POLITECNICO DI MILANO
Master of Engineering of Computing System
Department of Electronic and Information

Investigation of Power Capping Techniques for better Computing Energy Efficiency

NESCT LAB
Politecnico di Milano

Relator: Marco D. Santambrogio
Correlator: Giovanni Squillero
Davide B. Bartolini
Filippo Sironi

Graduation Thesis of:
Qianwen Gao, matricola 780391

Accademic Year 2013 - 2014

Abstract

Green computing is becoming a very hot topic. To achieve environmentally responsible use of computers and related resources, improving energy efficiency of computing is very crucial. Power capping techniques provides the possibilities of energy saving for computing devices. As the name implies, power capping refers to the practice of limiting the instantaneous power draw a computing device. It is widely used to fulfill power constraints and save energy. What's more, they are also widely used for power budget management and thermal control for data centers and computing devices.

Three very typical power capping techniques are presented in this thesis. They are: Dynamic Voltage and Frequency Scaling(DVFS), Idle Cycle Injection(ICI), and Clock Cycle Modulation(CCM). To have a systematic understanding of these three power capping techniques and their energy saving potential, I decided to gather run time data during the execution of benchmarks with each power capping technique. I recorded the power trace, the energy consumption and the execution time of each chosen benchmark with each power capping technique. Then, these real runtime data are analyzed to give a view of the characteristics of different power capping techniques. The comparison of their energy efficiency is particularly stressed out. I also investigated the reason of their different behavior and tried to conclude them generally.

From the data gathered, I found that DVFS, ICI and CCM are with very different features and they have different effects on the total energy consumption of the workload. After analysis and comparison, I concluded that DVFS is the only power capping technique provide the possibility of energy saving. ICI doesn't have significant influence on the total energy consumption of a workload. Using CCM will consume more energy.

However, due to the different control features of them, they are suitable for different applications. To further investigate their usage, I studied the related works using these three power capping techniques, then summarized the methods and mechanisms according to different applications. For each power capping technique, I summarized its advantages and discussed its possible future works for better utilization.

This work is able to provide a clear comparison of DVFS, ICI and CCM and give a hint for the future development of power management systems to achieve the the best use of different power capping techniques with better energy efficiency. Researchers could consider the characteristics of different power capping techniques, select the most appropriate ones according to their situations. Companies in IT industry can replicate the methods mentioned in this work to gather data about their workload, find their own best power capping configuration and benefit from it.

This thesis is structured as follows:

- Chapter 1 gives some background knowledge of the power capping techniques. I presented the challenges regarding to power and energy using in different areas of computer science and how to use power capping techniques to cope with them. I gave a typical example of industry use of power capping and state the reason to chose to focus specially on power capping technologies of processors. The three different technologies analyzed in this work are also introduced in this chapter.

- In chapter 2, the methods used in this work will be described in detail including the system environment, the benchmarks, the measurements and the statistic method, and how to apply power capping with the help of different tools.

- Chapter 3 presents the results of the experiments. The data are analyzed and evaluated. From the data, I extracted and summarized the features of different power capping techniques and drew the conclusion systematically.

- In chapter 4, existing solutions from different researches and industrial applications are analyzed with respect to the experiment results. I stressed out how to exploit the advantages of all the power capping techniques and tried to look into the future improvement of them. A state-of-art energy saving solution is presented as an example of using power capping techniques to improve energy efficiency of computing.

- Chapter 5 concludes the work and its limitations to give a direction of how to improve this work to have more general results.

Contents

1 Introduction 1

1.1 Research background and motivations . . . 1

1.2 Different challenges and solutions . . . 2

1.3 Power capping on single machine . . . 4

1.4 Importance of the processor . . . 6

1.5 Power capping techniques . . . 7

1.5.1 Dynamic Voltage and Frequency Scaling . . . 7

1.5.2 Idle Cycle Injection . . . 10

1.5.3 Clock Cycle Modulation . . . 11

1.5.4 Comparison of general features . . . 12

1.6 Assumptions before experiment . . . 12

2 Setup of the experiments 15

2.1 System environment . . . 15

2.2 Benchmarks . . . 17

2.2.1 Integer Benchmarks . . . 17

2.2.2 Floating Point Benchmarks . . . 18

2.3 Measurement, statistic and repetition . . . 20

2.4 Tools for power capping . . . 21

2.4.1 DVFS . . . 21

2.4.2 Idle cycle injection . . . 23

2.4.3 Clock cycle modulation . . . 24

3 Results evaluation and analysis 25

3.1 Experiment results . . . 25

3.1.1 DVFS . . . 26

3.1.2 Idle cycle injection . . . 32

3.1.3 Clock cycle modulation . . . 37

3.2 Comparison of different techniques 43
3.2.1 Influence on power 43
3.2.2 Influence on energy 44
3.2.3 Influence on performance 45
3.3 Further experiments 47
3.3.1 DVFS 47
3.3.2 Idle cycle injection 48
3.3.3 Clock cycle modulation 50
3.4 Conclusion 50

4 Related work of power capping techniques 53
4.1 Thermal control 54
4.1.1 DVFS thermal control 54
4.1.2 ICI thermal control 55
4.1.3 CCM thermal control 56
4.2 Energy saving 57
4.2.1 Extend battery life for mobile devices 57
4.2.2 Improve energy efficiency of HPC devices and data centers 58
4.3 Power optimization and allocation 60
4.4 Future work 61
4.4.1 Proper use of each power capping technique 61
4.4.2 A state-of-art energy saving solution 62

5 Summary 65

Bibliography 66

Chapter 1

Introduction

In this chapter, the background knowledge of the work is introduced. To understand the industrial usage of power capping, I investigated the challenges for achieving energy efficiency in warehouse-scale data centers, high performance computing devices and mobile devices. Power capping techniques can be applied in different ways to cope with these challenges. With a real life example, the reason to focus on the case of single server is stated. With statistics of power and energy consumption of a single server, I decided to emphasize improving energy efficiency of the processor. In the last section of this chapter, three power capping techniques for the processor will be introduced in details. Qualitative comparison between the power capping techniques will be made. Assumptions are proposed, and they need to be verified by the data of the experiments.

1.1 Research background and motivations

Energy Conservation is unquestionably of great importance to all of us, since we rely on energy for everything we do every single day. Energy supplies are limited and energy production is not always using renewable resources. A large portion of the energy we use is derived from fossil fuel.

According to the report[1] issued by EPA (Environmental Protection Agency), servers and data centers consumed 61 billion kWh in 2006. According to the prediction of the report, the energy consumption should be double by now. The amount of energy consumed by mobile and desktop computing equipment is definitely not negligible[2]. Globally, 488 million smart-phones, which have capabilities similar to computers, were sold in 2011, up from 174 million in 2009. From 2005 to 2009, the proportion of primary household computers that were laptops doubled, rising from 22% to 44%. More than 14% of primary household computers were used 10 hours or more a day in 2009. With the increasing energy consumption year by year, IT industry is at the forefront of the growing and highly intense public debate over power consumption and carbon footprints. Recognition of the importance of power and energy in the field of computing systems has never been greater. Energy efficiency means using less energy to accomplish the same task. Increasing energy efficiency of computing is beneficial in many ways.

From the point of view of the IT companies, energy costs have begun to eclipse the cost of physical hardware[3]. Reducing energy bill is a great way to lower operating costs and increase a company's bottom line. Energy savings return on investment, can add up very quickly. To maintain a healthy and sustainable development of IT technology, a company must find ways to use energy wisely. Improving energy efficiency of computing devices is therefore a very meaningful area to research on.

Power capping attracted a lot of research efforts as a widely adopted method for power allocation. There are some researches[4][5] aimed at promoting energy efficiency with power capping techniques. Experiments[6] had been done to prove DVFS (A power capping technique) can improve energy efficiency of processor, but there exist few research mentioned and compared the energy efficiency of other techniques such as idle cycle injection and clock modulation. Therefore, we wanted to gather real data of using these techniques, compare their performance, try to figure out their characteristics and analyze their usage systematically.

According to many previous researches[7][8], running at the maximum computing capacity or race-to-idle is the best strategy to achieve better energy efficiency. However, the processors are always evolving, it is not necessarily true that race-to-idle is always the best. It will be meaningful to verify this conclusion under the latest hardware and computing architecture.

1.2 Different challenges and solutions

Energy efficiency and power management are always very crucial topics no matter for warehouse-scale data centers, high performance computing device or mobile devices. For different situation, power capping can be applied in different manners to cope with their specific challenges.

Power management and energy efficiency is a challenge for modern enterprise data centers and cloud computing systems. Companies continue to increase computing capabilities to meet their growing business requirements[9]. Warehouse-scale datacenters host popular online services such as searching, social networking, webmail, video streaming, online maps, automatic translation, big-data analytics, and storage platforms. In the past ten years, operators have scaled the capabilities of cloud services by building larger data centers that can host tens to hundreds of thousands of multi-core servers[10]. However, the greatest immediate concerns about high-density servers are their power requirements[11]. Data centers require large amounts of power but the power budgets are always limited. Therefore, most of modern computing centers have to leverage power capping on individual machines and use a central power management system to enforce power constraints. In economic point of view, electronic bills have become a significant expense for today's data centers. Therefore, the energy efficiency of a computing center influences its cost directly. Power capping techniques can be used to increase their energy efficiency. Modern data centers operating under cloud computing model are hosting a variety of applications. For workloads that are not constrained by processor performance(e.g., I/O-intensive workload, memory-intensive workload), it is possible to throttle back the server processor without affecting overall performance. Therefore, energy efficiency is increased and cost of production is reduced.

Power provisioning and energy consumption become major challenges also in the field of high performance computing (HPC). Computer scientists across the globe are working towards creating the first exascale supercomputer[12]. However, energy consumption is likely to prove a major barrier to achieving this. Simon McIntosh-Smith of the University of Bristol, UK, reported at the workshop[13] that the average power consumption of the top ten powerful HPC systems has increased five times over the last five years, while the average power consumption of the top 500 powerful HPC systems has increased more than three times over the same period. Energy costs over the lifetime of an HPC installation are in the range of the acquisition costs. Ignoring power consumption as a design constraint will result in a HPC system with high operational costs and diminished reliability, which translates into lost productivity. At the same time, reduction of power for HPC system is very likely to influence the performance of the machine. For this reason, there are many researches[14][15][16] about trade-off between performance and power consumption of HPC systems according to different application purpose and environment. In these researches, power capping methods are mentioned, experiment and measurement are made to find a good trade-off configuration.

Looking at mobile, battery-powered devices, energy has always been extremely crucial since it directly determines battery life. Battery life is one of the most important issues for customers buying a new mobile handset. Nowadays mobile devices are becoming more powerful by each generation. Music and video players, built-in GPS receivers, high data rate for Internet connection, and high resolution cameras are just a few examples. To support these functions, mobile devices are becoming more energy demanding but at the same time, customers ask for longer operational time. Battery life is the weakest point of smart-phones. According to a battery satisfaction survey[17], only 34.3% of the sample questioned is sufficiently satisfied or satisfied with the battery life of their smart-phones. To cope with this issue, energy saving applications and systems are proposed[18]. According to the usage pattern and runtime information, the mobile device can automatically adjust the power draw due to the processor to reduce the energy consumption. For example, when an application does not need to run at the highest performance, the processor may reduce the frequency and voltage so as to reduce the power consumption. Other components (e.g., LCD, Memory) can be power-capped too with the goal of extending battery life.

1.3 Power capping on single machine

In the case of mobile devices or single HPC device, we need to power cap the components of a single device. For data centers, Power capping can be applied at different levels[19][20] such as single machine level, computing cluster level and computing center level. In the case of data centers, power capping on single server is very important because power capping is a key element for implementing power shifting[21], which is the dynamic setting of power budgets for individual servers such that a global power cap for the cluster is maintained.

To have a better understanding of industrial usage of power capping for data centers, we can take the Cisco Unified Computing System[22] for instance. The entire system consists of power cap groups. A power cap group is a collection of servers that share a common power allocation or budget, which is then applied in a dynamic manner to servers within the group. Power cap groups follow a simple set of rules:

• All servers within a single chassis are part of the same power cap group. There can be multiple chassis in a single power cap group.

• The chassis within a given power cap group do not have to be physically contiguous. All chassis in a power cap group do not have to be connected to the same distribution circuits.

• There is a special server (single point of management) managing the total power budget and dividing the power budget to all the power cap groups.

• Within each power cap group, machines are with different relative priorities set by the service team.

The system structure is shown in figure 1.1.

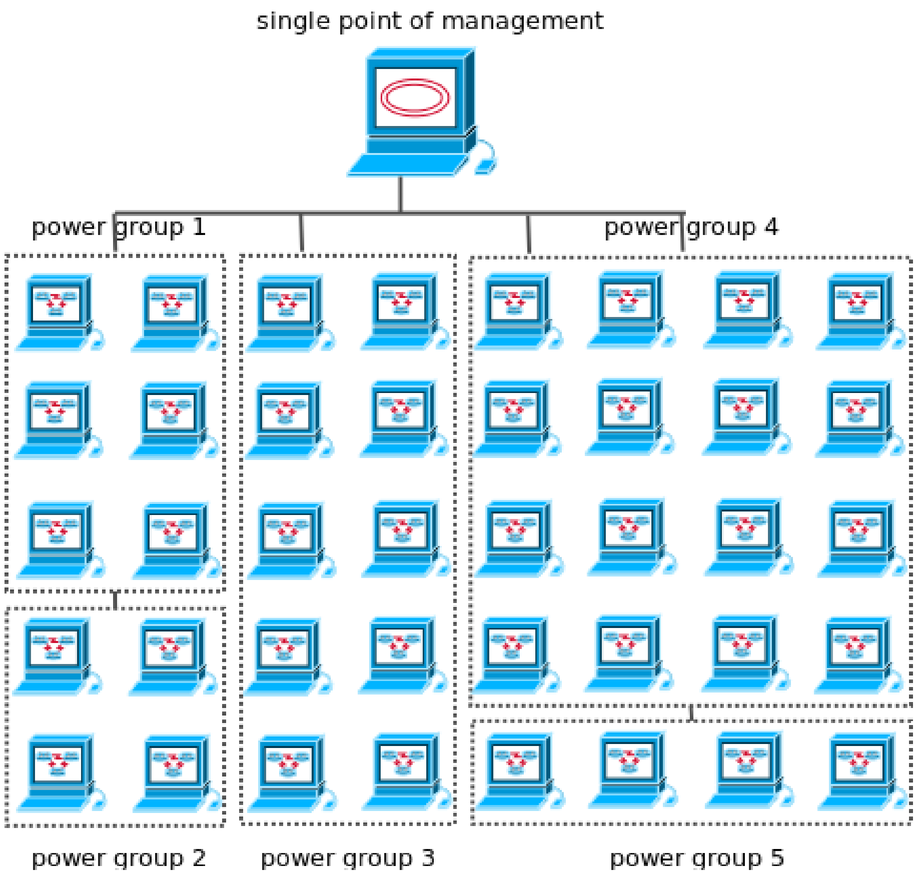


Figure 1.1: Cisco Power Cap Group

From the single point of management, total power budget is divided and power cap is applied at at group level. Then, the service team sets relative priority within power group. The last step is power capping in single machines level according to their priorities.

With this example, we can understand that if we trace down the power capping method for computing centers, power capping on single machine is basic for implementation. Therefore, we decided investigate the power capping of single machine and focus on power capping in component level.

1.4 Importance of the processor

The processor is one of the greatest power consumers of a computing device. In many common configurations, the processor is responsible for one third of the power a server consumes[23] as is shown in figure 1.2.

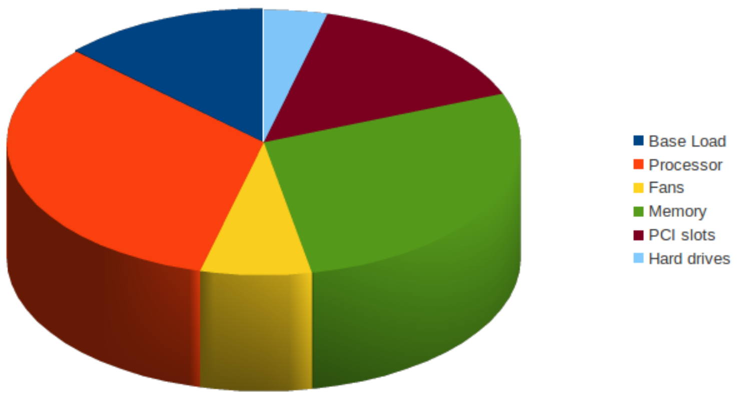


Figure 1.2: Statistic of Server Power Use

The processor also indirectly drives the power consumption of other server components. A busy processor naturally increases the workload of both the memory and peripherals. The heat generated by the increased workload makes the cooling system work harder. Naturally, from the point of view of energy, the processor is usually the component that consumes the most significant portion of the total energy.

In this work, I will focus on the most important component for of energy and power consumption: the processor. I will analyze three well-known power capping technologies for processors which are DVFS, idle cycle injection and clock cycle modulation.

1.5 Power capping techniques

The processor can make use of different power capping techniques to manage its power. With power capping techniques, CPU power can be guaranteed under a certain value for all operating conditions.

Power capping techniques are not only used to manage power, they are also used for thermal control. Thermal control is crucial to real-time systems as excessive processor temperature can cause system failure, performance degradation due to hardware throttling or even permanent damage[24]. Being aware of the importance of thermal control, different solutions[25][26] using power capping techniques are proposed by researchers.

