ParaDIME: Parallel Distributed Infrastructure for Minimization of Energy

Oscar Palomar, Gulay Yalcin, Santhosh Kumar Rethinagiri, Adrian Cristal, Osman S. Unsal and Gina Alioto
Barcelona Supercomputing Center, Barcelona Spain
E-mail: Firstname.Lastname@bsc.es

1. Introduction

The increasing power and energy consumption of modern computing devices are critical to technology minimization and the associated gains in performance and productivity. On the one hand, we expect technology scaling to finally come face-to-face with the problem of "dark silicon" (only segments of a chip can function concurrently due to power restrictions), which will push us to use devices with completely new characteristics. On the other hand, as core counts increase, the shared memory model based on cache coherence will severely limit code scalability and increase energy consumption. Therefore, to overcome these problems, we need new computing paradigms that are radically more energy efficient.

These problems of energy consumption scale beyond the computing node to the data center where consumption is independent of the computing load of the system [1]. The nodes keep state in memory and on local disk which means that they cannot be turned off even if the load is low. Moreover, making individual servers energy-proportional without any vital architectural changes. In ParaDIME, we will explore scheduling between the computing nodes by employing a mechanism for maintaining the state that allows switching off nodes when their load is low. At the same time, data centers have high power consumption even when they are idle due to the fact that CPU loads are often kept between 10%-30% [1] in order to be able to react to sudden peaks in load. In ParaDIME, we will employ a mechanism that raises the CPU load to 90% while at the same time adhering to Service Level Agreements (SLAs).

The high level objectives of the ParaDIME Project can be summarized as follows:

- Objective 1: To develop an energy-aware programming model driving an associated ecosystem, the ParaDIME Computing Node / Stack (applications, runtime and architecture) that combines energy efficient SW programming and HW design methodologies and utilizes new emerging devices at the limit of CMOS scaling to radically decrease energy consumption; to quantify the energy savings from employing these methodologies by running several real-world power-hungry applications to stress test this Computing Node / Stack.
- Objective 2: To build a reference Data Center (Infrastructure as a Service or IaaS) platform that incorporates new energy conscious workload scheduling techniques utilizing information from the runtime to radically decrease energy consumption; to quantify the energy savings from employing these techniques by running several real-world power-hungry applications to stress test this Data Center platform.

2. Project flow

ParaDIME stands for Parallel Distributed Infrastructure for Minimization of Energy. As the name states, this project is defined for minimization and optimization of energy consumption for the data center. The Figure 1 presents a global view of the project which is based on several energy minimization methodologies. In the Figure 1, there are two computing nodes are shown, First node is the prototype developed with the simulator and second node is the real hardware. The various energy efficient methodologies, which are proposed in this project at different level are given below:

- Programming model level
  - Message passing
  - Error detection/recovery
  - Approximate computing
- Runtime level
  - Energy aware scheduling
- Architectural level
  - Approximate computing
  - Task specific accelerators
  - Operation below safe Vdd
- Device level
  - Emerging device

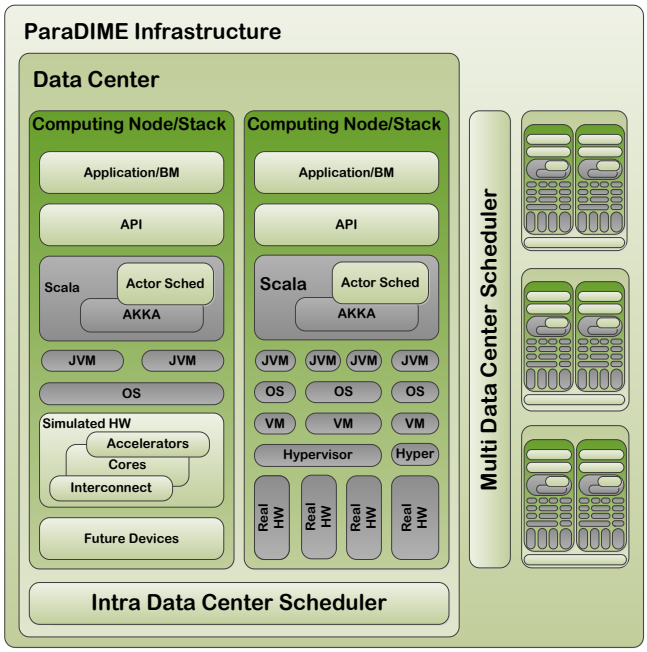

Figure 1: The ParaDIME Infrastructure

In this paper, we will discuss about the work in progress of combining different error detection mechanism and transactional memory for energy efficient computing below safe operation voltage [1] which is one of the several crucial methodologies described in the project to minimize the energy. By decreasing the voltage, the occurrence of failures increases drastically and without mechanisms for reliability, the systems would not operate any more [1]. For reliability we need (1) error detection and (2) error recovery mechanisms. According to our preliminary results, using reliability schemes combined with transactional memory for error recovery gains energy by 54 % while providing a reliability level of 100 %.

