The Case for Resurrecting Distributed Virtual Shared Memory

B M Saif Ansary, Masters Student, Virginia Tech, bmsaif86@vt.edu
Advisor: Binoy Ravindran, Virginia Tech, binoy@vt.edu

Computing platforms are becoming increasingly parallel and heterogeneous, moving towards combinations of multiple CPUs and multiple accelerators that are increasingly coupled. The vast amount of applications written within the shared memory programming model cannot run as-is on these architectures. We propose to design an operating system and a set of compiler tools that enable shared memory applications to run seamlessly on such heterogeneous platforms. Hereafter we discuss the mechanisms we foreseen should be implemented in the operating system.

Nowadays single processor execution speed has reached a plateau and the parallelism per chip is constrained by power. Moreover inter-chip communication overhead represents scalability problem [1]. Therefore, new techniques are being explored by the architecture community to continue achieving faster computations: heterogeneity, via the integration of multi-core and many-core accelerators [2], where many-core accelerators are OS-capable; faster interconnects, via photonics technology [3] have emerged which can mitigate inherent communication bottlenecks between systems. Therefore, we foresee the necessity for the system software to support and exploit such heterogeneity, and the opportunity, due to new and faster interconnects, to resurrect distributed virtual shared memory.

Multi-threaded programming is a common practice. However, on a heterogeneous platform, different code sections of a program can be made to run faster if the computational patterns are carefully identified and mapped onto the best processor. For example, serial code can be mapped on a general purpose processor while SPMD can run on a special purpose accelerator (e.g., GPU). To exploit heterogeneity, languages such as CUDA, OpenCL have been developed. However, we argue that such languages hinder programmability, in fact programmers have to adopt a new programming model, different from the familiar shared memory. That requires performing memory transfers explicitly, specifying the compute kernel etc. which greatly increase the complexity of programming.

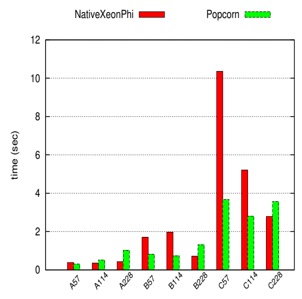
We propose system software that alleviates this extra effort in programming, exploits the architecture heterogeneity, while providing shared memory abstraction to the programmer. Such a software requires an operating system that provides distributed shared memory and strives to hide the underlying heterogeneity [4]; enabling tasks to migrate in the heterogeneous system.We believe that deploying distributed shared memory (DSM) on faster interconnects will exhibit low overheads, therefore it is worth exploring this solution, investigating inter-processor islands messaging. Our contribution is twofold. First we introduce a low latency, high throughput, inter-processor island messaging framework. Secondly a dynamic page-coherency protocol that provides shared memory abstraction amongst individual OS capable processor island.

Our target system is based on multicore CPUs (x86) and OS-capable accelerator cards, such as Intel Xeon Phi. Each processor island has different advantages over the other: Xeon Phi is better in vector processing, the host CPU is better in general data processing. Each island is an independent system connected through PCIe. The host CPU and Xeon Phi are individually cache-coherent shared memory islands, we consider them as different nodes. Each other memory can be accessed in different modes, by Remote memory access (RMA) and by Direct Memory Access (DMA), therefore representing two different memory configurations. Previous work, such as COSH [2], requires the programmers to explicitly manage this memory diversity. Our mechanism provides transparency by implementing DSM.

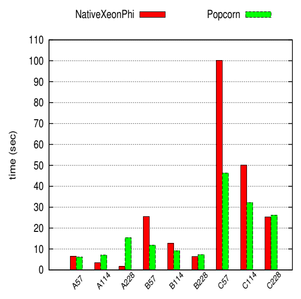
Our prototype implements a page coherency protocol inspired by MESI and a further page ownership protocol. The node on which the page is originally opened is the owner of that page. When threads migrate to different nodes, the pages are sent on demand. The remote node queries the owner node for the page. The ownership is transferred to the node which makes the latest modification. The ownership transfer distributes the workload of managing the page thus allowing the protocol to scale. We observe RMA is expensive compared to local access so for read-only pages we use DMA copy instead of RMA. This hybrid approach allows better performance compared to RMA only. We keep the pages consistent by adopting techniques used in distributed memory, through exchange of messages. The protocol is configurable for level of consistency (strict, weak). We implemented the system and ran NAS Parallel Benchmark (NPB) extensively, and found the performance numbers are better than the native execution in some cases.Figure 1 shows the comparison between native Xeon Phi vs popcorn performance.

For Xeon Xeon Phi setup DMA based copy approach out-performed RMA approach on all cases, due to high latency of PCIe bus, therefore we used the first technique for our evaluations. Evaluations are done with three benchmarks, Integer Sort(IS) Embarrassingly Parallel(EP) and Conjugate Gradient(CG). IS is random memory access , EP has no synchronizations and CG is irregular memory access with inter-thread communication. All of them stress the messaging and page-coherency protocols.

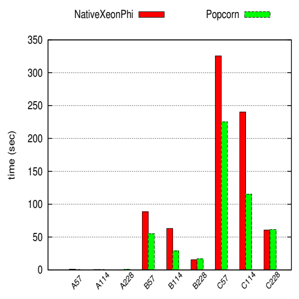
We intend to present our contributions in poster using graphic representation of the system architecture. We also plan to present further benchmark results that show how different solutions scales based on the application memory access pattern.



(a) NPB IS



(b) NPB EP



(c) NPB CG

Figure 1: Comparison between different NPB benchmarks on Xeon Phi native vs Popcorn on different number of threads(57,114,228) and data size(A,B,C)

References

[1] Andrew Baumann, Paul Barham, Pierre-Evariste Dagand, Tim Harris, Rebecca Isaacs, Simon Peter, Timothy Roscoe, Adrian Schpbach, and Akhilesh Singhania *The Multikernel: A new OS architecture for scalable multicore systems* SOSP 2009.

[2] Andrew Baumann, Chris Hawblitzel, Kornilios Kourtis,Tim Harris, Timothy Roscoe *Cosh: clear OS data sharing in an incoherent world* USENIX TRIOS 2014.

[3] Heck M.J.R, Hui-Wen Chen, A.W. Fang *Hybrid Silicon Photonics for Optical Interconnects* IEEE Journal of Selected Topics in Quantum Electronics, 2010

[4] Antonio Barbalace, Alastair Murray, Robert Lyerly and Binoy Ravindran *Towards Operating System Support for Heterogeneous-ISA Platforms* 4th Workshop on Systems for Future Multicore Architectures, 2014