PRiME
run smarter, live longer

Research Brief

Power-efficient, Reliable, Many-core Embedded systems www.prime-project.org

Scalable Performance Optimization for Power-constrained Many-Core Systems

PRiME *is a five year EPSRC funded research programme (2013-2018), in which four UK universities address the challenges of* ***power consumption*** *and* ***reliability*** *of future high-performance embedded systems utilising many-core processors.*

Many-Core Systems

Technology scaling has led to the use of many interconnected cores on a single chip for current and future generations of computing systems. This has enabled unprecedented levels of computing performance, with application parallelization and architectural support.

However, these higher levels of integration, together with increased operating frequencies, have led to exponential increases in power consumption and power density. Hence, limiting these uncontrolled increases in power consumption in a scalable manner is a key challenge for many-core applications.

Scalable Power Efficiency

PRiME addresses this challenge through a **power-efficient and scalable runtime framework**. Power efficiency is achieved through constraining the power budget for each application and optimizing performance within these constraints. Scalability is achieved through modular agent-based runtime controls.

System Description

Figure 1 shows a block diagram of the PRiME runtime framework which addresses the scalable power-efficiency challenge. Fundamental to this framework is an application power budget constraint - specified through an application programming interface (API).

For a given power budget, the application performance is optimized through two sets of runtime agents:

- thread agents (TAs)
- core agents (CAs)

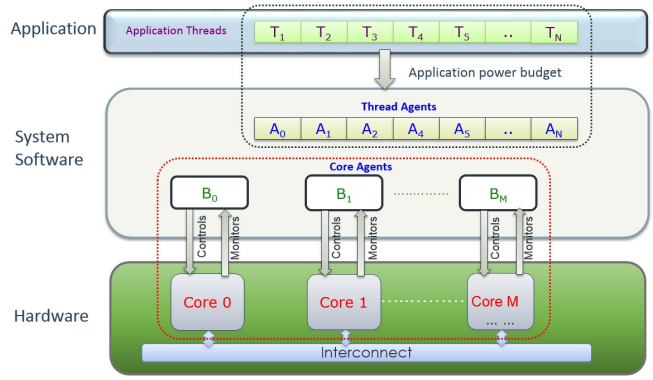

Figure 1. Power constrained & scalable runtime framework

The TAs, generated automatically from the application threads, distribute the overall power budget per thread using **economic trading principles**. The CAs are implemented as power governors per core and optimize the Dynamic Voltage & Frequency Scaling (DVFS) controls depending on the TA power budgets.

As a result of these modular agent-based controls, the framework achieves fast converging performance optimization with small runtime overheads, ensuring scalability for the runtime management of many cores.

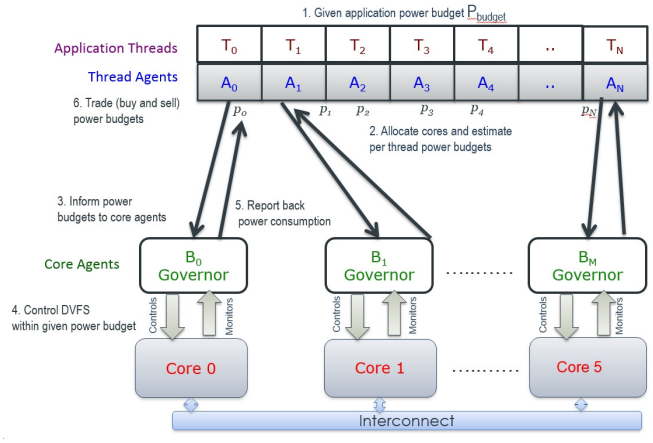

Figure 2. Key steps in the runtime framework

Figure 2 shows the key steps in the runtime framework. Initially a power budget (P_{budget}) is specified by the application (*step 1*). Based on this power budget, the number of cores to be allocated for the application is determined using a **runtime power/performance model**. The model uses regression based learning and prediction of the best

number of cores for a given power budget. During this time, a lightweight management thread agent (TA) is generated with each application thread (having similar core affinity/mapping). Each TA initially estimates the per thread power budget depending on its workload (*step 2*) and the overall power budget. Such per thread power budget is then communicated to the core agent (CA) (*step 3*). Each CA controls its DVFS based on this power budget (*step 4*). The actual power consumption from each core is measured/estimated and is communicated back to the corresponding TA (*step 5*).

