Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Gonçalo Marques Pestana

Energy Efficiency in High Throughput Computing

Tools, techniques and experiments

Master's Thesis
Espoo, 1 December, 2014

DRAFT! — November 19, 2015 — DRAFT!

Supervisors:	Professor Jukka K. Nurminen
Advisor:	Zhonghong Ou (Post-Doc.)

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Gonçalo Marques Pestana		
Title: Energy Efficiency in High Throughput Computing Tools, techniques and experiments			
Date:	1 December, 2014	**Pages:**	49
Major:	Data Communication Software	**Code:**	T-110
Supervisors:	Professor Jukka K. Nurminen		
Advisor:	Zhonghong Ou (Post-Doc.)		
abstract			
Keywords:	energy efficiency, scientific computing, ARM, Intel, RAPL, tools, techiques		
Language:	English		

Acknowledgements

I wish to thank all students who use LaTeX for formatting their theses, because theses formatted with LaTeX are just so nice.

Thank you, and keep up the good work!

Espoo, 1 December, 2014

Gonçalo Marques Pestana

Abbreviations and Acronyms

2k/4k/8k mode	COFDM operation modes
3GPP	3rd Generation Partnership Project
ESP	Encapsulating Security Payload; An IPsec security protocol
FLUTE	The File Delivery over Unidirectional Transport protocol
e.g.	for example (do not list here this kind of common acronymbs or abbreviations, but only those that are essential for understanding the content of your thesis.
note	Note also, that this list is not compulsory, and should be omitted if you have only few abbreviations

Contents

Chapter 1

Tools and techniques for energy measurement

The recent scientific applications have to process and store considerable volumes of data. It is expected that the volume of data will increase considerably in the future, as technology improvements and requirements increase. This fact increases considerably the costs with energy in a HTC system. Thus, energy consumption has become a major concern amongst the scientific community.

It is critical to understand how energy is used by the HTC systems and its components, in order to develop solutions for improving energy efficiency in HTC. The learnings, techniques and tools can be further implemented in non-HTC systems whith the same results.

This chapter outlines and describes tools and techniques for measuring energy consumption in any machine. Our main goal is to use these tools and techniques to study energy efficiency of HTC systems, more specificcally systems running on top of x86 and ARM architectures. The tools and techniques described in this chapter were used widely during the study of energy efficiency comparison described in the following chapters.

The results of this section were publish in the conference proceedings of the 16th International workshop on Advanced Computing and Analysis Techniques in physics research (ACAT'2014). The article was called *Techniques and tools for measuring energy efficiency of scientific software applications* [13] and can be found on Appendix A, at the end of this document.

The big picture

During this study, we will consider two different granularities at which is possible to measure the energy consumption of a computing system. The two granularities are coarse and fine granularity and they differ on the type of system components that are taken into consideration when measuring energy consumption.

The coarser granularity takes into account the behavior of the whole node. This is usually investigated when engineering and optimizing computing centers. Alternatively, a more detailed approach is to look into the components which make up the active parts of a node, in particular the CPU and its memory subsystem since these are responsible for a sizeable fraction of the consumed power. They are also the place where the largest gains in terms of efficiency can be obtained through optimizations in the software [13].

Coarse grain or external measurements

As stated by [13], If one is simply interested in the coarse power consumption by node, external probing devices can be used: monitoring interfaces of the rack power distribution units, plugin meters and non-invasive clamp meters (allowing measurement of the current pulled by the system by induction without making physical contact with it). They differ mostly in terms of flexibility. Their accuracy is typically a few percent for power, whereas their time resolution is in the order of seconds. This features are enough to optimize electrical layout of the datacenters or to provide a baseline for more detailed studies.

Fine grain or internal measurements

A alternative approach takes into account the internal structure of a computing element of an HTC system, as shown in figure 1.1. Nowadays, almost all board manufacturer provides on-board chips which monitor energy consumption of different components of the system. These allow energy measurements of fine grained detail, as it is possible to individually monitor energy consumption of components such as the CPU, its memory subsystem, and others. An example of this chip monitors is the Texas Instruments TI INA231 [**?**] current-churn and power monitor which is found on the ARMv7 developer board which we used for our studies. It is quite common in the industry.e Compared to external methods, these on-board components provide high accuracy and reasonably high precision measurements (millisecond level).

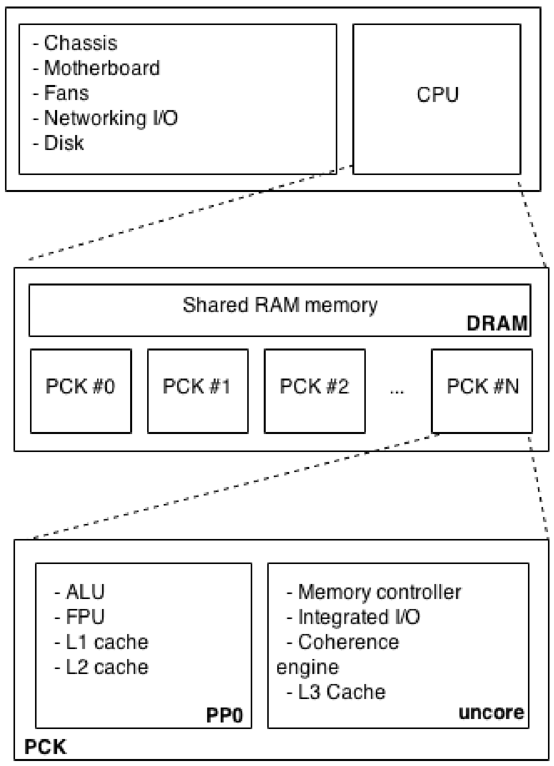


Figure 1.1: Components that contribute for power consumption in HPC. Taken from [13]

Tools for measuring energy consumption

1.0.1 Internal tools

Running Average Power Limit

Recent Intel boards provide a powerful way to measure fine grained energy consumption of their CPUs with the Running Average Power Limit (RAPL) technology. RAPL has been included on the SoC from the factory since the Sandy Bridge family of processors.

Contrary to other solutions (such as TI INA231, for example), which are implemented as discrete chips, RAPL is embedded as part of the CPU package itself and provides information on the CPUs own subsystems. In particular RAPL provides data for three different domains: **package** (pck), which measures energy consumed by the system's sockets, **power plane 0** (pp0), which measures energy consumed by the CPU core(s), and **dram**, which accounts for the sum of energy consumed by memory in a given socket, therefore excluding the on-core caches [**?**]. As for the discrete components case, the timing resolution of measurements is in the millisecond range [**?**]. This is fine enough to allow exploiting such data to build an energy consumption sampling profiler for applications, similar to how performance sampling profilers work (see section **??**). Finally, in addition to power monitoring of the sockets, RAPL can limit the power consumed by the different domains. This feature, usually referred as power capping, allows the user to define the average power consumption limit of a domain in a defined time window and allows more accurate independent measurements of the non limited components.

TI INA 231

There are chips that provide similar power monitoring capabilities than RAPL, but for any architecture and technology. An example is the Texas Instrument (TI) INA231 [**?**]. The TI INA231 is a power monitor that reports current, voltage and power consumption of CPU, DRAM and cores of the SoC which is attached. Some vendors include the TI INA231 in their boards from origin, relying on an external technology to provide the same capabilities as RAPL monitoring.

Similarly to RAPL, the sampling resolution is high (at the sampling rate of microseconds) and the error is relatively low. On the other hand, the TI INA231 does not have capping capanilities.

1.0.2 External tools

There are different techniques and tools to measure the power consumed by the whole system 1.1. Usually when the system is part of a server rack, the rack itself offers an API to sample the power consumed by each of the systems and overall rack power consumed. For simpler systems, it is recommended to use either power sockets with power meters or an external power meter.

The resolution and errors of external tools vary according to their nature. The resolution can range from microseconds - when the sampling is done digitally - to seconds - when the sampling is done by a person. It is recommended to use API and digital based external tools in order to achieve the best resolution and error possible.

Conclusion

There are many different methodologies and tools to measure energy consumption of computing systems. The differences between the existent options range from granularity, accessibility, error and resolution and also from hardware or software approaches. The decision for which tools and techniques should be relied upon for the measurements is important when conducting research on energy efficiency. The decision is directly dependent on the goals of the experiment and the availability of the tools according to the technology to be used.

Most of the techiques and tools outlined in this section were used in our experiments and are further discussed in the Experiments and Analysis chapters. In addition, the original paper which dwelves into the techniques and tools in more detail and was published at the ACAT'2014 conference proceding is attached as Appendix A.

Chapter 2

Experiments

We performed experiments with different hardware setups. The experiments consisted on running simulations of HPC workload while measuring the energy consumed by CPU and by the whole machine. The main goal is to compare the energy efficiency of ARM and Intel architectures. To attain that goal, we compared the results of the experiments to evaluate the potential of ARM architectures to perform HPC tasks, in comparison to the Intel architectures.

The software used to run the computing tasks widely used in production and research at the CMS experiment. In order the results to be as realistic as possible, we used the CMSSW framework [ref] and ParFullCMS [ref] simulations, which are widely used in production at CERN.

We organized the experiments in 2 sets. The conditions under which the experiments were conducted were similar. Due to hardware and software limitations and availability, it was not possible to completely reproduce the experiment conditions across all the sets. However, we believe that the differences will affect the final results only to a resonable degree, making it possible to scientifically compare the results. This and other considerations will be discussed further in the Analysis chapter.

The tools and techniques used to perform the energy consumption measurements were based on the study presented on the previous chapter. The setups of the experiments, methodology and tools used to perform the energy measurements during the experiments are explained and detailed in the following sections.

Thoughout this chapter, we will label *set of experiments* as experiments conducted with the same hardware and software configuration. The degrees of freedom of each experiment are the number of events and number of threads processing the workload.

For each setup, we outline the hardware, software setups and the used energy measurement tools. During this chapter and throughout the rest of the thesis, we will describe each batch of experiments as first set of experiments (1SE) and second set of experiments (2SE).

The remainder of this chapter divided in two section. Firstly, we will outline the most relevant characteristics of the architectures used during the experiments. Secondly, we describe the setup of the experiments and methodology used to perform the experiments.

2.1 Hardware

The focus of this work is to compare energy efficiency of ARM architectures and Intel based processors under similar workload. Our hardware choice was conditioned to the machine availability when the study was conducted. In addition, we also aimed at comparing similar conditions and workloads across all the set of experiments.

