metric or heterogeneous processors [19, 30, 35, 48, 49]. However, many of these systems focus on power efficiency, so they seek to identify which threads gain the most efficiency. Chameleon could use these techniques to identify which threads would perform best on a powerful execution object. Mogul et al. investigate the use of ACMPs for operating systems [42], but focus on static assignment of OS threads to simple cores; Chameleon could implement this policy using its property mechanism. Most similar to Chameleon is Luo et al.'s work on the use of helper threads and cache resizing on an ACMP [38]. Similar to Chameleon, this work determines when to allocate resources to speed sequential threads, but focuses again on identifying which threads gain the most benefit. ACS accelerates critical sections on an ACMP [55]. However, the latency of Chameleon's reconfiguration is still too long to help individual critical sections.

Gang scheduling. Chameleon's cluster scheduling is similar to gang scheduling in that it will try to schedule multiple CPUs simultaneously. Past work on gang scheduling [45, 27, 16, 6] focuses primarily on time-sharing gangs of threads. While Chameleon can do this, there is little benefit because performance improves by running time-sliced threads in parallel on separate cores. Thus, Chameleon is most effective when there are idle threads to be used opportunistically, or when a sequential thread has higher priority than competing workloads.

Tesselation uses a form of gang scheduling to provide a *cell* of processors to an application, which is similar to Chameleon's execution objects [11]. However, it provides cores for software process to use, while Chameleon provides cores to hardware for an enhanced CPU.

Support for reconfigurable hardware. Recent work on scheduling for reconfigurable hardware has largely focused on embedded and real-time systems [52, 29, 17]. In these environments, precise models of the transition costs and the execution time of code on different hardware are needed. These systems also place mandatory requirements on scheduling, so flexible tradeoffs like Chameleon's taxation are not used. In contrast, Chameleon focuses largely on best-effort workloads and must rely on admission control to meet performance goals.

Several projects discuss OS support for introducing reconfigurable logic onto a processor [13, 36, 54]. However, OS support for these systems focuses on efficiently allocating the reconfigurable logic to specific functions rather than on thread scheduling.

Windows 7's support for core parking [41], which coalesces threads onto a single core to disable the remaining cores, is similar to Chameleon's scheduling of threads on the execution object. It is also used to balance threads between hyperthreads. However, core parking targets all threads at a specific subset of CPUs, rather than context switching between configurations.

7. Conclusions

Dynamic processors will lead to new opportunities for improving performance, reliability, and power consumption by reconfiguring the set of running processors. Existing operating systems cannot react to changes fast enough to fully utilize reconfiguration, and do not have scheduling mechanisms to take advantage of them. Chameleon extends Linux to enable rapid reconfiguration through processors proxies, allowing use of reconfiguration even for short periods. It abstracts the reconfiguration abilities of the hardware with execution objects and nodes, which expose the new capabilities of the hardware to programmers and the scheduler. Chameleon's cluster scheduling with taxation allows sequential code to use idle cores and provides a flexible tradeoff between single-thread performance and parallel performance.

We plan to extend Chameleon for other uses of dynamic processors. Chameleon focuses on performance benefits of dynamic processors, but they should also promise power efficiency and reliability, which demand different scheduling policies. In addition, we plan to investigate other forms of dynamic processors that may not fit Chameleon's model. For example, processors with dark silicon accelerators cannot be used unless other cores are powered off.

Acknowledgements

This work is supported in part by National Science Foundation (NSF) grant CNS-0834473. We would like to thank our shepherd, Angela Demke Brown, and the anonymous reviewers for their invaluable feedback. Swift has a significant financial interest in Microsoft.

References

[1] N. Aggarwal, P. Ranganathan, N. P. Jouppi, and J. E. Smith. Configurable isolation: Building high availability systems with commodity multi-core processors. In *Proc. of the 34th ISCA*, June 2007.

[2] J. Allarey, V. George, and S. Jahagirdar. Power management enhancements in the 45nm intel core microarchitecture. *Intel Technical Journal*, 12(3):169–178, oct 2008.

[3] M. Annavaram, E. Grochowski, and J. Shen. Mitigating amdahl's law through epi throttling. In *Proc. of the 32nd ISCA*, pages 298 – 309, June 2005.

[4] L. A. Barroso, K. Gharachorloo, R. McNamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith, R. Stets, and B. Verghese. Piranha: A scalable architecture based on single-chip multiprocessing. In *Proc. of the 27th ISCA*, pages 282–293, June 2000.

[5] A. Baumann, P. barham, P.-E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schupbach, and A. Singhania. The multikernel: A new os architecture for scalable multicore systems. In *Proc. 22nd SOSP*, Oct 2009.

[6] M. Bhadauria and S. A. McKee. An approach to resource-aware co-scheduling for cmps. In *Proc. of the 24th ICS*, pages 189–199, 2010.

