

Dynamic Program Stirring on Multiple Cores: How Hardware Performance Monitors Can Help Regulate Performance, Power, and Temperature Simultaneously

Matthew Curtis-Maury and Dimitrios S. Nikolopoulos
Center for High End Computing Systems
Department of Computer Science, Virginia Tech
{mfcurt,dsn}@cs.vt.edu

Christos D. Antonopoulos
Department of Computer Science
College of William and Mary
cda@cs.wm.edu

The utility of hardware performance monitors stems from the insight they provide into the interaction between software and hardware. It is this interaction that determines many interesting properties of an application's execution. Among these properties, of primary interest to our research are the IPC, as it determines the performance of an application; the instantaneous power consumption; and the temperature of various processor resources. In particular, we are interested in how these properties scale with the number, configuration, and frequency scale of execution cores on multicore platforms. Previous work has shown the potential for accurate prediction of power [3], performance and scalability on multiple cores [2], and temperature [1] using HPMs.

We have focused on making predictions of the performance of a parallel application on a varying number of processors, processors cores, and core-level hardware threads. We exploit the iterative nature of parallel programs by recording event counts during an initial iteration of each program phase on a specific hardware configuration. We then make predictions by multiplying the event counts by coefficients that have been derived offline using regression. Two hurdles to accurate prediction are finding an effective performance model and selecting the correct sets of events. We have found effective solutions to these issues that allow for prediction accuracy approaching 90%.

Our method of performance prediction allows an application to have access to predictions early at runtime, so an application can adapt itself to use the optimal configuration for the remainder of execution. Adapting to use fewer processors, or *concurrency throttling*, has the potential to simultaneously reduce both execution time, by alleviating scalability bottlenecks, and energy consumption, by reducing the number of processors actively drawing current. We have experimented with this technique, optimizing for performance as well as power-performance, and we have seen improvements of up to 22% for execution time and 26% for energy consumption compared to using all available execution contexts.

In our ongoing work, we are considering extensions to our prediction and adaptation model to characterize program phases for DVFS and respond to thermal emergencies in processor cores. Our goal is to use the complete arsenal of actuators (DVFS, concurrency control, thread placement) for simultaneous PPT (Power-Performance-Thermal) control of the microprocessor and to maximize power savings and system reliability under rigid performance guarantees. To this end we attempt to derive performance, power, and thermal predictions for each program phase in response to adjustments in each of the actuators by sampling event counters during the same phase while running on a few strategically selected hardware configurations. To maximize the impact of our policies, we need to consider variable program granularity for using our actuators. While

concurrency throttling and thread-level scheduling and migration apply naturally to parallel loops and threaded function calls, DVFS for power control can operate at finer or coarser granularity, while DVFS for thermal control needs to extend across multiple phases as temperature is slow to change. Since each prediction can best be made using different sets of events, there may be many sets of events being recorded at any time, and they may be changed on different timescales.

The need for multidimensional and multi-grained event-based prediction and adaptation is our challenge statement for HPMs. Requiring multiple independent sets of events sampled during overlapping intervals at runtime presents a problem for current HPMs as it requires a potentially large number of HPM registers and it may suffer due to co-recording limitations that exist on some systems. Architectural support for multiple *contexts* of HPMs in the same thread would be helpful if each context could operate independently of the others. In this case the events for each prediction could be assigned to their own contexts and would not have to consider what else was being recorded. Ideally, each context could be configured, started, collected, and stopped without affecting the other contexts. Another possible way to overcome these limitations is to perform multiplexing of the HPM registers. Further, if multiplexing could be handled by the hardware, then the overhead would be reduced. This solution might be a potential way to get more functionality out of a limited die area. In the absence of enhanced HPM functionality, multidimensional program prediction and adaptation become challenging and time-consuming software problems that require advanced statistical and machine learning methods to converge to viable solutions.

References

- [1] S. W. Chung and K. Skadron. Using on-chip event counters for high-resolution, real-time temperature measurements. In *Proc. of IThERM*, June 2006.
- [2] M. Curtis-Maury, J. Dzierwa, C. Antonopoulos, and D. Nikolopoulos. Online Power-Performance Adaptation of Multithreaded Programs using Hardware Event-Based Prediction. In *Proc. of the 20th International Conference on Supercomputing*, June 2006.
- [3] C. Isci and M. Martonosi. Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data. In *Proc. of the 26th ACM/IEEE Annual International Symposium on Microarchitecture*, pages 93–104, November 2003.