RC23666 (W0507-124) July 14, 2005
Computer Science

Dynamic Processor Overclocking for Improving Performance of Power-Constrained Systems

Juan Rubio, Karthick Rajamani, Freeman Rawson, Heather Hanson, Soraya Ghiasi, Tom Keller
IBM Research Division
Austin Research Laboratory
11501 Burnet Road
Austin, TX 78758

Dynamic Processor Overclocking for Improving Performance of Power-Constrained Systems

Juan Rubio, Karthick Rajamani, Freeman Rawson, Heather Hanson,
Soraya Ghiasi, Tom Keller
Power Aware Systems Group
IBM Research, Austin, TX 78758
{rubioj,karthick,frawson,hansonh,ghiasi,tkeller}@us.ibm.com

Abstract

Power and thermal constraints are proving to be limiting factors for computer system performance. Conventional approaches estimate a design power and cooling limit based on consumption for a specific, worst-case workload while operating at a nominal frequency. The system is then constrained to operate at or below this frequency even for workloads that have lower power and cooling requirements. As a consequence real workloads pay a performance cost for operating at a design point defined by the worst-case workloads.

In this work, we propose *Dynamic Processor Overclocking* as a way of reducing the performance impact of the chosen design point for workloads that have sufficient slack in their power and cooling requirements. Dynamic Processor Overclocking utilizes the Dynamic Voltage and Frequency Scaling (DVS) capabilities of processors to obtain increased performance in power-constrained systems. While DVS has been widely studied in recent years, solutions exploiting it have focused on saving energy. In contrast, our proposal incorporates DVS for performance enhancement.

The paper focuses on the evaluation of the effectiveness of Dynamic Processor Overclocking using the SPEC CPU2000 benchmarks as a representative set of programs. We identify the workload characteristics that impact the benefit obtained from overclocking. We also investigate the relationship between the severity of the power constraints and application characteristics that determine the overclocking benefits. Our work presents a new approach to improving performance in power-constrained systems, and the insight and methodology required for analyzing its effectiveness for interesting workloads.

1 Introduction

The last twenty years have seen a steady increase in the performance of computer systems at a cost efficiency unmatched by any other industry. But the increasing capacity and density of components in the systems has resulted in a steady growth in power supply and cooling requirements. This growth in power and cooling requirements has now reached a point where power and thermal constraints are barriers to further efficient growth of computing performance.

A computer system or a component may be *power-limited* by:

- Power delivery: a limit in the amount of power that can be supplied to a module or system. This limitation can occur because of factors such as a limit in the capacity of the wires that feed current to the module, or the maximum current a specific voltage regulator design can deliver without a hazardous voltage drop.
- Power dissipation: a limit in the rate at which power (in the form of heat) can be removed from a system. This limitation is directly connected with the characteristics of the cooling system (e.g., thermal resistance and capacity of the heatsink, and airflow through the system).

In the case of a component such as a processor chip a power-delivery limitation can be from a limitation of the DC supply or voltage regulator module (VRM) providing the power to the chip; and a power-dissipation limitation may be from a cost-efficient heat-sink and fan design and how it matches to the Thermal Design Point (TDP) specification for the processor. The peak processor power consumption (and heat) is related to peak switching activity in the processor. So the conventional approach to ensure that a chip meets the constraints is to fix the frequency of the chip (and the voltage needed to support it) at a value low enough to limit the switching power to a value that meet the constraints. Alternatively, the power and cooling systems are designed for specific capacities tied to an estimate of the switching activity for the chip operating at a target frequency (and voltage).

However, the switching activity and associated power consumption are highly workload dependent. Applications that cause intense core activity either from a high degree of instruction-level parallelism (e.g., high instructions retired per cycle) or higher speculative execution tend to have higher core power consumptions than those that do not. At the chip-level, high level of on-chip cache activity can also increase power consumption. As a consequence it is impossible to adopt a workload-agnostic approach for determining the power consumption even at a fixed frequency.

1.1 Power-Constraints Impede Performance

Chip vendors typically test their chips at a desired operating frequency for a special (set of) worst-case workload(s) that they expect no practical workload can match in terms of power consumption, appropriately de-rate the power consumption for realistic workloads and provide the de-rated number to system designers building the cooling and power supply sub-systems. However, most workloads fall short of the vendor-specified power consumption, but have to pay the price in performance by using the fixed performance level (frequency) that is guaranteed to be safe by the vendor specifications. This fact has been known for quite a while, as it is reflected by the popularity of static overclocking solutions that run the chips at frequencies higher than qualified by the chip vendor. The limitation in performance for real workloads is equally applicable when system designers have to optimize (lower) their cooling and power supply designs for cost considerations and have to then statically pick a lower operating frequency to meet the constraints.

1.2 Dynamic Processor Overclocking to Boost Performance

In this paper, we advocate an adaptive approach to reduce the performance impact seen by real workloads when limited to operate at a frequency determined from worst-case workloads. Our proposal is to utilize voltage and frequency scaling capabilities of processors to boost application performance through *Dynamic Processor Overclocking* (DPO). Dynamic voltage scaling (DVS) technology was initially adopted in processors designed for embedded devices such as IBM PowerPC 405LP [1], and Intel PXA (Xscale) processors [2]. The technology was used for energy savings in battery operated environments.

Even as variations of the technology moved to larger system environments as with the Intel PentiumM [3] for laptops and the IBM PowerPC 970[4, 5] in desktop and server systems, current systems exploit DVS to mainly save power when the system utilization falls below pre-specified thresholds. The newly announced Intel Montecito [6, 7] processor adopts a dynamic approach to processor clocking similar to what we advocate, but managed by an embedded microcontroller, Foxton, in the processor.

With Dynamic Processor Overclocking, the system boosts the processor frequency to values higher than the nominal whenever the power/cooling constraints are not violated. For workloads that are not operating at the cooling/power supply limits this can often result in real performance increase. The focus of this paper to evaluate the benefits of Dynamic Processor Overclocking and establish the relationship between the workload and related system characteristics which determine the benefits.