[7] C. Bienia and K. Li. PARSEC 2.0: A new benchmark suite for chip-multiprocessors. In *Proc. 5th Workshop on Modeling, Benchmarking and Simulation*, June 2009.

[8] N. Brookwood. Amd fusion. family of apus: Enabling a superior, immersive pc experience. `http://sites.amd.com/us/Documents/48423B_fusion_whitepaper_WEB.pdf`, Mar 2010.

[9] J. Charles, P. Jassi, A. N. S, A. Sadat, and A. Fedorova. Evaluation of the Intel Core i7 Turbo Boost feature. In *Proc. International Symposium on Workload Characterization*, Oct. 2009.

[10] S. Chaudhry, R. Cypher, M. Ekman, M. Karlsson, A. Landin, S. Yip, H. Zeffer, and M. Tremblay. Simultaneous speculative threading: A novel pipeline architecture implemented in sun's rock processor. In *Proc. of the 36th ISCA*, June 2009.

[11] J. A. Colmenares, S. Bird, H. Cook, P. Pearce, D. Zhu, J. Shalf, S. Hofmeyr, K. Asanovic, and J. Kubiatowicz. Resource management in the tessellation manycore os. In *HotPAR*, 2010.

[12] J. Corbet. CFS group scheduling. `http://lwn.net/Articles/240474/`, 2007.

[13] M. Dales. Managing a reconfigurable processor in a general purpose workstation environment. In *Proc. Design, Automation and Test in Europe*, 2003.

[14] K. J. Duda and D. R. Cheriton. Borrowed-virtual-time (bvt) scheduling: supporting latency-sensitive threads in a general-purpose scheduler. In *Proc. 17th SOSP*, 1999.

[15] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger. Dark silicon and the end of multicore scaling. In *Proc. of the 38th ISCA*, June 2011.

[16] D. G. Feitelson and L. Rudolph. Gang scheduling performance benefits for fine-grain synchronization. *JPDC*, 16(4):306 – 318, 1992.

[17] W. Fu and K. Compton. Scheduling intervals for reconfigurable computing. In *Proc. 16th FCCM*, Apr. 2008.

[18] R. Grant and A. Afsahi. Power-performance efficiency of asymmetric multiprocessors for multi-threaded scientific applications. In *Proc. 20th IPDPS*, Apr. 2006.

[19] B. Hamidzadeh, Y. Atif, and D. J. Lilja. Dynamic scheduling techniques for heterogeneous computing systems. *Concurrency: Practice and Experience*, 7(7):633–652, 1995.

[20] L. Hammond, B. Hubbert, M. Siu, M. Prabhu, M. Chen, and K. Olukotun. The stanford hydra cmp. *IEEE Micro*, pages 71–84, March-April 2000.

[21] J. L. Henning. Spec cpu2006 benchmark descriptions. *Computer Architecture News*, 34(4):1–17, 2006.

[22] M. D. Hill and M. R. Marty. Amdahl's law in the multicore era. *IEEE Computer*, pages 33–38, July 2008.

[23] H. Hoffmann, J. Eastep, M. D. Santambrogio, J. E., Miller, and A. Agarwal. Application heartbeats: A generic interface for specifying program performance and goals in autonomous computing environments. In *Proc. 7th ICAC*, 2010.

[24] Intel Corp. 82093AA I/O ADVANCED PROGRAMMABLE INTERRUPT CONTROLLER (IOAPIC). `http://www.intel.com/design/chipsets/datashts/29056601.pdf`, 1996.

[25] Intel Corp. Thermal protection and monitoring features: A software perspective. `http://www.intel.com/cd/ids/developer/asmo-na/eng/downloads/54118.htm`, 2005.

[26] E. Ipek, M. Kirman, N. Kirman, and J. F. Martinez. Core fusion: Accomodating software diversity in chip multiprocessors. In *Proc. of the 34th ISCA*, June 2007.

[27] Y. Jiang, X. Shen, J. Chen, and R. Tripathi. Analysis and approximation of optimal co-scheduling on chip multiprocessors. In *Proc. of the 17th PACT*, 2008.

[28] M. T. Jones. Inside the linux 2.6 completely fair scheduler, Dec 2009. http://www.ibm.com/developerworks/linux/library/l-completely-fair-scheduler/.

[29] H. Kooti, E. Bozorgzadeh, S. Liao, and L. Bao. Transition-aware real-time task scheduling for reconfigurable embedded systems. In *Proc. Design, Automation and Test in Europe*, Mar. 2010.

[30] D. Koufaty, D. Reddy, and S. Hahn. Bias scheduling in heterogeneous multi-core architectures. In *Proc. EuroSys*, 2010.

[31] V. Krishnan and J. Torrellas. Hardware and software support for speculative execution of sequential binaries on a chip-multiprocessor. In *Proc. ICS*, pages 85–92, July 1998.

[32] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-isa heterogeneous multi-core architectures: The potential for processor power reduction. In *Proc. of the 36th MICRO*, Dec. 2003.

