Seamless POSIX/OpenMP Thread Migration Across ISA Boundaries

Robert Lyerly – Ph.D. Student, Advisor: Binoy Ravindran
Bradley Department of Electrical and Computer Engineering, Virginia Tech
{rlyerly, binoy}@vt.edu

1 Problem and Motivation

The proliferation of heterogeneous processors has spurred new interest in system software design [19, 16, 4]. Recently, several works have proposed thread migration between heterogeneous instruction set architecture (ISA) CPUs to obtain better performance and energy efficiency [8, 4, 20, 3]. These systems allow developers to write compiled shared-memory POSIX applications and let the system deal with inter-ISA migration details. Currently, they only migrate threads across ISAs at equivalence points [21]. This forces the scheduler to wait long periods for the to-be-migrated thread to reach an equivalence point. Figure 1 shows a histogram of the number of instructions between consecutive migration points in NPB's [2] CG benchmark. This bi-modal distribution is common in many applications – there are plenty of migration opportunities when calling short functions, but for long functions the scheduler must wait a significant number of instructions before migrating the thread. This prevents the scheduler from quickly adapting execution of workloads, which is especially important as compiled applications are increasingly co-located in datacenters where the scheduler must meet strict performance goals.

Additionally, none of these systems simultaneously exploit compute resources across multiple processors for a single application. Figure 2 shows speedup for PARSEC's [5] vips benchmark using different numbers of threads on an 8-core hyperthreaded Intel Xeon and a 96-core Cavium ThunderX versus sequential execution on the Xeon. The benchmark scales with thread counts, meaning it could potentially gain performance by distributing work across both machines simultaneously. However, it is not clear how to balance the work distribution in consideration of architectural characteristics and system software overheads.

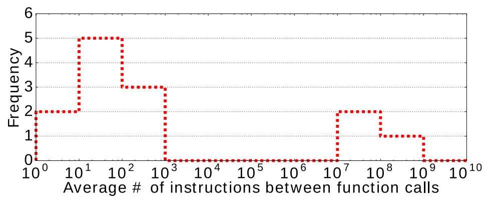

Figure 1: Histogram of number of instructions between migration points in CG, class A.

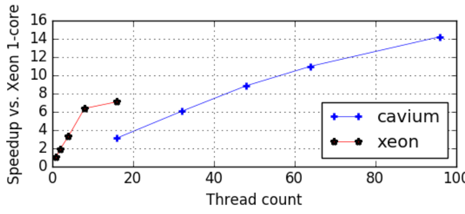

Figure 2: Speedup of vips on an 8C/16T Intel Xeon and a 96C Cavium ThunderX versus running sequentially on Xeon.

These limitations must be overcome using a combination of new compiler and runtime techniques to better leverage heterogeneous-ISA systems. This would give the system better control in adapting workloads to meet performance goals, and for boosting the performance of highly parallel applications.

2 Background and Related Work

Previous works studied process migration in heterogeneous-ISA platforms using state transformation techniques [1, 18, 22], but incur extensive overheads as they must translate the entire virtual address space between ISA-specific formats. More recently, Popcorn Linux implemented thread migration in heterogeneous-ISA systems with minimal state transformation [4, 3]. However, Popcorn Linux can only migrate threads at equivalence points, and does not run threads of a single application simultaneously across multiple processors. DeVuyst et al. [8] and Venkat and Tullsen [20] similarly implement single-threaded process migration in a simulated heterogeneous-ISA chip multiprocessor. They migrate threads outside of equivalence points using a complex emulation environment, which can cause overheads from warming the code cahce and contention on emulator data structures [7].