Power capping can reduce CPU's power consumption. However, power capping is not always saving energy[27]. In other words, processors working with less power are not always more energy efficient. From the definition of power and energy, we know they are strictly related with each other. If CPU power can be reduced properly without heavy influence on performance, the energy consumed by the processor can be saved. Therefore, the key point is to find the most suitable power capping technique with the right power restriction level.

1.5.1 Dynamic Voltage and Frequency Scaling

Dynamic voltage and frequency scaling is a power management technique in computer architecture where the voltage used in a component is increased or decreased, the frequency of a microprocessor can also be automatically adjusted "on the fly", depending upon circumstances. Dynamic voltage scaling to increase voltage is known as overvolting; dynamic voltage scaling to decrease voltage is known as undervolting. The voltage required for stable operation is determined by the frequency at which the circuit is clocked, and can be reduced if the frequency is also reduced. Therefore, the voltage and frequency are usually scaled together. DVFS can be used to reduce the active power or reduce the heat generated by the chip.

The general power model[28] of a CPU core with DVFS is:

$$P_{core} = CfV^2 + P_{static}$$

where C is the capacitance of the transistor gates, f is the operating frequency and V is the supply voltage. The decrease of V can lead to a significant reduction in power consumption because of the quadratic relationship shown above. P_{static} consists of various leakage currents. To simplify, we use:

$$P_{core} = P_{dynamic} + P_{static}$$

to present power consumption of a single core. DVFS reduces $P_{dynamic}$ part of P_{core}.

Processors nowadays are mostly multi-core. Therefore, power consumption of the entire CPU package should consider all the cores, also other components other than the cores (e.g., cache). Therefore, the power consumption model of the CPU package become:

$$P_{package} = \sum_{i=1}^{n} P_{core_n} + P_{uncore}$$
$$P_{package} = \sum_{i=1}^{n} \{P_{dynamic_n} + P_{static_n}\} + P_{uncore}$$

where n is the number of cores, P_{uncore}stands for the power consumed by uncore component.This power model will be used for analyzing the behavior of DVFS in chapter 4.

In principle, DVFS for a processor can be performed at various levels of granularity[29] :

• Per-chip DVFS, as shown in figure 1.3a , uses the same power delivery network to reach every core, and consequently, binds each core to the same DVFS schedule.

• Per-core DVFS, illustrated in figure 1.3b, uses a separate voltage regulator for each core and therefore allows every core to have an independent DVFS schedule.

• Cluster-level DVFS, shown in figure 1.3c, uses multiple of on-chip regulators to drive a set of DVFS domains, or clusters, so that one or more cores are associated with each cluster.

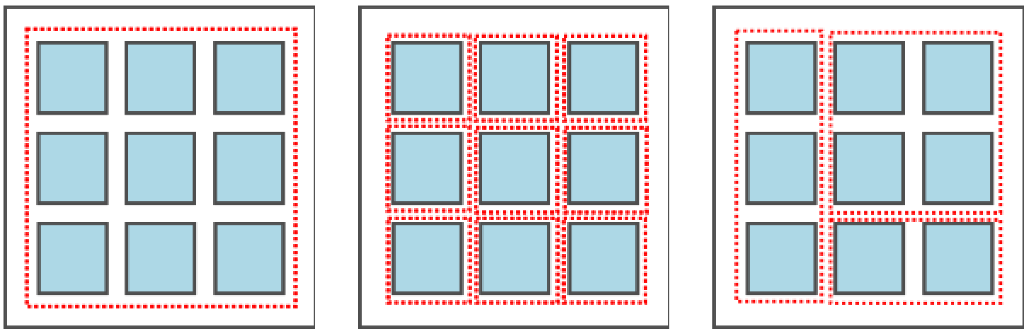

(a) Per-chip DVFS (b) Per-core DVFS (c) Per-cluster DVFS

Figure 1.3: DVFS Levels

For each frequency island, one voltage regulator is required. Per-core and cluster-level DVFS require more than one regulator. Due to the scales involved, it is essential for all of these regulators to be located on-chip. However, on-chip regulators now still incur significant space and power overhead by introducing large inductors and capacitors. Therefore, the majority of commercial processors now provide only per-chip DVFS.

DVFS can lead to significant reduction in the energy required for a computation, particularly for memory-bound workloads. Weiser et al. were the first to propose the use of DVFS to reduce the energy consumption of computer processors.[30]

Other researches[31][32] has attempted to leverage DVFS as a means to improve energy efficiency by lowering the CPU frequency to get a lower running power. But apparently, energy can be saved only if the power consumption is reduced enough to cover the extra time it takes to run the workload at the lower frequency. With the decreasing of operating frequency, the ratio between dynamic power, static power uncore power is also changing. When dynamic power is only a very little portion of the entire CPU power consumption, DVFS will give shrinking potential for saving energy[7]. Therefore, the key point to use this technique for better energy efficiency is to find the configuration where $P_{package_0} \times T_0 > P_{package} \times T$. $P_{package_0}$ is the power consumption of package without any DVFS. T_0 is the execution time accordingly. $P_{package}$ and T are the values after DVFS.

1.5.2 Idle Cycle Injection

Idle cycle injection is a scheduler-level mechanism to force idle the processor during its execution time.

This technique is implemented by setting the processor into C-states. CPU operating states (C-states) are the capability of an idle processor to turn off unused components to save power. When a processor runs in the C0 state it is working. A processor running in any other C-state is idle. Higher C-state numbers represent deeper CPU sleep states with more components shut down and longer wake up time. Table 1.1 gives an overview of the most common C-states.

Mode	Name	Definition
C0	Active	Operational state. CPU fully turned on.
C1	Auto-halt	Stop CPU main internal clocks via software. Bus interface unit and APIC keep running at full speed.
C2	Stop-clock	Stop CPU main internal clocks via hardware. Maintain all software-visible states, but may take longer to wake up through interrupts.
C3	Deep-sleep	Stop all CPU internal clocks. The processor does not need to keep its cache coherent, but maintains other states.

Table 1.1: C-states

Setting the processor into C-states for a certain percentage of its execution time can reduce average power draw and result in cooler but longer execution[33]. Therefore, force idling of the CPU can be used to avoid overheating. This power capping technique provides fine-grained per-thread policy control which allows user to target specific workloads. For instance, the user can target only key heat-producing workloads. As a result, idle cycle injection is the choice of several recently proposed thermal control systems such as ThermOS[34] and Dimetrodon[25].

This technique is relatively new and few research mentioned the energy efficiency of it. Not difficult to tell, CPU in idle status consumes less power.

However, the force idling of CPU will lead to longer execution time of a task, which may increase the total energy needed for complete the task. Therefore, if idle cycle injection can save energy become an interesting question.

1.5.3 Clock Cycle Modulation

Clock Cycle Modulation is a modulation technique that conforms the width of the clock pulse, formally the pulse duration, based on modulator signal information. Its main use is to allow the control of the power supplied to electrical devices.

The phrase 'clock duty cycle' does not refer to the actual duty cycle of the clock signal. Instead it refers to the time period during which the clock signal is allowed to drive the processor chip. By using the stop clock mechanism to control how often the processor is clocked, processor power consumption can be modulated.

Clock duty cycle is expressed in percentage. It describes the proportion of 'on' time to the regular interval or 'period' of time. It can be understood as the percentage of active clock signal of the CPU. Lower duty cycle usually corresponds to lower power, because clock is off for longer time. The average value of voltage and current fed to the load is controlled by turning the switch between supply and load on and off at a fast pace. The longer the switch is on compared to the off periods, the higher the power supplied to the load is.

The mechanism is simple. A signal with 100% duty cycle would deliver 100% of the voltage. It would be like a DC power supply. By applying clock cycle modulation, the area of active clock signal is decreased as shown in figure 1.4. The power delivered to the load is reduced.

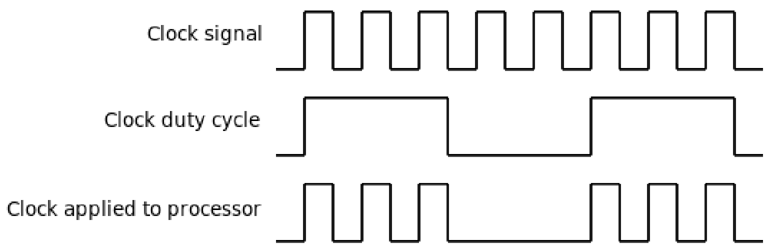


Figure 1.4: Energy saving of CCM

The total power delivered to the connected load each time, is the area under the positive state of the signal. By altering the clock cycle, we can alter the power delivered by the supply. And because the wave form is a square wave, the power supplied each time is calculated as: $P_{delivered} = P_{supplied} \times ClockCycle$

The main use of clock cycle modulation is to allow the control of the power supplied to electrical devices, especially to inertial loads such as motors. This technique can be also used to control the power supply of CPU but it is less mentioned with respect to the previous two techniques. Theoretically, clock cycle modulation will reduce the CPU power and it might increase energy efficiency in some degree. However, there are some informal discussion[35][36] about this technique state that it is hardly useful and doesn't provide significant power saving.

1.5.4 Comparison of general features

These three power capping techniques have different features. CCM is the most easy to be used power capping technique because it can be implemented by simply writing a register. CCM can only be done separately for each core but usually it is done chip-wide. CCM is with very little control latency.

To implement DVFS, a special driver control the core frequency is required. Most of commercial CPUs only provide chip-wide DVFS, but it is possible to have per-chip DVFS. DVFS is with a little control latency.

ICI is the power capping technique with a more complex mechanism. It requires software control. It is a fine grained technique which can be applied to specific threads. It is very flexible. Different strategies can be used for the implementation of ICI. The control latency of ICI is usually not considered because the most important index is the percentage of idle time during the execution.

1.6 Assumptions before experiment

Before the experiments, some assumptions can be made about what can be expected from experiment data. Race-to-idle[8] is an algorithm to improve energy efficiency of processor. If this will happen in our experiments, we may expect that the most energy efficient configuration is always without any power capping.

Regarding DVFS, there are different opinions about if it saves energy. We may expect that race-to-idle is the most energy saving solution for DVFS, or a task running at a DVFS configuration consumes less energy than running at the maximum capacity of CPU.

There is no analysis about energy efficiency of using ICI, but CPU in idle status still consumes energy and using ICI will extend the execution time of tasks. Therefore, we can consume that race-to-idle is more energy efficient than using ICI.

For CCM, the majority opinion is that it is not very effective for power saving and energy saving. We may expect applying CCM to the processor will consume more energy than without.

Chapter 2

Setup of the experiments

In this chapter, the set up process of the experiments will be described in details including the system environment, benchmarks, measurement tools, statistic methods, repetition and the tools we used to leverage power capping on the processor.

2.1 System environment

In this work, all experiments were performed on a host equipped with a CPU of haswell system architecture. The machine is with a quad core (8 threads) CPU: Intel i7-4770K, and 32 GB of RAM memory. The operating system is Debian GNU, Linux kernel 3.12.1. The base Board of the machine is made by ASUSTeK COMPUTER INC, P9D-C Series, Version: Rev 1.xx.

The maximum frequency of CPU is 3.5GHz and the cores provide 15 different frequency levels. I use the configuration of 3.5GHz without any power capping as our standard configuration. Data gathered under this configuration are used to compare with the data gathered under power capping.

During the experiment, I used model specific registers(MSR), a set of various control registers for debugging, program execution tracing, computer performance monitoring, and toggling certain CPU features. Reading and writing these registers is handled by the rdmsr and wrmsr instructions respectively. As these instructions are privileged, they must be executed by the operating system. In our work, MSRs were used for power monitoring, idle cycle injection and clock cycle modulation. On Linux, there is a kernel module called MSR driver to read and write MSRs. The MSR driver is not auto-loaded. On the modular kernel, it is necessary to use the following command: *modprobe msr* to load it explicitly before use.

More details of the processor can be found in table 2.1.

vendor_id	GenuineIntel
cpu family	6
model	60
model name	Intel(R) Xeon(R) CPU E31270 v3 3.50GHz
voltage	1.8 V
external clock	100 MHz
physical id	0
cache size	8192 KB
siblings	8
cpu cores	4
apicid	5
initial apicid	5
fpu	yes
fpu_exception	yes
cpuid level	13
wp	yes
flags	fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm ida arat epb xsaveopt pln pts dtherm tpr_shadow vnmi flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm
bogomips	6983.67
clflush size	64
cache_alignment	64
address sizes	39 bits physical, 48 bits virtual

Table 2.1: Processor information

2.2 Benchmarks

In this work, I chose to use SPEC CPU2006 as the reference benchmark. SPEC CPU2006 is SPEC's industry standardized, CPU-intensive benchmark suite, stressing a system's processor, memory subsystem and compiler. SPEC designed CPU2006 to provide a comparative measure of compute-intensive performance across the widest practical range of hardware using workloads developed from real user applications. It is designed to provide performance measurements that can be used to compare compute-intensive workloads on different computer systems, SPEC CPU2006 contains two benchmark suites: CINT2006 for measuring and comparing compute-intensive integer performance, and CFP2006 for measuring and comparing compute-intensive floating point performance.

Even though some researchers indicate that SPEC benchmarks doesn't represent the real workload of computers since all the benchmarks in the set are single-thread and very compute intensive, which is not always the case in real life, we still decide to use this tool because it is the best to stress the capacity of CPU. In this way, we are able to understand the features of the three power capping techniques more clearly. To investigate the case of multiple-thread, we will run multiple instances of benchmarks on CPU to give a simulation.

To have a more general experiment result, I chose six benchmarks in total, three integer and three floating point with different programing languages and applications.

2.2.1 Integer Benchmarks

The three chosen interger benchmarks are: gcc, h264ref and xalancbmk.

- gcc: C Language optimizing compiler. It generates code for an AMD Opteron processor. The benchmark runs as a compiler with many of its optimization flags enabled. The input files are 9 preprocessed C code (.i files), output files are x86-64 assembly code files. It is written in C.

- h264ref: Video Compression. h264ref is a reference implementation of H.264/AVC (Advanced Video Coding), the latest state-of-the-art video compression standard. This standard replaces the currently widely used MPEG-2 standard, and is being applied for applications such as the next-generation DVDs (Blu-ray and HD DVD) and video broadcasting. h264ref use two different files for input. Both are raw uncompressed video data in YUV-format. The output for the reference run consists of 6 files includes the encode log from the foreman sequence, the output log files and binary format files for all inputs are verified at the end of each run. It is written in C.

- xalancbmk: XSLT processor for transforming XML documents into HTML, text, or other XML document types. This program is a modified version of Xalan-C++, an XSLT processor written in a portable subset of C++ . You use the XSLT language to compose XSL stylesheets. In structural terms, an XSL stylesheet specifies the transformation of one tree of nodes (the XML input) into another tree of nodes (the output or transformation result). This benchmark takes an XML document and an XSL Stylesheet as input. The output is an HTML document. It is written in C++.

2.2.2 Floating Point Benchmarks

The 3 chosen floating point benchmarks are: bwaves, milc, namd.

- bwaves: Computational Fluid Dynamics. bwaves numerically simulates blast waves in three dimensional transonic transient laminar viscous flow. The initial configuration of the blast waves problem consists of a high pressure and density region at the center of a cubic cell of a periodic lattice, with low pressure and density elsewhere. Periodic boundary conditions are applied to the array of cubic cells forming an infinite network. Initially, the high pressure volume begins to expand in the radial direction as classical shock waves. At the same time, the expansion waves move to fill the void at the center of the cubic cell. When the expanding flow reaches the boundaries, it collides with its periodic images from other cells, thus creating a complex structure of interfering nonlinear waves. These processes create a nonlinear damped periodic system with energy being dissipated in time. Finally, the system will come to an equilibrium and steady state. The input file describes the grid size, flow parameters, initial boundary condition and number of time steps. The three data sets, test, train and ref, differ only in grid size and number of time steps. The output are 3 files: the L2 norm of dq vector after final time step, the residual for convergence after each time step and the cumulative sum of iterations for convergence for every time step. The benchmark is written in Fortran language.

- milc: Physics / Quantum Chromodynamics (QCD). The MILC Code is a set of codes developed by the MIMD Lattice Computation (MILC) collaboration for doing simulations of four dimensional SU(3) lattice gauge theory on MIMD parallel machines. The code is used for millions of node hours at DOE and NSF supercomputer centers. The program generates a gauge field, and is used in lattice gauge theory applications involving dynamical quarks. Lattice gauge theory involves the study of some of the fundamental constituents of matter, namely quarks and gluons. In this area of quantum field theory, traditional perturbative expansions are not useful. Introducing a discrete lattice of space-time points is the method of choice. It takes one input file with different parameters and generate one output file used to verify correctness. It is written in C.

- namd: Scientific, Structural Biology, Classical Molecular Dynamics Simulation. s derived from the data layout and inner loop of NAMD, a parallel program for the simulation of large biomolecular systems. namd was a winner of a 2002 Gordon Bell award for parallel scalability, serial performance is equally important to the over 10,000 users who have downloaded the program over the past several years. Almost all of the runtime is spent calculating inter-atomic interactions in a small set of functions. The input file is a 92224 atom simulation of apolipoprotein A-I. The output file contains various checksums on the force calculations. It is written in C++.

2.3 Measurement, statistic and repetition

At run time, a script named meter will log the power consumption of the CPU in cores and the whole package every second. Accumulated energy comsumption of the cores and the package can be gathered by reading the MSR accordingly[37]:
MSR_PP0_ENERGY_STATUS and MSR_PKG_ENERGY_STATUS.