[33] M. Laux. Solaris processor sets made easy. `http://developers.sun.com/solaris/articles/solaris_processor.html`, 2001.

[34] C. Letavec and J. Ruggiero. The n-queens problem. *INFORMS Transactions on Education*, 2(3), 2002.

[35] T. Li, D. Baumberger, D. A. Koufaty, and S. Hahn. Efficient operating system scheduling for performance-asymmetric multi-core architectures. In *Proc. of SC2007*, Nov. 2007.

[36] E. Lübbers and M. Platzner. Reconos: Multithreaded programming for reconfigurable computers. *ACM Trans. Embed. Comput. Syst.*, 9, October 2009.

[37] P. Marcuello, A. Gonzalez, and J. Tubella. Speculative multithreaded processors. In *Proc. of the 1998 ICS*, pages 77–84, July 1998.

[38] Y. Luo, V. Packirisamy, W.-C. Hsu, and A. Zhai. Energy efficient speculative threads: dynamic thread allocation in same-isa heterogeneous multicore systems. In *Proc. 19th PACT*, pages 453–464, 2010.

[39] P. E. McKenney, J. Appavoo, A. Kleen, O. Krieger, , R. Russell, D. Sarma, and M. Soni. Read-copy update. In *Proc. of the Ottawa Linux Symposium*, July 2001.

[40] Microsoft Corp. New NUMA Support with Windows Server 2008 R2 and Windows 7. `http://archive.msdn.microsoft.com/64plusLP`, 2008.

[41] Microsoft Corp. Processor power management in windows 7 windows server 2008 r2. `http://download.microsoft.com/download/3/0/2/3027D574-C433-412A-A8B6-5E0A75D5B237/ProcPowerMgmtWin7.docx`, Jan. 2010.

[42] J. C. Mogul, J. Mudigonda, N. L. Binkert, P. Ranganathan, and V. Talwar. Using asymmetric single-isa cmps to save energy on operating systems. *IEEE Micro*, 28(3):26–41, 2008.

[43] Z. Mwaikambo, R. Russell, A. Raj, and J. Schopp. Linux kernel hotplug CPU support. In *Proc. of the Ottawa Linux Symposium*, pages 181–194, 2004.

[44] J. T. Oplinger, D. L. Heine, and M. S. Lam. In search of speculative thread-level parallelism. In *Proc. 8th PACT*, Oct. 1999.

[45] J. Ousterhout. Scheduling techniques for concurrent systems. In *Proc. of the 3rd ICDCS*, pages 22–30, 1982.

[46] R. Raman, M. Livny, and M. Solomon. Matchmaking: distributed resource management for high throughput computing. In *Proc. HPDC*, pages 140 –146, July 1998.

[47] S. K. Reinhardt and S. S. Mukherjee. Transient fault detection via simultaneous multithreading. In *Proc. of the 27th ISCA*, pages 25–36, June 2000.

[48] J. C. Saez, M. Prieto, A. Fedorova, and S. Blagodurov. A comprehensive scheduler for asymmetric multicore systems. In *Proc. EuroSys*, 2010.

[49] J. C. Saez, D. Shelepov, A. Fedorova, and M. Prieto. Leveraging workload diversity through os scheduling to maximize performance on single-isa heterogeneous multicore systems. *J. Parallel Distrib. Comput.*, 71:114–131, January 2011.

[50] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, and C. Moore. Exploiting ilp, tlp, and dlp with the polymorphous trips architecture. In *Proc. of the 30th ISCA*, pages 422–433, June 2003.

[51] S. Siddha and V. Pallipadi. Chip multi processing aware Linux kernel scheduler. In *Proc. of the Ottawa Linux Symposium*, pages 337–348, 2006.

[52] S. P. Smith. Dynamic scheduling and resource management in heterogeneous computing environments with reconfigurable hardware. In *International Conference on Computer Design*, 2006.

[53] W. Stanek. Windows Server 2008 R2: A primer. `http://technet.microsoft.com/en-us/magazine/ee677582.aspx`, Nov. 2009.

[54] C. Steiger, H. Walder, and M. Platzner. Operating systems for reconfigurable embedded platforms: online scheduling of real-time tasks. *IEEE Trans. Computers*, 53(11), Nov. 2004.

[55] M. A. Suleman, O. Mutlu, M. K. Qureshi, and Y. N. Patt. Accelerating critical section execution with asymmetric multi-core architectures. In *Proc. 14th ASPLOS*, 2009.

[56] Texas Instruments. OMAP 5 platform. `http://www.ti.com/ww/en/omap/omap5/omap5-OMAP5430.html`, 2011.

[57] P. Wells, K. Chakraborty, and G. Sohi. Mixed-mode multicore reliability. In *Proc. of the 14th ASPLOS*, Mar. 2009.

[58] X. Zhang, S. Dwarkadas, and K. Shen. Hardware execution throttling for multi-core resource management. In *Proc. USENIX ATC*, 2009.