The ARM machines used were a single-board ARM processor developed by Odroid [11] and a server class ARM processor by Boston Viridis [12]. The Intel machines used were part of the microarchitectures family Sandy Bridges and Intel Bonnell. In the following sections, we will describe the hardware architecture, features of the hardware used to run the experiments and where the hardware is commonly used outside the scope of this study.

2.1.1 ARM architecture

2.1.1.1 Boston Viridis server

The Boston Viridis server is one of the first ARM architecture based servers where the processors, IO and networking are fully integrated in one single chip. According to the vendor, the server is intended to perform in a web server, cloud and data analytics environment with outstanding power performance [12].

The Boston Viridis server used in this study (which we will label as ARM_viridis throughout the rest of the document) consists of a chassis with twelve racks, each with an energy card. Each energy card contains four nodes 2.1. A node is an ARM based CPU fabricated by Calxeda. The block diagram of a ARM_viridis node is represented on 2.2shows the architectures of the EnergyCore used. We can notice that the SoC has an energy management engine that will further on allow us to sample the energy consumed by the node.

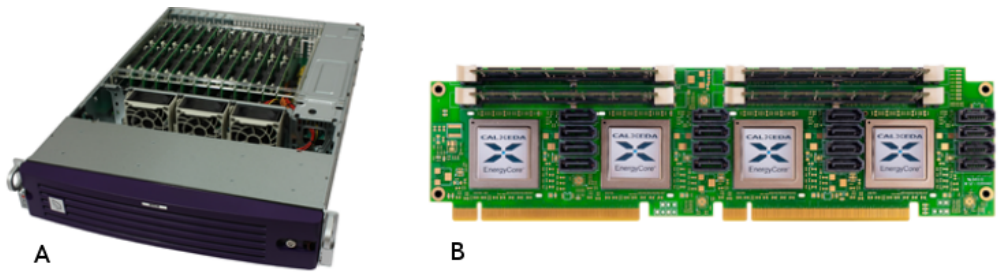


Figure 2.1: A. Viridis Server chassis with 12 energy card in it. B. Energy card with 4 nodes. Taken from [12]

Each ARM_viridis node contains four ARM A9 Cortex core with a clock speed up to 1.4MHz. A memory controller and L2 cache is includede on the chip. In addition, a couple of energy management blocks and IO controllers complete the Calxeda EnergyCore processor 2.2. These energy measurement blocks were used to perform part of the energy measurements with this setup.

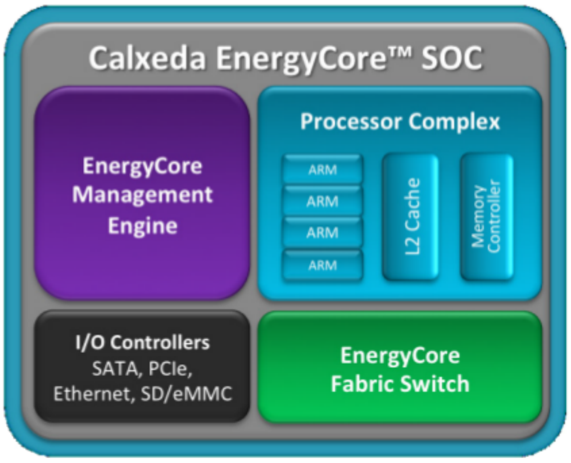


Figure 2.2: Block diagram of Calxeda EnergyCore. Taken from [12]

According to [2], the ARM A9 Cortex is a popular and mature general purpose core for low-power devices. It was introduced in 2008 and it remains a popular choice in smartphones and applications enabling the Internet of Things (IoT) [2]. The ARM A9 Cortex supports the ARMv7A instruction set architecture. A detailed study of the ARMv7A internals is out of scope of this work. More detailed specifications about the internals of the ARMv7A instructions set can be found in [2].

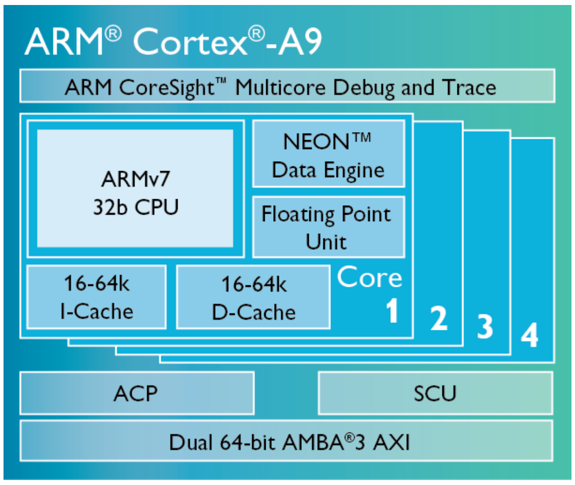


Figure 2.3: Block diagram of Cortex A9. Taken from [2]

2.1.1.2 ODROID-XU3 development board

The ODROID-XU3 [11] is an open-source development board produced by Hardkernel. They claim that the ODROID-XU3 is a "new generation of computing device with more powerful, more energy efficient hardware and smaller form factor" [11]. At the time of these experiments, the ODROID-XU3 was mostly used for testing and platform development and it was not intended to run in production scenarios. Throughout this document, the ODROID-XU3 described in this section will be called ARM_odroid.

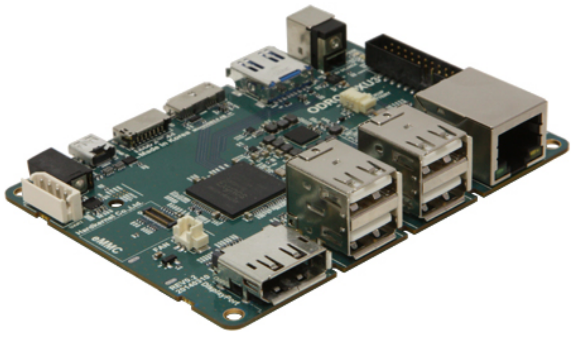


Figure 2.4: ODROID-XU3 development board. Taken from [11]

The ODROID-XU3 processor has four Samsung Exynos-5422 Cortex A15 and four Cortex A7 cores, with 2GB of LPDDR2 RAM. Only four cores are working at the same time and they are scheduled based on the big.LITTLE technology. The big.LITTLE technology [1] automatically schedules workloads across cores based on performance and energy needs. The vendor claims that the big.LITTLE technology can achieve energy savings from 40% to 75%, depending on the performance scenario [1]. It is important to note that, even though the CPU contains eight cores, only four of then are working at a given moment. The block diagram of the ODROID-XU3 can be seen in 2.5.

The ODROID-XU3 has a Texas Instrument power monitor chip (TI INA231) embedded from origin. The TI INA231 provides an API to read the energy consumed by the cores and DRAM at a sampling rate of microseconds. These readings can be easily triggered and read through software and consist of an accurate way to make fine-grained energy consumption measurements. We assumed that the measurements made by the TI INA231 can be compared to the RAPL technology by Intel.

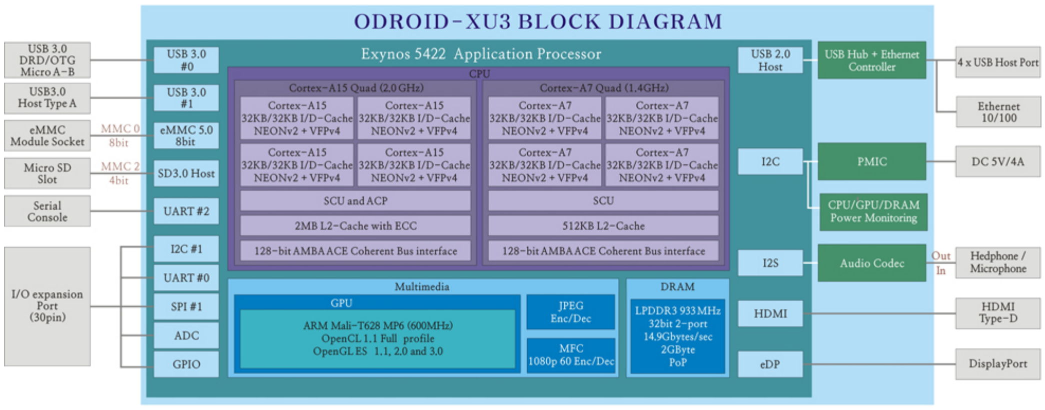


Figure 2.5: ODROID-XU3 block diagram. Taken from [11]

2.1.2 Intel x86 architecture

Across the different experiments, we have used three different machines running on top of x86 Intel instruction sets to compare with the ARM based machines. The Intel x86 machines are the most widely used solutions for server and workstation applications.

The Intel Xeon that we used had RAPL enable (refer to Chapter X), which allowed us to measure energy consumption accurately at a fine-grained level. Since the ATOM and QUAD machines did not have RAPL technology enabled, we used a clamp power meter to measure the energy consumed by the CPU at a given time.

The different types of measurements within the same architecture and its possible affect on he final result are discussed in the Analysis section.

describe more about Intel, its features (hyper threading, wich affects the results for example), its microarch families and where/how

have them been used in production
ex: "The Intel Atom is a brand name for a line of ultra-low-voltage CPUs by Intel. On the other hand, the x86 Intem Quad is brand name for a high performance family of Intel CPUs"

2.2 Experiments setup

2.2.1 First set of experiments

Hardware specifications

For the first set of experiments, we used three machines with different hardware setups. The three machines differ in architecture and general purpose. The ARM_virdis is a server rack with CPU consisting of ARMv7 processors produced by Boston Labs [ref]. We ran the same workloads in a x86 Intel Atom (Intel_atom) and Intel Quad (Intel_quad) for comparison. The Intel Atom is a brand name for a line of ultra-low-voltage CPUs by Intel. On the other hand, the x86 Intem Quad is brand name for a high performance family of Intel CPUs.

Below, we outline the most important specifications of the hardware setups we used for the experiments.

