Thermal-Aware Scheduling for Real-Time Applications in Embedded Systems

Adam Lewis, Soumik Ghosh, and N.-F. Tzeng

Center for Advanced Computer Studies,University of Louisiana, Lafayette, Louisiana 70504
{awlewis,sxg5317,tzeng}@cacs.louisiana.edu

Introduction

The emergence of multi-core and many-core architectures in embedded systems has increased the energy and thermal challenges faced by such systems. As the technology for microprocessors moves toward the nanometer scale, power density becomes one of the major constraints to the performance improvement of embedded processors. Real-time embedded signal processing tasks on multi-core systems have a high potential to thermally stress the die and negatively impact processor performance. This potential problem is crudely regulated by Dynamic Thermal Management (DTM) in commercial processors. DTM is typically realized by gauging the chip temperature at run-time to enable dynamical adjustment of the processor voltage and frequency (DVFS); it slows down the processor upon die overheat. However, real-time embedded systems often cannot tolerate the implied performance degradation entailed in DVFS techniques should any job fail to complete within its deadline as a result of processor slowdown. Consequently, DTM alone cannot avoid real-time application failures completely in the event of die overheat.

This work investigates operating-system assisted, hardware performance counter-based techniques for thermal control in embedded systems running real-time applications. Such a technique schedules the processor workload in a way that reduces the thermal adverse impacts as much as possible. It makes use of thread adjustment scheduling on multi-core processors to ensure active tasks observing their real-time deadlines with lowered performance overhead resulting from fewer DTM events.

Thermal-aware scheduling relies on a mechanism to obtain energy consumption and thermal loading on a system. Our scheduling technique derives energy consumption information from a model based upon bus traffic and subsystem transactional traffic [3]. The model needs to keep track of several key micro-architectural events within the processor and its associated peripherals. It includes a bus-based energy model, which is dictated by HyperTransport/FSB transactions amongst cores, caches, memory, and peripherals, given that such transactions capture real-time system activities and consequently, energy consumption. Our scheduler employs this model and associated hardware performance counters to regulate the thermal load of real-time applications on embedded systems. It is being implemented as an extention to the Power-Aware Dispatcher (PAD) [6] (to be introduced in OpenSolaris [7]).

Multiplexing Performance Counters

Modern processor cores support between 30 to 500 PeCs (depending upon the processor) but only permit 2 to 18 of these counters to be read at one time (again depending upon processor). Furthermore, certain combinations of PeCs may not be permitted to be collected at the same time [5].

Using PeCs to gain insight into architectural events (in our case bus transactions)is complicated by the fact that the types of events, number of events, and use cases varies widely, not only across architectures, but across systems sharing the same Instruction Set Architecture (ISA). For instance, the Intel and AMD implementations of performance counters have very little in common in spite of the processor families using the same ISA [6][1].

As an example, to derive the thermal impact of application on cache, the scheduler needs to collect nine PeCs to calculate shared and unshared cache activity on the AMD Opteron processor: $RetiredInstructions$, $DCAccesses$, $DCRefillsL2$, $DCRefillsFromSystem$, $ICFetches$, $ICrefillsFromL2$, $ICRefillsFromSystem$, $L2RequestsTLB$, and $L2MissesTLB$. The model must calculate the L2 miss rate and L2 miss ratio using the following set of formulas with the PeCs as inputs:

$$L2MissRate = L2misses/RetiredInstructions$$
$$L2MissRatio = L2misses/L2Requests$$

These measures indicate cache activity which our model uses as an estimator for energy consumption and potential for a DTM event.

Note that we have to collect nine different PeCs even though the processor can only collect four of these counters at a time. For real-time situations, time multiplexing is the most popular solution to this problem [2]. In this approach the performance counters are reconfigured for different sets of counter events at regular time intervals. However, time multiplexing introduces issues with reconfiguration overhead and time alignment of samples.

Scheduler Design

The foundation of our scheduler is the model for run-time energy consumption detailed in [3]. This model provides a system-wide view of the energy consumption by making use of hardware performance counters to relate system power consumption to its overall thermal envelope.

