Harnessing Energy Efficiency of Heterogeneous-ISA Platforms

Sharath K. Bhat
ECE, Virginia Tech
skbhat@vt.edu

Ajithchandra Saya
ECE, Virginia Tech
ajith6@vt.edu

Hemedra K. Rawat
ECE, Virginia Tech
hkr1990@vt.edu

Antonio Barbalace
ECE, Virginia Tech
antoniob@vt.edu

Binoy Ravindran
ECE, Virginia Tech
binoy@vt.edu

Abstract

With the emergence of both power and performance as primary design constraints, energy efficiency has become the new design criteria. A platform with heterogeneous-ISA processors can provide multiple power-performance execution points needed for a varied mix of workloads. We argue that a new system software architecture is needed to obtain maximum energy efficiency on such heterogeneous-ISA platforms. We present our system software, a replicated-kernel operating system and a compiler framework, and quantify the advantages of such a system software on ARM-x86 using simulations. Based on our experimental observations, we propose a scheduling approach which considers system and application runtime characteristics along with platform profiles to maximize energy efficiency.

Categories and Subject Descriptors

D.4.8 [**Operating Systems**]: Performance—*Simulation*; C.1.3 [**Processor Architectures**]: Other Architecture Styles—*Heterogeneous (hybrid) systems*

Keywords

Power Consumption, Performance, Energy Efficiency, Replicated-kernel OS, Heterogeneous Platform

1. INTRODUCTION

Computer architectures are evolving towards greater heterogeneity to provide various power-performance execution points for different mixes of workloads in all areas of computing. With the hitting of power wall and the emergence of multicore processors, power has become a primary design constraint [8]. Therefore, minimizing energy consumed by a workload has emerged as the new design criteria.

As computer architects strive for providing platforms with multiple non-overlapping power-performance execution points, having specialized cores with different power-performance characteristics is an attractive solution indeed. Today, heterogeneity can be found in the form of microarchitectural features within a single-ISA, such as ARM big.LITTLE [12], or among overlapping ISAs, as in Intel Xeon-Xeon Phi [10]. The other form of heterogeneity is the one in which different processor cores implement completely different ISAs. A CPU-GPU setup is a traditional example for such a case, but GPUs cannot run an operating system, and are therefore being exploited as devices by the main processor. However, a heterogeneous system with different ISA processors that are OS-capable [2] can provide greater power and performance benefits as shown in [13].

Today, ARM and x86 are the two most popular architectures known for power efficiency and high performance respectively. Thus, conceiving a heterogeneous platform with ARM and x86 cores with varying microarchitectural features can provide a broad range of power-performance execution points. Even if such platform is not available on the market as of today, an ARM-x86 system can be created by interconnecting an ARM and an x86 processor through PCIe [7].

The prospects of a heterogeneous-ISA platform is exciting but providing the necessary software support to fully exploit heterogeneous systems is a daunting task and opens a whole new area of research. Traditional SMP operating systems like Linux are designed to run on a single ISA and need a major redesign to support such systems. Moreover, to ensure optimal utilization of the hardware, an application needs to be meticulously crafted considering the underlying architecture. This hinders programmability: it is hard and time consuming for a programmer to manually identify and annotate various parts of the application as demonstrated in CPU-GPU setups. Thus, providing a familiar programming model without losing the power-performance benefits of the underlying architecture is crucial for the success of such platforms.

In this paper, we present a new system software architecture for heterogeneous-ISA platforms. Moreover, based on experimental observations, we describe the parameters that should be considered while scheduling application threads to obtain energy benefits. Specifically, we make the following contributions.

- Introduce a new system software design for heterogeneous-ISA platforms.
- Provide a new measurement framework to profile an application's energy efficiency at function level.

HotPower '15, October 04-07, 2015, Monteray, CA, USA

DOI: http://dx.doi.org/10.1145/2818613.2818747

- Create a system simulator to show the energy benefits provided by such a design.
- Propose static and dynamic parameters that need to be considered for scheduling application's threads.

The paper is organized as follows. We discuss the need of a new system software for emerging platforms in Section 2 and present our design in Section 3. Section 4 and Section 5 discuss the new measurement framework and simulator respectively to show the benefits of such a design. Section 6 analyzes the parameters to consider while scheduling to maximize energy benefits and Section 7 concludes.