Intel_ATOM

kernel & sys: Linux cernvm 2.6.32431.5.1.el6.x86_64

OS: Scientific Linux release 6.5 (Carbon)

CPU: 4x Intel^{TM} Atom^{TM} CPU D525 1.8GHz

Memory (MemTotal): 3925084 kB (4GB)

For more detailed specs refer to [6]

Intel_QUAD

kernel & sys: Linux cern-vm 2.6.32-431.5.1.el6.x86_64

OS: Scientific Linux release 6.5 (Carbon)

CPU: 4x Intel^{TM} Core^{TM}2 Quad CPU Q9400 2.66GHz

Memory (MemTotal): 7928892 kB (8GB)

For more detailed specs refer to [7]

ARM_Viridis

kernel & sys: Linux 3.6.10-8.fc18.armv7hl.highbank

OS: Fedora release 18 (Spherical Cow)

CPU: 4x Quad-Core ARM^{TM} CortexA9^{TM} processor 1.4GHz

Memory (MemTotal): 4137780 kB (4GB)

For more detailed specs refer to [3]

Software and workload

We used the CMSSW framework in the generation-simulation mode (GEN-SIM). The workflow performs a Monte Carlo simulation of 8 TeV LHC Minimum bias event using Pynthia8 (generation step), followed by Simulation with Geant4 (simulation step). For more information about the CMSSW framework and its limitation on ARM, refer to Chapter X. At the time of the experiments, the CMSSW port for ARM had limitations on the multithreading support. We wanted to study the energy consumption of each hardware setup given different core load. Thus, we spinned up different processes instead of threads. The core-load levels used were 1/4, 1/2, 1 and 2 processes per number of physical cores.

Metrics

The energy efficiency metric used in this study is the ratio of performance per power consumed (in Watts). Performance consist on the average of events computed per second. Considering this metrics for comparing energy consumption, we consider a system to be as energy efficient as higher the ratio $nr\_of\_events/s/W$ is.

Given the hardware disparities of the setups we had in place to run our experiments, we used the performance (average fd events computed per second) as a way to uniform the results.

Tools for measuring energy consumption

For this set of experiments, we performed physical measurements using an external clamp meter. The clamp was a Mini AC/DC Clamp meter Mastech MS2102 AC/DC (see Figure 2.6). The clamp meter supports a maximum of 200A current, which was enough for our experiments. In addition, it presents an accuracy of +-2.5%. For more specifications about the clamp used, refer to [REF].

Figure 2.6: Mastech MS2102 clamp meter used to measure energy consumption. Taken from [] - cite Mastech website

Machine codename	**Architecture**	**CPU**	**N° active cores**	**RAM**	**Notes**
ARM_viridis	Quad-Core ARM^{TM} CortexA9^{TM}	ARMv7 32b (A7)	4	2 GB	Server class ARM processor with ipmitools
Intel_ATOM	Intel Bonnell^{TM}	Atom D525	4	4GB	No internal measurement tool
Intel_QUAD	Intel Sandy Bridge^{TM}	Quad CPU Q9400	4	8GB	No internal measurement tool

Table 2.1: Summary of the 1SE specifications

2.2.2 Second set of experiments

Hardware specifications

For the second set of experiements, we again used three machines with different hardware setups. As in the 1SE, the three machines differ in architecture and general purpose. The ARM_virdis, which was used in the 1SE, was also used during the second set of experiments. In addition to ARM_viridis, we also resort to another machine powered by an ARM CPU. The ARM_odroid is a development board manufactured by HardKernel [ref] and it is intended to provide a cheap and easy way to develop hardware and software in a ARM architecture. To represent the Intel architecture we used Intel_xeon, a machine from the Intel Sandy Bridge family and powered by an Intel R5-2650 CPU. This machine was part of a server rack and it was intended for high performace scientific computation in a production scenario.

Below, we outline the most important aspects of the hardware setups we used for the experiments.

ARM_Viridis

kernel & sys: Linux 3.6.10-8.fc18.armv7hl.highbank

OS: Fedora release 18 (Spherical Cow)

CPU: 4x Quad-Core ARM^{TM} CortexA9^{TM} processor 1.4GHz

Memory (MemTotal): 4137780 kB (4GB)

For more detailed specs refer to [3]

ARM_odroid

kernel & sys: Linux 3.10.24 LTS

OS: Ubuntu 14.04.3 LTS (Trusty Tahr)

CPU: 2x A15 and/or A7 cores(big.LITTLE technology) - A7 at 1.4GHz and A15 at 2GHz

Memory: 2GB

For more detailed specs refer to [10]

Intel_xeon

kernel & sys: Linux cern-vm 2.6.32-431.5.1.el6.x86_64

OS: Scientific Linux release 6.5 (Carbon)

CPU: 4x Intel^{TM} CPU E5-2650 2GHz

Memory: 252GB

For more detailed specs refer to [8]

Software and workload

We used the CMSSW's mode ParCullCMS for generating the workload. The ParFullCMS mode is a multi-threaded Geant4 [14] benchmark. It uses a complex CMS geometry for the event simulation and has the advantage of being multithreaded in both Intel and ARM architectures. As in the first set of experiements, we measured the energy consumed by the machine under different physical core loads. The core-load levels used were 1/4, 1/2, 1 and 2 threads per number of physical cores.

Metrics

As in the 1SE, the energy efficiency metric used in this study is the ratio of performance per power consumed (Watts). The hardware setups used in the 2SE differ in specs and features. Therefore, we used the this metric as a way to uniform the results.

Tools for measuring energy consumption

For the 2SE, we performed both internal and external measurements in the Intel_xeon and ARM_odroid. On the ARM_viridis, we performed only internal measurements given the lack of a tool that would performe with the same degree of accuracy than the tools used for Intel_xeon and ARM_odroid. All the tools used to measure energy consumption were embeeded in the hardware setup of the machines.

For the ARM_odroid, we used a Texas Instrument power monitor chip (TI INA231) for internal measurements. The TI INA231 allowed us to sample the energy consumed by the cores and DRAM at a frequency rate of

Machine codename	**Architecture**	**CPU**	**N° active cores**	**RAM**	**Notes**
ARM_odroid	Quad-Core ARMv7^{TM}	A15 and or A7 cores(big.LITTLE technology)	4	2 GB	Development board with TI INA231 chip
ARM_viridis	Quad-Core ARM^{TM} CortexA9^{TM}	ARMv7 32b (A7)	4	2 GB	Server class ARM processor with ipmitools
Intel_xeon	Intel Sandy Bridge^{TM}	CPU E5-2650	32	252 GB	System on a rack with RAPL

Table 2.2: Summary of the 2-SE specifications

microseconds. For the extrenal measurements on the ARM_odroid, we used an external plug-in power monitor with a computer interface for sampling and storing the results.

For the Intel_xeon machine, we used the Running Average Power Unit (RAPL) technolgy to perform internal measurements. The RAPL allowed us to sample the energy consumed by the CPU's package, DRAM and cores. For the external measurements, we used an API provided by the server rack's PDU. This API provides a measure sampling rate of around 1 second.

For the ARM_viridis, we used the capabilities of the Intellegent Platform Management Interface (IPMI) [9] included in the server from origin. The IPMI is a chip that runs as a separate subsystem and is attached to the motherboard. The ARM_viridis implementation of IPMI provide several capabilities, namely interlal hardware energy monitoring. We leveraged the IPMI tools to perform internal energy consumption of the ARM cores during the experiments

2.3 Summary

We have performed several experiments under different hardware setups. Our main goal was to understand how the ARM and Intel architectures perform under similar workloads from an energy consumption standpoint.

In this chapter we outlined the setup of the machines used during the experiments.

The hardware setups were chosen given their similarity and possibility of a reliable comparison and hardware availability. It is important to note that both ARM_viridis and ARM_odroid machines are much more recent than the compared Intel hardware. All the ARM machines used were still a technology that was yet to find production stability at the moment of the experiments. On the other hand, the Intel architecture used in this study was widely used in real HPC applications at the time of this study.

For this study, we assume that the RAPL, the internal TI INA231 chip and the IPMI tools for internal energy consumption measurement are similarly accurate and would produce the same results if interchanged.

Chapter 3

Results

This chapter presents the results of the experiment descibed in the later chapter. The plots and figures in this chapter can be divided into 3 sections. The first section outlines the results of the first set of experiments. The second section outlines the results of the second set of experiments. The last section outlines the results that aim at studying the CMSSW framework is a Non-Uniform Memory Access (NUMA) environment.

In the next chapter, we analyse the results based on the content of this chapter.