The difference between two consecutive reading after one second is the energy consumed in this second in the unit of Energy Status Units(ESU)[37]. ESU is an unsigned integer represented by MSR_RAPL_POWER_UNIT bit 12:8. Default value is 10000b, indicating energy status unit is in 15.3 micro-Joules increment. The unit of energy consumed in one second should be convert to Joule by dividing the ESU. According to the definition of power, $Energy(Joule) = Power(Watt) \times Time(s)$ the power is of the same number as the energy consumed in one second. The reason to log the power consumption every second is to get the power trace of benchmarks. The total energy consumed by the task in Joule is the sum of the energy consumed in each second of runtime.

The results of experiments will be presented graphically. To achieve a better precision of the tests made, I repeated each test for 10 times. The result of 10 tests of one benchmark in same configuration will be presented together with a boxplot or shown as a cluster. I will compare the execution time, energy consumption, power consumption for the same benchmark in different configurations.

The performance of a benchmark under a specific configuration is defined as:

$$Performance = \frac{T_0}{T}$$

where T is the is running time of benchmark under current configuration and T_0 is the time used when the benchmark is running at the maximum frequency and no power capping.

To compare the energy efficency of different configurations, the performance and energy comsumption of the benchmark will be shown together. The result of different power capping techniques will also be compared to give a macro point of view of their difference. The graphs will be explained in details in Chapter 3.

2.4 Tools for power capping

In this work, I used different tools to leverage power capping on the processor. For each power capping technique, I used different tools. In this section, I will introduce the tools used in details.

2.4.1 DVFS

Dynamic Voltage and Frequency Scaling of a CPU is done by setting it into different power-performance states. They are also called P-states.

While a device or processor operates, it can be in one of these several states. P0 is always the highest-performance state while P1 to Pn being successively lower-performance states with voltage and frequency scaled. The number of P-states is processor-specific and the implementation differs across the various types. Usually it is not more than 16.

CPU frequency scaling is implemented in Linux kernel, the infrastructure is called CPUfreq. Since kernel 3.4 the necessary modules are loaded automatically.

Starting with Linux kernel 3.9, the new P-state power scaling driver is used automatically for modern Intel CPUs. An important tip is that when choosing an appropriate CPUfreq driver, always choose acpi-cpufreq over p4-clockmod. Because p4-clockmod driver reduces the clock frequency of a CPU, but no the voltage. Instead, acpi-cpufreq reduces voltage along with CPU clock frequency.

CPUfreq "governor" is used to manage the frequency setting of CPU. In order to set CPU in different P-states by ourselves, I used the "userspace" P-state governor because this governor allow the user to decide what specific speed the processor shall run at.

The command used to set P-states is: *cpufreq-set -c $CORE -f $FREQ* where *CORE* is the index of the core (from 0 to 7) and *FREQ* is the target frequency in KHz. An important issue need to be noticed is that the CPU used for the experiments provides only per-chip DVFS. Therefore, even if it seems possible to set the cores in different P-states by the command, the real frequency of cores will be modified only if all the cores are set in the same P-state.

The processor provide 15 levels of P-states. I ran benchmarks under all avaliable P-states. The details of the P-states are in the table 2.2.

P state	Frequency
0	3500000KHz
1	3300000KHz
2	3100000KHz
3	2900000KHz
4	2700000KHz
5	2500000KHz
6	2300000KHz
7	2100000KHz
8	2000000KHz
9	1800000KHz
10	1600000KHz
11	1400000KHz
12	1200000KHz
13	1000000KHz
14	800000KHz

Table 2.2: DVFS P-states

2.4.2 Idle cycle injection

Idle cycle injection is done by set CPU into idle states. CPU operating states (C-states) is a set of possible idle states when processor to turn off components to save power. A processor running in C0 state is working normally. A processor running in any other C-state is idle. Higher C-state numbers represent deeper CPU sleep states, more signals and circuits are turned off and more time the CPU will take to wake up. C-states and P-states can vary independently of one another.

On modern Intel processors (Nehalem or later), package level C-state residency is available in MSRs. These MSRs are:
MSR_PKG_C2_RESIDENCY, MSR_PKG_C3_RESIDENCY,
MSR_PKG_C6_RESIDENCY, MSR_PKG_C7_RESIDENCY .

With the support of these MSRs, Linux kernel developers have created an Intel PowerClamp driver, which is an experiment with idle injection for Intel hardware. In our experiment, we used this tool to do the idle cycle injection.

The reason to choose Intel PowerClamp driver is because it introduced the method of synchronizing idle injection across all online CPU threads. It is conceived as such a control system, where the target set point is a user-selected idle ratio based on power reduction. With this tool, forced and controllable C-state residency can be achieved with a single simple command.

The PowerClamp driver is registered to the generic thermal layer as a cooling device. Similar to MSR module, it is necessary to use the following command: *modprobe intel_powerclamp* to load it before use. The command used to inject idle time is:
sudo echo $PERCENT > $PATH/cur_state
Where PERCENT is the percentage of idle time, and PATH is path of the cooling device PowerClamp stands for. Usually it is with the form of:
/sys/class/thermal/cooling_deviceN
N is the number for PowerClamp cooling device, it will appear only after the PowerClamp module is loaded.

To have a clear view of the trend, we ran the benchmarks under 9 different idle ratios which are: 0%, 5%, 10%, 15%, 20%, 25%, 30%, 40%, 50%.

2.4.3 Clock cycle modulation

Pentium 4, Intel Xeon and Pentium M processors also support software-controlled clock modulation.[37] The stop-clock duty cycle is controlled by software through IA32_CLOCK_MODULATION MSR. In this work, CCM is realized by writting this MSR.

In this MSR, bits 0-4 are used to enable software-controlled clock modulation and to select the clock modulation duty cycle. Bit 4 is the flag to enable On-Demand Clock Modulation. CCM will be activated only when the on-demand clock modulation enable flag is set to 1. Bits 1-3 Selects the degree on-demand clock modulation

The possible configuration of CCM is given in the table 2.3.

Duty Cycle Field Enoding	Decimal Presentation	Percentage of active clock signal
000B	0	No CCM (100%)
001B	18	12.5% (Default)
010B	20	25.0%
011B	22	37.5%
100B	24	50.0%
101B	26	63.5%
110B	28	75%
111B	30	87.5%

Table 2.3: CCM configurations

I gathered data in all possible configurations listed.

The command to do clock cycle modulation is:

taskset -c $CORE wrmsr -p $CORE 0x19a $LEVEL

where CORE is the core index from 0 to 7, LEVEL is the level of decimal presentation of the degree of duty cycle modulation. 0x19a is the physical address of IA32_CLOCK_MODULATION MSR.

As we can see from the command, each processor core can modulate to a programmed duty cycle independently. However, since the experiments for DVFS and ICI were both done in the scale of the entire processor. To give a better comparison, I applied CCM for all the four cores .

Chapter 3

Results evaluation and analysis

In this chapter, the result of the experiments are shown graphically. For each benchmark, I presented the total energy consumption of every power capping configuration to find out the most energy efficient one. I also presented the average power consumption of benchmarks to show the power change with different configuration of each power capping technique. I analyzed performance change according to energy and power change for all the techniques to find the technique saves more energy or power with least negative influence on performance. The power trace of benchmarks' will be logged to emphasis the features of DVFS, ICI and CCM.

Since the results I got in one power capping technique for different benchmarks have very similar features, to avoid repeating, for each power capping technique, I took only one benchmark(gcc) to explain in detail. The results of different benchmarks for the same test are very similar. After the experiments, I gave a comparison of performance for all the techniques and discuss the features of them.

To get a more general result, I did more experiments with multi-thread workload and different operating frequency. Finally, I discussed their advantages and disadvantages under different situations and made a final conclusion.

3.1 Experiment results

The results of the first set of experiments is shown in this section. In this set of experiments, I ran one instance of a benchmark on the CPU. For each power capping configuration, I recorded its execution time, power trace and calculated its energy consumption. The purpose is to find the most energy efficient configuration of each technique for a specific workload (the benchmark) and try to have a general idea of their features.

3.1.1 DVFS

Figure 3.1 is about the power usage of gcc under different DVFS P-states. X axis is the frequency in GHz. From left to right, frequencies on X axis are for different P-states from P0 to P14 as indicated in table 2.2. Y axis is the average power consumed by gcc in Watt. Green stands for the average power consumption of the entire CPU package while red stands for only the cores.

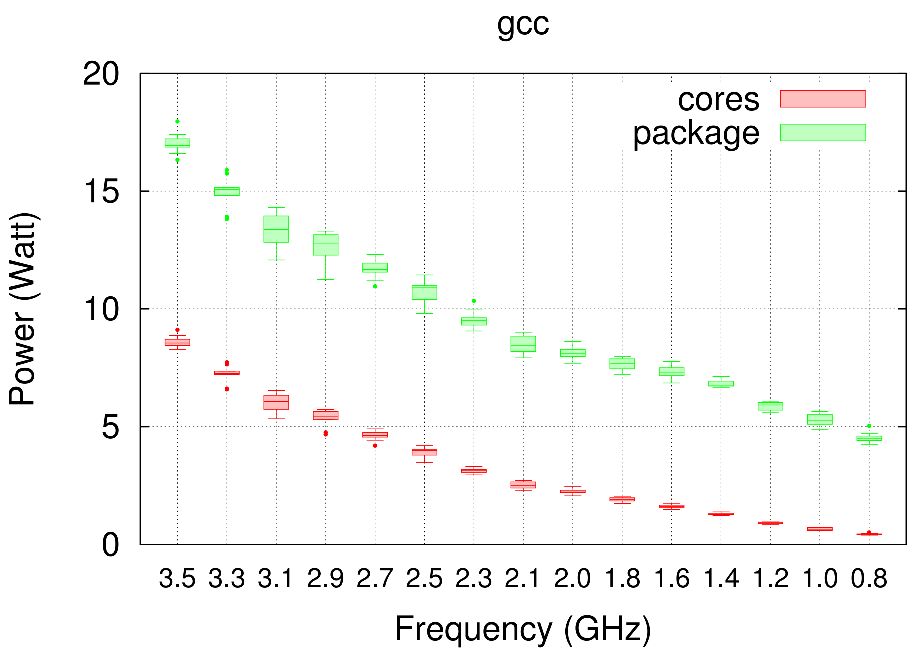


Figure 3.1: gcc DVFS power usage

We can see the power consumption of the package and the cores are both decreasing. Minimum power consumption is achieved at P-state P14 with lowest frequency. According to the power model of DVFS we mentioned before, $P_{package} = P_{core} + P_{uncore}$; $P_{core} = P_{dynamic} + P_{static} = CfV^2 + P_{static}$.

With the decrease of f and V, dynamic power consumed by core is decreased and total power of package decrease in a similar trend.

Figure 3.2 is about the energy usage of gcc under different DVFS P-states. X axis is the frequency in GHz from P-state P0 to P14. Y axis is the energy consumed by gcc in Joule. Green stands for package and red stands for the cores.

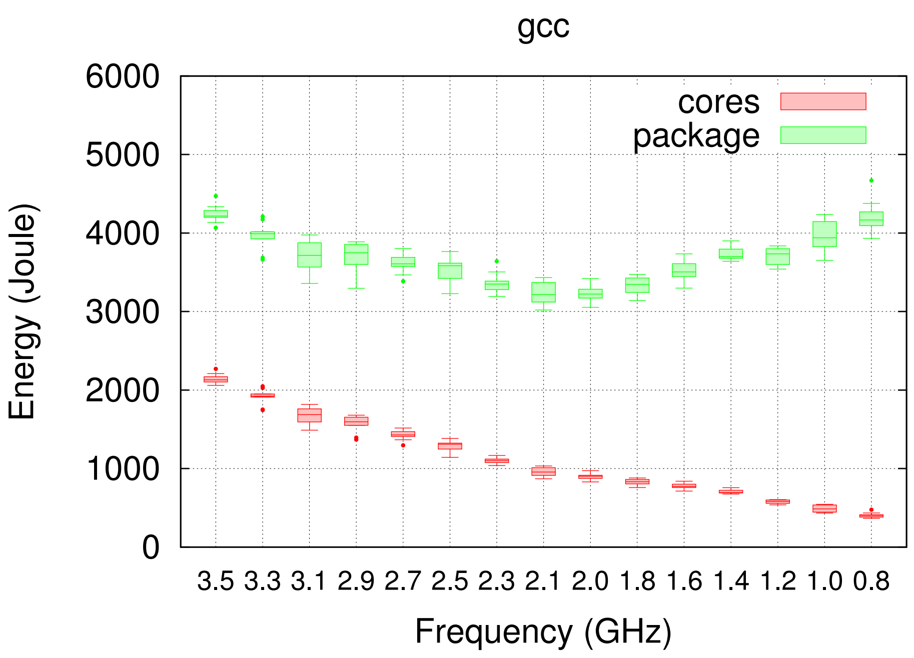


Figure 3.2: gcc DVFS energy usage

From the graph we can see that the energy consumption for package and the cores have different trends. For package, the total energy consumption kept decreasing until the lowest point which is obtained at P7 with frequency of 2.1GHz. Then it went up gradually. For the cores, the energy consumed kept decreasing with the frequency. The lowest energy consumption is obtained with P14.

According to my analysis, the reason to lead this result is following: $Energy = Power \times Time$ If power decrease, the time of execution become longer. For cores, the dynamic power decrease is enough to recover the time spend for longer execution. For the package, after P7, static power and uncore power become much more significant than dynamic power, the power saved by the reducing dynamic power is not enough to recover the longer time. Therefore the total power consumed by the package increased.

Figure 3.3 is performance VS. power. X axis is the power in Watt. Y axis is the performance. 100% performance is achieved at P-state P0, with the least amount of execution time. Performance at other configuration is defined as the ration between the minimum execution time in P0 and the execution time in current configuration.

All the experiment results are listed here in cluster to give a better approximation. From up to down, each cluster is for a P-state from 0 to 14. Red stands for the cores, black stands for the package.

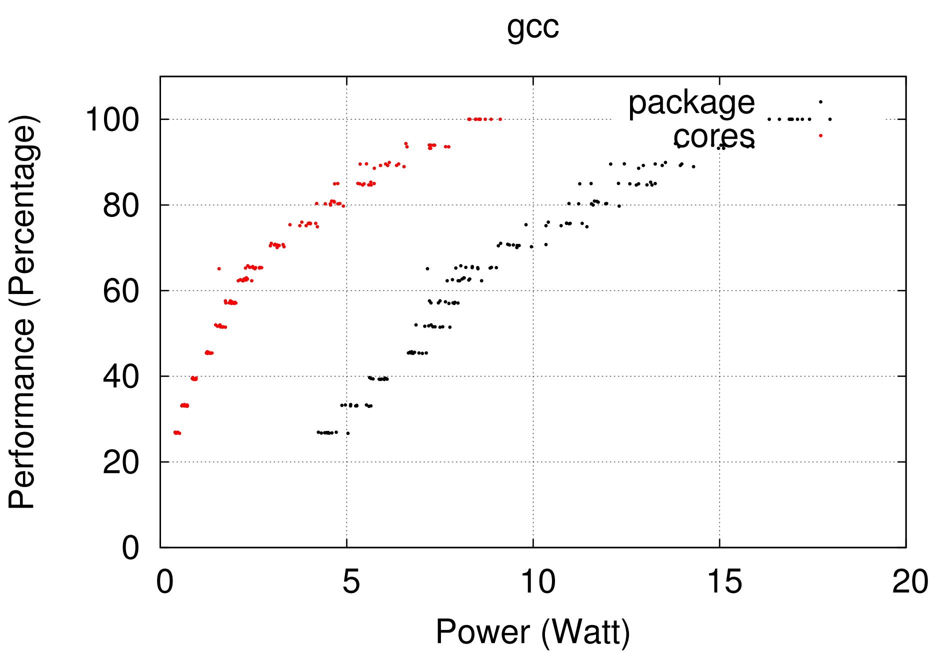


Figure 3.3: gcc DVFS performance to power

We can see clearly that with the growth of power, the performance was improving. With the increase of power, the performance improvement became less significant.

Figure 3.4 is the performance VS. energy. X axis is the energy in Joule, Y axis is the performance. The definition of performance is the same as the previous graph. Red stands for cores and black for the package.

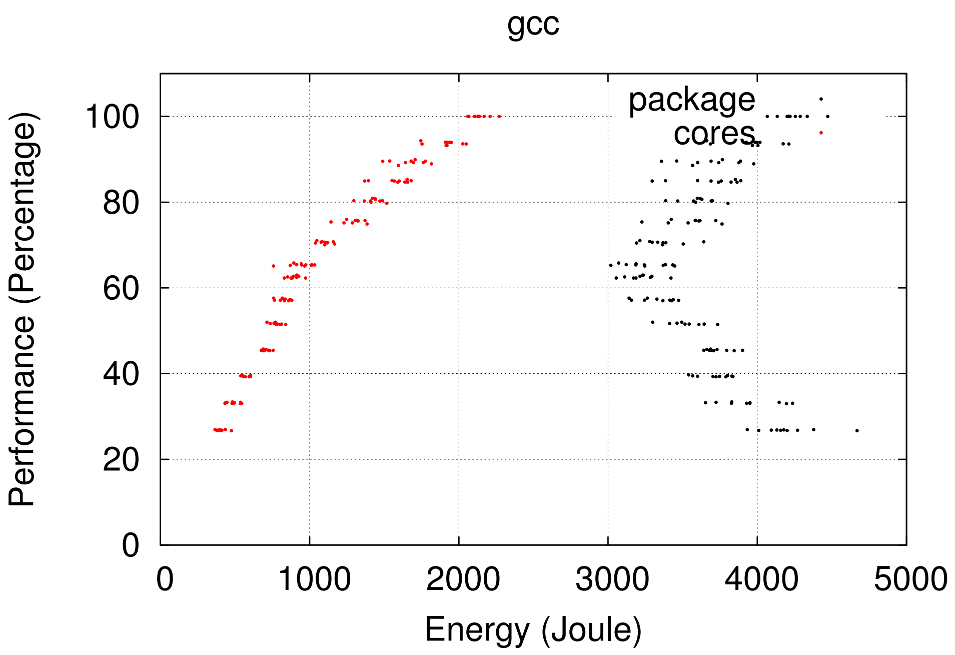


Figure 3.4: gcc DVFS performance to energy

Performance VS. Energy has an interesting shape for the total package. The best energy efficiency point is achieved at P-state P7. We can see that lowering performance doesn't always mean saving energy for the CPU package. If we only consider the cores, running the benchmark at lower performance saves energy.