2. MOTIVATION

Previous research [13] has shown the power and performance benefits of having multiple different-ISA cores on a single chip. However, such work uses simulations and does not discuss the kind of system software needed by the platform. Also, the discussion focuses on processor cores and their microarchitectural differences. However, computing platforms are designed for a wide variety of target markets. For instance, power-efficient low-performance servers are designed with ARM processors, whereas a server machine based on Intel Xeon or Xeon Phi processors is designed for high performance. On each platform, along with a processor other components like peripherals, memory, storage etc. are carefully selected to design an ecosystem with a consistent power-performance profile. In [16], two server platforms with different power-performance profiles are paired to investigate an energy-efficient architecture. The authors run separate system software on each architecture. When migrating an application from one server to the other, the application is terminated on the origin server and re-started on the destination server; only its persistent data is maintained consistent, e.g., the data saved on networked file system. This results in either high migration overheads to checkpoint/restart the application, or limits the approach to only certain classes of applications that are stateless, i.e., without a volatile state.

The lack of an efficient system software solution to encompass both heterogeneous chip multiprocessors and server platforms acts as a limiting factor in harnessing the complete energy efficiency provided by such platforms. The traditional approach of porting an operating system to a new platform doesn't apply to heterogeneous-ISA architectures. Linux, as an SMP operating system, has been ported to overlapping-ISA platforms [11], but porting to fully heterogeneous-ISA platforms is not possible – the complete ISA incompatibility requires per-ISA executable code. On the other hand, running independent operating systems as in [16] is too restrictive. In this paper, we try to bridge this gap by exploiting the replicated-kernel OS model and extending Popcorn Linux [3] on ARM-x86. Compared to other multikernel approaches like [4], Popcorn Linux can run standard applications without any modification as it is compatible with Linux.

3. THE SYSTEM SOFTWARE

A replicated-kernel OS design [3] provides an effective solution for heterogeneous-ISA platforms. On such platforms, a different kernel runs on each ISA. Kernels communicate using explicit message passing to maintain a single operating system state despite the ISA differences. This provides a single operating environment, which also enables application migration among kernels.

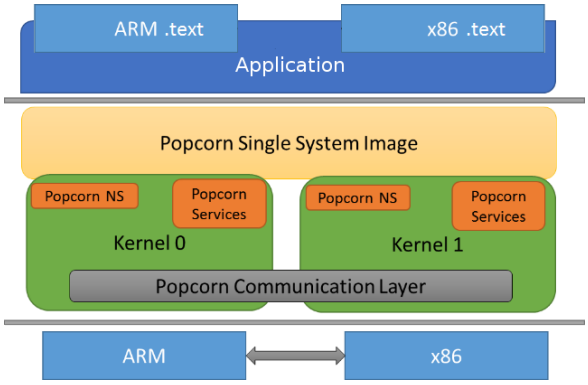

Figure 1: System software architecture for an ARM-x86 heterogeneous platform

For such an operating system, the applications are compiled to transparently run on every architecture in the platform and support migration among different-ISA processors. A compiler and linker toolchain is provided to create binaries with multiple code sections, one per ISA, all of which operate on the same address space (fixing the same data types, alignments, and address space layout).

The architecture of the system software for an ARM and x86 platform is shown in Figure 1. Such architecture extends the Popcorn design introduced in [2] to a fully heterogeneous-ISA platform. Both architectures run an instance of the Linux kernel natively compiled for each architecture. These two kernels communicate with each other using explicit message passing via the Popcorn Communication Layer. This layer enables all distributed services between kernels, including Popcorn Namespaces (NS) and Popcorn Services, that create a single operating environment (single system image). Moreover, each kernel supports the loading of heterogeneous binaries. Such application binaries, as depicted in Figure 1, have multiple code sections (*.text*) – each compiled with architecture-specific optimizations for a particular ISA. The two key functionalities of Popcorn Linux that have been extended by our initial prototype are described in the following.

Task Migration.

The goal of the system software is to provide the application developer with the familiar SMP interface on the ARM-x86 heterogeneous-ISA platform. Therefore, as tasks can seamlessly migrate among cores in an SMP OS, seamless task migration between ARM and x86 processors is provided by the operating system using the task migration service in Popcorn Linux [2]. The task state is packed into a message and sent to the destination ISA where it is correctly mapped to compensate for the ISA difference. Currently, migration points in the application code have been inserted at function boundaries to provide safe places from which to migrate [2]. Function boundary migration is achieved by packing the function arguments at a specific address in global data section in an ISA-independent manner.

