September 30, 2014

Exploration of the Energy and Thermal Behaviors of Emerging Architectures

Final Report

September 30th, 2014

Sudhakar Yalamanchili[1] and Hyesoon Kim[2]

[1]School of Electrical & Computer Engineering and [2]School of Computer Sciences
Georgia Institute of Technology

1 Introduction

The goal of this project is to study the energy and thermal behaviors of future applications executing on emerging architectures at the node level. Our approach is to construct models of the energy dissipation in candidate node architectures and drive models with benchmark codes made of representative kernels and applications. Given that future systems may contain several hundreds to thousands of cores with large memory footprints, traditional modeling and simulation techniques are impractical. Our approach builds on several recent developments at Georgia Tech to ensure fast, high-fidelity models of energy consumption that can scale to the size of future systems.

2 Overview

The project started in May of 2013. The progress to date is best understood in the context of the larger, long-term vision for the application-driven modeling and analysis environment for HPC applications. This environment consists of three main components: i) compiler-based instrumentation tools to acquire implementation-neutral energy and performance data; ii) an automated data analysis toolset and data repository for constructing analytic models for energy and time; and iii) a macro-scale simulation environment for applying the models to large scale (100,000s cores) system models. This program focuses on developing Part i and extending to Georgia Tech's Eiger modeling environment for Part ii, with the final goal of integration with the SST/Macro architecture modeling environment. The long-term vision for this project is shown in Figure 1. We wish enable energy/power debugging of applications in the same vein as performance debugging for HPC applications. Note that several elements of this vision (Eiger and the SST/Macro components) were supported by, and continue to be supported by, Sandia National Laboratories (Livermore). Our goal is to ensure the collective efforts are synergistic.

3 Progress

The primary engineering implementation took the form of extending the scope and capabilities of LANL's Byfl compiler-based instrumentation framework. A high level view is shown in Figure 2. Specific progress to date includes the following.