The rest of the paper is organized as follows. The following section gives a brief overview of some related work. Section 3 provides a broader description of our DPO proposal. Following that, we give a brief overview of our experimental platform based around the Intel Pentium M processor and our evaluation methodology for analyzing DPO's performance impact. We then go over the results of our evaluation and end with the conclusions in Section 6.

2 Related Work

There have been a number of efforts over the years examining the implementation and effectiveness of dynamic voltage and frequency scaling for saving power in embedded systems [8, 9, 10, 11, 12, 13]. Performance-oriented explorations include attempts to quantify and/or reduce the performance loss encountered in an energy-saving adoption of DVS. In contrast, our approach targets performance increase from DVS in a power-constrained environment. Some of the newer works that look at power management and its impact on performance in a non-embedded-systems context include the following.

Miyoshi, et al., introduce the concept of *critical power slope* which can be used to identify the energy efficient operating points in a system [14]. Although Miyoshi applies critical power slopes for determining which operating points to use to save energy, the same technique can be extended to cover situations when the processing frequency is increased to increase performance.

Felter, et al, examine the possibility of maximizing performance in the presence of reduced, fixed power budgets by dynamically changing the allocation of power between the processor and memory [15]. For example, during memory intensive phases, more power can be allocated to the memory subsystem and less to the processor.

Kotla, et al., instead use dynamic program classification and scheduling techniques to adjust the frequency and voltage of a system to minimize the power consumption for existing performance [16]. Ghiasi, et. al., introduces an operating systems scheduler which dynamically places tasks on the processor which most closely matches the application's ideal frequency setting [17] in a multi-processor system with heterogeneous-frequency processors. Both rely on performance counter-based prediction to guide decision making.

Uht and Vaccaro's TEAPC uses a feedback control system to adapt to changes in system temperature by decreasing or increasing the frequency of the processor clock but makes no attempt to exceed the nominal frequency in situations where the temperature remains below their threshold under full CPU load [18].

Weissel and Bellosa also use a feedback control system to develop an energy efficient scheduler, but instead adapt to changes in the composition of the instruction mix [19]. They develop a matrix in which the rates of memory requests and instructions per cycle map to a frequency.

Lee and Skadron develop a performance counter-based temperature model which can model system temperature while an application is running [20]. Such a model could be used to guide changes in frequency similar to TEAPC.

Annavaram, et al., use energy per instruction to guide processor configuration in a four-processor system [21].

A program with a high degree of thread-level parallelism can show a speed-up when run on a system consisting of four low frequency cores rather than a single high frequency core. Although this work does focus on performance improvement, it is applicable only to situations in which the degree of thread-level parallelism varies and the number of available processors can be dynamically varied.

3 Dynamic Processor Overclocking

Although manufacturing technology and circuit design considerations both limit processor clock speed, increasingly the first-order limitation for a particular processor family on a specific platform is power. System vendors set the nominal clock frequency of the processor in a system by using a worst-case test of the machine's power, even thought there are many programs that could run at higher clock frequencies without crossing this power limit. This slack in the power consumption for many workloads creates an opportunity to extract more performance for these workloads by dynamically increasing the processor clock beyond the nominal value dictated by the worst-case test. This paper studies both (a) when is it possible to run at a higher clock speed without exceeding the power limits, and (b) the value of doing so. Using the processor performance counters to measure execution events of the applications, we identify the characteristics that determine the extent of performance benefits from higher clock frequencies and those characteristics that cause the application to become power-limited.

3.1 Definition

Dynamic Processor Overclocking (DPO) increases the processor frequency to values that are within its range of correct execution but which exceed the power limitation imposed on the processor by the system environment. It does so dynamically using the standard mechanisms that are increasingly provided by the power management features of modern microprocessors, and it increases the clock speed only when the power consumed by running the current workload remains below the defined power limit. For many programs and workload mixes, this means that the processor often runs at a higher than nominal frequency since the executing work requires less power than the worst-case workload used to set the nominal frequency.

Figure 1 illustrates the behavior of a system using dynamic processor overclocking with two possible frequency settings, f_1 and f_2, such that $f_1 > f_2$. The top two graphs show that the power constraint is exceeded during portions of the system timeline when the processor uses only f_1 but never when it uses only f_2. The third graph depicts the behavior when the processor dynamically chooses between f_1 and f_2 by picking the higher value of

f_1 whenever the power is low enough and f_2 otherwise.

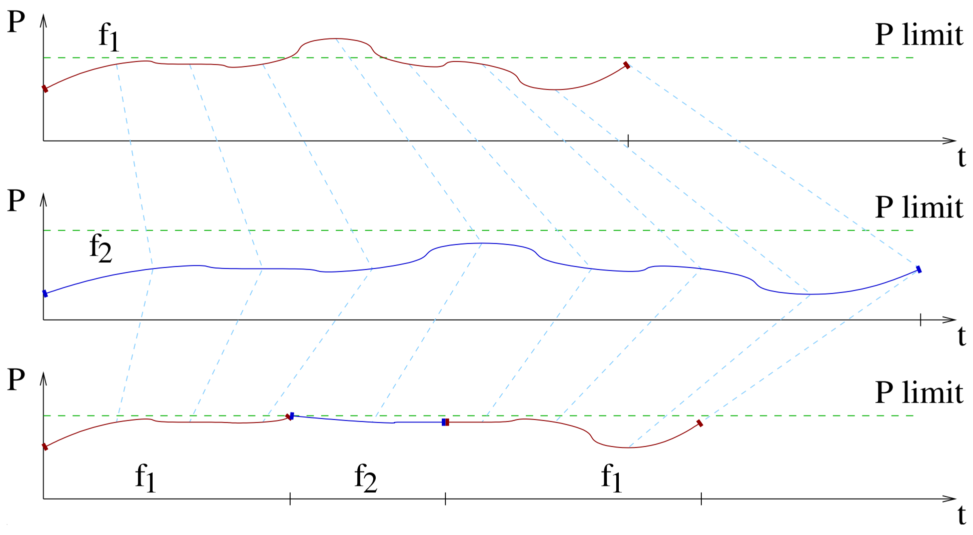

Figure 1: Example of Dynamic Processor Overclocking

.

