,” *Protein Science*, vol. 21, no. 1, pp. 107–121, 2012.

[6] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. Von Laszewski, V. Nefedova, I. Raicu, T. Stef-Praun, and M. Wilde, “Swift: Fast, reliable, loosely coupled parallel computation,” in *Services, 2007 IEEE Congress on*. IEEE, 2007, pp. 199–206.