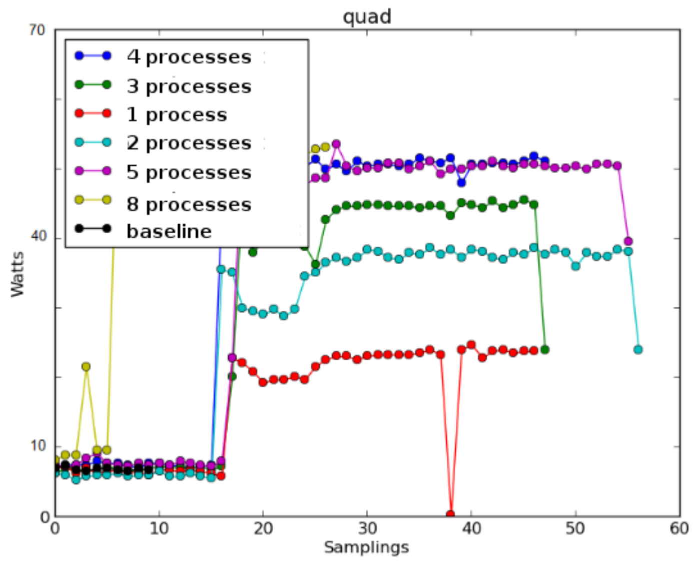


Figure 3.1: All stages of the CMSSW experiments on Intel_quad

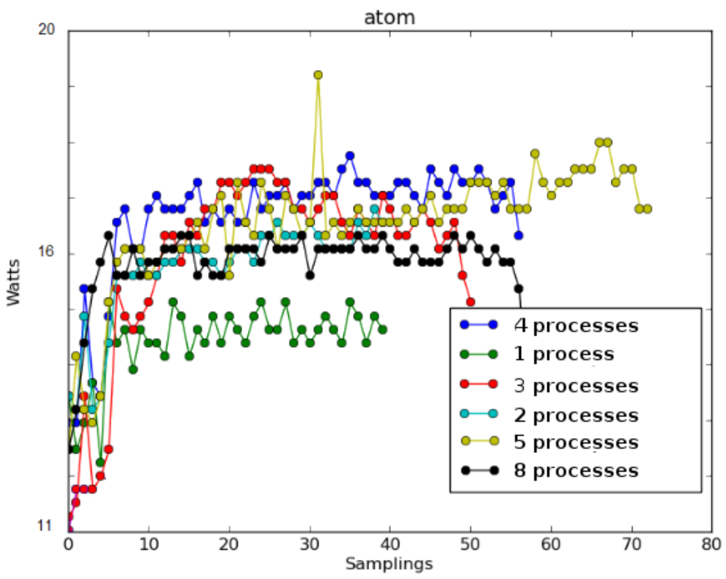


Figure 3.2: All stages of the CMSSW experiments on Intel_atom

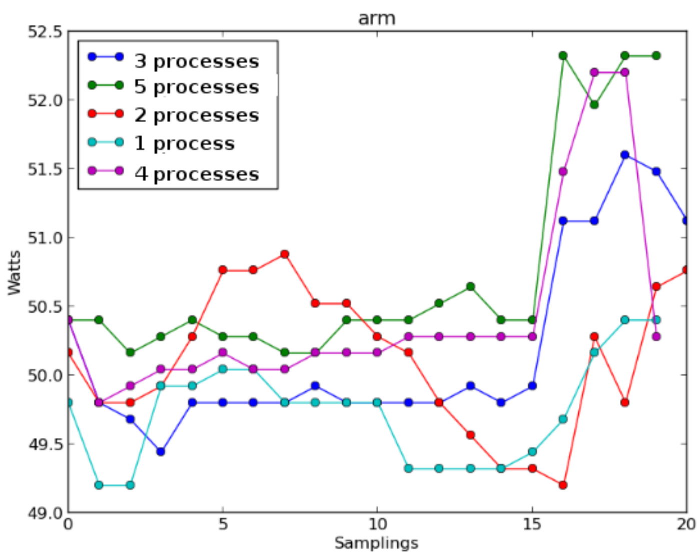


Figure 3.3: All stages of the CMSSW experiments on ARM_viridis

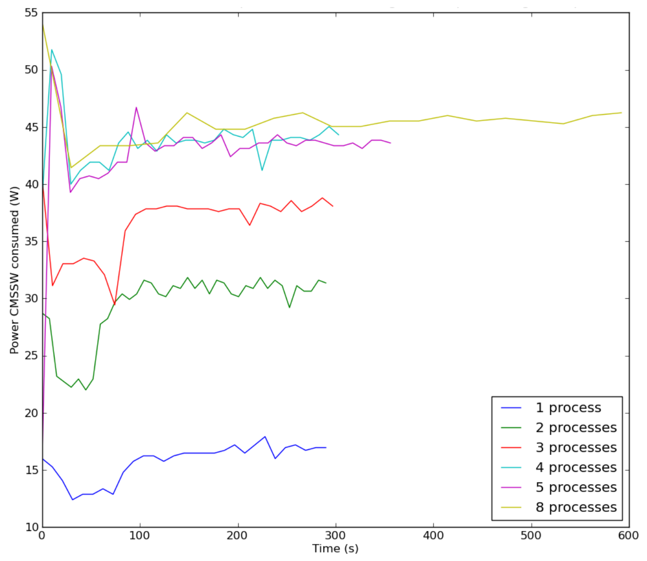


Figure 3.4: CMSSW experiments on Intel_quad - event processing stage

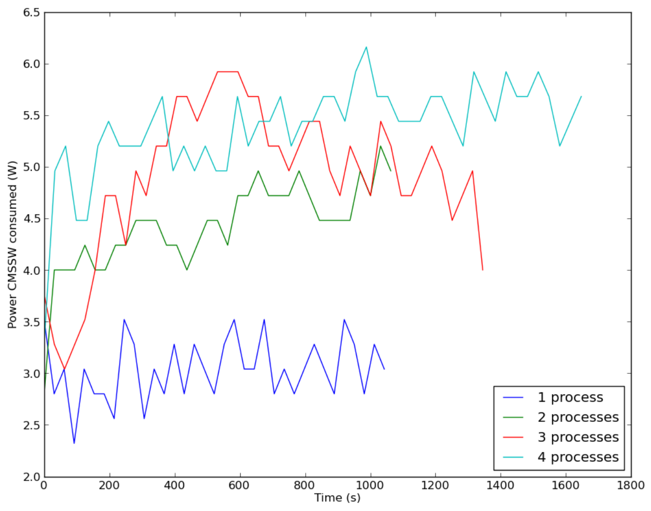


Figure 3.5: CMSSW experiments on Intel_atom - event processing stage

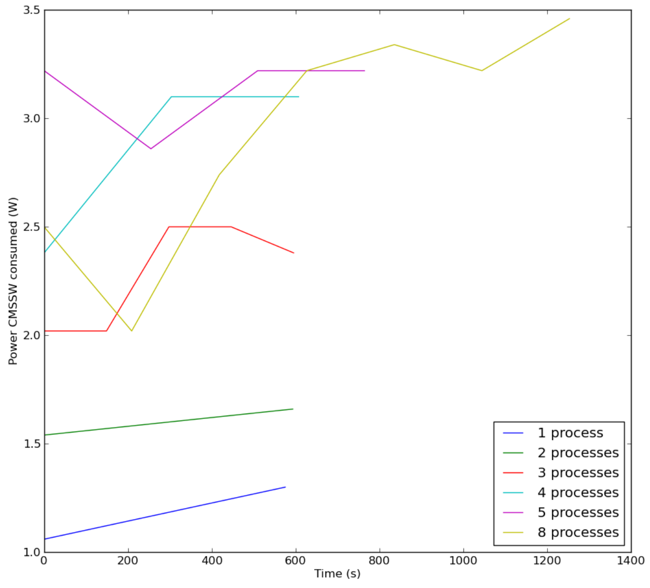


Figure 3.6: CMSSW experiments on ARM_viridis - event processing stage

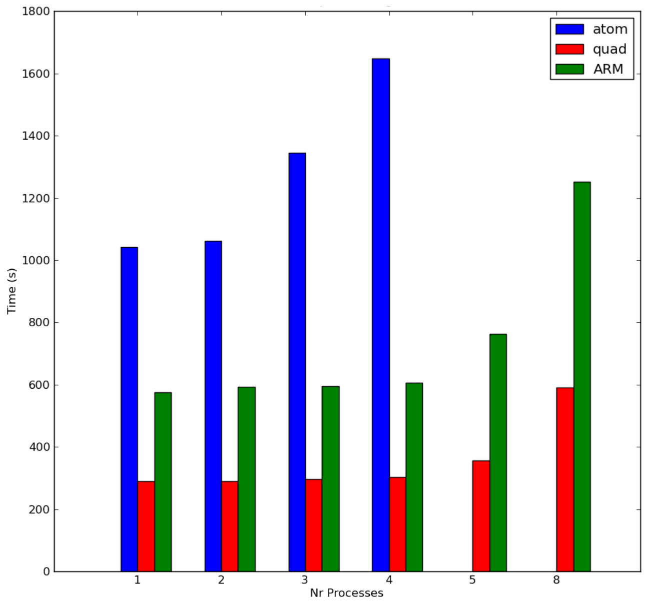


Figure 3.7: Processing time comparison

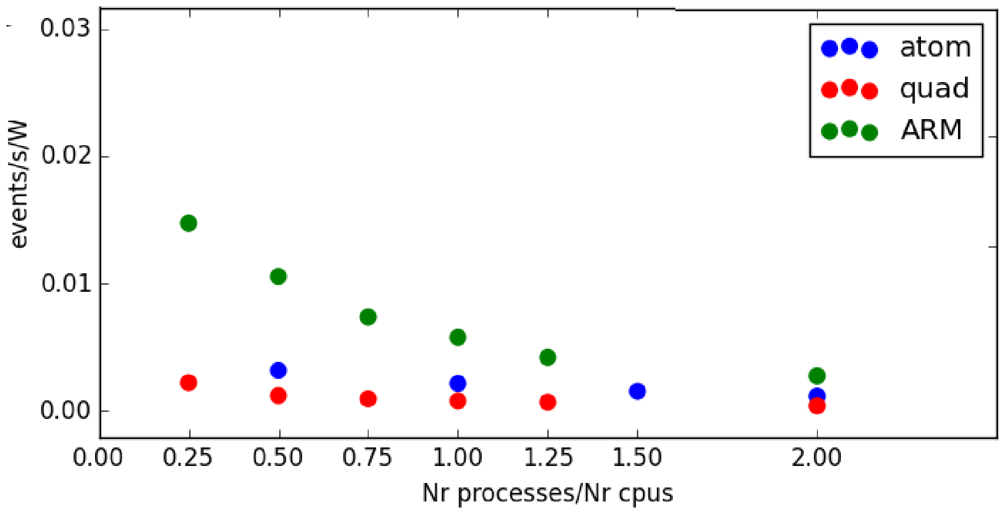


Figure 3.8: Energy efficiency comparison for the first set of experiments - External measurements

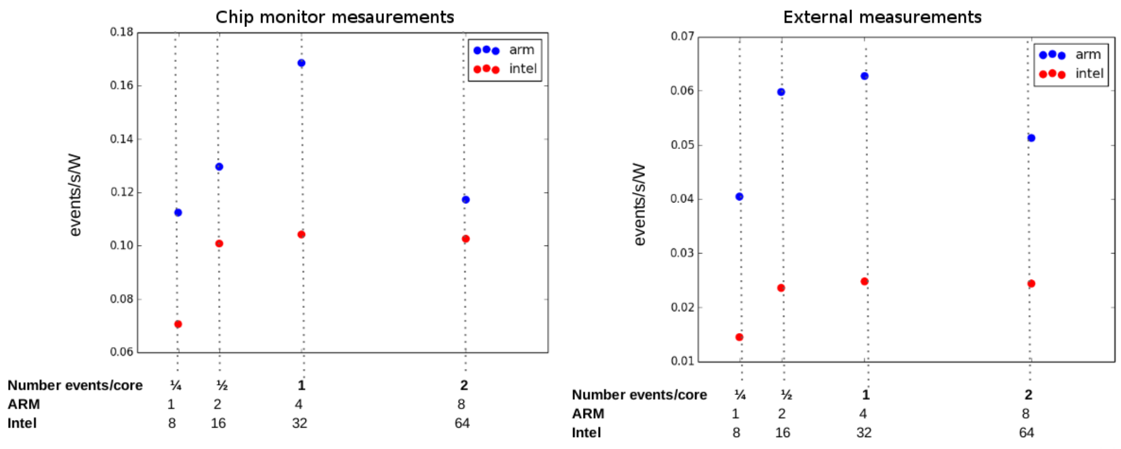


Figure 3.9: Multithreaded ParFullCMS comparison between Intel_xeon and ARM_odroid