3 Approach and Uniqueness

In order to migrate across ISAs, a thread's registers and stack must be transformed between ISA-specific formats at equivalence points. We propose a lightweight mechanism for migrating application threads at arbitrary locations using hardware transactional memory (HTM) [14]. HTM uses the CPU's cache subsystem to execute developer-specified transactions, i.e., code which executes atomically as seen by other threads. The transaction executes like normal code, except all results are buffered in hardware. These results are either committed to the cache at the end of a transaction or rolled back when a conflict is detected (e.g., two threads write to the same memory address). HTM's checkpointing can be utilized for migrating threads at arbitrary program locations. At equivalence points, threads start a transaction. When the thread reaches the next equivalence point, it commits the results to memory and starts a new transaction. However if the scheduler requests migration during the transaction, the thread is rolled back to the previous equivalence point for state transformation between ISA-specific formats. This allows the scheduler to quickly migrate threads without expensive ISA emulation. Challenges arise when dealing with the limitations of the HTM system, e.g., dealing with capacity aborts (buffer capacities are reached).

Work-splitting across multiple architectures has been shown to achieve large performance gains in CPU/GPU systems [13, 11, 17]. However, these systems have programmability limitations (low-level compute languages, no I/O functionality) and limited ability to automatically distribute work across processors (developer-guided data partitioning). Heterogeneous-ISA CPU platforms alleviate these issues while still benefiting from architectural diversity. Using existing thread migration techniques, threads could be forked and migrated across machines for parallel computation. As threads perform work, an OS mechanism like Popcorn's page migration service would migrate data on-demand for the computation. Thus, developers could transparently scale parallel computation across several devices. The system must intelligently load-balance the distribution of work, e.g., dynamically grabbing batches of computation from a work pool (like OpenMP's [15] dynamic scheduler) or statically partitioning work based on compute characteristics (memory access patterns, scalability). The system must also take into account software overheads, e.g., page migration costs and page contention across separate machines.

4 Results and Contributions

Figure 3 shows the minimum, average and maximum state transformation latencies for several NPB applications using our custom runtime. Latencies are significantly smaller than a typical scheduling quantum (16.7ms), leading to low thread migration overheads compared to PadMig [9], a Java-based thread migration framework. Figure 4 shows that when running IS (sorting on x86, verification on ARM), Popcorn Linux is twice as fast. The language VM's data serialization costs are significant compared to our state transformation runtime.

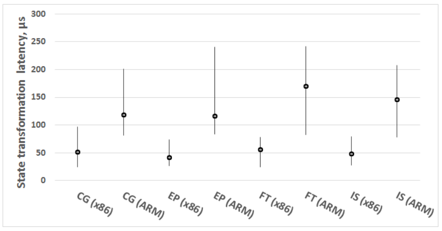

Figure 3: Minimum, average and maximum state transformation latency for several NPB benchmarks on Xeon and Cavium processors.

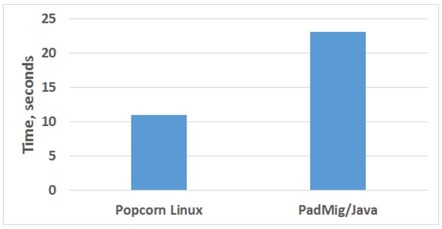

Figure 4: Time to run IS on x86 and verify on ARM using Popcorn Linux and PadMig

While our runtime efficiently transforms state, there are still several limitations which prevent these systems from fully utilizing heterogeneous platforms. Faster migration response times and cross-ISA work distribution would allow users to transparently and effectively utilize emerging heterogeneous-ISA platforms for compiled shared-memory POSIX/OpenMP applications.

References

[1] G. Attardi, A. Baldi, U. Boni, F. Carignani, G. Cozzi, A. Pelligrini, E. Durocher, I. Filotti, W. Qing, M. Hunter, et al. Techniques for dynamic software migration. In *Proceedings of the 5th Annual ESPRIT Conference (ESPRIT'88)*, volume 1. Citeseer, 1988.

[2] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, et al. The nas parallel benchmarks. *The International Journal of Supercomputing Applications*, 5(3):63–73, 1991.

[3] A. Barbalace, R. Lyerly, C. Jelesnianski, A. Carno, H.-R. Chuang, V. Legout, and B. Ravindran. Breaking the Boundaries in Heterogeneous-ISA Datacenters. In *Proceedings of the Twenty Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XXII, 2017. To Appear.