Figure 3.5 is the power trace of gcc under maximum frequency without any power capping. gcc has an uniformly fluctuating power trace. The benchmarks chosen in this work have different power traces with their own features.

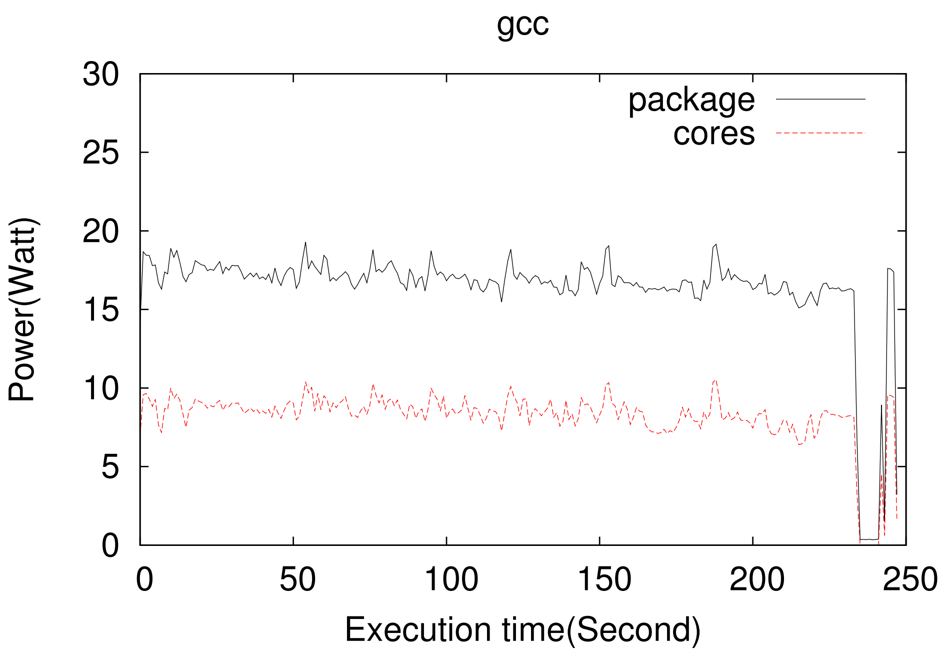
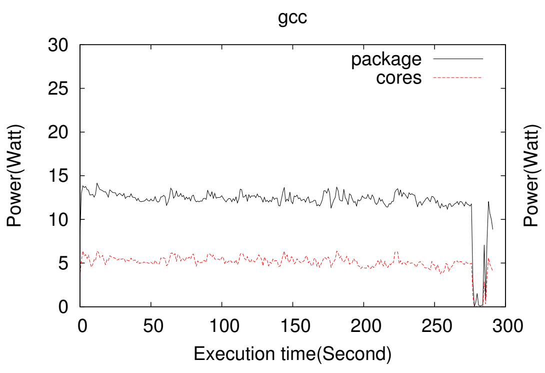
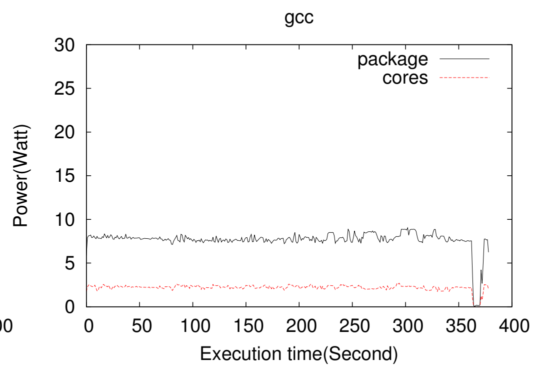


Figure 3.5: gcc power trace at max frequency

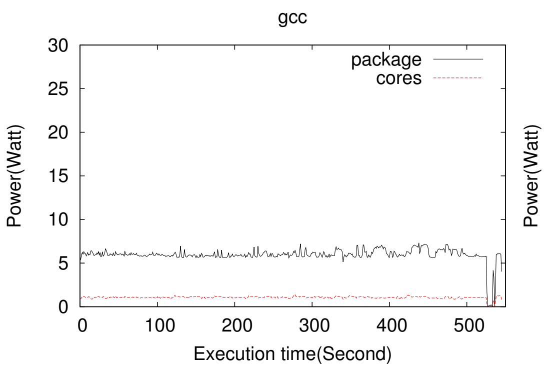
To have a better view of how DVFS affects the power and execution time, figure 3.6 shows the power trace of gcc under different DVFS P-states. Figure 3.6a is the power trace at P3 with frequency 2.9 GHz.Figure 3.6b is the power trace at P7 with frequency 2.1 GHz. Figure 3.6c is the power trace at P11 with 1.4 GHz. Figure 3.6d is the power trace at P14 with 800 MHz.



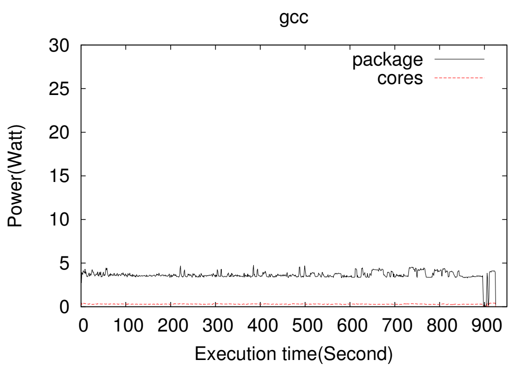
(a) Frequency 2.9 GHz



(b) Frequency 2.1 GHz



(c) Frequency 1.4 GHz



(d) Frequency 800 MHz

Figure 3.6: gcc power trace DVFS

From the graphs, we can see the power trace of gcc became smoother and longer when CPU frequency and voltage were reduced. DVFS affects power consumption of cores greatly. At the lowest frequency, the cores consume very little energy. . From the fact that the gap between power traces for package and the cores was decreased when applying DVFS, we can infer that DVFS also reduce the sum of uncore and static power consumption.

3.1.2 Idle cycle injection

I took gcc as the example benchmark for idle cycle injection. Figure 3.7 is the graph about the power usage. X axis is the percentage of idle time of the CPU. Y is the power consumption in Watt. Red stands for cores and green stands for package. Since all the experiment result are very uniformed, the color is difficult to see. The group of data with higher power consumption value is for the complete package.

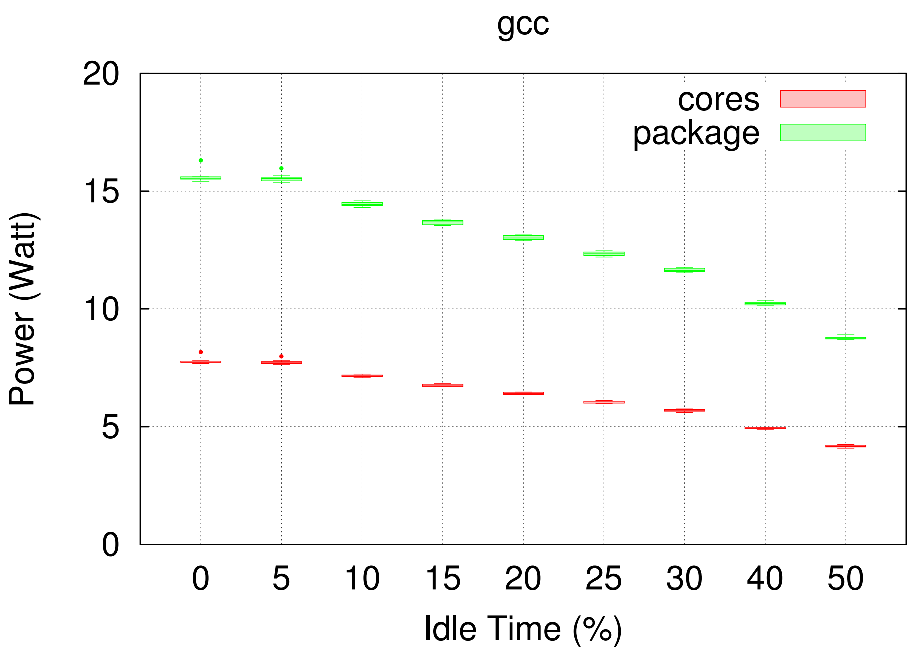


Figure 3.7: gcc ICI power usage

we can see that the power is decreasing gradually with little slopes for both the cores and the whole package. The package power is having a slightly bigger gradient respect to the cores power.

Figure 3.8 is the energy consumption graph for gcc ICI. X axis is the percentage of idle time of the CPU. Y is the energy consumption in Joule. The group pf data with higher energy consumption is for cores while lower one is for cores.

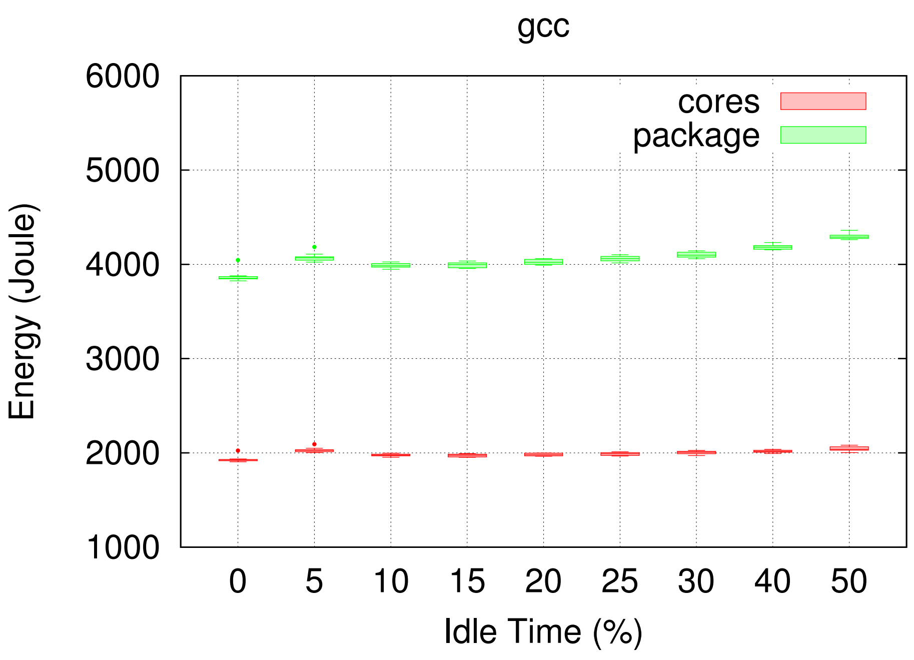


Figure 3.8: gcc ICI energy usage

It is very interesting that for both cores and package, the energy consumption change is very insignificant. The total energy consumed by the benchmark is keeping more or less an equilibrium. For the complete package, there is only a very slight trend of raising. Therefore, I proposed an assumption here, that is for the same workload, same frequency and voltage configuration, the idle cycle injection doesn't affect the total energy consumption of the workload. To further prove this assumption, I launched more tests. Details of more tests is given in section 3.2.

Figure 3.9 is performance VS. power. X axis is the power in Watt. Y axis is the performance. 100% performance is achieved at max frequency without any ICI, the least amount of execution time. Performance at other configuration is defined as the ration between the minimum execution and the execution time in current configuration. With more percentage of ICI, the performance is lower. Each cluster of data is from one same configuration. Red stands for cores, black stands for package.

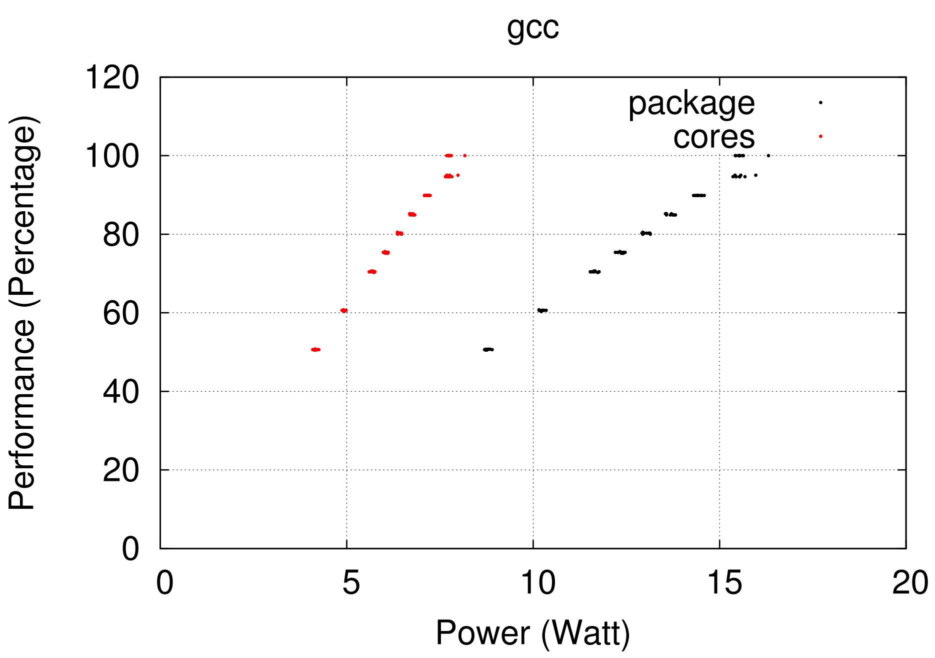


Figure 3.9: gcc ICI performance to power

This is another interesting fact about ICI. With the increasing of power, the performance is growing almost at a linear trend.

Figure 3.10 is the performance VS. energy. X axis is the energy in Joule, Y axis is the performance. The definition of performance is the same as the previous graph. Red stands for cores and black for the package.

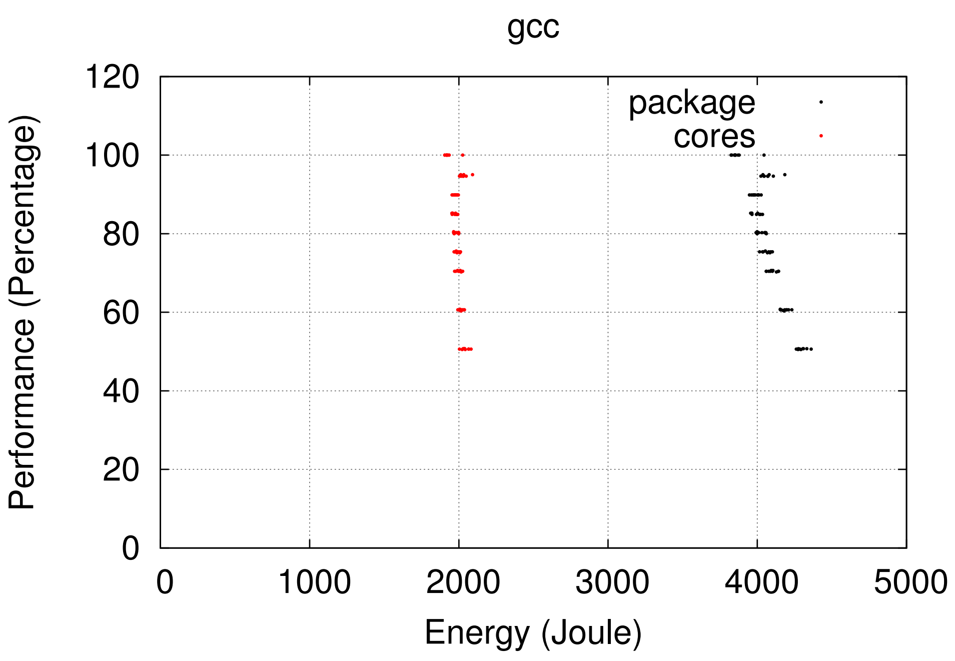


Figure 3.10: gcc ICI performance to power

For ICI, we can see that the each configuration will consume similar amount of energy. The reason for this is that with more percentage of ICI, the execution time become longer and the power become less. The product of them keeps the same. Therefore, to improve a system only for energy efficient reason, ICI is not the best choice.

Figure 3.11 is the power trace of gcc under maximum frequency without ICI. Figure 3.12a is the power trace with 10% of idle cycle injection. Figure 3.12b is with 25% of idle time. Figure 3.12c is with 40% and figure 3.12d is with 50%.