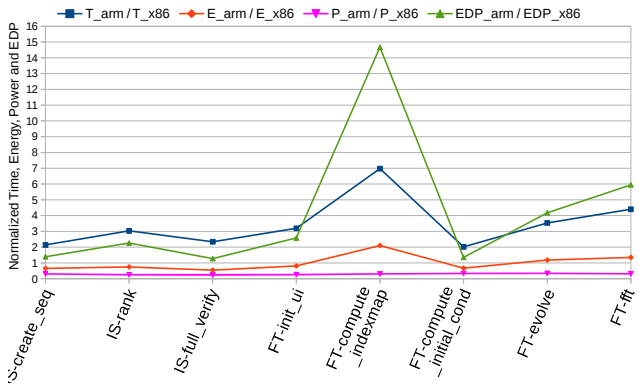

Figure 2: Function level energy profile of NPB IS and FT benchmarks

Application Execution.

To execute an application, a heterogeneous binary is created. Every symbol is aligned at the same virtual address on both the ARM and x86. Function addresses are also aligned. The operating system aliases different-ISA code sections to the same virtual memory range. The memory consistency protocol and the page replication algorithm of Popcorn Linux [2] provide a consistent view of an application's address space across kernels. This enables the familiar SMP shared memory interface for application developers, thereby improving programmability. Further, the migration of application threads at any machine instruction can be enabled by using low-overhead techniques like dynamic binary translation as shown in [6].

4. ENERGY EFFICIENCY ANALYSIS

In order to estimate the energy efficiency provided by emerging heterogeneous platforms, we conducted a thorough evaluation of various classes of workloads from PARSEC [5] and NPB [1] benchmarks on both ARM and x86 machines running vanilla Linux (version 3.12). The experiments were conducted using Intel Xeon E5-1650 v2 (*x86_64*) and Applied Micro X-Gene 1 (*aarch64*) machines. The same setup has been used for all the experiments presented in this paper. Only a subset of the results is reported due to space constraints.

Function Level Profiling.

Experiments were conducted to investigate the energy benefits at the programming language function level. For this purpose we developed a library to measure the system energy consumption at a program's function level granularity. The library provides a uniform interface across various platforms to measure energy so that the applications need not be modified for profiling across multiple platforms. On x86 the library uses the PAPI [14] library (version 5.4.1) that supports measuring the energy consumed using the model specific register (MSR) Linux driver. However, on ARM there was no such support available. We developed a kernel module which can profile the energy consumption by periodically monitoring the power sensors through a kernel thread using the I2C driver interface.

For reliable measurements on each platform, we shutdown the unused services and ensured that the system does not run any other application. We selected OpenMP-based applications whose threads execute homogeneous computations. We used the gprof [9] utility to selectively choose those functions which account for significant application execution time. The selected functions are instrumented using library calls to gather the energy consumption for every function call. When an application is run, the library interface collects the function-level characteristics such as execution time, call frequency, and the energy and average power consumed.

Figure 2 shows the ratio of ARM-to-x86 for various metrics such as performance, energy, power, and energy-delay product (EDP) for functions of interest across multiple applications. The points on the energy plot with a value of less than 1 indicate that ARM consumes less energy for those functions compared to x86. Furthermore, if the same functions show a minimum EDP it implies that the energy savings can be achieved with minimum performance loss if those functions were migrated to ARM. Since all functions consume on average less power when running on ARM than on x86, exploiting ARM helps reducing the overall system power consumption. If further reduction in the power consumption is needed, it can be achieved by moving those functions like *FT-fft* which have an energy ratio closer to 1 on to the ARM side. This ensures power reduction but at the expense of performance and energy benefits. All other functions can be executed on x86 itself for improved energy efficiency.

5. SYSTEM SIMULATION

Although the experimental results show the promising energy and power benefits of using an ARM-x86 platform, the proposed system software has overheads that need to be considered to estimate the overall system benefits. We have created a system simulator that takes as input the profiling information of an application, such as per-function time and power consumption, and the system software overheads, such as thread migration latency and per page memory consistency latency, on both ARM and x86. The simulator is a user-space program that implements an oracle scheduler, which supports different scheduling policies including energy minimization and energy-delay minimization. It has been implemented as a user-space program.

The simulator provides a comprehensive tool to evaluate all kinds of platforms: shared-memory cache coherent and non-cache coherent processors, as well as processors that do not share memory. To account for the memory consistency part of system software overhead, we exploit an offline analysis tool from [2] that collects the number of instances of such overheads via application profiling. The tool is implemented as an LLVM pass and tracks all memory accesses within functions while simulating the page coherency protocol of Popcorn Linux.

Results.