[4] A. Barbalace, M. Sadini, S. Ansary, C. Jelesnianski, A. Ravichandran, C. Kendir, A. Murray, and B. Ravindran. Popcorn: Bridging the Programmability Gap in heterogeneous-ISA Platforms. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, pages 29:1–29:16, New York, NY, USA, 2015. ACM.

[5] C. Bienia. *Benchmarking Modern Multiprocessors.* PhD thesis, Princeton University, January 2011.

[6] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti. Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 301–314, New York, NY, USA, 2011. ACM.

[7] E. G. Cota, P. Bonzini, A. Bennée, and L. P. Carloni. Cross-isa machine emulation for multicores. In *Proceedings of the 2017 International Symposium on Code Generation and Optimization*, CGO '17, pages 210–220, Piscataway, NJ, USA, 2017. IEEE Press.

[8] M. DeVuyst, A. Venkat, and D. M. Tullsen. Execution migration in a heterogeneous-isa chip multiprocessor. In *Proceedings of the Seventeenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XVII, pages 261–272, New York, NY, USA, 2012. ACM.

[9] J. Gehweiler and M. Thies. Thread migration and checkpointing in java. *Heinz Nixdorf Institute, Tech. Rep. tr-ri-10-315*, 2010.

[10] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen. Comet: Code offload by migrating execution transparently. In *Proceedings of the 10th USENIX Conference on Operating Systems Design and Implementation*, OSDI'12, pages 93–106, Berkeley, CA, USA, 2012. USENIX Association.

[11] D. Grewe and M. F. P. O'Boyle. A static task partitioning approach for heterogeneous systems using opencl. In *Proceedings of the 20th International Conference on Compiler Construction: Part of the Joint European Conferences on Theory and Practice of Software*, CC'11/ETAPS'11, pages 286–305, Berlin, Heidelberg, 2011. Springer-Verlag.

[12] Khronos OpenCL Working Group. The OpenCL Specification, March 2016. `https://www.khronos.org/registry/OpenCL/specs/opencl-2.2.pdf`.

[13] C.-K. Luk, S. Hong, and H. Kim. Qilin: Exploiting parallelism on heterogeneous multiprocessors with adaptive mapping. In *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 45–55, New York, NY, USA, 2009. ACM.

[14] T. Nakaike, R. Odaira, M. Gaudet, M. M. Michael, and H. Tomari. Quantitative comparison of hardware transactional memory for blue gene/q, zenterprise ec12, intel core, and power8. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, pages 144–157, New York, NY, USA, 2015. ACM.

[15] OpenMP Architecture Review Board. OpenMP Applicatoin Programming Interface, November 2015. `http://www.openmp.org/wp-content/uploads/openmp-4.5.pdf`.

[16] P. Rogers. Heterogeneous system architecture overview. In *2013 IEEE Hot Chips 25 Symposium (HCS)*, pages 1–41, Aug 2013.

[17] T. R. W. Scogland, W. C. Feng, B. Rountree, and B. R. de Supinski. Coretsar: Core task-size adapting runtime. *IEEE Transactions on Parallel and Distributed Systems*, 26(11):2970–2983, Nov 2015.

[18] P. Smith and N. C. Hutchinson. Heterogeneous process migration: The Tui system. *Software-Practice and Experience*, 28(6):611–640, 1998.

[19] H. Sutter. Welcome to the jungle. *URL http://herbsutter. com/welcome-to-the-jungle*, 2011.

[20] A. Venkat and D. M. Tullsen. Harnessing isa diversity: Design of a heterogeneous-isa chip multiprocessor. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, pages 121–132, Piscataway, NJ, USA, 2014. IEEE Press.

[21] D. G. von Bank, C. M. Shub, and R. W. Sebesta. A unified model of pointwise equivalence of procedural computations. *ACM Trans. Program. Lang. Syst.*, 16(6):1842–1874, Nov. 1994.

[22] M. V. Yalamanchili and R. M. Hyatt. Heterogeneous process migration: issues and an approach. In *Proceedings of the 35th Annual Southeast Regional Conference*, pages 275–281. ACM, 1997.