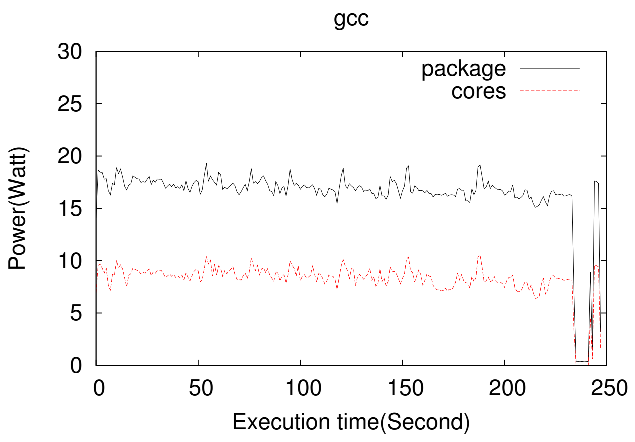


Figure 3.11: gcc power trace at max frequency without ICI

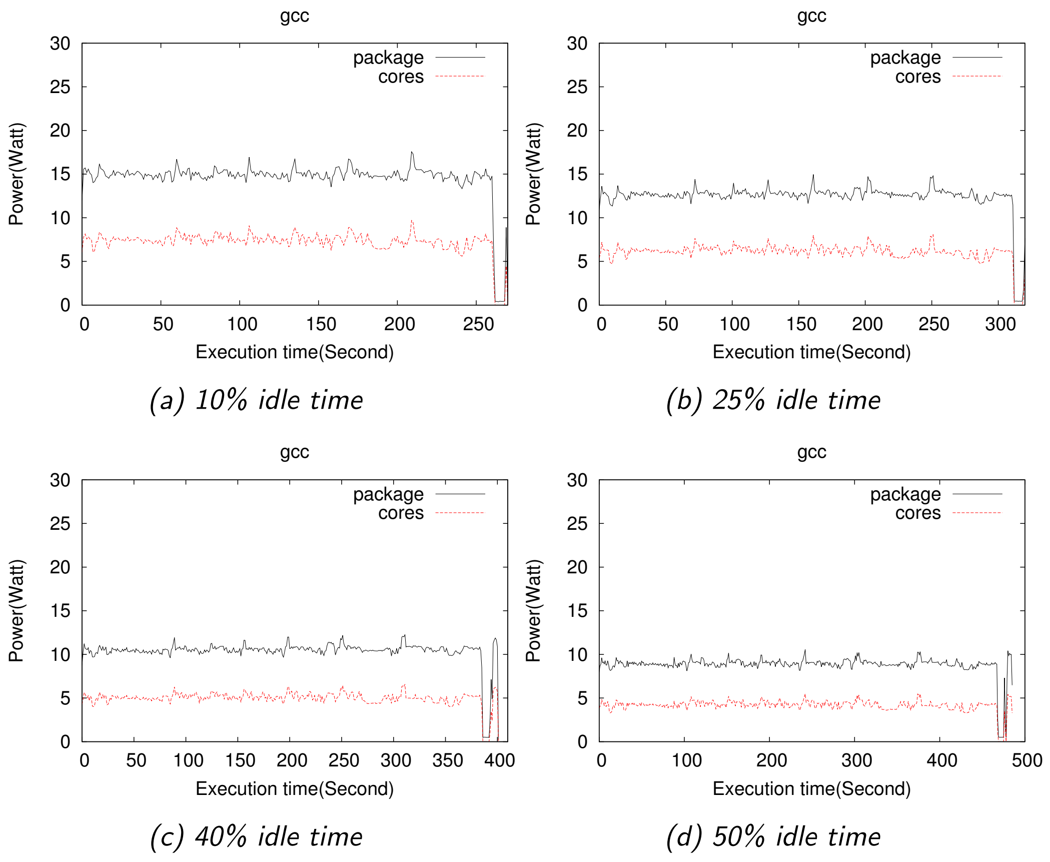


Figure 3.12: gcc power trace ICI

From figure 3.12 gcc power trace of ICI, we can see that ICI reduce the average power consumption of cores and the package. The ratio of average cores power to average package power was roughly fixed for different percentage of idle time. Different from DVFS, the shape of power trace was well preserved rather than significantly smoothed.

3.1.3 Clock cycle modulation

For clock cycle modulation, I took gcc as example as well. Figure 3.13 is its power usage. X axis is the percentage of active clock signal as indicated in table 2.3. Y axis is the power consumed by the benchmark in Watt.

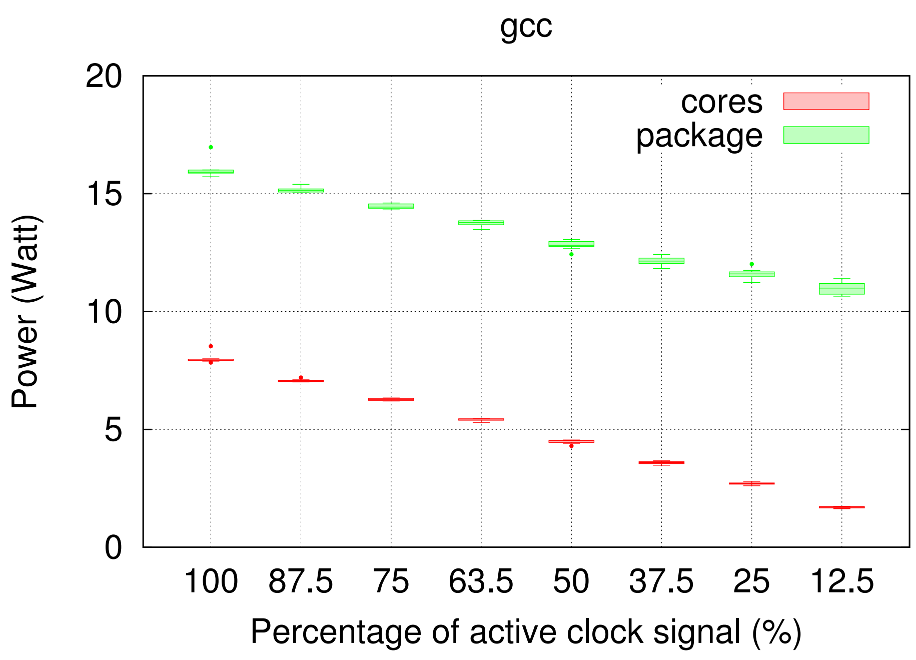


Figure 3.13: gcc CCM power usage

From the graph, we can see the power consumption is decreasing gradually for smaller percentage of active clock signal. The cores and the package have very similar trend.

Figure 3.14 is gcc CCM energy usage. X axis is the percentage of active clock signal. Y axis is the energy consumption in Joule. Green is for package, with higher value than the red for cores.

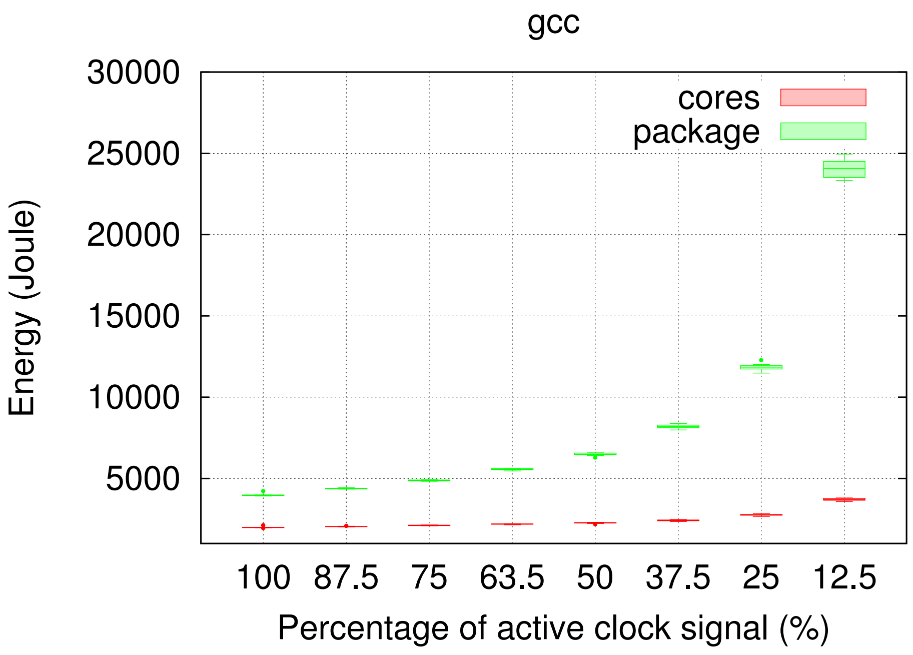


Figure 3.14: gcc CCM energy usage

We can see an interesting shape for the energy consumption. The energy consumed by cores and package are both growing in an exponential fashion. The total change for the cores is not very significant. At the least percentage of active clock signal, the energy consumed by the package increase 5 times more than no CCM. The reason for this result is that with CCM, the time to complete the workload is increasing exponentially. The energy consumption with CCM is increasing instead of decreasing. It is obviously not the best technique to save the energy.

Figure 3.15 performance vs power. X axis is power in Watt and Y axis is the performance in percentage. The minimum execution time is achieved at the highest frequency(3.5GHz) without any CCM. Performance is defined as the ratio between minimum execution time and the current execution time. Performance decrease for smaller percentage of active clock signal and each cluster is for the same configuration. Red stands for cores and black stands for the package.

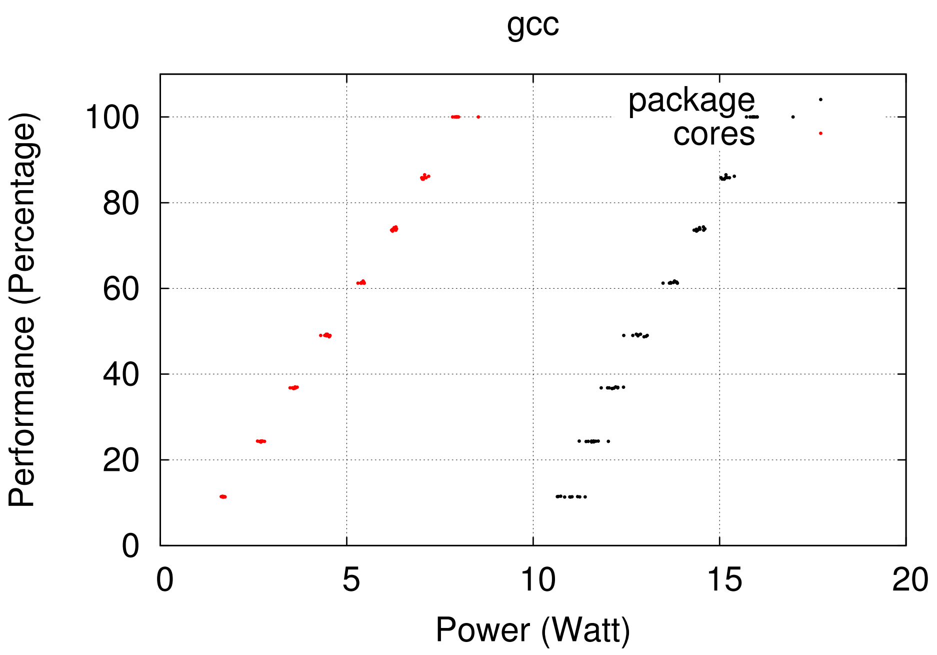


Figure 3.15: gcc CCM performance to power

From the graph, we can see performance increase with power in more or less an linear fashion for both cores and package.

Figure 3.16 is the performance vs energy. X axis is energy in Joule and Y axis is performance defined as before. Performance decrease for smaller percentage of active clock signal. Each cluster is for a same configuration. Red stands for cores, black for package.

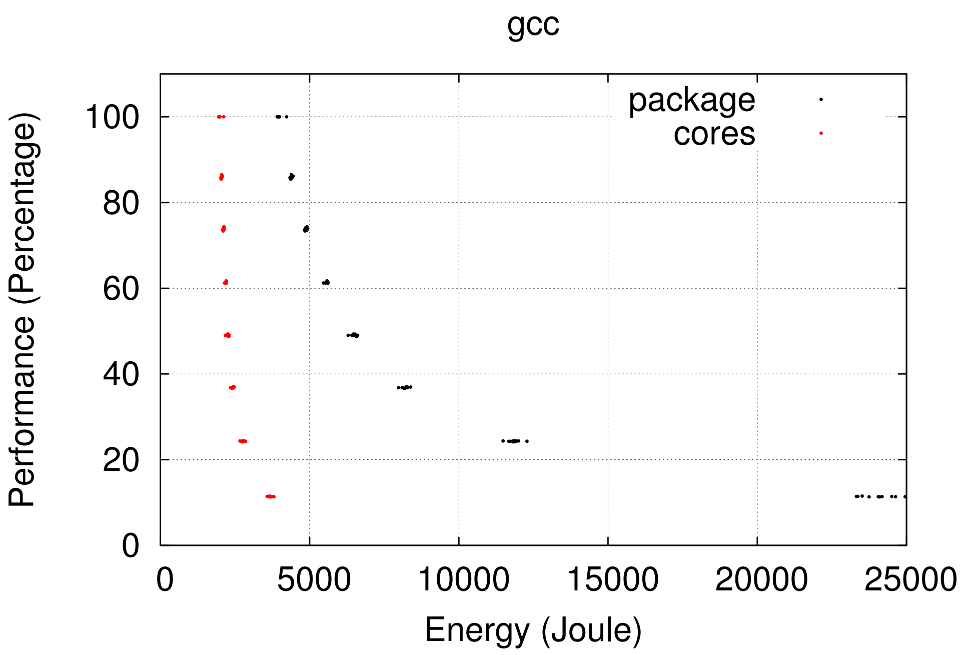


Figure 3.16: gcc CCM performance to energy

From the graph, we can see that for both cores and the package, more CCM will lead to worse performance and more energy consumption at the same time. We can say that in energy efficient point of view, CCM, especially with a very small percentage of active clock signal, is very inconvenient.

The figure 3.17 is the power trace of gcc under maximum frequency without CCM. Figure 3.18a is power trace of gcc under CCM with 87.5% of active clock signal. Figure 3.18b is gcc's power trace with 63.5% of active clock signal. Figure 3.18c is with 37.5% of active clock signal. Figure 3.18d is with 12.5% of active clock signal.

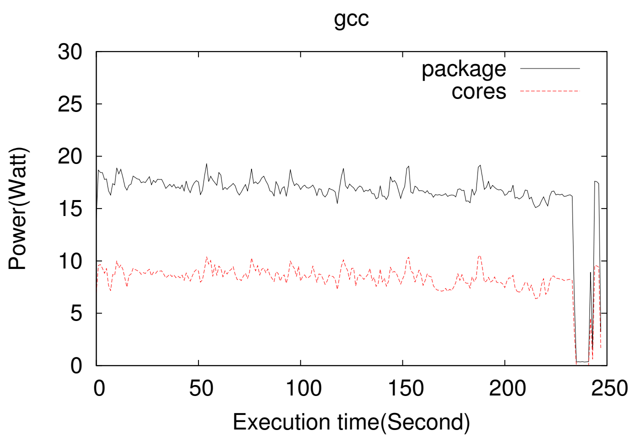
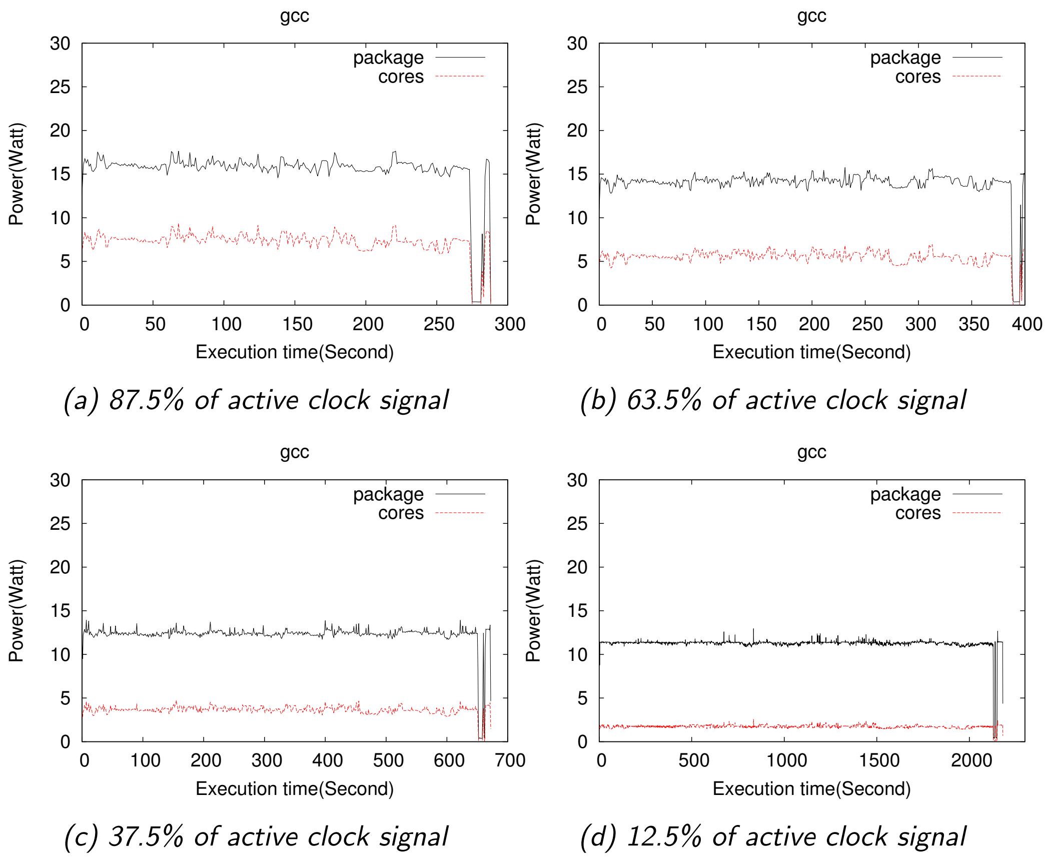


Figure 3.17: gcc power trace at max frequency



(a) 87.5% of active clock signal *(b) 63.5% of active clock signal*

(c) 37.5% of active clock signal *(d) 12.5% of active clock signal*

Figure 3.18: gcc power trace CCM

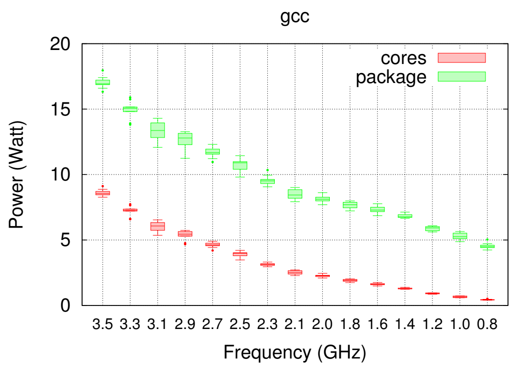
From the power trace of different CCM configurations, we can see that the power trace is smoothed, the gap between the trace of cores and the trace of package become bigger when the percentage of active clock signal is less. From these graphs, we can understand that clock cycle modulation has negative effect on power saving for uncore components of the processor. For uncore components, the energy consumption is significantly increased with CCM because both execution time and power are increased. It's a very important why CCM was not improving the energy efficiency of the processor.