3.2 Possible Implementations

For a DPO implementation, it is important to correctly determine when it is safe to overclock the processor. This requires some idea of the current power consumption – either a good real-time power feedback or real-time access to good proxies for power. The proxies could be a direct proxy, such as the current consumed by the processor or indirect proxies such as reasonable extrapolations of processor power from certain dedicated performance counter data.

A DPO implementations may be either in-band or out-of-band. An in-band implementation requires the software running on the processor to have access to the real-time power feedback or its proxies. An out-of-band implementation could be either completely in hardware or with additional firmware support. It is easier to implement strong timing guarantees in an out-of-band implementation, improving the reliability of the dynamic processor overclocking mechanism. Such an implementation also avoids the complexity of changing or getting the owners to change a number of different operating system implementations. Montecito's Foxton microcontroller is an example of an out-of-band hardware implementation for dynamic frequency adaptation [7].

4 Evaluation Setup

4.1 Processor and system board

To study the value of DPO, this paper uses a real machine rather than simulation. The specific platform studied here is a Pentium M-based system running Windows XP. The 90 nm Pentium M processor “Dothan” has a 32 KB primary instruction cache, 32 KB primary data cache, and a 2MB, 8-way unified secondary cache [3]. The processor has programmable performance counters to track events such as the number of instructions decoded or memory references during program execution.

The processor supports dynamic voltage and frequency scaling and with drivers developed by us we can adjust the processor frequency between its full operating range of 600 MHz to 2.0 GHz and the corresponding supply voltage range of 0.988 V to 1.34 V; during frequency transitions, the processor stalls for up to 10 μs. The dynamic voltage and frequency adjustment allows the system to operate throughout a range of power and performance levels. The processor chip is paired with an Intel 855GME chipset and 512 MB of DDR SDRAM memory on a Radisys uni-processor motherboard [22].

4.2 Power Measurement

The hardware discussed in this section allows us to measure and record the power consumption of the processor, which is then used to determine what speed the processor can use for the current phase of execution.

We added a sense resistor between each voltage regulator module (VRM) and the processor and placed data acquisition probes to monitor processor supply voltage and current levels. Figure 2 shows the system under test and the ribbon cables that connect the probe points to the data acquisition system. We collect and analyze power data on a separate computer to avoid interference with workloads executing on the system under test. A National Instruments data acquisition system samples current and voltage values and interfaces with a Pentium III system that executes a custom program in LabView software to capture the data in a trace file.

4.3 Performance Measurement

In addition to application execution time, we monitor performance events throughout execution for each benchmark. We developed software for performance data collection that first configures event counters and sets the processor’s frequency and voltage and then spawns the benchmark under test. During benchmark execution, the

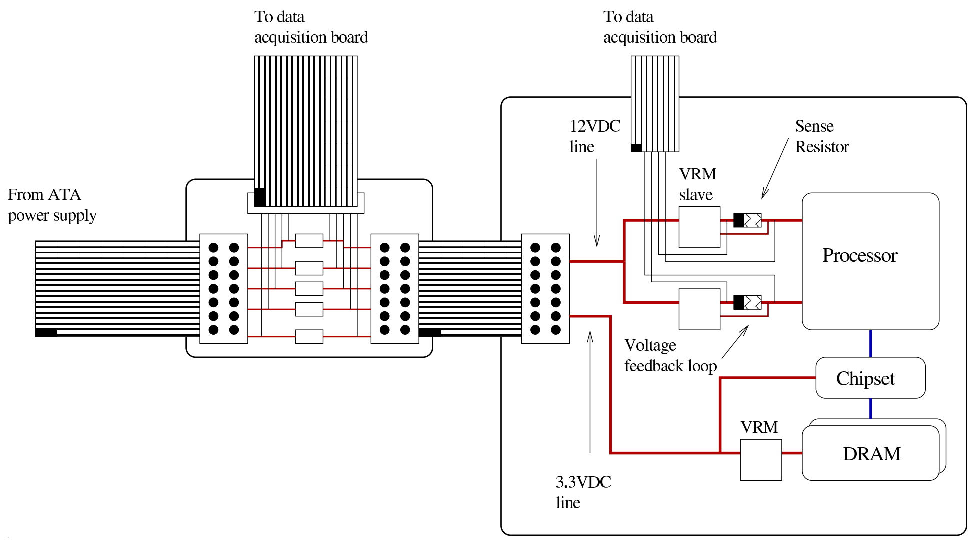

Figure 2: Experimental platform: processor under test with sense resistors and data acquisition probes

Counter Name	Description
DCU_MISS_OUTSTANDING	Number of cycles for outstanding misses to the data cache
RESOURCE_STALLS	Number of cycles with resource-related stalls
L2_LD	L2 cache loads: count prefetched and demand loads
L2_LD	L2 cache loads: count demand loads only
INST_RETIRED	Number of instructions retired (used for IPC)
INST_DECODED	Number of instructions decoded
L2_RQSTS	Number of requests for the L2 cache
BUS_TRAN_MEM	Number of completed memory transactions

Table 1: Performance Counters

software reads performance counter values with the RDPMC instruction at approximately 15 ms intervals (limited by the resolution of the timer mechanisms in Windows XP) and then generates an output file of timestamps and counter values.

The Pentium M performance registers hold two performance counter values. Some counters use additional bits to specify variations of the data collected, such as including or excluding prefetched lines in a cache access count [23]. We executed the benchmark suite multiple times to capture events for a total of eight performance counter types. Table 1 lists the performance counters used in this study. The collection of counters provides a view of processor activity, memory pressure, and efficiency for each benchmark.

4.4 Workloads

We characterized the full suite of SPEC CPU2000 benchmarks, both integer and floating-point applications [24] for this dynamic processor overclocking study. We used the "reference" input set with the `runspec` script supplied by the SPEC organization to run each benchmark in the suite with appropriate options and input files. Some programs, such as `gzip`, are executed multiple times with different input files. In those cases, we follow the SPEC standard of summing the individual runs to produce a total result for the benchmark. The study uses highly optimized binaries compiled for execution on Windows XP.