The simulator generates an optimal schedule by mapping the functions to a suitable architecture as determined by a policy. The graphs in Figure 3 and Figure 4 show the energy benefits of the heterogeneous ARM-x86 platform (without shared memory) with respect to homogeneous ISA execution for IS and FT applications (class C) using energy minimization and energy-delay minimization policies respectively. The energy minimization policy results in an energy

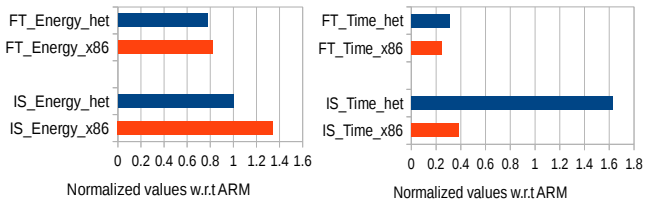

Figure 3: Simulation: Energy Minimization policy

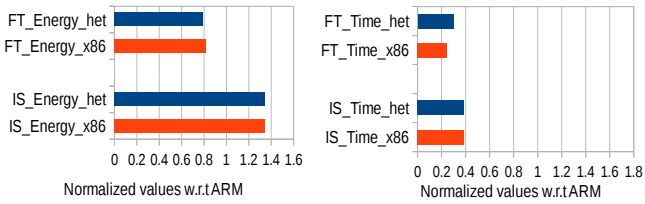

Figure 4: Simulation: Energy-Delay Minimization policy

saving of 22.4% with a speed up of 68.7% for FT on the proposed system software architecture and platform compared to ARM and a 5.3% energy saving with a performance loss of 27.3% compared to x86. In the case of the energy-delay policy FT achieves an energy saving of 21% with a speed up of 70% compared to ARM and 3.7% energy savings with a performance loss of 21.9% compared to x86. Note that for an application like IS, though there are energy savings, the migration can result in a loss of performance. However, when simulating an ARM-x86 setup with shared memory, these performance penalties are almost canceled. The *IS_Time_het* value for energy minimization falls below 1, while the *FT_Time_het* value becomes lower than *FT_Time_x86* for EDP minimization.

6. DISCUSSION

In our heterogeneous setup without shared memory, Intel Xeon E5-1650 typically consumes $60 - 75$W while APM X-Gene 1 consumes around $15 - 22$W. An application like Ferret, from the PARSEC benchmarks [5], which has a huge working set and an irregular memory access pattern turns out to be more energy efficient when running on ARM. This is because the x86 processor consumes more power compared to ARM even with features like dynamic voltage and frequency scaling (DVFS) enabled during the memory bound phase of an application. Energy efficiency of x86 mainly depends on how fast it can execute the workload. Since an irregular memory access pattern stretches the execution time, the benefits of a faster execution are lost, thus making x86 less energy efficient. The performance is bounded by the memory access latency. Migrating such applications to ARM reduces the overall energy consumption. We argue that such migrations not only improve the energy efficiency of that workload but also improve the energy efficiency of the co-running applications which are memory intensive and have a regular memory access pattern. This is due to the fact that the co-running applications will experience less cache interference and also can use spare CPU cycles left behind by the migrated application.

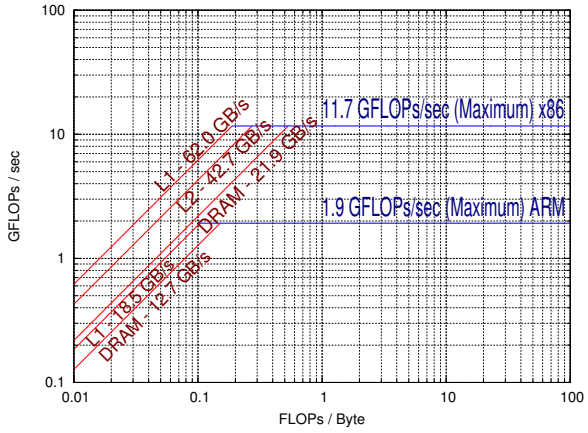

Figure 5: The Roofline model [15] for the selected ARM and x86 platforms

Roofline.