3.2 Comparison of different techniques

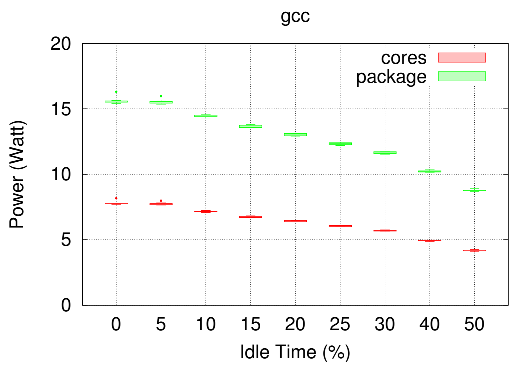
To have a clear over view of the three different power capping techniques, comparisons are made between them with respect to power, energy and performance.

3.2.1 Influence on power

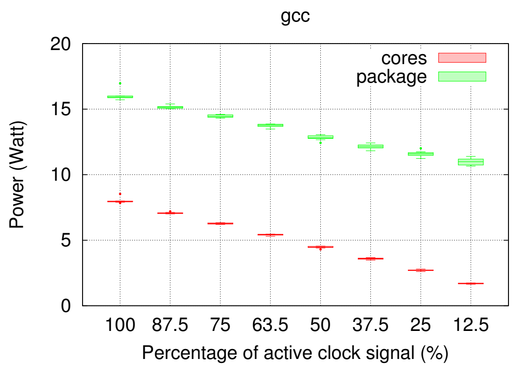
Figure 3.19 is the comparison between the three power capping techniques with respect to their influence on the power of CPU package and cores. From the figures, we can see that to have the same package power achieved by DVFS at



(a) DVFS effect on power



(b) ICI effect on power

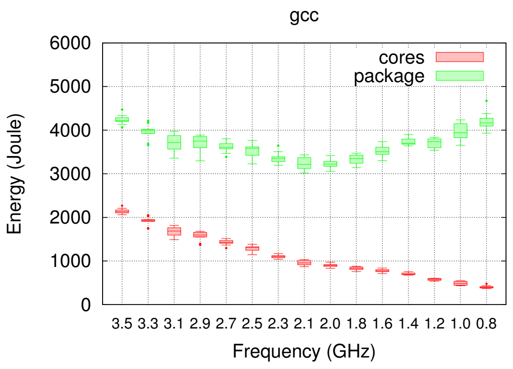


(c) CCM effect on power

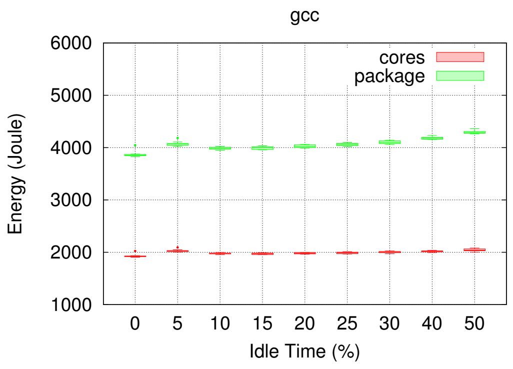
Figure 3.19: Comparison of effect on power

3.2.2 Influence on energy

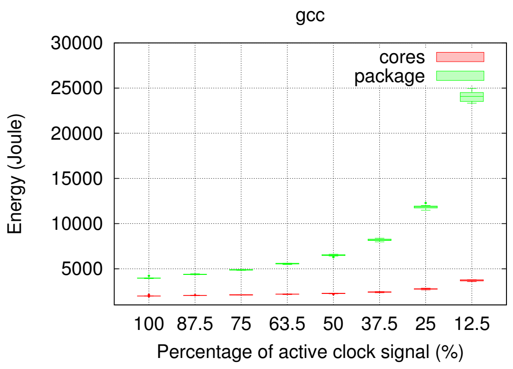
In figure 3.20 we can see clearly the different influence on energy consumption of the power capping techniques. DVFS is the only technique provides the possibility of energy saving. ICI doesn't influence the energy consumption strongly and applying CCM will consume more energy during the execution time.



(a) DVFS effect on energy



(b) ICI effect on energy



(c) CCM effect on energy

Figure 3.20: Comparison of effect on power

3.2.3 Influence on performance

The comparison of different techniques' effects on power consumption and performance is in figure 3.21, The X axis is the in Watt. Y axis is the performance, defined as minimum execution time/ current execution time. Each cluster of a same color is from the same configuraion. Red stands for cores of DVFS. Orange stands for package of DVFS. Green is core of ICI. Blue is package of ICI. Brown is cores of CCM and black is package of CCM.

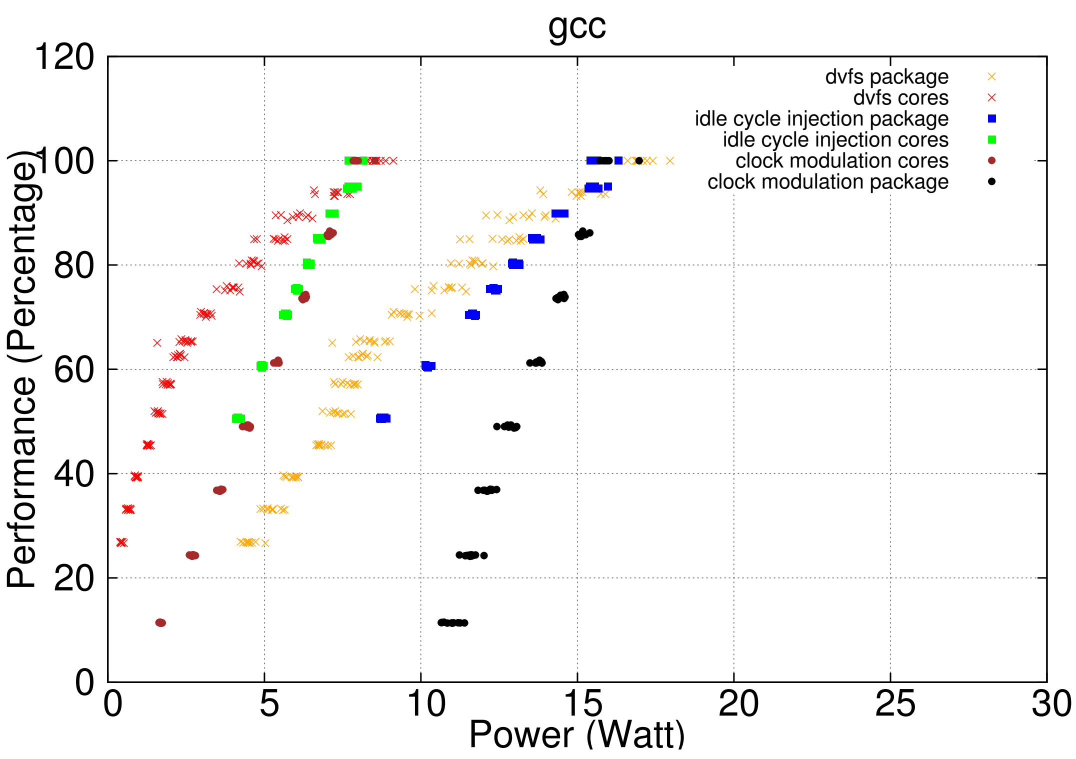


Figure 3.21: gcc performance to power comparison

From the graph, we can see for all the techniques, higher power leads to better performance. Using DVFS consumes the least power to achieve the same performance respect to others. For only the cores, ICI and CCM has similar behavior but ICI is slightly better. For the entire package, it is obvious that ICI overperformed CCM. From the graph we can understand using DVFS can achieve lower power with the least influence on performance. ICI is the following, then CCM.

Figure 3.22 is the comparison with respect to energy consumption and performance. X axis is energy in Joule, Y axis is the performance. Cluster with same color belong to the same configuration. Different colors stands for different techniques for cores or package. The details are the same as previous graph.

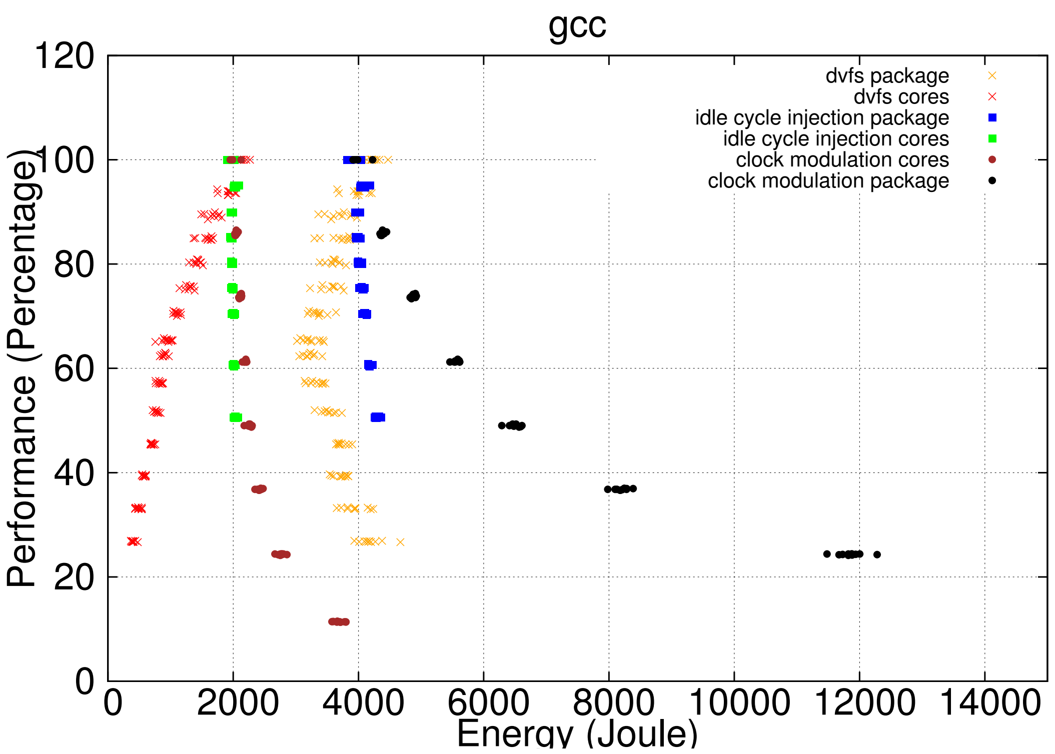


Figure 3.22: gcc performance to energy comparison

We can see that for DVFS, the energy required to achieve same performance as the other two techniques is much lower. For the cores, using DVFS to lower its performance can save energy. It is also true for the entire package until a turning point. For ICI, the energy keeps the same for all performance. For CCM, sacrificing performance increase the energy consumption to finish the workload. Therefore, we can conclude that for in general, DVFS has the best energy efficiency. It can finish the workload with the same performance but less energy. Then ICI is the second with stable energy consumption. CCM is not suitable for saving energy.

3.3 Further experiments

From the previous experiments, I concluded that DVFS is the only technique introduced here that can improve energy efficiency of the processor. ICI doesn't have great influence on CPU's energy efficiency and CCM has negative effect on energy efficiency.

To complement my experiment results, I decided to launch more tests with multiple threads and different operating frequencies.

3.3.1 DVFS

From the pervious experiments, I assumed that DVFS can improve the energy efficiency of the processor and there is a best configuration for energy saving. From previous test, it's always P7 or P8. Since we were always running one single thread benchmark, it was not very convincing. To get more proofs, I ran multiple instances of gcc on each cores.

Firstly, I run four instances on each core and we found that the most energy efficient point become P11 or P12. Then I ran eight instances on the CPU, with two threads for each core. I found the same result. When the CPU gets busier, using DVFS can significantly decrease the energy consumption. And when CPU is in less power consuming states, DVFS will be less effective in energy saving.

To further prove this assumption, I ran eight gcc instances, but with clock cycle modulation of 12.5% active clock signal. I found that the best energy saving point changed to P7. Therefore, I concluded that with lower average power consumption, DVFS become less effective in energy saving.

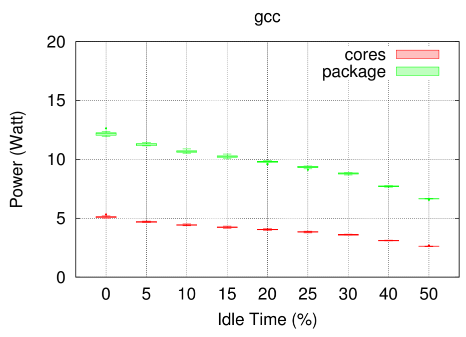
I did another experiment running 8 gcc threads, with 50% of ICI instead of CCM. The result is the same. The most energy efficient configuration change to P9 instead of P11 and P12.

3.3.2 Idle cycle injection

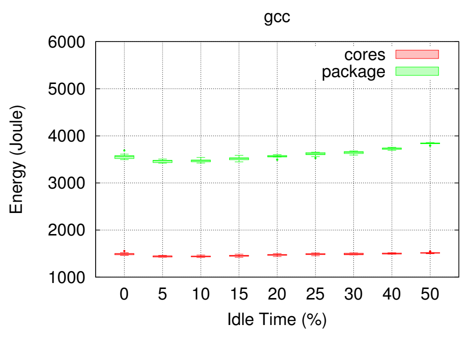
We did the assumption that, under a specific voltage and frequency, ICI will not influence significantly the total energy consumed by a workload. We tried to prove it with more tests.

As we previously introduced, P-states and C-states are independent, they can vary without influencing each other. To avoid repetition, we will only list the gcc case as example. Graphs for other benchmarks are also having very similar features. We omit them here.

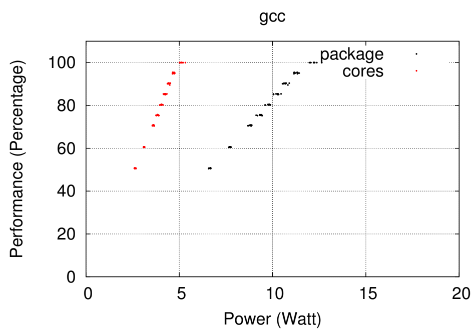
Figure 3.23 is the group of graph for gcc ICI in 2.9GHz, DVFS P3 state:



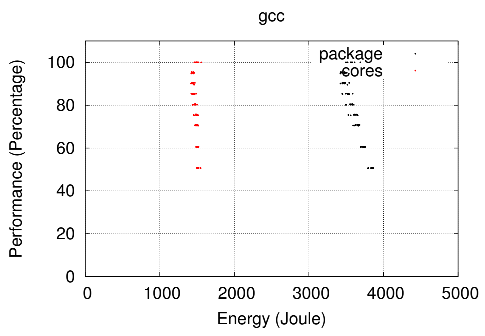
(a) Power usage



(b) Energy usage



(c) Performance to power

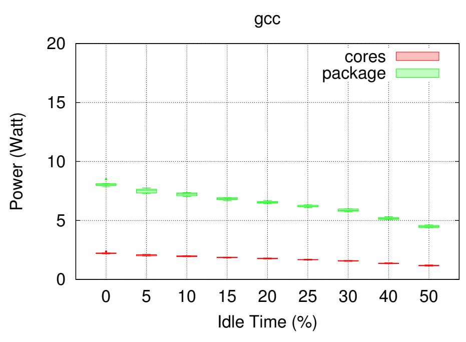


(d) Performance to energy

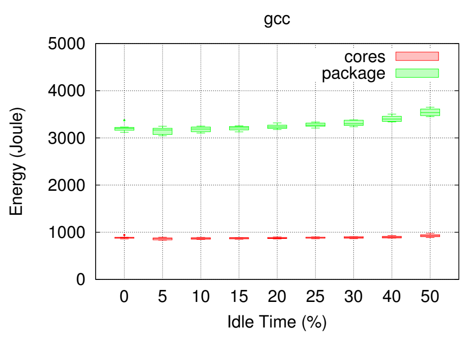
Figure 3.23: gcc ICI at 2.9GHz

We can see that the total energy consumed with respect to under 3.5GHz is decreased. For different ICI configuration, the total energy is not changing too much. Performance vs power still keep a linear growth approximately. The general feature is very similar with the result under 3.5GHz.