At this point, the TAs can carry out **economic trading** between themselves to ensure that an optimized distribution of the overall power budget is achieved (*step 6*). E.g., when the measured/estimated power consumption is less than the TA power budget, the excess power budget is sold (i.e. traded out) to another TA with less power budget than required. Similarly, when the measured/estimated power consumption is higher than the TA power budget, the required extra power budget is bought (i.e. traded in) from a TA with more power budget than required. These steps (steps 2-6) are repeated per decision frame continuously until the performance optimization is achieved within the given power budget (P_{budget}).

Results

Fig 3 shows the experimental setup with application (PARSEC benchmarks), system software (TAs implemented through *pThread*-based runtime library and CAs implemented as Linux power governors) and hardware (12 Intel cores with 9 DVFS options each).

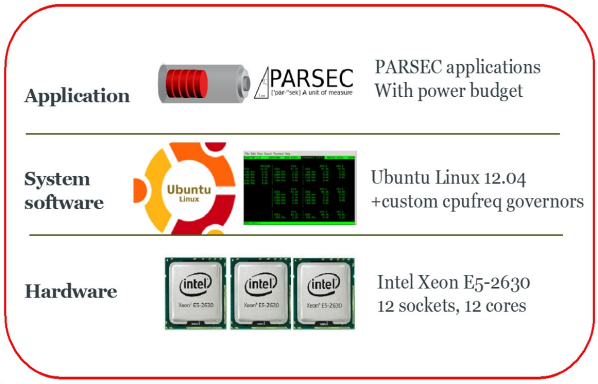

Figure 3. Experimental Set-up

Fig. 4 shows the results of a Raytrace application used as a case study to evaluate the runtime framework.

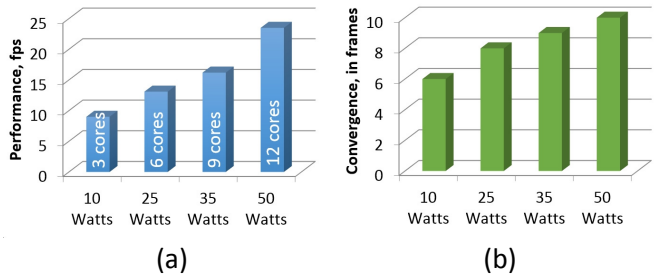

Figure 4: (a) Performance of the power budget constrained Raytrace application using PRiME's runtime approach (with the number of allocated cores). Power budgets are shown in the X-axis and the corresponding performances (in fps) are shown in the Y-axis, and (b) convergence time (in frames) of the runtime with increasing power budgets

The PRiME runtime achieves higher performance with increased power budgets. This is because with higher power budgets more cores are allocated for the parallel application with optimized DVFS controls. The performance is comparable to that of a linux *ondemand* governor running with the same allocated cores. However, in practice the *ondemand* governor cannot constrain core allocations and hence causes uncontrolled power increase. The PRiME runtime also benefits from excellent convergence (in frames) even with large number of cores. For example, for 12 allocated cores with 9 DVFS options (typically 9^{12} optimization space for global optimization) the optimization is carried out within 10 frames, meaning that the technique is eminently scalable.

Future Work

Future work will focus on further development and improvement of the power/performance model.

More information

Visit the PRiME programme web site, including the opportunity to sign-up for programme updates, or contact the PRiME Impact Manager:

Gerry Scott
Email: gerry.scott@soton.ac.uk
Tel: +44 (0)23 8059 2749

www.prime-project.org