4.5 Methodology

The experiments reported here use two selected frequencies of 1.8 GHz and 2.0 GHz. These are available, well-defined frequencies on the Dothan part that are easy to select. They are sufficiently far apart to have a noticeable power difference, assuming the appropriate voltage scaling, as well as a measurable effect on the performance of programs that are sensitive only to the clock speed of the processor. But they are close enough together to represent a realistic overclocking across a range of processors and processor families.

Since the machine used in the experiments is based on commercially available parts, we have direct access to information about its design limits. However, for the purposes of some of the experiments that we describe, we determined a power limit using a worst-case methodology. We assume that 1.8 GHz is the nominal frequency and the 2 GHz is the overclocking frequency. A highly optimized version of the Linpack benchmark run at 1.8 GHz provides a value for the assumed processor power limit. The peak power from the worst-case Linpack test of 16.5 W is the power limit used in much of our analysis of dynamic processor overclocking although we report some results with higher and lower power limits.

For each run, the instrumentation gathers 1000 processor voltage and current samples per second. System software adjusts the frequency to the desired value, configures the performance counters, and creates a process that will run the benchmark with the highest user-level priority. This software is also responsible for gathering performance samples of instructions executed and elapsed cycles every 15 ms. It also raises the voltage of a general purpose I/O pin (*marker*) to indicate the power instrumentation that a managed application is running. The voltage of the marker pin will be lowered once the application signals its completion.

Using the marker signal, post-processing software combines the out-of-band power traces and the in-band performance traces into traces with the elapsed time in milliseconds since the marker was raised, the number of instructions in the interval and the power in Watts for the given window. Since both input traces have different sampling rates, this process uses linear interpolation to obtain performance sub-samples that map to the corresponding power samples. The result is a performance-power trace, where each sample represents a window of execution, which is identified by a dynamic instruction count id, incremental time in milliseconds, incremental performance event count, and processor power consumption. To estimate the benefit of DPO, we compare the performance-power trace obtained at 1.8 GHz, with that obtained at 2.0 GHz. We go through both traces using the instruction id as the correlating index. So for each chunk of instructions, we select the 2.0 GHz frequency if the power consumed by those instructions is under the power limit at 2 GHz and 1.8 GHz otherwise. The result is a indication of the time it would take to execute those instructions in a DPO environment. If the frequency is different than the frequency used in the previous interval, we add a penalty of 10 us to account for the change in frequency. This process is repeated for each chunk of instructions in the trace, at which point we obtain the duration of the application.

For the purposes of this study, the post-processing approach has a number of advantages. It does not require additional extensions to Windows XP in the form of other device drivers. It allows the experimental environment to collect performance counter information for explanatory and analysis purposes rather than trying to use it to determine the power. Moreover, the emulation and post-processing approach provides information about the limiting case as it provides a frequency setting based on perfect knowledge of which frequency can be chosen for each sample. With post-processing we are able to consider a larger range of scenarios, including some that may exceed the power limits of the experimental platform. Finally, post-processing reduces the amount of variability and eliminates one potential source of experimental error.

5 Results of Dynamic Processor Overclocking

The Pentium M processor was designed with power consumption in mind, so it can operate very close to its peak frequency without violating most systems' peak power limits. We studied a number of different peak power limits and analyzed performance counter and power information to identify the characteristics which determine system behavior for DPO. We consider the performance gain at different power constraints as well as the amount of time spent at 2.0 GHz for each power limit.

In the discussion that follows, no differentiation is made between SPECCPU2000 integer and floating point benchmarks except in certain situations in which it is necessary to do so to explain the differences in results.

5.1 Results for a Fixed Power Budget of 16.5 W

To understand the advantage of DPO, we used our post-processing code to impose a power limit of 16.5 W. As indicated previously, we selected this value because it is the maximum power consumed by a highly optimized version of Linpack [25], while running at 1.8 GHz. Selecting this value allows us to explore the advantage of dynamic selecting a higher frequency (2.0 GHz) when the applications do not apply a heavy burden on the power and cooling system. Linpack is commonly used as a worst case for power as its function `daxpy` tends to produce a high simultaneous switching activity in most components of the processor.

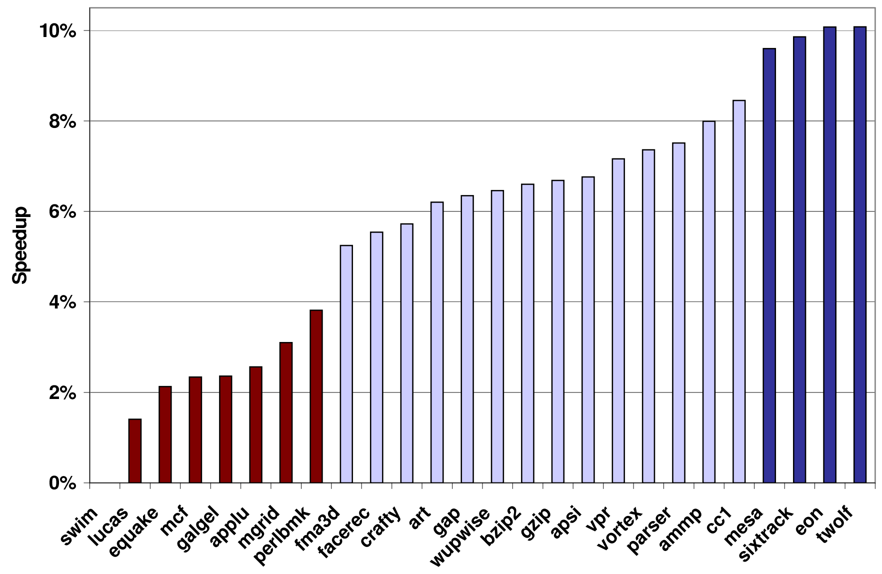

Figure 3: Classification of SPEC CPU 2000 applications based on their speed-ups from Dynamic Processor Overclocking with a 16.5 W peak power limit.