3. Operation below Safe Vdd

In our previous studies, we extended the gem5 simulator in order to implement FaulTM, a replication-based error detection scheme combined with an error recovery mechanism based on transactional memory [2]. In the ParaDIME project, we started with this FaulTM-simulator which has the capability of simulating both sequential and parallel applications. It measures the error detection overhead, the error recovery overhead and the reliability performance. The equation (1) given below is used to calculate the probability of the transaction fault (*PTXf*), where P_{ALU} denotes the error rate of the execution unit and s denotes the size of a transaction.

$$PTXf = 1 - (P_{ALU})^{s} \quad (1)$$

Thus assuming that the failing probability of a transaction is *PTXf* and the energy required for re-execution is $E_{TX;}$ we calculate the energy spent for recovery ($E_{recovery}$) is given by the following equation (2).

$$E_{recovery} = E_{TX} * PTXf * (1/(1-PTXf)) \quad (2)$$

4. Results

In this section, we analyze the feasibility of applying the error detection schemes with TM-based error recovery. We are specifically interested in how much we can lower the voltage while still providing high error detection capability. For the evaluation we consider the following two scenarios: 1) We investigate the energy overhead of the error detection schemes and the combined error detection and recovery and 2) combination of different error detection schemes. Our preliminary results are shown in the Figure 2 and Figure 3.

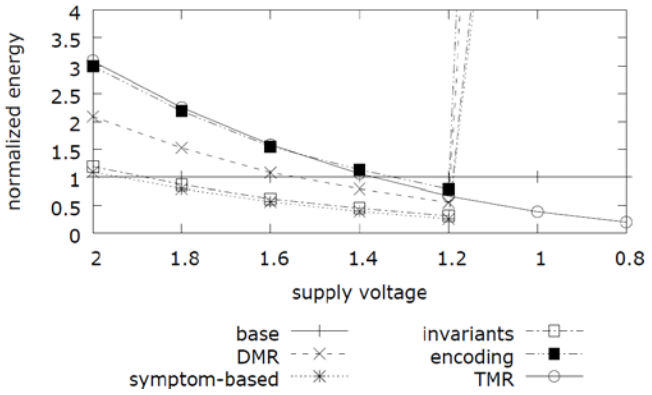

Figure 2: Energy for transactions with 100 instructions

In Figure 2, we summarize the performance of all applications in the SPLASH benchmark by averaging their energy consumption. The energy consumption is normalized to the error-free base case in which 2V supply voltage is used. From this graph (Figure 2), we can observe that when a transaction consists of 100 instructions, Double Modular Redundancy (DMR) starts to outperform the base-case, when Vdd is 1.4V (up to 28% reduction) or 1.2V (up to 54% reduction). Due to the increase in the fault rate, the probability of faults causing rollbacks repeatedly becomes significantly high. Thus, the energy consumption of DMR increases drastically after this voltage level.

There is a trade-off between energy efficiency and reliability, as we can see for DMR and symptom-based error detection and TM recovery. Thus, we can for example combine symptom-based error detection and DMR for consuming less energy, but providing full reliability for critical parts. In Figure 3, we analyzed the energy overhead of this combination in comparison to the base case and DMR only for a transaction size of 100 instructions. We assume that 30, 50 or 70% of the application are only secured by symptom-based error detection. With this combination it is possible to lower the V_{dd} to 1 V (in comparison to 1.2 V with DMR only) and still be more efficient than the base case. Specifically, we reduce the energy consumption by 66\% in comparison to the base case.

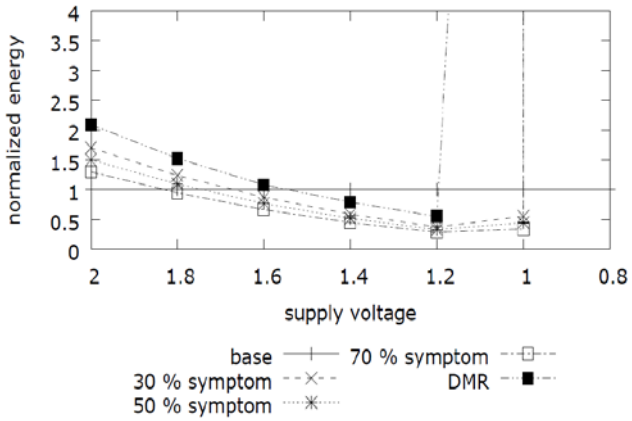

Figure 3. Combination of different error detection schemes

5. Conclusion

To improve the energy-efficiency of modern CPUs, one can reduce the supply voltage of cores. Reducing the supply voltage increases however the likelihood for wrong executions of programs. In this paper, we proposed using transactional memory (TM) for rolling back the effects of wrong executions. To reduce energy consumption, one needs an error detection scheme that has both a sufficient coverage and a low overhead. Based on our evaluation, we conclude that one can reduce the energy consumption of CPUs, in particular, if we have efficient hardware support for TM and for error detection. An open question remains with respect to how to effectively protect the TM itself against transient errors caused by lowering Vdd. Future work of this methodology will focus on the development of SW (programmer-driven) and HW methods to operate below the safe Vdd limit using selective replication to detect and correct sporadic errors and also devise a probabilistic model for near- and far- future devices that provides failure rates for SRAM cells when Vdd is below the safe limit in order to build a statistical distribution.

Reference

[1] Combining Error Detection and Transactional Memory for Energy-efficient Computing below Safe Operation Margins. Gulay Yalcin, Anita Sobe, Alexey Voronin, Jons-Tobias Wamhoff, Derin Harmanci, Adrián Cristal, Osman Unsal, Pascal Felber, Christof Fetzer, In 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2014) - Feb 2014 (accepted and will appear in the proceedings).

[2] FaulTM: error detection and recovery using hardware transactional memory. *Gulay Yalcin, Osman Unsal, and Adrian Cristal. 2013. In Proceedings of the Conference on Design, Automation and Test in Europe (DATE '13). EDA Consortium, San Jose, CA, USA, 220-225.*