Beyond DVFS: A First Look at Performance Under a Hardware-Enforced Power Bound

Submitted for blind review
January 20, 2012

Abstract

Dynamic Voltage Frequency Scaling (DVFS) has been the tool of choice for balancing power and performance in high-performance computing (HPC). With the introduction of Intel’s Sandy Bridge family of processors, researchers now have what at first glance appears to be a far more attractive option: user-specified, dynamic, hardware-enforced processor power bounds. In this paper we provide a first look at this technology in the HPC environment and detail both the opportunities and potential pitfalls of using this technique as a replacement for DVFS.

As part of this evaluation we measure power and performance for single-processor instances of several of the NAS Parallel Benchmark suite and focus on the behavior of a single benchmark, MG, under several different power bounds. We quantify the well-known manufacturing variation in processor power efficiency and show that, in the absence of a power bound, this variation has no correlation to performance. We then show that execution under a power bound translates this variation in efficiency into variation in performance. If our sample of 64 processors is representative, a cluster composed of processors at or above median efficiency would run up to 5% faster at the same power than a cluster drawn from the full distribution.

1 Introduction

Power has now become the primary performance problem in high-performance computing (HPC). Up to this point, Dynamic Voltage/Frequency Scaling has been the method of choice for investigating the tradeoff between power and performance in HPC applications. Running the processor at a lower clock frequency required less voltage, but both the performance impact as well as the amount of power and energy saved was highly application dependent. While research has made great strides in modeling these effects, to our knowledge no machine in the Top 500 list of supercomputers makes use of DVFS to save power or energy.

Power clamping provides what may turn out to be a compelling alternative to DVFS. The user simply specifies a time window and a power bound and the hardware guarantees that over each window the average power will be no more than the specified bound. Both the window size and bound may be modified at runtime. So far as processor power is concerned, system designers and operator can now control exactly how much power is consumed per processor across the entire system.

However, there are subtleties that must be accounted for if power clamping is to gain widespread acceptance in the HPC community. In this paper, we explore the variations in power efficiency across processors and how this variation is translated into variations in performance while under a power bound. Small variations in processor power consumption inevitably introduced in the manufacturing process are well-known [4] and do not affect processor speed. Placing a power bound on a processor moves the variation from power (the processor now operates at a specified number of watts) to performance (less efficient processors will run more slowly at the specified
power bound).

This is of particular interest in the HPC domain. Up to this point, machines have been specified with the assumption that variation in power can be tolerated while variation in performance cannot: a few inefficient processors will not upset the power provisioning of a machine room, but a single slow processor can constrain the performance of an entire cluster. Under a power bound, those few inefficient processors are transformed into slow processors. This paper quantifies that effect on real hardware for several codes in the NAS Parallel Benchmark Suite [13].

2 Overview

Up until now, most research in power-aware supercomputing focused on trading a loss of performance for energy savings. While an interesting problem in its own right, it did not match well with the goal of supercomputer stakeholders, which is how to make an existing machine run as fast as possible. In effect, these stakeholders were asked to consider doing less work per unit time in order to save some amount of someone else’s money. These approaches have not gained any traction in the wider community.

However, we have reached a point for the largest supercomputers where the amount of electricity that can be brought into the machine room will be the limiting factor on the amount of work that can be done. At this point, buying additional homogeneous nodes will no longer increase performance because there will be no power available to bring them online.

The processor architecture community has already reached this point and we propose to adopt their strategy. The most recent processors from both AMD and Intel are overprovisioned with respect to power: not all cores can run simultaneously at the highest possible frequency, and the user effectively buys capacity that will always remain unused. What the user buys instead is flexibility, either to run all cores at a slower frequency or a handful of cores at a faster frequency.

We foresee future clusters designed the same way. For problems that benefit from the largest number of processors, all nodes will run simultaneously at either a low CPU frequency or at a low power bound). For problems that perform best given a smaller number of faster nodes, the user or operator will schedule a smaller number of nodes with a higher power draw and turn off the remaining nodes. Utilization will no longer be measured as a percentage of node-hours but rather as a percentage of maximum kilowatts.

Making this approach a reality requires solving problems far more difficult than relatively straightforward problem of saving energy. The user is presented with what is effectively a dynamically reconfigurable cluster and must determine not only the optimal number of nodes but also how much power should be assigned to each node. In short, we have moved from an energy savings problem to a power scheduling problem.

As we detail in the discussion section, both DVFS and power clamping have their strengths when brought to bear on the scheduling problem: power clamping provides a hard, tight bound on power but performance modeling under the power bound appears to be quite difficult. DVFS provides a loose, soft bound, but performance under DVFS is now well-understood. The experimental results that follow are the first necessary steps to understanding performance under a power bound.