Figure 3 shows the speed-up achieved by using DPO with a 16.5 W power limit. The benchmarks are broken down into three broad classes. The benchmarks in the leftmost class, `swim` through `perlbmk`, show less than a 4% speed-up. The center class covers the benchmarks with greater than a 4% speed-up and less than a 9% speed-up. The benchmarks in the rightmost class see speed-ups greater than 9%. These results indicate that for all applications, except `swim`, it is possible to gain some performance by using dynamic overclocking. In particular, a system in which the nominal frequency is limited to 1.8 GHz by Linpack, DPO improves performance by a small amount.

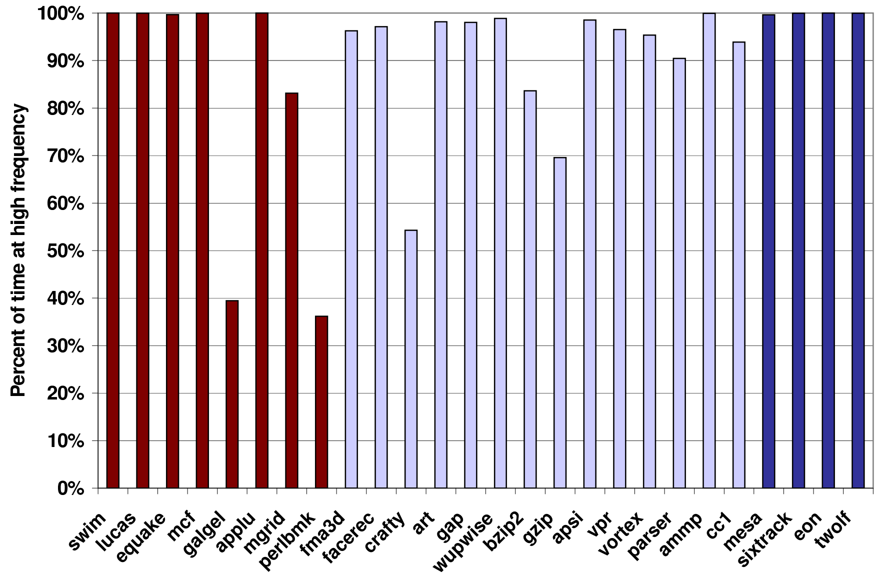

Figure 4: Fraction of the total run that used the 2.0 GHz frequency with a power limit of 16.5 W.

Figure 4 provides a partial explanation for the results shown in Figure 3. It highlights the fact that most benchmarks are able to spend a significant amount of time at 2.0 GHz even with a power constraint of 16.5 W. Applications which spend less time at 2.0 GHz are power-limited. Each category introduced by Figure 3 is analyzed in more detail in the following subsections.

Tables 2 and 3 present the core and memory performance of the SPEC CPU 2000 benchmarks according to micro-architecture metrics. These tables also show the relative ranking of each benchmark, which will help understand the benefit obtained from an increase in frequency.

5.1.1 Low benefit from DPO (below 4%)

Eight benchmarks show little benefit from using DPO at 16.5 W. These benchmarks can be divided into non-power-limited and power-limited programs.

The non-power-limited benchmarks – `swim`, `lucas`, `equake`, `mcf`, and `applu` – spend 99% or more of their execution time at 2.0 GHz, but show little performance gain. These benchmarks exhibit a number of distinct characteristics which separate them from the power-limited group. They typically have a high values for BUS_TRAN_MEM/cycle and RESOURCE_STALLS/cycle. At the same time, `swim`, `equake`, and `mcf` all have

Table 2: Relative ranking of the SPEC CPU 2000 benchmarks in terms of architecture performance events.

DCU_M_O per cycle		RESOURCE STALLS per cycle		INST RETIRED per cycle		INST DECODED per cycle	
art	2.29	mcf	0.876	sixtrack	1.41	crafty	1.67
mcf	1.17	swim	0.845	crafty	1.35	gzip	1.49
swim	1.12	applu	0.795	perlbmk	1.33	sixtrack	1.44
fma3d	1.1	art	0.792	mesa	1.28	perlbmk	1.41
equake	0.89	equake	0.788	vortex	1.21	mesa	1.33
apsi	0.84	lucas	0.703	wupwise	1.19	parser	1.32
applu	0.55	ammp	0.677	gap	1.12	bzip2	1.27
ammp	0.52	apsi	0.671	galgel	1.12	twolf	1.25
mgrid	0.47	fma3d	0.634	gzip	1.1	vortex	1.23
vpr	0.46	mgrid	0.604	eon	1.06	wupwise	1.23
galgel	0.38	facerec	0.518	parser	1.01	gap	1.2
wupwise	0.3	sixtrack	0.48	mgrid	0.97	eon	1.17
vortex	0.28	wupwise	0.463	bzip2	0.95	galgel	1.13
gzip	0.27	galgel	0.455	cc1	0.91	vpr	1.13
bzip2	0.27	gap	0.443	twolf	0.9	cc1	1.09
cc1	0.27	vpr	0.44	facerec	0.9	mgrid	0.97
lucas	0.21	twolf	0.422	vpr	0.75	facerec	0.9
parser	0.19	parser	0.338	ammp	0.74	ammp	0.79
facerec	0.19	vortex	0.336	apsi	0.69	apsi	0.7
twolf	0.18	bzip2	0.311	lucas	0.58	lucas	0.58
gap	0.09	eon	0.309	fma3d	0.55	fma3d	0.57
perlbmk	0.04	mesa	0.277	applu	0.44	equake	0.44
crafty	0.03	gzip	0.271	equake	0.42	applu	0.44
sixtrack	0.01	cc1	0.266	art	0.36	art	0.38
mesa	0.01	crafty	0.134	swim	0.18	mcf	0.24
eon	0	perlbmk	0.097	mcf	0.16	swim	0.18

a high number of DCU_MISS_OUTSTANDING/cycle. These characteristics constrain the benchmarks so that they show sub-linear performance enhancements when the frequency is raised 10% from 1.8 GHz to 2.0 GHz. This result is consistent with those from an earlier study by Kotla et al. [26].