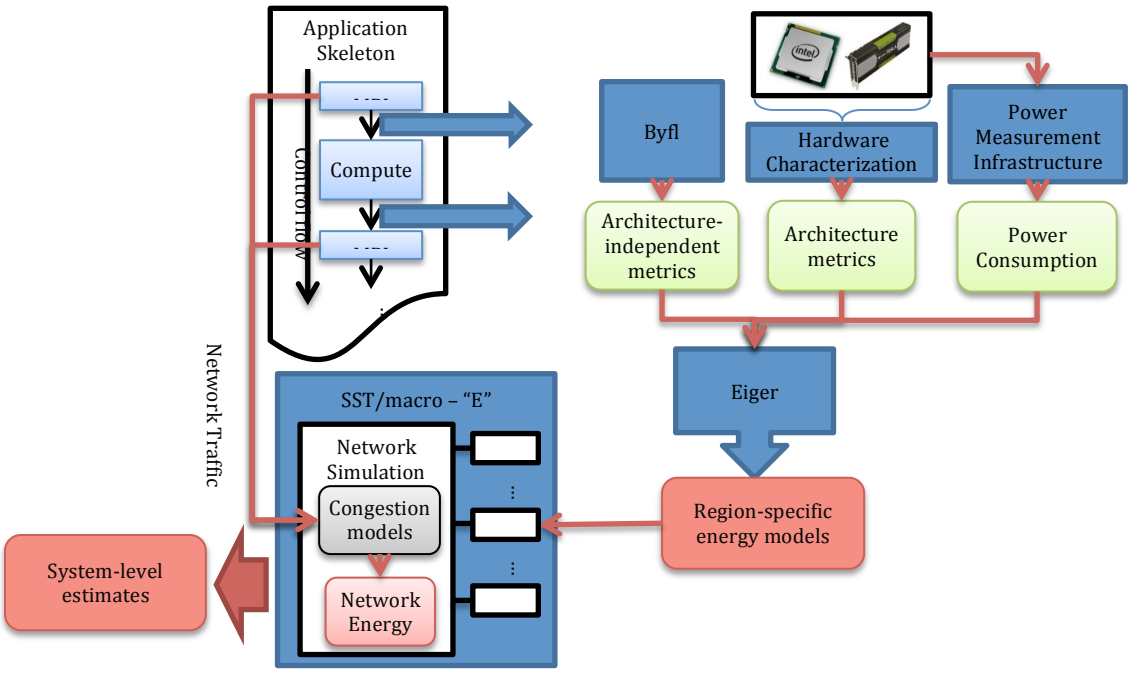

Figure 1 Long Term Vision

3.1 Front-End Tools:

The first few months of the project constructed x86 and ARM instruction tracing tools from application binaries executed with full system emulators. These traces drive instruction level power models being developed in this program. This tool chain development was set aside for the moment while we worked with extensions based on LANL's Byfl infrastructure as described below.

3.2 Statistical Approaches to CPU Power/Energy Modeling:

This part of the project deals with statistically modeling the power behavior of applications. This phase relies on measuring the on-chip power and correlating it with other performance metrics collected from hardware performance counters and compiler instrumentation. We used Intel's RAPL (Running Average Power Limit) interface along with the Linux perf tool and PAPI (Performance Application Programming Interface) for collecting raw data, including power data, from Intel Ivy Bridge systems. The data is imported into Eiger, an automated framework for creating models based on regression, principal components analysis (PCA), and other machine learning techniques. Using a diverse set of applications and their profiled data, we have created first order models that tie into the larger goal of characterizing energy behavior of applications to provide insight for programmers and researchers.

We constructed models based on performance counters and compiler generated instrumented data. As a result, once these models are refined and validated to be robust, they can be used as building blocks for building application-level models that abstract away some of the hardware and architecture-dependent details in favor of compiler-visible performance behaviors. To that end, we have added support for

integrating Eiger with Byfl, enabling Byfl data to be pushed into Eiger's database, so Eiger can create models based on architecture-independent application characterization.

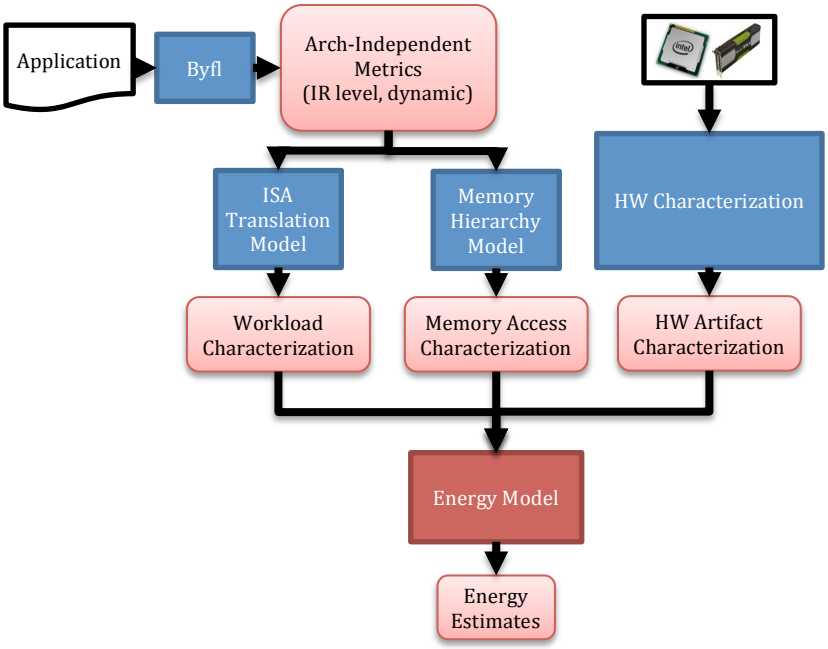

Figure 2 Key Elements of this Program

We are currently studying the process of generating accurate cache models using Byfl+Eiger, and we are also in the process of implementing an LLVM Pass in Byfl that can interface with RAPL directly. Eiger can then be used to construct analytic energy models at code segment granularity. Preliminary work towards this end is in submission to a SC14 workshop.