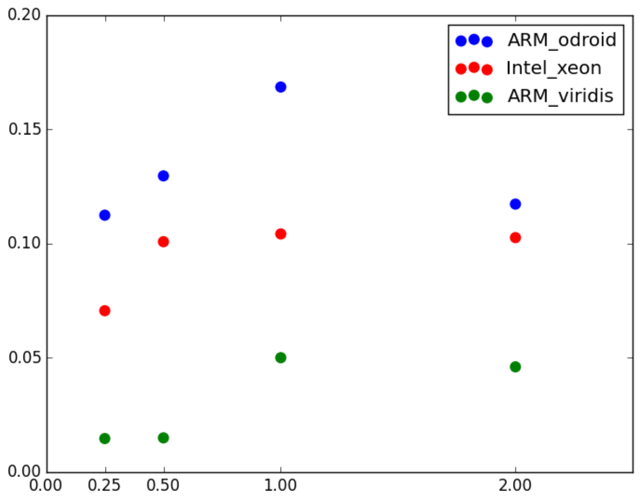


Figure 3.10: Multithreaded ParFullCMS comparison between Intel_xeon, ARM_viridis and ARM_odroid

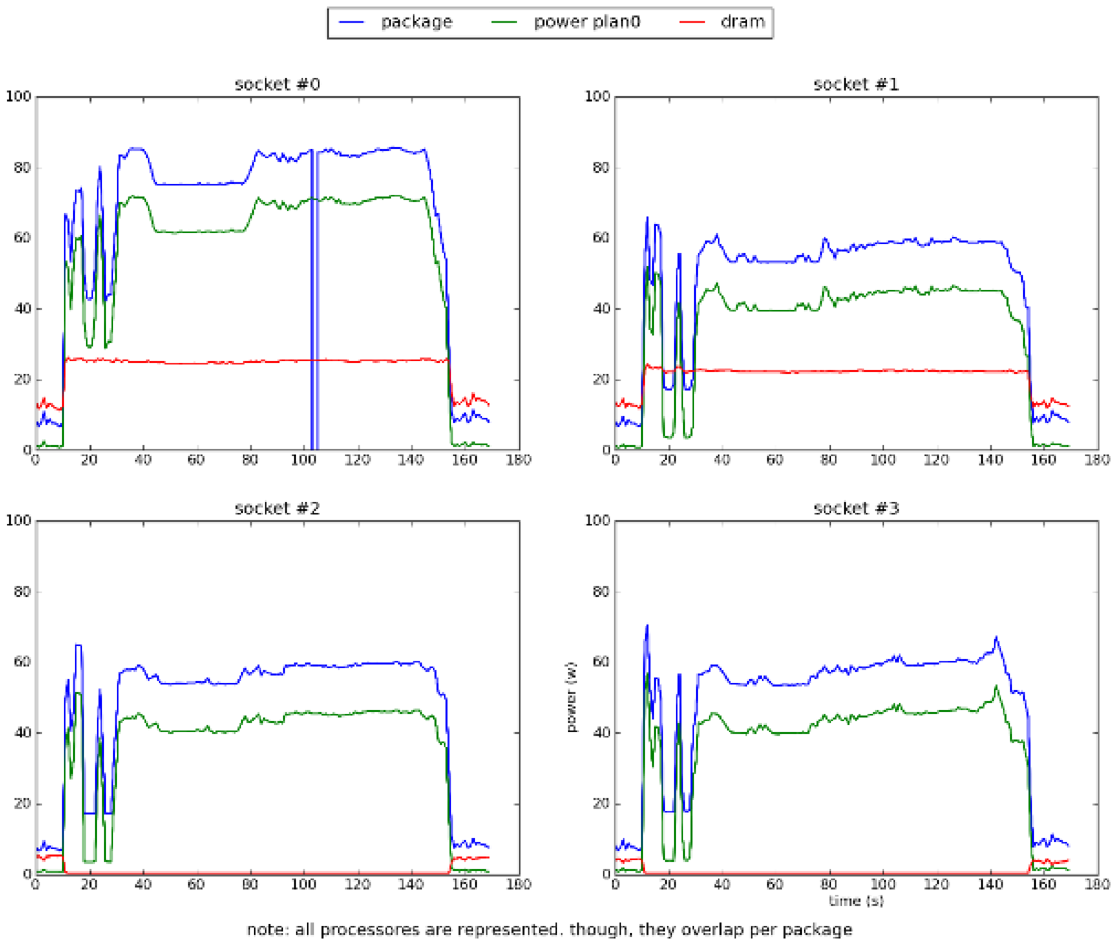


Figure 3.11: RAPL measurements of NUMA nodes - 16 processes with no explicit binding

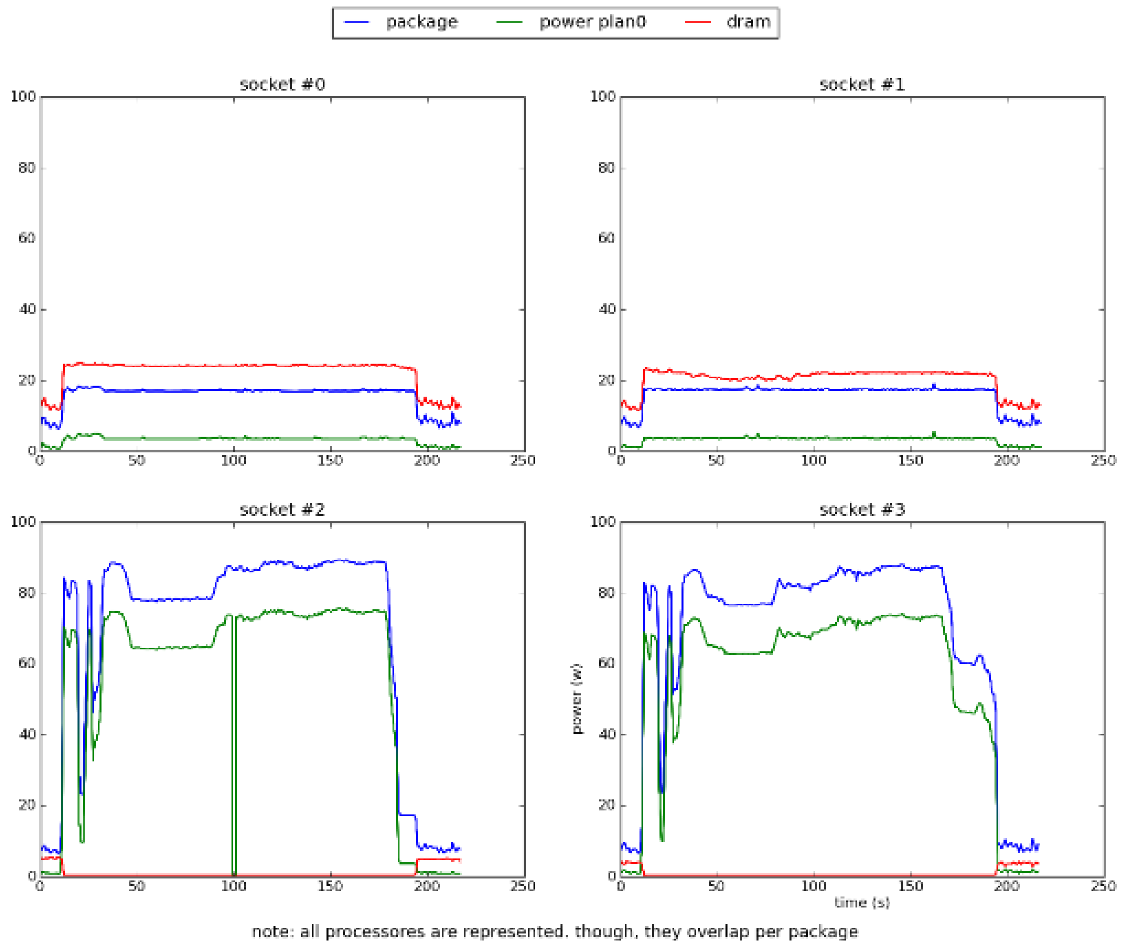


Figure 3.12: RAPL measurements of NUMA nodes - 16 processes. Explicit binding on node #2 and node #3 binding

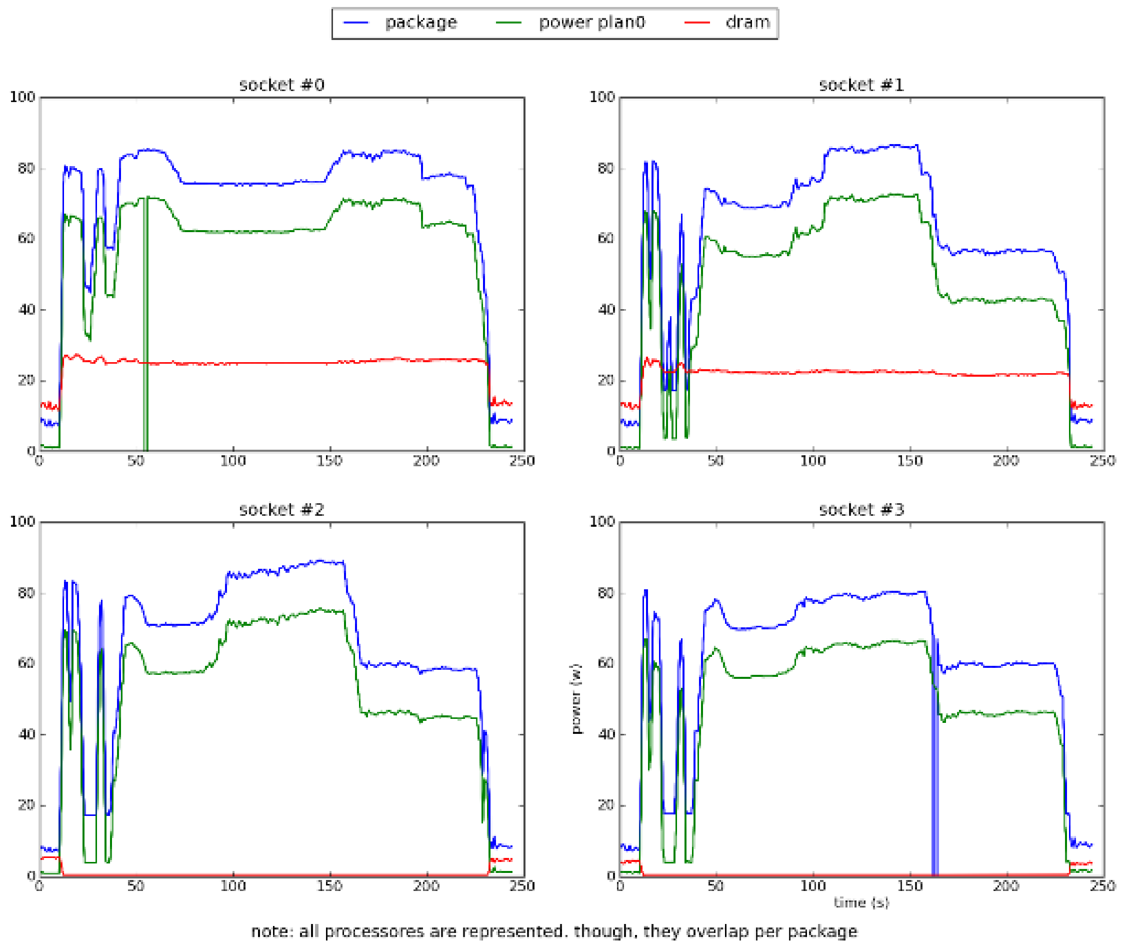


Figure 3.13: RAPL measurements of NUMA nodes - 32 processes with no explicit binding