The benchmark `mgrid` is slightly power-limited and is considered separately from both the non-power-limited and power-limited groups. It spends 17% of its time at 1.8 GHz. Despite the fact that it spends the remainder of its time at 2.0 GHz, the non-power limited portions do not scale linearly with frequency because of high DRAM activity, which is on its critical path. L2_LD_demands/cycle match BUS_TRAN_MEM/cycle, which suggests most L2_LD demand accesses see the DRAM latency because not many L2 requests are satisfied in L2. It also has fairly high RESOURCE_STALLS/cycle.

Power-limited benchmarks display different characteristics. `galgel` spends 39% of time at 2GHz for only 2.36% improvement. It has a high instruction throughput combined with high L2 activity which explain its high

Table 3: Relative ranking of the SPEC CPU 2000 benchmarks in terms of architecture performance events.

L2_LD		L2_LD		L2_RQSTS		MEM_RQSTS	
		demand					
per cycle		per cycle		per cycle		per cycle	
art	0.0914	art	0.0565	art	0.0572	swim	0.0122
gzip	0.0438	gzip	0.0301	gzip	0.0305	equake	0.0116
galgel	0.0389	twolf	0.0223	twolf	0.0248	lucas	0.0109
mcf	0.036	galgel	0.0193	galgel	0.0227	mgrid	0.0108
twolf	0.0314	mcf	0.0184	vpr	0.0195	mcf	0.0095
equake	0.0259	vpr	0.0182	mcf	0.0185	applu	0.0094
swim	0.0253	ammp	0.0125	cc1	0.0152	fma3d	0.0076
mgrid	0.0229	parser	0.0121	swim	0.0147	art	0.0068
vpr	0.0223	swim	0.0119	vortex	0.0144	wupwise	0.0064
parser	0.0191	equake	0.0108	parser	0.0131	facerec	0.0061
cc1	0.0189	cc1	0.0107	ammp	0.0128	gap	0.0050
ammp	0.0171	mgrid	0.0105	apsi	0.0128	apsi	0.0039
vortex	0.0143	apsi	0.01	mgrid	0.0121	vpr	0.0033
apsi	0.0138	vortex	0.0092	equake	0.0112	galgel	0.0033
wupwise	0.0126	bzip2	0.0082	bzip2	0.0105	ammp	0.0032
facerec	0.011	fma3d	0.0052	crafty	0.0098	bzip2	0.0029
bzip2	0.011	lucas	0.005	lucas	0.0088	vortex	0.0029
fma3d	0.0103	crafty	0.0046	applu	0.0068	parser	0.0020
lucas	0.01	facerec	0.0045	facerec	0.0061	cc1	0.0019
applu	0.01	wupwise	0.0045	fma3d	0.0061	mesa	0.0011
gap	0.0051	applu	0.0044	wupwise	0.005	gzip	0.0009
crafty	0.005	gap	0.0026	perlbmk	0.0047	perlbmk	0.0006
perlbmk	0.004	perlbmk	0.0025	gap	0.0039	sixtrack	0.0002
mesa	0.0014	mesa	0.001	mesa	0.0018	crafty	2.62E-05
sixtrack	0.0013	sixtrack	0.0005	sixtrack	0.0006	twolf	5.49E-06
eon	0.0004	eon	0.0002	eon	0.0005	eon	2.295E-06

power consumption. It had the second highest rates of L2_REQUESTS/cycle and L2_LD/cycle. `perlbmk` shows a nearly linear speed-up for the time it spends at 2.0 GHz, but it spends only 36% of time at 2 GHz for 3.81% improvement. It has a very high INST_RETIRED/cycle and INST_DECODED/cycle with few resource and data cache unit stalls. These results indicate that it makes very efficient use of the pipeline.

Power-limited benchmarks obviously would benefit from a larger power budget. If the power budget were unlimited, `galgel` would show a speedup of 6.61% and `perlbmk` would show an almost linear speedup of 9.44%.

It is worth noting that the low benefit category is dominated by floating point benchmarks. Only two integer benchmarks appear in this class – `mcf` in the non-power-limited category and `perlbmk` in the power-limited category.

5.1.2 Moderate benefit from DPO (between 4% to 9%)

Fourteen benchmarks fall into the moderate benefit class. These are again separated into non-power-limited and power-limited subclasses.

The non-power-limited benchmarks are similar to the non-power-limited benchmarks for the low benefit class, but generally do not have extreme characteristics or have their performance limited in fewer ways. Some benchmarks have moderately high RESOURCE_STALLS/cycle – `fma3d`, `facerec`, `wupwise`, and `gap` – which limit their performance. `fma3d` and `art` both have moderately high RESOURCE_STALLS/cycle and high DCU_MISS_OUTSTANDING/cycle but are able to achieve moderate speedups despite this. In particular, `art` makes effective use of the L2 cache, thus reducing the number of memory requests. Finally, `cc1`, `parser`, `ammp`, `bzip2`, `vortex`, `vpr`, and `apsi` are able to gain partial benefit from the higher frequency because as they have a relatively lower number of DCU_MISSES_OUTSTANDING/cycle and/or a lower number of RESOURCE_STALL/cycle.

`bzip2`, `parser`, and `cc1` spend less time at 2.0 GHz than the other members of this subclass and could instead be considered as partially power-limited benchmarks. They are, on average, able to achieve an additional 0.47% to 1.1% if provided with unlimited power.

The power-limited benchmarks are characterized by a high number of INST_DECODED/cycle. Both `crafty` and `gzip` show a roughly linear improvement in performance. `crafty` spends only 54% of time at 2 GHz, but gains a 5.72% improvement. `gzip` spends only 63% of time at 2 GHz for a 6.68% speedup. If power were unlimited, `crafty` would achieve a 10% speed-up, while `gzip` improves its performance by 9.37%.