3 Intel’s Running Average Power Limit (RAPL)

With the Sandy Bridge family of processors, Intel introduced both onboard power meters and power clamping. In this section we give a technical overview informed by our practical experience in getting these tools to work. To the best of our knowledge, the only documentation for these features is in chapter 14.7 of Intel’s Software Developer’s Manual[9]. We stress that we are experimenting with new processors in a pre-production environment and issues raised here may well have been solved by the time this paper is read. In the following we abbreviate Watts as W and Joules as J.
3.1 Interface

Users measure and control processor power using several machine-specific registers, or MSRs. Intel provides two privileged instructions, `readmsr` and `writemsr`, as the interface to these registers. Instead of writing a specialized kernel driver, users and developers may make use of the `msr` kernel module. This exports a file interface at `/dev/cpu/N/msr` that, given suitable file permission, can be used to read and write any MSR on the node. This approach has significant security implications and should only be used for development in a trusted environment.

3.2 Architectures

Intel separates the Sandy Bridge family into two classes, client (family=0x06, model=0x2A) and server (family=0x06, model=0x2D), with the server-class processor receiving the Xeon designation. The two architectures share a subset of RAPL features. We only use Xeon processors in this work.

3.3 Domains

The Sandy Bridge architecture supports three power domains on each architecture. Both architectures support package (PKG) and Power Plane 0 (PP0) domains, while the server adds a separate DRAM domain and the client adds a second power plane (PP1). The documentation provides little information to distinguish what circuitry falls into which domain. For example, “PP1 may reflect to uncore [the unified core abstraction of last-level cache]” and “Generally, PP0 refers to the processor cores” exhaust the descriptive documentation of these two domains.

Our testbed does not yet support measure or control of the DRAM domain. Across the whole of the NAS Parallel Benchmarks the power ratio between the PKG and PP0 domain remained nearly constant. In this work we limit ourselves to experiments based on measuring and controlling the PKG domain.

3.4 Units

We have not been able to locate any documentation describing the accuracy of the time, power and energy measurements available. Precision is architecture-specific and is provided by reading the MSR `RAPL_POWER_UNIT` register. Our architecture reports power clamping will be performed in units of 0.125W over time windows with units of 0.000977 seconds. Energy measurements are reported in units of 0.0000152J.

3.5 The PKG Domain

The `POWER_LIMIT` set of MSR reports the architecture-specific power envelopes and maximum clamping time window for each domain. In our case, the lowest power bound supported by the PKG domain is 51W. The `thermal spec power` is rated at 115W and the maximum power is rated at 180W. The maximum time window for power clamping is 0.0459 seconds.

3.6 Power Clamping

The Sandy Bridge processor does not provide a power bound in the strictest sense. Rather, the user specifies a time window and a maximum average power for that window and the processor guarantees that the average bound will be met. Intuitively, longer windows may allow better performance for applications that utilize the CPU in bursts; if the burst exceeds the window size, the processor will have to be throttled. Determining how the size of this window affects machine room power provisioning will be an area of our future research, especially since we can expect highly synchronized load spikes from parallel scientific codes.

The PKG domain is unique in that it provides for two separate clamping windows. If a user provides a higher bound for a smaller window and a lower bound for a larger window, this may provide finer control over application performance. Note that a lower bound for the smaller window would make the larger window superfluous. All experiments in this paper were conducted using a single window of the smallest possible size (0.000977 seconds). This is because we have no application-specific knowledge, and smaller windows best avoid potential power spikes in this absence of knowledge.
There are two modes provided for power clamping, enabled and clamping. Setting the former causes the processor to respect the minimum performance level request by the operating system. Setting both allows the processor to override the OS if that is necessary to meet the power bound. In our experience, setting only the enabled bit did not change the power profile of any benchmark. All experiments reported in this paper set both bits high.

3.7 Other Interfaces

Each domain has a separate, read-only, 32-bit energy meter. As the unit of joules is so small, this will roll over within hours. The PP0 and PP1 domains expose a policy interface; this may only be useful on client architectures. The PKG and DRAM domains expose a counter that records the number of seconds spent below the performance level requested by the operating system. This could be potentially useful when the power bound is expected to be reached only sporadically. In our work we expect the clamping to be operating more or less continuously, and so we have not investigated this counter further.

4 Experimental Results

We performed the experiments detailed in this section on a dedicated 32-node partition of the Zanfordel TLCC2 cluster at Lawrence Livermore National Laboratory. Each node contains two Sandy Bridge 8-core processors. We configured the NAS Parallel Benchmarks to use class C problem size and 8 MPI ranks. We chose this benchmark because it consumed the greatest amount of power of all of the NAS Suite, and because it executes for a reasonable length of time while keeping all eight cores busy. Experiments that used a hardware power bound were configured to use a 1 millisecond window on the PKG domain with clamping enabled.

Figure 1 shows power variation across the 64 processors in our testbed partition. We measure average power on each processor for selected NAS Parallel Benchmarks. We order the processors by maximum power draw. Three processors are annotated: the two least-efficient processors are on nodes 49 and 50, and the most efficient processor is found on node 48. The MG benchmark consumes the greatest amount of power, with roughly ten watts separating the least- and most-efficient processors.

The data in Figure 2 show the results from the MG benchmark. We compiled the MPI version of the benchmark to use eight MPI ranks, one for each core in the machine. We created a PAPI profiling library to set up the necessary MSRs immediately after the program returned from MPI_Init and measured and reset the MSRs immediately before calling MPI_Finalize. We read total joules from the PKG domain and divided this over elapsed time to arrive at average watts.