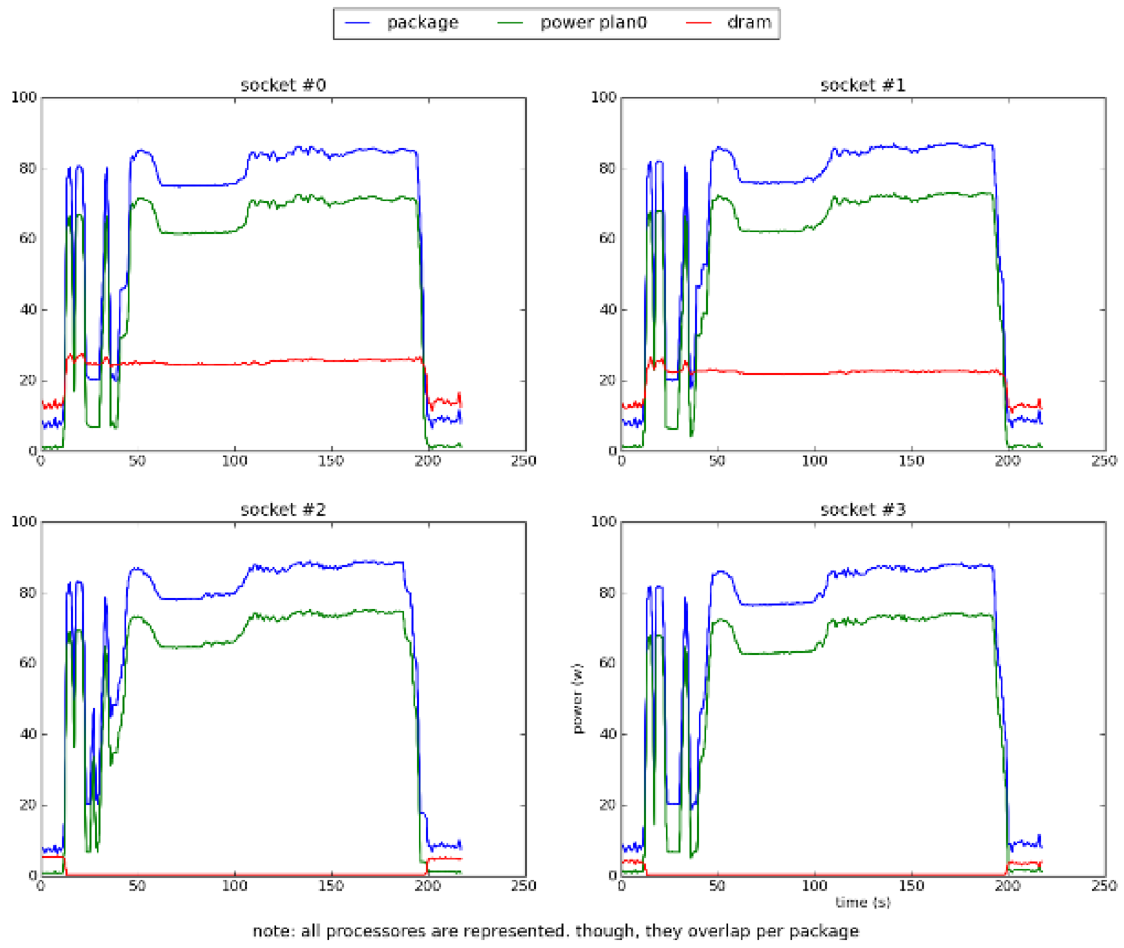


Figure 3.14: RAPL measurements of NUMA nodes - 32 processes. Processes distributed evenly explicitly - 8 processes per node.

Chapter 4

Analysis

In this chapter, we present our analysis based on the results shown in the last chapter. The scope of the analysis presented in this section is twofold: to compare the platforms from an energy efficiency perspective and analyze the tools and techniques used on the different experiment sets.

Whereas the first two sections analyze the energy efficiency of the platforms studied and the particularities of the tools and techniques used, the last section covers the results and issues which arose when using RAPL to measure the energy consumption in a NUMA environment and how the CMSSW framework performs in such conditions.

In the final of this section, we outline the highlights of the analysis for each set of experiments.

4.1 First Set of Experiments

In figures 3.1, 3.2 and 3.3 it is plotted the energy measurements from the beginning until the end of the event generation-simulation by the CMSSW. The energy measurements were done using a meter clamp. The energy measured is represented in the Y-axis and the X-axis represents the time of the experiment in samplings. For each experiment, a sample corresponds to the same time.

In the figures 3.4, 3.5 and 3.6, we trimmed out the initialization stage and connection stage of the workload and only show the event processing stage. Whereas the Y-axi represents the energy measured in Watts, the X-axis represents the time spent until the correspondent energy sampling.

Finally, the figures 3.7 and **??** compare the time spent by each of the hardware setups to process the workflow and the power consumption efficiency of the different setups, respectively.

CMSSW stages

Based on the figures 3.1, 3.2 and 3.3, we can distinguish three different patterns of energy consumption during the expriement. We refer to each pattern as being part of a different CMSSW stage. The stages can be better identified when plotting the memory workload and the CPU usage(see Figures of memory usage in GDrive - Add them or our of scope??).

The first stage consists is the initialization process. During this stage, the memory is the main module being used and thus, it is out od the scope of this work to analyse this stage in depth.

The second stage is the connection phase. The goal of this stage is to fetch the metadata from the CERN servers that allow the event generation-simulation. The metadata is needed to perform the reconstruction of the events. Once again, during this stage the CPU load is low when compared to the memory workload.

The third stage corresponds to the event processing. This last stage is CPU intensive and it has the most relevant data to to our study, since we our goal is to compare the energy efficiency of the different CPUs. The event processing stage alone is represented by the figures 3.4, 3.5 and 3.6.

1. Add memory plots? – easier to identify the 3 stages but out of scope

Relative importance of the stages

The most important stage when studying the energy efficiency of workload with the CMSSW is the last stage. There are three main reasons for that: Firstly, the CMSSW configuration at CERN has caches that speed up considerably the second stage [refs], thus reducing the energy consumed in the connection stage. Secondly, the first and second stages are not CPU intensive. Lastly, the processing stage is the only one that the energy consumption is directly porpotional o the amount of events. Therefore, given any large amount of data to be processed, the last stage will consume so much more energy than the former stages that the first two stages will become irrelevant in terms of overall energy consumption. Therefore, we focus our energy consumption analysis on the event processing stage only. The event processing stage alone is represented by the figures 3.4, 3.5 and 3.6

Overcommiting CPU and energy efficiency

We consider a CPU to be overcommited when it has to process more threads or processes than the physical cores available.

If we consider each hardware setup individually, the time needed for running the three stages of the experiment is roughly the same, if the CPU is not overcommitted. When the number of processes exceed the number of available cores, the time to process the events increases since there are no available cores to process the events concurrently. In the overcommitted situation, the time increase follows the ratio *nr_of_processes/nr_of_cores_available*. For example, if the number of processes running is 6 and the number of cores available is 4, the time needed to process the events increases roughly 2/3 compared to when the CPU is not overcommitted.

In terms of energy consumed by the CPU, we do not find any outstanding difference in terms of overall energy efficiency by comparing CPUs that are overcommited vs non overcommited, as we can seen in the Figure 3.8. However, we expect that if the ratio *nr_of_processes/nr_of_cores_available* is large enough, it can affect negatively the energy performance given the energy overhead spent when the jobs are being swapped.

Time comparison

When comparing the time taken by the different architectures to process the same task (Figure 3.7), the pattern is evident. Regardless the number of processes, the Intel_quad architecture is faster than Intel_atom and ARM_viridis and ARM_viridis is faster than Intel_atom. This fact is due to the architectures characteristics and its specifications, most notably the CPU clock speed.

Energy efficiency comparison

Given the metrics used in this study (see Metrics section in the Experiments chapter), it is clear that systems are proportionally energy efficient with its ratio performance per watts. Therefore, by analyzing the Figure 3.8, it is evident that given the architectures and its configurations, ARM architecture outperforms in terms of energy efficiency its concurrence in all considered scenarios. In addition, we conclude that between Intel architectures, Intel_atom is more energy efficient than the Intel_quad.

Measuring tools: external monitoring

For this set of experiments, the external samples were acquired and recorded manually. This factor had a visible impact on the resolution of the measurements. Clearly, the all the plot show spikes and rough transitions between samples. Moreover, the error tends to increase proportional to the human interaction with the experiment. Therefore, we conclude that it is more effective to use digital and automated ways to sample and log the data acquired during the measurements.

Measuring tools: software-based monitoring

We used software mesaurement tools to get an estimated energy consumption by the memory and other system components. In this particular set of experiments, the memory energy measurements done with software were of particular help to distinguish the different stages, which existence was unknown before the experiment. The software-based tools can be used as a decision support and for learning about unknown and unexpected system behaviours. Thus, even if the output does not directly show information about energy consumption of the system, it can be important to support and explain expected - and unexpected - behaviors.

4.2 Second Set of Experiments

Energy efficiency comparison between Intel_xeon and ARM_odroid

In the Figure 3.9, we can see the energy efficiency comparison of Intel_xeon and ARM_odroid. The righmost plot represents the internal energy measurements, whereas the leftmost plot represents the external energy measurements. As in other energy efficiency comparisons in this study, we used the metrics *nr_of_events/s/W* to represent the energy performance of the measured systems.

The main conclusion from 3.9 is that ARM_odroid outperforms Intel_xeon in both internal energy efficiency and external energy efficiency.