Experimental Methodology and Tools used in this phase: (1) *RAPL*: Intel introduced the Running Average Power Limit (RAPL) features with the Intel microarchitecture. Although primarily intended to control or limit power usage on chip, this feature can also be used for measuring power and energy consumed by the processor. RAPL interfaces consist of non-architectural model specific registers (MSRs). (2) *PAPI* and *perf*: The Performance API (PAPI) project specifies a standard application programming interface (API) for accessing hardware performance counters available on most modern microprocessors including RAPL counters. The linux perf tool also provides access to the same counters. PAPI provides more fine-grained control for measuring counters in code sections, whereas the perf tool is more convenient for measuring application-wide or

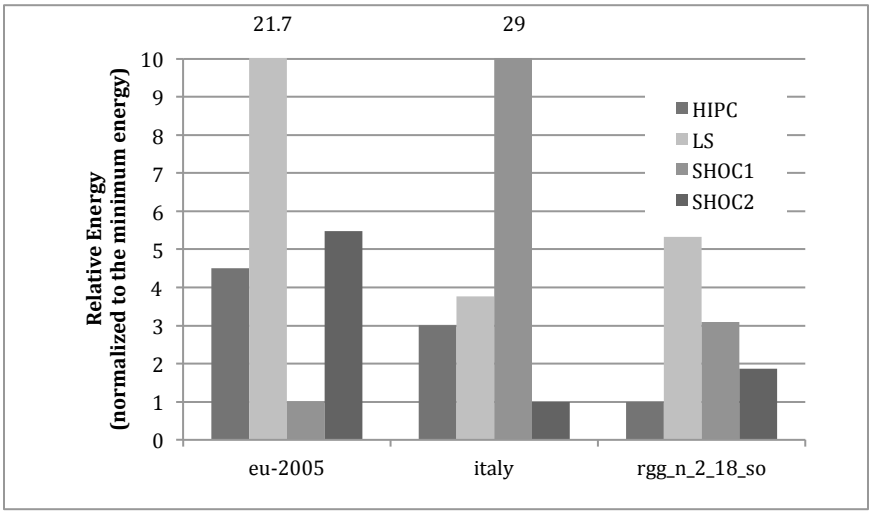

Figure 3 Impact of Input Data Sets on the Energy Behavior of BFS

system-wide counters.

Applications that are used in this study: Mantevo Mini App Suite, SPLASH2, OmpSCR, SSCAv2, and PARSEC.

3.3 High-Level Power Modeling for Accelerators:

We have set up a power measurement infrastructure to measure the power of a NVIDIA K-40 GPU. We have measured power for several graph algorithms (BFS, SSSP, MST, etc.), especially focusing on different implementations of BFS algorithms. We discovered that the power efficiency for the same functionality varies significantly depending on implementations and input sets—see Figure 3. We plan to investigate this issue in more detail. This relationship between algorithms and energy/power efficiency of architectures will be used to understand the energy behavior of irregular algorithms on accelerators (in this case GPUs) as a prelude to the development of high-level energy models for different types of accelerators.

3.4 Energy Auditing of Applications:

We have implemented a tool for associating application functions with their energy consumption. Analogous to time profiling tools like `gprof`, this tool breaks down the overall energy consumption of an application at the function granularity. We use two techniques for associating energy counts with functions: (1) we inject the measurement functions at the beginning and end of every function call, precisely measuring energy; (2) we sample the execution, periodically halting and associating energy for the sample with the current function. The first technique, while more accurate, has a higher execution-time overhead. The second is lighter weight, but it may alias the energy counts for short-lived functions. This tool leverages the Byfl infrastructure for injecting instrumentation into applications at compile-time and the PAPI interface for reading the RAPL energy counters.

4 Summary

We are currently completing the integrated Byfl-Eiger infrastructure that will add analytic models for energy developed here and demonstrate the ability to audit the energy expenditures of an application, e.g. by function or procedure. This capability can be used to provide compile time feedback to developers on the energy consequences of algorithm and data structure decisions.

This year the research has focused on measurement based approaches to energy/power modeling, and integrating this capability into the existing infrastructures (Byfl and Eiger). Longer term we wish to create a comprehensive energy auditing/debugging infrastructure that parallels that of performance debugging developed over the last few decades. At the moment our vision for achieving this is captured in Figure 1.