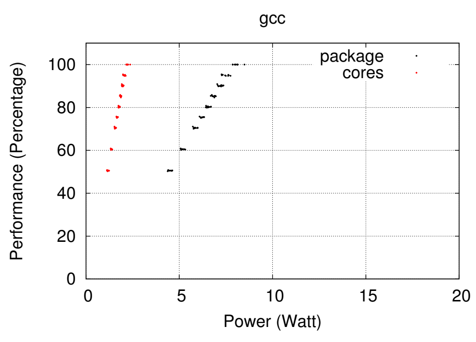
Figure 3.24 is gcc ICI in 2.0GHz. P8 state.



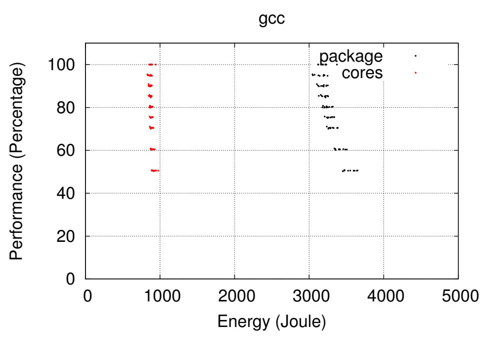
(a) Power usage



(b) Energy usage



(c) Performance to power



(d) Performance to energy

Figure 3.24: gcc ICI at 2.0GHz

Under different ICI percentage, the total energy keeps more or less stable with a slight increase. The general feature is consistent with the previous test cases.

We also launched the experiment for ICI with 8 instances of gcc running at the same time, the result is that with multiple thread, ICI still keeps a relatively stable energy consumption for the same workload.

3.3.3 Clock cycle modulation

From the data I got from CCM, I found that CCM was not suitable for energy saving since it increase the power consumption of uncore components and penalize the execution time too much. However, it might be possible that when the CPU cores are extremely busy that its power consumption is much more significant than uncore components, using CCM may save energy.

Therefore, I ran 8 instances of gcc on the CPU to verify our assumption. However, the result is that even in this case, using CCM still cost more energy than without.

3.4 Conclusion

From all the experiments, we can conclude the features of DVFS, ICI and CCM.

Using DVFS to lower the frequency and voltage can save energy when the CPU is in a very power consuming status. The most energy saving configuration change according to the average power consumption of CPU. DVFS can be used together with ICI and CCM. With more percentage of CPU idle time or less active clock signal, the most energy efficient configuration is tend to be with a higher frequency and voltage. To achieve the maximum energy efficiency, it is necessary to adjust the DVFS configuration according to runtime information about the workload and power consumption. For mobile devices, DVFS can be a very effective technique to extend battery life it can be used properly.

As power capping techniques, DVFS has a big advantage to lower the power with less influence in performance. It is also the most energy efficient compare with the others. However, it is less used for thermal control compare with ICI because in most of the real situation, it can be only used per-chip. Theoretically, DVFS has some latency when switching to another voltage and frequency. However in our experiment, from the power trace of DVFS, we can see the latency was almost not noticeable.

Using idle cycle injection to the entire CPU doesn't influence the amount of energy required to finish a task significantly. ICI can not be used as an energy saving tool for servers or mobile devices.

ICI can be used for thermal control without energy overhead. Since ICI can have fine grained control (per-thread), it is possible to do ICI only to the kind of work will produce significant amount of heat. Therefore, we can say it is a very ideal tool for thermal control.

Due to the fine grained feature, ICI can be very versatile. It can be implemented with different strategies and for different use. For example, it is possible to implement "smart" power capping using ICI without influencing execution time of the tasks with higher priority. Due to the same reason, the ICI strategies can be more different and complicated. In our work, we used indiscriminate ICI provided by PowerClamp. For ICI with a specific purpose, a different strategy is required.

Using CCM will increase the energy needed to finish a task. It also increase the power consumption of uncore components of the CPU.

CCM is used as a thermal control method in Intel CPU. However, using CCM is with a big energy overhead, and it is very inconvenient to have high level CCM with very little percentage of active clock signal. Like DVFS, CCM can be only applied per-chip for most of commercial CPUs. Therefore, even if CCM is available and can be applied easily to the processor by writing MSR, it is not recommended as the best power capping technique.

As a power capping technique on CPU, CCM might be not very efficient as the other ones in terms of energy saving and power capping, but CCM is with a big variety of use rather than power capping the CPU. It can be used to encode information for transmission[38], or control of the power supplied of electronic motors and LED displays[39].

Chapter 4

Related work of power capping techniques

DVFS, ICI, CCM, these three power capping techniques were frequently used and researched. They were used for thermal control, energy saving and power optimization.

In this chapter, I introduced some existing solutions using these power capping techniques. Three thermal control solutions using different power capping techniques will be introduced. I analyzed their energy efficiency with respect to our experiment results, discuss their advantages and disadvantages. From the experiment results, we understood DVFS is the only technique provide the possibility of energy saving. I also presented different energy saving solutions for mobile devices, HPC devices and data centers using DVFS. Regarding to power optimization and allocation of computing centers, most of existing solution used a central system or stratified system to manage the power plan, then power capping on every single server. Some examples is given of typical systems with this architecture. Since ICI is a relatively new technique, few existing solution used it. I explored the possibility to use ICI for power optimization in computing centers.

At the end of the chapter, I discussed about how to exploit the advantages and avoid the disadvantages of each power capping technique. For all the techniques, I discussed their potential improvement with respect to their shortcomings and look into the future for a better use of them.

4.1 Thermal control

During the active time, the CPU consumes electrical energy. A Part of energy loses in the form of heat due to the impedance of the electronic circuits. A higher power consumption of CPU tends to lead a bigger increase of CPU temperature. For the same processor, if the average CPU power draw is decreased, the amount of energy transformed into heat per time unit is decrease and its temperature is controlled. Therefore, all the power capping techniques mentioned in this work can be applied for thermal control of the processor.

4.1.1 DVFS thermal control

When using DVFS for thermal control, it is possible to decrease not only the processor's temperature, but also its energy consumption especially when the processor is in a very power consuming status.

However, the gross grain control of DVFS may unfairly penalize heterogeneous workload. To cope with this drawback, Globally Asynchronous Locally Synchronous (GALS) architecture[40] is proposed. In GALS architecture, the chip is split into several frequency islands. Moreover, each frequency island can be supplied with a different voltage becoming a Voltage and Frequency Island (VFI). GALS architectures are suitable for fine grain power management as the power consumption of the whole platform depends on the supply voltage and the clock frequency applied to each VFI.

With GALS architecture, nonlinear and asymmetric thermal control using DVFS [26] is proposed. The proposed strategy implements a chopped scheme on top of a robust DVFS approach in order to limit the temperature increase. Firstly, they analyzed the relation between VFI power consumption of and its thermal aspects. Power-thermal model of the processor is given. Then, depend upon an internal control law, the system may decide to cool down after analyzing current thermal situation with information of power consumption. As a result, the frequency of a VFI goes from one level to another with a given transition time and dynamics.

To further improve the precision of DVFS thermal control, workload prediction can be added. In the research: Consistent Runtime Thermal Prediction and Control Through Workload Phase Detection [41] described the mechanism in details. They devised an off-line analysis algorithm that learns a set of thermal models as a function of operating frequency and globally defined workload phases. During run time, by using the measurements of the processors performance counters, they designed a methodology capable of detecting workload phase transitions and associating them with changes in thermal behavior. Afterwards, the current workload phase is identified and the corresponding state-space model is pulled from the lookup-table. Finally, the system selects the highest operating frequency that does not lead to thermal violations.

With workload information, thread migration can be used for further improvement. In addition to aggregate heat production, there can be significant temperature variance across different regions of the die, and thus one also must worry about more localized hot spots at particular portions of the chip. To cope with this issue, a group of researchers from Princeton University proposed a thermal control system[42] with thread migration policies managed by the operating system. Knowing information of the workload, the operating system can migrate the threads to balance heat production of the processor.

However, even thought there are many researches about workload prediction, most of them are with off-line machine learning. All of them can't be always accurate especially for very specific workloads. Without workload prediction, thread migration requires knowledges about the workload before execution. In reality, it is not common to have information about the workload before execution. Most important of all, fine grain DVFS control of the processor exist only in research environment. Per cluster DVFS(GALS) and per core dvfs are not avaliable for commerical CPUs.

4.1.2 ICI thermal control

In the previous chapters, I mentioned several times that ICI is very suitable for thermal control. The first research using idle cycle injection for thermal control is Dimetrodon proposed by Harvard University and AMD Research. Their solution used flexible, per-thread ICI technique. With per-thread control, hot processes will run with idle cycle injection. At the same time, the cool processes can run uninterrupted while system-level temperatures are lowered. They compared their solution to the ones using hardware techniques in a commodity operating system on real hardware under throughput and latency-sensitive real-world workloads. The result was that Dimetrodon was able to reduce CPU temperature with the less influence on execution time than hardware thermal control solutions using DVFS or CCM. They stated that Dimetrodon is a complementary system that can be used in conjunction with DVFS thermal control system. This combination is useful especially when temperature predictions of machines are inaccurate, or when a data center experiences uniform temperature increases.

With the idea of using ICI for thermal control, ThermOS[34] was raised as a complete thermal management system with control strategies. Firstly, they proposed the thermal model of the processor:

$$T(k+1) = aT(k) + bI(k)$$

where $T(k)$ and $T(k+1)$ are the temperatures at the k-th and *k + 1* -th sample instants, respectively; $I(k)$ is the idle time between the k-th and *k + 1*-th sample instants; while a and b are parameters defining the temperature behavior. With this model, the system can estimate the current thermal status without only relying on sensors. Finally, with a formal feedback control system, ThermOS apply idle cycle injection to decrease thermal emergencies according to real time policies (thread priority etc.).

In ThermOS, thread migration is also mentioned. They proposed the idea of scheduling existing idle tasks as a substitution of ICI. They stated that the overhead(3 to 30μs) for thread migration is acceptable and does not compromise the efficiency of ThermOS. With this possibility, it is possible to implement ICI without actually injecting extra idle cycles in the processing time.

ICI is a very ideal tool for fine grained thermal control without introducing big energy overhead. It can be also used together with DVFS without influence each other. We may expect a wide use of ICI for thermal control in future.

4.1.3 CCM thermal control

CCM is used as thermal control method for Intel processors with IA-32 architecture[37]. With a simple MSR writing, the active clock signals can be reduced immediately and the processor temperature can be controlled. CCM is very easy to use. Therefore, even if CCM is the least energy efficient technique we introduced, it can be found in most of the commercial processors as a mechanism for thermal control. Few recent research mentioned CCM as a thermal control method.

4.2 Energy saving

Applying DVFS on the processors properly can be used to improve energy efficiency of computing devices and data centers. In this section, we will present different energy saving solutions for mobile devices, HPC devices and data centers.

4.2.1 Extend battery life for mobile devices

An Energy Conservation DVFS Algorithm for the Android Operating System[43] is a very typical research for the purpose of extending mobile battery life. In this research, they mentioned the fact that traditionally, users may choose to use the lowest CPU frequency for saving energy. However, reducing the CPU frequency may not always reduce the energy consumption. The lowest energy consumption usually appears at some operating frequency (optimum frequency) other than the lowest one supported by the processor. These statements are consistent with our experiment results. With experiments, they found that a relationship exists between the optimum frequency and the memory access behavior which is represented by an index called the memory access rate (MAR). MAR is defined as the ratio of cache misses number to the number of instructions executed. A program with a lower MAR (which tends to be CPU-bound) will have a higher optimum frequency, whereas a program with a higher MAR (which tends to be memory-bound) will have a lower optimum frequency. With the experiment data, they built a model for the relation between MAR and optimum frequency by curve fitting. During run time, MAR will be obtained and operating frequency can be adjusted according to their model.

An improvement of the solution mentioned above is Unifying DVFS and Offlining in Mobile Multicores[44]. They integrated core offlining with DVFS for multicore mobile phones. They implemented medusa, an offline-aware frequency scaling governor, in the Linux kernel running on two Galaxy S4 smartphones. From their experiments, they found that running more cores at low frequency is more energy efficient than activate only one core using high frequency. During idle state of mobile phone, more cores will be put off-line. During the execution time of tasks, the system will increase the frequency of all the on-line cores, once a specific high frequency (calculated from their power model) is reached, more cores will be activated and the frequency will be adjusted.

To efficiently apply energy saving algorithms to mobiles with less delay, workload prediction can be useful. For a mobile device, workload is more predictable because it heavily depends on the owner's habit. Therefore, by profiling the using behavior and the context, workload prediction can be done more precisely. Mpower[18] implemented this mechanism for mobile energy saving and battery life prediction. MPower relies on a system architecture based on two main components: an Android client logging application running on the mobile device, and a remote server for the data storage and the model computation. The application on the mobile device gathers data related to energy consumption, e.g., battery level, CPU frequency, screen brightness, and sends them to a remote cloud server. Then, energy saving advises is sent back to the device and the currently estimated battery lifetime is shown by the MPower client.

4.2.2 Improve energy efficiency of HPC devices and data centers

To improve the energy efficiency of a HPC device, a very classical and widely accepted method is combining DVFS with thread migration. The main idea is predicting fast and slow threads at runtime and applying DVFS to cores executing fast threads. Thread Shuffling[45] is one of the researches using this method for parallel applications. This type of solutions rely on GALS architecture with multiple voltage frequency islands in the processor. In their work, meeting point thread characterization is used to identify the critical thread dynamically during program execution by checking the workload balance at intermediate points of a parallel section. The thread with higher criticality usually requires longer execution time. The idea of thread shuffling is to map threads with similar criticality degrees into the same core through thread migration and then apply dynamic voltage and frequency scaling to cores containing non-critical threads.

When we consider energy saving for HPC devices, DRAM power should be take into account as well as CPU. Actually, DVFS does not have any direct control over DRAM power. However, memory traffic is related with processing speed, and processing speed is controlled by DVFS. Consequently, DVFS can control CPU power as well as DRAM power. Energy-centric DVFS Controlling Method for Multi-core Platforms[46] proposed this argument and their solution: eDVFS aiming at minimizing total energy consumption. They observed that the energy efficiency of the can be improved further by adjusting CPU frequency according to the degree of contention in shared resources. Memory traffic is generated by misses in the last-level cache. If memory traffic exceeds a given threshold, decrease CPU frequency. If memory traffic is under a given threshold, move towards the direction in which the ratio between current power and Instructions Per Second is reduced. eDVFS reads the number of instructions, memory traffic and consumed power from performance counters in CPU periodically and estimating Instructions Per Second. The memory traffic threshold is decided during runtime. If the amount of memory traffic exceeds the threshold, the main memory access latency will rise sharply. However, this solution requires also information of workload. Even if the information can be gathered during runtime, there will be a significant operating latency.

For computing cluster data centers, much research has proposed power management techniques at two levels: host level and cluster level. At host level, processor's power is managed using DVFS, which adjusts the dynamic power portion of the total power. However, this achieves a limited amount of energy savings because of the bounded portion of the dynamic power. To realize more energy savings, algorithms exploiting live migration and consolidation can be implemented at cluster level. This allows adjusting the amount of idle power by keeping few hosts with high utilization and turning off unused ones. Thus, there is a chance to achieve energy saving by virtual machines (VMs) consolidation using virtualization technologies, which enable live migration and dynamic configuration of VMs. Virtualization allows agile management and guarantees performance isolation where a VM is a container of a job. Therefore, consolidation of VMs into few numbers of physical hosts and turning off unused hosts can achieve energy savings. Currently, virtualized platforms are used not only for web or commercial transaction applications but also in grids and HPC clouds that host HPC applications. The research: Energy Efficient Scheduling of HPC-jobs on Virtualized Clusters Using Host and VM Dynamic Configuration[47] proposed a dynamic configuration of VM to minimize energy by exploring the number of core in the multicore processor. However, unused cores can suffer energy wastage unless completely powered down (repowering may incur new overhead). Moreover, the frequent migration of VMs degrades the system performance, so the decision of migration should be taken with awareness of its cost and the system dynamism.

4.3 Power optimization and allocation

In todays data centers, precisely controlling server power consumption is an essential way to avoid system failures caused by power capacity overload or overheating due to increasingly high server density. Power control needs to be enforced at three levels: rack enclosure, power distribution unit(PDU), and the entire data center, due to the physical and contractual power limits at each level. SHIP[48] is a power capping solution of this architecture. First, the rack-level power controller adaptively manages the power consumption of a rack by DVFS of the processors of each server in the rack. Second, the PDU-level power controller manages the total power consumption of a PDU by manipulating the power budget of each rack in the PDU. Similar to the PDU-level controller, the data center-level controller manages the total power consumption of the entire data center by manipulating the power budget of each PDU. The power budget of each rack depends on its priority and utilization. From data center to PDUs, then to the rackets and servers, the power budget will be divided and distributed into each single server.

After each single server receive its power budget, a common strategy is executing at the highest performance within the allowance of power budget. The research: Coordinated Power-Performance Optimization in Manycores[49] proposed C-3PO, which optimizes the performance of manycore processors under a power constraint by controlling two software knobs: thread packing, and dynamic voltage and DVFS. For each program, C-3PO checks whether the per-core utilization of the previous epoch is below a threshold or not. If it is below threshold, the number of cores allocated to the program is reduced by x. Then,C-3PO calculates the surplus power by subtracting the estimated power consumption from the power budget. Finally, if surplus power is positive, C-3PO distributes the power consumption to programs by allocating additional cores or increasing the operating frequency. otherwise the scheduler drops the frequency level or decreases the number of allocated cores so that the power consumption stays within the power cap. In other words, C-3PO estimates the power needed by programs at runtime. Then salvage the power budget that is not contributing to performance and redistribute to programs that would benefit from it.