The benchmark was run 34 times on each processor, once without any power bound and once at each of the 33 power bounds ranging from 51W to 83W. Each point on the graph represents a single run and its associated power (given in average watts) and performance (given in seconds of execution time). We highlight the following special cases:

Unbounded power  At the bottom right corner of the graph there are a cluster of black crosses, a red empty circle and a blue empty triangle. These represent each of the 64 processors in normal operation, i.e. with no user-specified power bound. Note that these points are distributed horizontally: each processor took roughly the same amount of time to execute the benchmark. However, these data points are spread out over the power dimension, ranging from 77.74W for the most efficient processor (the blue empty triangle) to 85.36W for the least-efficient processor (the empty red circle).

We emphasize the lack of correlation between efficiency (as measured by power consumption) and performance (as measured by wall clock time). All 64 of these processors operate at the same clock frequency and have the same execution rate. However, some processors need more power than others to accomplish this.

Bounded power  Starting from the top left of the graph we show average power and execution time
Figure 1: 64 processors ordered by average power consumption over selected NAS Parallel Benchmarks.

while running under a user-specified power bound. We track the processors identified as most- and least-efficient based on non-bounded execution. These are consistently remain the most- and least- efficient at nearly allow measured bounds. However, efficiency is now expressed in term of time. Execution times range from 16.26 seconds for the most-efficient processor to 17.23 seconds for the least efficient.

Here we again emphasize the lack of correlation between efficiency and power consumption. Regardless of how efficient a processor may be, if it is operating under a user-specified power bound, its power consumption will match the bound precisely.

5 Discussion

In this paper we have laid out a vision for a new approach to power-aware supercomputing and demonstrated a new tool that may be helpful in achieving that vision. In this section we sketch out what may be some of the less obvious implications of computing under a power bound.

5.1 Processor-level performance modeling

The combination of high-resolution timers and on-board power meters makes processor-level performance modeling far more tractable. For a given static
computational load it is now straightforward to plot performance to an arbitrary power bound. If only a subset of the processors in a system are to be used the obvious optimization is to select first from the most-efficient processors.

Performance modeling becomes far more interesting when loads are no longer assumed to be static. For example, running a load-imbalanced application under a power bound might result in dramatic swings in execution rate: a state where all cores are busy and slowed to meet the power bound can suddenly become a state where a subset of cores are now idling and their power has been effectively contributed to the remaining cores, causing them to execute much faster. This would be a challenging problem for single-processor applications; modeling HPC applications would also have to account for these local effects influencing the behavior of remote processors.

5.2 Node-level performance modeling

HPC nodes are now universally multiprocessor. Assuming that available processors will fall within a range of efficiencies, is it more advantageous to install processors of similar efficiencies on a node or to arrange them so that one processor will be significantly more efficient than the other? The answer will depend on the cluster configuration as well as
the characteristics of the applications executed on the cluster. For example, an application that needs to maximize memory first and processor counts second would benefit from having unused, less-efficient processors distributed throughout all of the nodes. Applications that benefit from smaller numbers of faster processors would likely benefit from having the most efficient processors concentrated in the smallest number of nodes.

5.3 System performance modeling

If the system designer chooses to have heterogeneous nodes (with regard to processor efficiency), how are those nodes best distributed throughout the system? If nodes are segregated by efficiency this may provide superior network performance for small-node-count, high-node-power jobs but inferior performance for large-node-count, low-node-power jobs. There may also be an issue of rack capacity: a rack designed to match the expected median power draw may not perform as well if it is populated entirely with low-efficiency processors.

5.4 Runtime performance modeling

Once a system design has been implemented, the user must decide how to best make use of the system. To do so, the user must now not only choose the best number of nodes, but also select which nodes to use, which processors on the nodes, and finally how to go about balancing power consumption throughout the duration of the program. These decisions must be informed by the interplay of power and performance at the system, node and processor levels.

6 Related Work

Within-die power variation has been taken into account for scheduling Dynamic Voltage/Frequency Scaling (DVFS), both at the level of the package [12] (for the Pentium M architecture) as well as for individual cores [15, 8] (on the AMD Opteron architecture) as well as for more exotic architectures [5, 3]. Herbert et al. [7] took the next step by combining DVFS with work shifting to prioritize use of the most efficient cores. Power variation can be expected to change as the processor ages, thus requiring continuing characterization during the lifetime of the processors [1]. Rangan et al. [14] observe that variation across cores can be evidenced by variation in the highest frequency each core can support; rather than using the minimum frequency across all cores they recommend classifying the processor using the average core frequency and present a system that effectively masks the core heterogeneity. Research on global power control has also assumed the existence of a DVFS interface[10].

And alternative to DVFS, per-core power gating (effectively shutting off individual cores) has been explored for data center workloads [11]. Cebrían et al. [2] combine DVFS with several additional architectural-level techniques, such as instruction criticality analysis, pipeline throttling, and power-token throttling. Davis et al. [4] have examined the effects of variability in power models used to characterize large-scale clusters. Finally, the physical consequences of overcommitting available power in datacenters have been studied [6].

References