Energy performance and overcommited CPUs

It is noticeable that ARM_odroid has a significant energy performance decline when its cores are overcommited. It is also interesting to see that the energy performance decline in the ARM_odroid is relatively larger on the internal energy measurements. One of the reasons we found in our raw results to explain this phenomenona is the large increase of time taken to process the events when the cores are overcommited. Thus, even if the cores are consuming the same Watts per second during the event processing stage, the energy efficiency will decrease with the time taken to process the events.

On the other hand, the energy performance of Intel_xeon does not seem to be significantly affected when overcommited. This phenomenon is explain by the fact that Intel_xeon took roughly the same time to process the events when using one core per event and half a core per event.

We believe that the different results between ARM_odroid and Intel_xeon discussed below are due to the fact that ARM_odroid is a development board and it does not implement sophisticated techniques such as Hyper Threading Technology (HTT) by Intel [4]. According to Intel, HTT delivers two processing threads per physical core, which allows highly threaded applications to be processed faster. It is expected that if the ratio of *nr_of_threads/core* would be larger than 2, energy efficiency of Intel_xeon would start to decline.

We believe that if we would overcommit Intel_xeon with more than 4 threads per core, the time to process the workload would increase, which would be followed by a degradation of energy performance.

Measurement tools and techniques

The internal measurement tools used in ARM_odroid and Intel_xeon provide a fine grained resolution to the core level. The TI INA231 and RAPL chips can isolate the pp0, which consists of ALU, FPU, L1 cache and L2 cache when performing energy measurements.

On the other hand, as stated in [5], the lower resolution that IPMI tools offers for internal measurements include energy consumed by the 0P9V, 1P8V, VDD and Vcore rails, which includes the system on the chip, DRAM, Temperature Sensors, and ComboPHY Clock. The components that are measured by the IPMI tools at each energy sample are shown in the Figure **??**.

As a result of this measurement discrepancy, 3.10 shows that ARM_viridis performs worse than any other machine. We believe that this result can be misleading, due to the fact that the tools used to measure the energy consumed by each of the setups measure different components in the CPU. We believe that if components measured in the ARM_viridis would be same as the components measured on the ARM_odroid and Intel_xeon measurements, we would obtain a different result. Namely that ARM_viridis would, at least, perform better that Intel_xeon from an energy consumption perspective. Given the actual setup, we can not scientifically directly compare the results.

Comparison between First set of experiments and Second set of experiments

When we compare the main results of the 1SE (Figure 3.10) and 2SE (Figure 3.9) we may be inclined to conclude that the setups in the 2SE presented an overall more efficiency than the setups the 1SE. Again, the used measurement

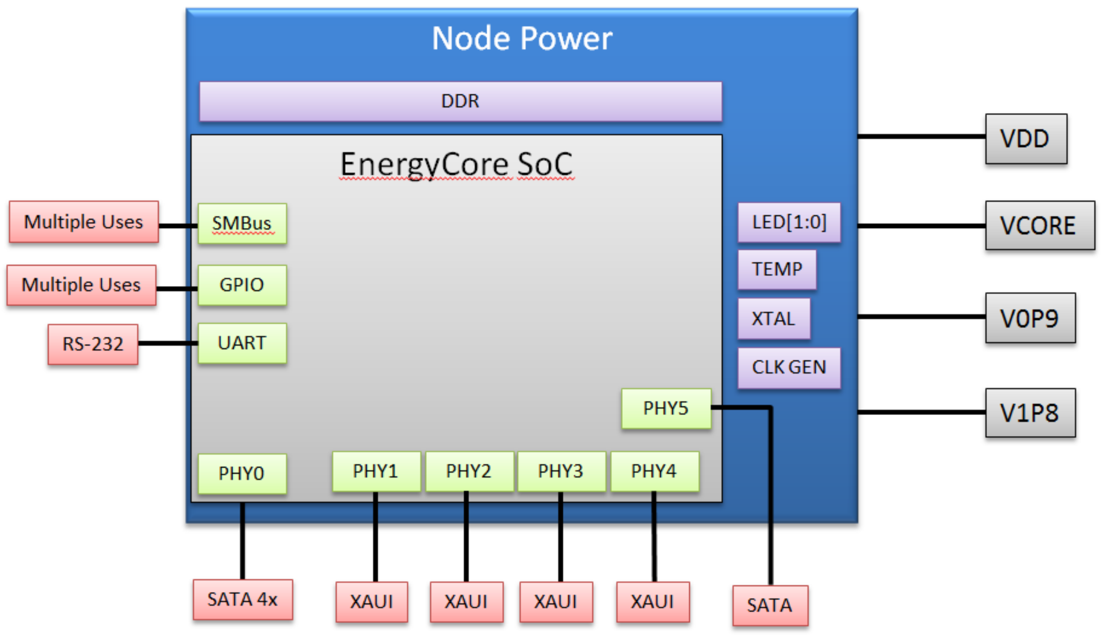


Figure 4.1: Representation of the Power Node measured by IPMI tools [make one myselfd and change!]

tools play an important role and should not be disregarded when analysing the results. In the 1SE, we only performed external measurements. Thus, we discard the possibility to compare the 1SE results with the results of the internal measurements of the 2SE. As for the external measurements performed in both set of experiments, the tools for measuring the energy consumption of both experiments have disctinct resolution and grain. In the 1SE, we used the clamp meter for measurements in all setups. As for the 2SE, we used embeeded and computer-assisted tools to perform the external measurements. This discrepancy of tools, its resolutions and errors, make it difficult to compare the results of the 1SE and 2SE.

However, we can conclude that ARM architecture outperforms the Intel architectures in each and every experiment, regardless the measurement tools and methodologies used.

4.3 RAPL in a NUMA environment

Intel Xeon, using RAPL to measure energy consumed by the different nodes, with different types of binding

Should I include this in the thesis? I worked briefly on this at CERN and couldnt conclude anything because RAPL in the Intel_xeon was not 100 per cent compatible with the NUMA architeture used. Also, it goes a bit out of scope of the thesis. Maybe I should drop this part?

4.4 Conclusions

The main conclusion of our experiments is that given the setups used in our study, ARM architecture outperforms Intel in terms of energy efficiency.

We learned that the gen-sim mode of CMSSW has different stages and the most relevant from an energy consumption point of view is the latest one, when the events are processed.

In addition, we leanerd that the tools used to make the experiments play a crucial role in the whole experiment. It is important to assure that the measurement tools and methodologies in use are compatible and suitable to produce results that can be compared. This aspect can be hindered based on the availability of hardware and measurement tools.

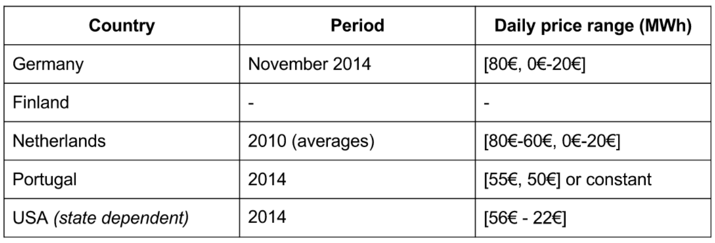
Chapter 5

HTC in a dynamic energy price market

One of the ways to utilize the conclusions from the experiments conducted in this study is to actively lower the energy bill in HTC. One approach could be to schedule jobs between more energy efficient but slower ARM architectures and the less energy efficient but faster Intel machines. This approach makes sense in a multi energy price ecosystem. A multi energy price ecosystem is an energy market where the prices float accorading to the overall power grid usage.

In this chapter, we present a study about the potential of an algorithm that schedules workload to machines with different energy and computation performance, based on the the daily dynamics of energy price. The main goal is to leverage computing heterogeneity to achieve the optimal ratio between work produced and price paid.

5.1 Dynamic eletricity pricing model



Country	**Period**	**Daily price range (MWh)**
Germany	November 2014	[80€, 0€-20€]
Finland	-	-
Netherlands	2010 (averages)	[80€-60€, 0€-20€]
Portugal	2014	[55€, 50€] or constant
USA *(state dependent)*	2014	[56€ - 22€]

Figure 5.1: Examples of dynamic power energy pricing in different markets

We will use a simplified dynamic pricing model based on the empirical research of such models in real power grids. Based on some examples obtained, we can conclude that not all the countries adopt a dynamic eletricity pricing model 5.1. For the sake of this study, we will consider a hypothetical case where the dynamic pricing model works as in 5.2. The red line in 5.2 represents the eletricity price along the day. For the sake of simplicity, during this study we define that the price of eletricity can is 60 euros/Mwh during 12 hours per day and 20 euros/Mwh during the remaining 12 hours of the day.

As we can see based on the 5.2 and 5.1, our simplified energy model presents a similar pattern to the energy prices of some countries.

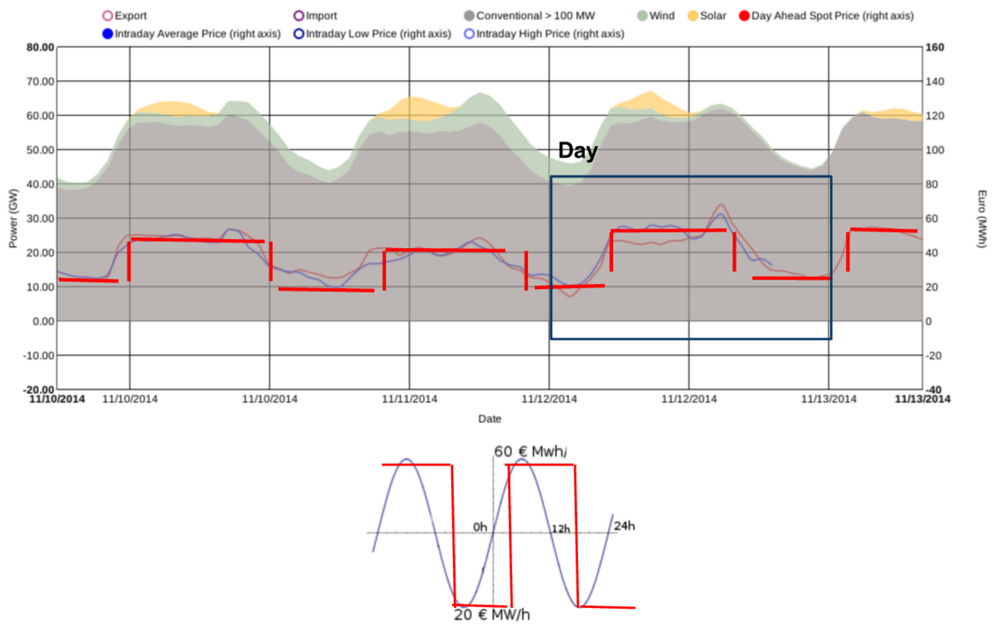


Figure 5.2: Simplified pricing model based on the Germany energy market

5.2 Scheduling algorithm

Problem formulation

The main goal of the algorithm is to lower the electric bill in dynamic eletricity markets. The algorithm schedules HPC workload among nodes with different energy profiles, depending the energy price at the time. In addition, the algorithm should ensure that a certain minimum amount of workload is processed. Thus, the algorithm input can be defined as (see also 5.3):