5.1.3 High benefit from DPO (above 9%)

Only four benchmarks show a speed-up of greater than 9% for DPO at 16.5 W, two floating point benchmarks, `mesa` and `sixtrack`, and two integer benchmarks `twolf` and `eon`, spend all but a small fraction of their time at 2.0 GHz and achieve nearly linear speed-ups with the increase in effective frequency. They are not power-limited due to lower number of both INST_DECODED/cycle and L2_RQSTS/cycle. They also suffer little impact from the latency of accessing higher levels of memory hierarchy at 2.0 GHz because they have a low number of low DCU_MISS_OUTSTANDING/cycle and BUS_TRAN_MEM/cycle.

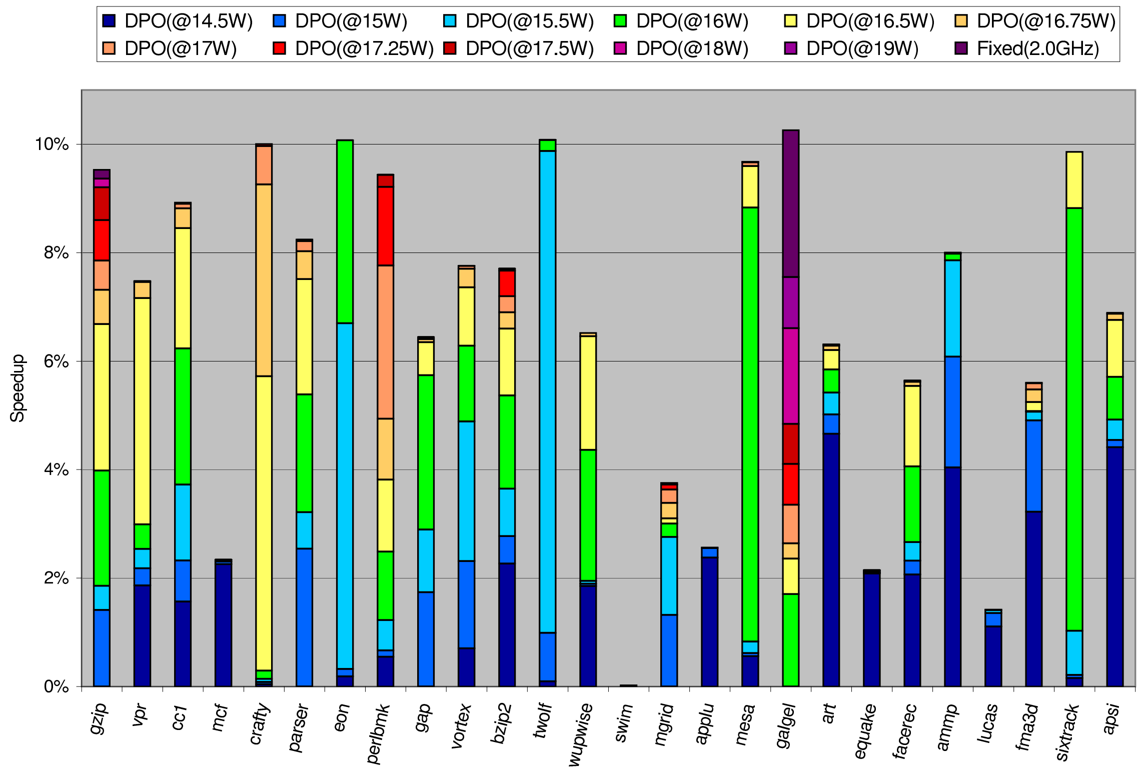

Figure 5: SPEC CPU 2000 performance using Dynamic Processor Overclocking (DPO) with different power limits.

5.2 Impact of Power Budget On Speed-up

Figure 5 shows the speedup which can be gained when DPO is run with different power budgets. Each segment within a column shows the incremental speed-up over the lower power budget. For example, the benchmarks `swim`, `lucas`, `equake`, `mcf` and `applu` showed little speed-up and were not power-limited at 16.5 W. They actually gain most of their performance benefit with a power budget of only 14.5 W. At the other extreme, `galgel` requires a power budget of 19 to equal its maximum speed-up when running at 2.0 GHz with an unlimited budget. In fact, six benchmarks receive a significant performance improvement with a larger power budget – `gzip`, `crafty`, `parser`, `perlbmk`, `bzip2`, `galgel` – but the remainder see only a small or no speed-up. In particular, `gzip`, `perlbmk`, and `galgel` have the property that their performance gain is directly proportional to the relaxation of the imposed power limit.

5.3 Summary

Our results show that the benefits of Dynamic Processor Overclocking (DPO) are closely linked to extent of power constraint on the system and the characteristics of the application. Our analysis is presented in two phases. First, we examined the impact for a specific processor power budget determined as the peak power consumed by a power-intensive workload (Linpack) at the maximum operating frequency (greater than the nominal). This helped us examine in detail the relationships between the application characteristics and dynamic processor overclocking in the context of a power constraint. Next, we examined the impact of the value of the power constraint by examining the benefits for multiple budgets. Our conclusions from both can be summarized as follows:

- The application characteristics that determine the extent to which a workload can benefit from overclocking are related to how efficient the application makes use of the memory hierarchy and processor pipeline. Applications that are impacted by the higher latency of the higher levels of the memory hierarchy and encounter resource stalls when using the pipeline are limited in how much benefit they get out of overclocking, e.g. `swim`, `lucas`, `equake`, `applu`, `mcf`.
- As a corollary of the above, applications that are not constrained by the memory hierarchy characteristics or by resource limitations in the processor pipeline are best suited to take advantage of overclocking, e.g. `eon`, `twolf`, `mesa` and `sixtrack`.
- However, in addition to the above application characteristics the extent of the system-imposed power constraint has a profound impact on the benefit gained from overclocking. For example, `gzip`, `crafty`, `perlbmk` and `galgel` need significantly relaxed power constraints to attain their true benefit from overclocking. These applications are characterized by heavy use of the processor pipeline (high rate of decoded instructions) and the memory hierarchy (high rate of cache requests, not limited by latency of the hierarchy) that increase their power requirements. So while, these could scale their performance with higher frequencies they require additional power margins because of their heavy usage rate leading to higher power consumption.