To strengthen our hypothesis and have a better picture of the performance gap between ARM and x86 processors at various degrees of memory intensity, we evaluated both platforms using the Roofline model [15]. Figure 5 shows the maximum single core performance in GFLOPS/s achievable on each platform at different degrees of operational intensity [15]. A similar graph has been obtained with an integer-only compute kernel (a new compute kernel was implemented for such evaluation). The performance gap between ARM and x86 is wider in compute bound phases and becomes narrower as an application starts becoming more memory intensive. At low operational intensity values, such as 0.1 in Figure 5, it is more energy efficient to move the application to ARM as the performance on x86 has reduced thereby increasing the energy consumed. This confirms our previous argument that it is energy efficient to run memory-bound phases on ARM and compute-bound phases on x86. Thus, for a memory intensive application with a regular memory access pattern, it will still be energy efficient to run the application on x86. Note that performance counters can be used to monitor cache misses, which can then be used to determine whether an application phase has a regular or an irregular memory access pattern.

Another factor that needs to be considered is the CPU cycles available on each CPU for an application. When multiple applications are co-running on the same system, the load increases thereby reducing the share of CPU cycles for each application. The performance achieved with the roofline evaluation assumes that all of a CPU's cycles are available for a single application. To account for multiple applications running on the system we can scale the performance value of each application based on its current CPU cycle share.

Scheduling.

With the above observations, we argue that an effective global scheduler that considers power and performance should also monitor per-thread memory traffic and can be designed with minimal runtime complexity. We propose that each processor in the platform maintains a set of its characteristic performance at various degree of operational intensity, for example obtained via profiling at boot-time. Such profiles

are known by all kernels. Moreover at runtime, each kernel periodically updates all the others about its load. Memory access pattern of the application will be estimated using performance counters. With these information the scheduler makes workload migration decisions between processors to improve energy efficiency.

7. CONCLUSION

In this paper, we discussed the inability of present software architectures to exploit the energy efficiency provided by heterogeneous systems with different power performance profiles. We then proposed a new system software based on replicated-kernel OS design. Energy benefits provided by our approach is demonstrated by constructing a simulator which shows up to 22.4% energy savings and 70% speedup compared to ARM. Through a detailed application analysis and system evaluation, we proposed a simple and effective workload scheduling approach which can improve energy efficiency in such platforms.

8. ACKNOWLEDGMENTS

This work is supported by the US Office of Naval Research under Contract N00014-12-1-0880.

9. REFERENCES

[1] D. Bailey, E. Barszcz, J. Barton, and et al. The NAS parallel benchmarks summary and preliminary results. In *Supercomputing'91*.

[2] A. Barbalace, A. Murray, R. Lyerly, and B. Ravindran. Towards operating system support for heterogeneous-isa platforms. In *SFMA'14*.

[3] A. Barbalace, B. Ravindran, and D. Katz. Popcorn: a replicated-kernel OS based on Linux. In *OLS'14*.

[4] A. Baumann, P. Barham, P. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schüpbach, and A. Singhania. The multikernel: A new os architecture for scalable multicore systems. In *SOSP'09*.

[5] C. Bienia, S. Kumar, J. P. Singh, and K. Li. The parsec benchmark suite: Characterization and architectural implications. In *PACT'08*.

[6] M. DeVuyst, A. Venkat, and D. Tullsen. Execution migration in a heterogeneous-ISA chip multiprocessor. In *ASPLOS-XVII*.

[7] Dolphin Interconnect Solutions. Express IX. www.dolphinics.com/download/WHITEPAPERS/Dolphin_Express_IX_Peer_to_Peer_whitepaper.pdf.

[8] H. Esmaeilzadeh, E. Blem, R. Amant, K. Sankaralingam, and D. Burger. Power challenges may end the multicore era. *CACM*, 56(2), Feb. 13.

[9] S. Graham, P. Kessler, and M. K. McKusick. Gprof: A call graph execution profiler. *SIGPLAN Not.*, 39(4), Apr. 04.

[10] Intel Corporation. Xeon Phi product family. http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-detail.html.

[11] D. Reddy, D. Koufaty, P. Brett, and S. Hahn. Bridging functional heterogeneity in multicore architectures. *SIGOPS OSR*, 45(1), Feb. 2011.

[12] A. Stevens. big.LITTLE processing with ARM Cortex-A15 & Cortex-A7. Technical report, 11.

[13] A. Venkat and D. Tullsen. Harnessing ISA Diversity: Design of a heterogeneous-ISA Chip Multiprocessor. In *ISCA'14*.

[14] V. Weaver, D. Terpstra, and et al. PAPI 5: Measuring power, energy, and the cloud. In *ISPASS'13*.

[15] S. Williams, A. Waterman, and D. Patterson. Roofline: An insightful visual performance model for multicore architectures. *CACM*, 52(4), Apr. 09.

[16] D. Wong and M. Annavaram. Knightshift: Scaling the energy proportionality wall through server-level heterogeneity. In *MICRO-45*.