The scheduler architecture is shown in Figure 1. The design intent is for our scheduler to be implemented as an extension of the existing power management infrastructure in

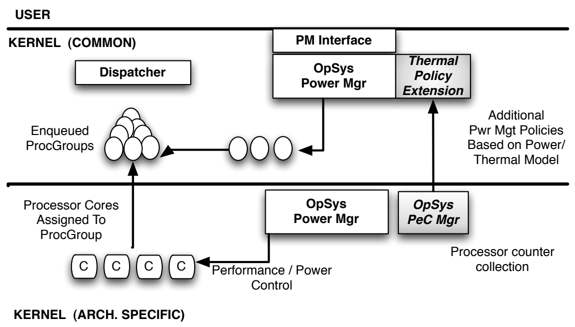

Figure 1. Thermal Enhancements To Power-Aware Dispatcher

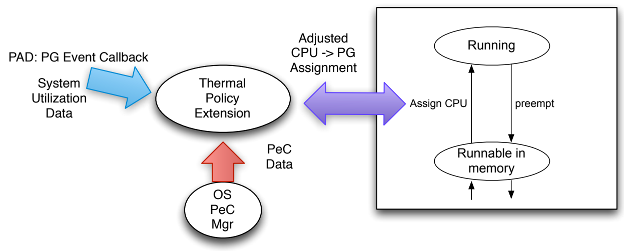

Figure 3. Thermal policy mgr/Dispatcher interaction.

the operating system. The design is being prototyped as an enhancement to the Power-Aware Dispatcher (PAD) [7] to be introduced in the OpenSolaris operating system [4]. A

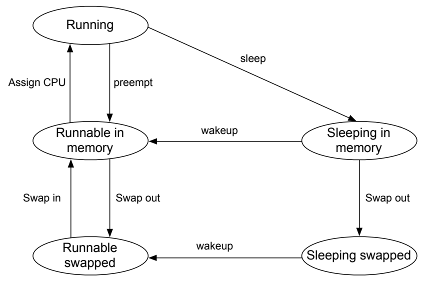

Figure 2. Solaris Dispatcher State diagram.

thread can be in any of the states indicated in Figure 2. Our enhancement to the PAD deals with the transition of state from being “Runnable in memory” to “Running”. The dispatcher decides which processor group (a collection of one or more processing cores) in which to assign the thread. The thread is assigned to a core in this group.

The Operating System Power Manager is responsbile to keeping track of the power events. A Thermal Policy Manager is added to the Power Manager to keep observe the power and thermal state of the system. The relationship between the Thermal Policy Manager and the Dispatcher is shown in Figure 2 and Figure 3. A data collection extension is added to the platform specific code in the OpenSolaris kernel that utilizes the Solaris libcpc library to collect the PeC data required by our model. The Thermal Policy Manager combines this information with the utilization information provided by the Dispatcher the overall thermal change in the next cycle and adjust the schedule of which processor group allowed access to the run queue in the next cycle.

Concluding Remarks

The OpenSolaris prototype of our scheduler proves the feasibility of using a model that uses bus traffic and subsystem transactional traffic to regulate thermal load of applications. We shall present performance measurements for a number of multiprocessor benchmarks focused on real-time high performance workloads and key general purpose benchmarks such as SPEC CPU2006 and SPECpower benchmarks.

References

[1] AMD. *Software Optimization Guide for AMD Family 10h Processors*, 3.06 08 edition, September 2005.

[2] R. Azimi, M. Stumm, and R. W. Wisniewski. Online performance analysis by statistical sampling of microprocessor performance counters. In *ICS ’05: Proceedings of the 19th annual international conference on Supercomputing*, pages 101–110, New York, NY, USA, 2005. ACM.

[3] A. Lewis, S. Ghosh, and N.-F. Tzeng. Run-time energy consumption estimation based on workload in server systems. In *Proc. of the 2008 Workshop on Power Aware Computing and Systems (Hotpower’08)*, 2008.

[4] R. McDougall and J. Mauro. *Solaris Internals: Solaris 10 and OpenSolaris Kernel Architecture*. Prentice Hall, 2nd edition, 2007.

[5] T. Mytkowicz, P. F. Sweeney, M. Hauswirth, and A. Diwan. Time interpolation: So many metrics, so few registers. In *MICRO ’07: Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 286–300, Washington, DC, USA, 2007. IEEE Computer Society.

[6] R. Singhal. Inside Intel Next Generation Nehalem Microarchitecture. In *Intel Developer Forum, Shanghai, China*, 2008.

[7] Sun Microsystems. OpenSolaris CPU Power Management - Project Tesla, June 2009.