6 Conclusions

We propose the use of *Dynamic Processor Overclocking* for boosting application performance in power-constrained systems. Dynamic Processor Overclocking (DPO) uses the DVS capabilities of a processor to obtain increased performance for an application by increasing the processor frequency when there is a slack with respect to the power constraint. In contrast to existing systems that use DVS solely for energy savings we adopt a DVS-based solution focused on performance improvement. Our work provides the first evaluation of a DVS-based solution for performance improvement. We evaluated the benefits of Dynamic Processor Overclocking with the SPEC CPU2000 benchmarks on a Pentium M-based platform for a modest exploitation in frequency increase of 11.1% (from 1.8 GHz to 2 GHz). The range of performance improvement ran from 0% to 10.1%. With application performance analysis using the processor performance counters, we are able to identify the application characteristics that determine the extent of performance improvement. We also show the relationship between the level of power constraint and the performance impact. Our work quantifies the value of DPO to tackle the performance limitations imposed by power constraints and provides an analysis approach for estimating the benefits of this solution for any new application.

Acknowledgment

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCH3039004.

References

[1] International Business Machines Corporation, "PowerPC 405LP Embedded Processor Product Brief." www.ibm.com, January 2003.

[2] Intel Corporation, "Intel XScale Microarchitecture Technical Summary."

[3] Intel, "Pentium M processor on 90 nm process with 2-MB L2 cache datasheet." http://www.intel.com/design/mobile/datashts/302189.htm, Jan. 2005.

[4] N. J. Rohrer, et al, "PowerPC 970 in 130nm and 90nm Technologies," in *IEEE International Solid-States Circuits Conference 2004*, February 2004.

[5] C. Lichtenau, et al, "PowerTune: Advanced Frequency and Power Scaling on 64b PowerPC Microprocessor," in *IEEE International Solid-States Circuits Conference 2004*, February 2004.

[6] C. McNairy and R. Bhatia, “Montecito: A Dual-core Dual-thread Itanium Processor,” in *IEEE International Solid-States Circuits Conference 2005*, March-April 2005.

[7] C. Poirier, R. McGowen, C. Bostak, and S. Naffzigr, “Power and Temperature Control on a 90nm Itanium-Family Processor,” in *IEEE International Solid-States Circuits Conference 2005*, March-April 2005.

[8] K. Govil, E. Chan, and H. Wasserman, “Comparing algorithms for dynamic speed-setting of a low-power CPU,” in *Proceedings of the ACM International Conference on Mobile Computing and Networking*, pp. 13–25, November 1995.

[9] T. Pering, T. Burd, and R. Brodersen, “Dynamic voltage scaling and the design of a low-power microprocessor system,” in *Power Driven Microarchitecture Workshop, attached to ISCA98*, June 1998.

[10] D. Grunwald, P. Levis, K. Farkas, C. Morrey, and M. Neufeld, “Policies for dynamic clock scheduling,” in *Proceedings of the Symposium on Operating System Design and Implementation*, October 2000.

[11] J. Lorch and A. Smith, “Improving dynamic voltage scaling algorithms with PACE,” in *Proceedings of the ACM SIGMETRICS 2001 Conference*, June 2001.

[12] K. Flautner and T. Mudge, “Vertigo: Automatic Performance-Setting for Linux,” in *Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 105–116, December 2002.

[13] Bishop Brock and Karthick Rajamani, “Dynamic Power Management for Embedded Systems.” IEEE International SOC Conference, September 2003.

[14] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar, “Critical power slope: Understanding the runtime effects of frequency scaling,” in *Proceedings of the International Conference on Supercomputing (ICS)*, 2002.

[15] W. Felter, K. Rajamani, C. Rusu, and T. Keller, “A performance-conserving approach for reducing peak power consumption in server systems,” in *Proceedings of the 19th ACM International Conference on Supercomputing*, June 2005.

[16] R. Kotla, S. Ghiasi, T. W. Keller, and F. L. Rawson, “Scheduling processor voltage and frequency in server and cluster systems,” in *Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS 2005)*, April 2005.

[17] S. Ghiasi, T. W. Keller, and F. L. Rawson, “Scheduling for heterogeneous processors in server systems,” in *Proceedings of the International Conference on Computing Frontiers (CF 2005)*, May 2005.

[18] A. K. Uht and R. J. Vaccaro, “TEAPC: Adaptive computing and underclocking in a real PC,” in *Proceedings of the 1st Watson $P = ac^2$ Conference*, Oct. 2004.

[19] A. Weissel and F. Bellosa, “Process cruise control: Event-driven clock scaling for dynamic power management,” in *Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES 2002)*, pp. 238–246, October 2002.

[20] K. S. Kyeong-Jae Lee, “Using performance counters for runtime temperature sensing in high-performance processors,” in *Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS 2005)*, April 2005.

[21] M. Annavaram, E. Grochowski, and J. Shen, “Mitigating Amdahl’s law through EPI throttling,” in *Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA-32)*, 2005.

[22] Radisys, "Endura LS855 product data sheet." http://www.radisys.com/oem_products/ds-page.cfm?productdatasheetsid=1158, Oct. 10 2004.

[23] Intel Corporation, "IA-32 intel architecture software developer's manuals, volume 3." Document Number: 25366816.

[24] Standard Performance Evaluation Corporation, "SPEC CPU2000 v1.2," Jan. 2 2002. http://www.spec.org/cpu2000/.

[25] A. Petitet, C. Whaley, J. Dongarra, and A. Cleary, "HPL - a portable implementation of the high-performance linpack benchmark for distributed-memory computers," tech. rep., Univeristy of Tennessee, Jan. 20 2004.

[26] R. Kotla, A. Devgan, S. Ghiasi, T. W. Keller, and F. L. Rawson, "Characterizing the Impact of Different Memory Intensity Levels," in *Proceedings of the IEEE International Workshop on Workload Characterization (WWC-7)*, October 2004.