4.4 Future work

In this section, how to use different power capping techniques properly according to the needs and situations is discussed. For each power capping technique introduced in this work, there are further improvements can be done.

From the experiment results, we can understand that proper use of DVFS will decrease energy consumption of the processor package. To exploit this advantage, the principles of using DVFS to lower CPU energy consumption is introduced with a state-of-art system.

4.4.1 Proper use of each power capping technique

For DVFS, to deal with its shortcoming of gross grained control, GALS architecture should be used. To have per-core DVFS with on chip regulators, researches[50] had been done. However, most of commercial processors don't actually provide it because it is relatively costly. Usually, the researches using DVFS for thermal control, power control or energy saving are with GALS architecture. Therefore, to put the research results in use, it is necessary to integrate per-core DVFS regulators on the processors with a lower cost.

With GALS architecture, using thread migration can further improve the performance of thermal control or energy saving solutions with DVFS. Thread migration strategies should be designed specifically for different purposes. To identify the characteristics of a thread, it is possible to analysis the instructions of it. However, to have a precise thread migration, the information is required before execution. Therefore, workload profiling and prediction can be used. Machine learning can be applied to gather workload information but workload prediction is very difficult. To cope with the uncertainties, DVFS systems can add feed back loops to adjust their behaviors at runtime.

Considering power control of data centers or single servers. To have an optimized division of power budget which maximize the system performance, power management system should be able to gather information of servers and allocate power according to the needs. For a single server, to make the full use of its power budget and achieve best performance, more precise power model should be proposed for estimating the optimum configuration. Feedback loops with little overhead can be added to adjust configurations at runtime.

ICI is a new power capping technique with a lot of potential for its fine grained feature and its compatibility with DVFS. Thread migration can be applied with ICI solutions by scheduling existing idle tasks as a substitution of injecting extra idle cycles in the processing time. Combining ICI and DVFS together, smarter solutions can be proposed. For example, a thermal control system can use DVFS to keep the average temperature of the chip within a limit then using ICI for further adjustments. A power capping system can do idle cycle injection to threads with lower priorities instead of using DVFS for the entire core to guarantee the performance of critical threads. However, to achieve more precise power capping with ICI, power model of ICI should be proposed, and feed back control loops can be added.

CCM is provided as a thermal control method for processors. It is the least energy efficient power capping method but it's very easy to use and with little latency. Because of these features, CCM can be useful to deal with thermal or power emergencies. Under specific conditions, systems can trigger CCM to protect its hardwares and maintain it's power consumption within power budget. However, it is necessary to prevent abusive use of CCM for computing devices or data centers.

4.4.2 A state-of-art energy saving solution

A system may have different strategies during the execution time of tasks. In the case of try to reduce production cost, energy saving is with the highest priority. Under some circumstances, a system may choose to maximize performance instead of saving energy. Another strategy can be choosing a trade off configuration that saves more energy than the maximum performance configuration, but has better performance than the most energy efficient configuration. Some systems have to follow constraints in power or performance. Therefore, a strategy and constraint controller is required to maintain the general rules of the system.

The problem of minimizing energy consumption of real-time tasks is very challenging task because the totally energy consumption of a task will be known only after its execution. Therefore, a prediction system to estimate the most energy efficient operating frequency is required. There is evidence[43] showing that the optimum operating frequency with best energy efficiency is related to the memory access rate which is the ratio of the total number of data and instruction cache misses to the number of instructions executed. According to the experiment result of this work, we may also conclude the the most energy efficient operating frequency is also related with the average power consumption of one specific CPU. With these information provided, a mathematical model should be proposed to predict the most energy efficient operating frequency. After optimum energy efficient frequency prediction, the system can calculate the target frequency according to the strategy and constraints of the system and use DVFS controller change the frequencies of cores.

For CPUs with multiple frequency island, thread migration can be used to achieve optimum solution for different types of workloads. For example threads with similar memory access rate can be assigned to the same core. For periodical tasks, it is possible to apply feedback loops for a more precise control.

Figure 4.1 is the graphical representation of the state-of-art system proposed.

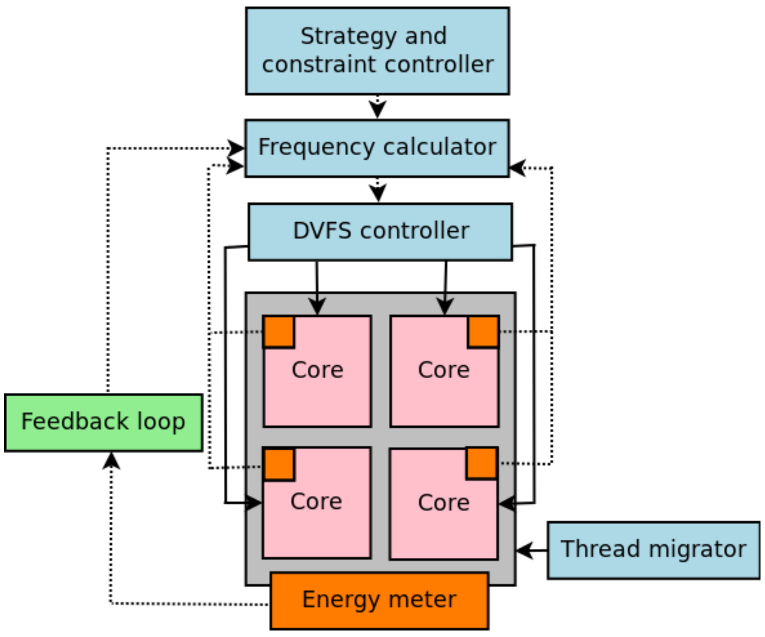


Figure 4.1: Energy saving system

Chapter 5

Summary

In this work, three very widely used power capping techniques were introduced: DVFS, ICI and CCM. With experiment data, I compared their general features, their energy efficiency, and their effectiveness of power capping with less performance degradation. I introduced the related works with different techniques for thermal control, energy saving, and power management. For each technique, I summarized its features and discussed the future works that could be done to use it more effectively.

The limitation of this work is that all the experiment data are from the same computer architecture. I launched multiple instances of single thread benchmarks instead of running a multi-thread program. However, this substitution is not strictly precise. To have a more general and precise conclusion, the experiments should be repeated with other computer architectures and with more variety of benchmarks.

This thesis can provide a clear overview power capping techniques and give advices of how to apply different power capping techniques properly according to different purposes. And by applying power capping techniques properly, IT industry will be greener in future.

In future, more experiment and analysis should be done with different computing architectures and applications to give a more systematic view of power capping techniques.

Bibliography

[1] ENERGY STAR Inc. Epa report to congress on server and data center energy efficiency. 2007.

[2] Univerisity of Michigan Center for Sustainable Systems. Patterns of use. *Green IT Factsheet*, 2013.

[3] C. Belady. In the data center, power and cooling costs more than the it equipment it supports. *Electronics Cooling*, 2007.

[4] Wan Yeon Lee. Energy-saving dvfs scheduling of multiple periodic real-time tasks on multi-core processors. In *Distributed Simulation and Real Time Applications, 2009. DS-RT '09. 13th IEEE/ACM International Symposium on*, pages 216–223, Oct 2009.

[5] Mehmet Basoglu, M. Orshansky, and M. Erez. Nbti-aware dvfs: A new approach to saving energy and increasing processor lifetime. In *Low-Power Electronics and Design (ISLPED), 2010 ACM/IEEE International Symposium on*, pages 253–258, Aug 2010.

[6] Etienne Le Sueur and Gernot Heiser. Slow down or sleep, that is the question. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference*, USENIXATC'11, pages 16–16, Berkeley, CA, USA, 2011. USENIX Association.

[7] Etienne Le Sueur and Gernot Heiser. Dynamic voltage and frequency scaling: The laws of diminishing returns. In *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*, HotPower'10, pages 1–8, Berkeley, CA, USA, 2010. USENIX Association.

[8] Antonios Antoniadis , Susanne Albers. Race to idle: new algorithms for speed scaling with a sleep state. In *Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms.*

[9] Luiz Andre Barroso. Warehouse-scale computing: Entering the teenage decade. *SIGARCH Comput. Archit. News*, 39(3), 2011.

[10] James Hamilton. Internet-scale service infrastructure efficiency. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 232–232, New York, NY, USA, 2009. ACM.

[11] Preston Gralla. Green it's new frontier: "power-capping" the data center. *GreenerComputing*, 2009.

[12] Matt Peckham. Riken plans exascale supercomputer 30 times faster than todays fastest in six years. *Time Tech*, 2013.

[13] Andrew Purcell. Making computing for big science more green. *International Science Grid This Week*, 2013.

[14] Yongpeng Liu and Hong Zhu. A survey of the research on power management techniques for high-performance systems. *Softw. Pract. Exper.*, 40(11):943–964, October 2010.

[15] O. Sarood, A. Langer, L. Kale, B. Rountree, and B. de Supinski. Optimizing power allocation to cpu and memory subsystems in overprovisioned hpc systems. In *Cluster Computing (CLUSTER), 2013 IEEE International Conference on*, pages 1–8, Sept 2013.

[16] Naoto Fukumoto Satoshi Imamura, Hiroshi Sasaki. Optimizing power-performance trade-off for parallel applications through dynamic core and frequency scaling. Technical report, Faculty of Information Science and Electrical Engineering, Kyushu University, 2012.

[17] Ilan Nass. Battery life: The smartphone achilles heel. *VIEWPOINTS*, 2013.

[18] A. Bonetto, M. Ferroni, D. Matteo, A.A. Nacci, M. Mazzucchelli, D. Sciuto, and M.D. Santambrogio. Mpower: Towards an adaptive power management system for mobile devices. In *Computational Science and Engineering (CSE), 2012 IEEE 15th International Conference on*, pages 318–325, Dec 2012.

[19] *Data Center Energy Efficiency with Intel Power Management Technologies.*

[20] *Power Capping on Intel Xeon Processors.*

[21] Charles Lefurgy, Xiaorui Wang, and Malcolm Ware. Power capping: A prelude to power shifting. *Cluster Computing*, 11(2):183–195, June 2008.

[22] *Power Management in the Cisco Unified Computing System: An Integrated Approach.*

[23] *HP Power Capping and HP Dynamic Power Capping for ProLiant servers.*

[24] Kevin Skadron Puyan Dadvar. Potential thermal security risks. Technical report, Department of Computer Science University of Virginia, 2005.

[25] Peter Bailis, Vijay Janapa Reddi, Sanjay Gandhi, David Brooks, and Margo Seltzer. Dimetrodon: Processor-level preventive thermal management via idle cycle injection. In *Proceedings of the 48th Design Automation Conference*, DAC '11, pages 89–94, New York, NY, USA, 2011. ACM.

[26] S. Durand and S. Lesecq. Nonlinear and asymmetric thermal-aware dvfs control. In *Control Conference (ECC), 2013 European*, pages 3240–3245, July 2013.

[27] David Meisner and Thomas F. Wenisch. Does low-power design imply energy efficiency for data centers? In *Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design*, ISLPED '11, pages 109–114, Piscataway, NJ, USA, 2011. IEEE Press.

[28] A. Castagnetti, C. Belleudy, S. Bilavarn, and M. Auguin. Power consumption modeling for dvfs exploitation. In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, pages 579–586, Sept 2010.

[29] T. Kolpe, A. Zhai, and S.S. Sapatnekar. Enabling improved power management in multicore processors through clustered dvfs. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, pages 1–6, March 2011.

[30] Mark Weiser, Brent Welch, Alan Demers, and Scott Shenker. Scheduling for reduced cpu energy. In *Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation*, OSDI '94, Berkeley, CA, USA, 1994. USENIX Association.

[31] Masaaki Kondo, Hiroshi Sasaki, and Hiroshi Nakamura. Improving fairness, throughput and energy-efficiency on a chip multiprocessor through dvfs. *SIGARCH Comput. Archit. News*, 35(1):31–38, March 2007.

[32] A.K. Mishra, S. Srikantaiah, M. Kandemir, and C.R. Das. Cpm in cmps: Coordinated power management in chip-multiprocessors. In *High Performance Computing, Networking, Storage and Analysis (SC), 2010 International Conference for*, pages 1–12, Nov 2010.

[33] Jonathan Corbet. Idle cycle injection. *LWN.net*, 2010.

[34] F. Sironi, M. Maggio, R. Cattaneo, G.F. Del Nero, D. Sciuto, and M.D. Santambrogio. Thermos: System support for dynamic thermal management of chip multi-processors. In *Parallel Architectures and Compilation Techniques (PACT), 2013 22nd International Conference on*, pages 41–50, Sept 2013.

[35] jlinkels@linuxquestions.org. Powersaving and clock modulation.

[36] Dmitri Besedin@linuxquestions.org. on demand modulation with celeron m.

[37] *Intel 64 and IA-32 Architectures Software Developers Manual.*

[38] Lie Xu and V.G. Agelidis. Vsc transmission system using flying capacitor multilevel converters and hybrid pwm control. *Power Delivery, IEEE Transactions on*, 22(1):693–702, Jan 2007.

[39] robotroom.com. Pulse width modulation for dc motor speed and led displays. In *Pulse-Width Modulation.*

[40] Umit Y. Ogras, Radu Marculescu, Puru Choudhary, and Diana Marculescu. Voltage-frequency island partitioning for gals-based networks-on-chip. In *Proceedings of the 44th Annual Design Automation Conference*, DAC '07, pages 110–115, New York, NY, USA, 2007. ACM.

[41] Ryan Cochran and Sherief Reda. Consistent runtime thermal prediction and control through workload phase detection. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 62–67, New York, NY, USA, 2010. ACM.

[42] James Donald and Margaret Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *Proceedings of the 33rd Annual International Symposium on Computer Architecture*, ISCA '06, pages 78–88, Washington, DC, USA, 2006. IEEE Computer Society.

[43] Che Wun Chiou Wen-Yew Liang, Po-Ting Lai. An energy conservation dvfs algorithm for the android operating system. *Journal of Convergence*, 2010.

[44] Aaron Carroll and Gernot Heiser. Unifying dvfs and offlining in mobile multicores. In *Real-Time and Embedded Technology and Applications Symposium (RTAS 2014)*, Berlin, Germany, April 2014.

[45] Qiong Cai, José González, Grigorios Magklis, Pedro Chaparro, and Antonio González. Thread shuffling: Combining dvfs and thread migration toreduce energy consumptions for multi-core systems. In *Proceedings of the 17th IEEE/ACM International Symposium on Low-power Electronics and Design*, ISLPED '11, pages 379–384, Piscataway, NJ, USA, 2011. IEEE Press.

[46] Shin-gyu Kim, Chanho Choi, Hyeonsang Eom, Heon Y. Yeom, and Huichung Byun. Energy-centric dvfs controling method for multi-core platforms. In *Proceedings of the 2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, SCC '12, pages 685–690, Washington, DC, USA, 2012. IEEE Computer Society.

[47] Ibrahim Takouna, Wesam Dawoud, and Christoph Meinel. Energy efficient scheduling of hpc-jobs on virtualize clusters using host and vm dynamic configuration. *SIGOPS Oper. Syst. Rev.*, 46(2):19–27, July 2012.

[48] Xiaorui Wang, Ming Chen, Charles Lefurgy, and Tom W. Keller. Ship: Scalable hierarchical power control for large-scale data centers. In *Proceedings of the 2009 18th International Conference on Parallel Architec-*

tures and Compilation Techniques, PACT '09, pages 91–100, Washington, DC, USA, 2009. IEEE Computer Society.

[49] Hiroshi Sasaki, Satoshi Imamura, and Koji Inoue. Coordinated power-performance optimization in manycores. In *Proceedings of the 22Nd International Conference on Parallel Architectures and Compilation Techniques*, PACT '13, pages 51–62, Piscataway, NJ, USA, 2013. IEEE Press.

[50] Wonyoung Kim, M.S. Gupta, Gu-Yeon Wei, and D. Brooks. System level analysis of fast, per-core dvfs using on-chip switching regulators. In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 123–134, Feb 2008.

List of Tables

1.1 C-states 10

2.1 Processor information 16
2.2 DVFS P-states 22
2.3 CCM configurations 24

List of Figures

1.1 Cisco Power Cap Group 5
1.2 Statistic of Server Power Use 6
1.3 DVFS Levels 9
1.4 Energy saving of CCM 11

3.1 gcc DVFS power usage 26
3.2 gcc DVFS energy usage 27
3.3 gcc DVFS performance to power 28
3.4 gcc DVFS performance to energy 29
3.5 gcc power trace at max frequency 30
3.6 gcc power trace DVFS 31
3.7 gcc ICI power usage 32
3.8 gcc ICI energy usage 33
3.9 gcc ICI performance to power 34
3.10 gcc ICI performance to power 35
3.11 gcc power trace at max frequency without ICI 36
3.12 gcc power trace ICI 36
3.13 gcc CCM power usage 37
3.14 gcc CCM energy usage 38
3.15 gcc CCM performance to power 39
3.16 gcc CCM performance to energy 40
3.17 gcc power trace at max frequency 41
3.18 gcc power trace CCM 41
3.19 Comparison of effect on power 43
3.20 Comparison of effect on power 44
3.21 gcc performance to power comparison 45
3.22 gcc performance to energy comparison 46
3.23 gcc ICI at 2.9GHz 48

3.24 gcc ICI at 2.0GHz 49

4.1 Energy saving system 63