Energy profile of the machines:

Energy efficiency of the machines (ev/s/W);

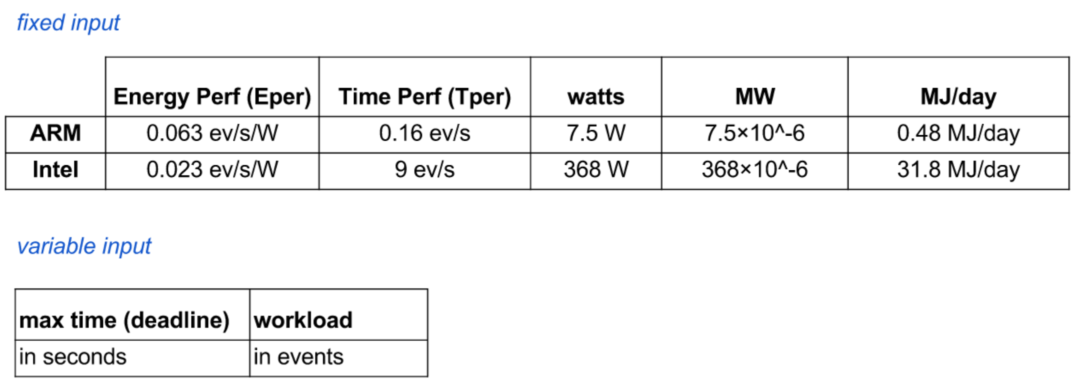
Time performance (ev/s);

Computing requirements:

Time deadline (s);

Nr. events to be processed (ev);

Therefore, given a set of machines with different energy profiles; the computing requirements (how many events must be processed in how much time); and the energy pricing dynamics during 24h, *what is the optimal machine scheduling that ensure the computing requirements and achieve the lowest price budget at the end of 24 hours* ?



fixed input

	Energy Perf (Eper)	Time Perf (Tper)	watts	MW	MJ/day
ARM	0.063 ev/s/W	0.16 ev/s	7.5 W	7.5×10^-6	0.48 MJ/day
Intel	0.023 ev/s/W	9 ev/s	368 W	368×10^-6	31.8 MJ/day

variable input

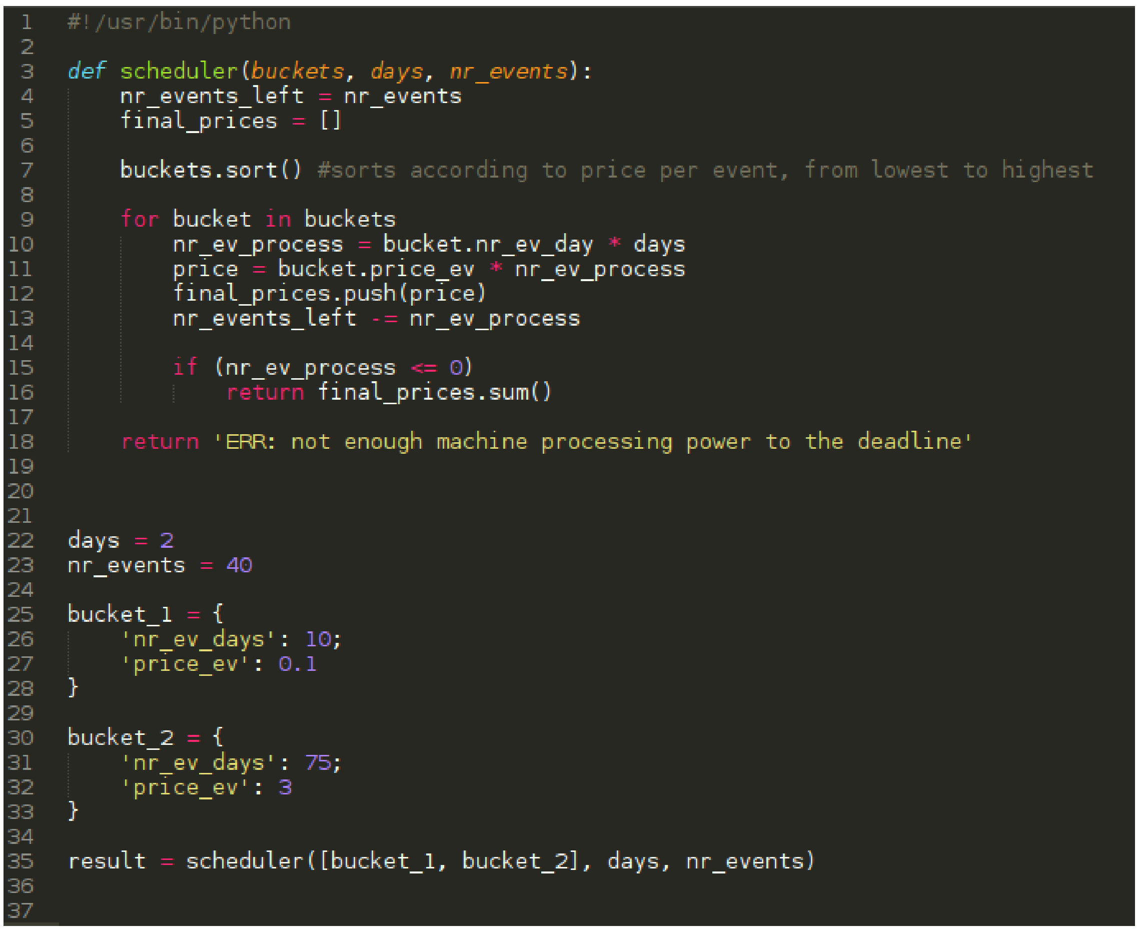
max time (deadline)	**workload**
in seconds	in events

Figure 5.3: Example input for the scheduling algorithm

The scheduling algorithm

5.3 Further developments

Well known algorithms such as job shop and others can be applied using the same rational. We expect that different algorithms present different results



```python
#!/usr/bin/python

def scheduler(buckets, days, nr_events):
    nr_events_left = nr_events
    final_prices = []

    buckets.sort() #sorts according to price per event, from lowest to highest

    for bucket in buckets
        nr_ev_process = bucket.nr_ev_day * days
        price = bucket.price_ev * nr_ev_process
        final_prices.push(price)
        nr_events_left -= nr_ev_process

        if (nr_ev_process <= 0)
            return final_prices.sum()

    return 'ERR: not enough machine processing power to the deadline'



days = 2
nr_events = 40

bucket_1 = {
    'nr_ev_days': 10;
    'price_ev': 0.1
}

bucket_2 = {
    'nr_ev_days': 75;
    'price_ev': 3
}

result = scheduler([bucket_1, bucket_2], days, nr_events)
```

Figure 5.4: Proposed scheduler algorithm written in Python

and that the nearly-optimal scheduling algorithm can be achieved with one well studied existing algorithm. We believe that to explore the potential of the heterogeneous HTC in a dynamic energy pricing market further on can show big savings in the future.

5.4 Conclusions

Our solution takes a different perspective when compared with related research. Studies like [15] and [17] do not take the dynamics of electrical pricing into consideration. However, their algorithm is already quite complex and proved NP-complete, to the point they have to come up with heuristic algorithms to apply it in the real world.

Therefore, our approach may have some novelty in a really narrow and still unexplored idea: to develop a scheduling algorithm for heterogeneous HPC that takes into consideration the nodes' energy profile, the dynamic electricity price and also, eventually, the tasks' energy profiling. The algorithm would schedule the jobs in order to minimize the energy consumption and energy bill (note: energy consumption and energy bill are not the same thing), while the deadline is met.

However, there are some open points that we still have might want to consider. First, as [16] mentions, it is important to insure that the hardware existent in the data center is used at its full potential, in order to not waste the investment made when it was purchased. Our solution, though, does not insure that since the idea is to power down/idle machines that are less power efficient in high-peak times. Secondly, from a practical perspective, if we consider only the scheduling between ARM and Intel architectures, it seems not likely that the data center will haver the same software running over both architectures at the same time, give the expertise and investment needed to have the application stack running properly in both architectures (as we witness with CERN's efforts). If we decide to abstract from that point and see the machine's architectures as a black box, then that's not a problem. Thirdly, comparing with other recent research works such as [15], our algorithm model seems to be over simplifying the problem to an extent that might hinder our purposes of creating a practical and energy efficient scheduling algorithm for heterogeneous HPC under dynamic electrical pricing.

- Use the greedy and jobshop models to schedule works across different energy profile machines.
- Use experiments done to characterize the machines
- Code scheduling algorithm

Bibliography

[1] Arm big.little technology. http://www.arm.com/products/processors/technologies/biglittleprocessing.php. Accessed: 2015-2-27.

[2] Arm cortex a9 website. http://www.arm.com/cortex-a9.php. Accessed: 2015-2-27.

[3] Cortexa9 specs. http://www.arm.com/products/processors/cortex-a/cortex-a9.php. Accessed: 2015-10-27.

[4] Hyper threading technology. http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html. Accessed: 2015-2-27.

[5] Hyper threading technology. http://wiki.bostonlabs.co.uk/w/index.php?title=Calxeda:Get_node_power_using_ipmitool. Accessed: 2015-2-27.

[6] Intel atom processor d525 specs. http://ark.intel.com/products/49490/Intel-Atom-processor-D525-1M-Cache-1. Accessed: 2015-10-27.

[7] Intel core2 quad processor q9400 specs. http://ark.intel.com/products/35365/Intel-Core2-Quad-Processor-Q9400-6M-Cache-2. Accessed: 2015-10-27.

[8] Intel xeon processor e5-2650. http://ark.intel.com/products/64590/Intel-Xeon-Processor-E5-2650-20M-Cache-2_00-GHz-8_00-GTs-Intel-QPI. Accessed: 2015-10-27.

[9] Ipmi tools. http://www.boston.co.uk/technical/2012/03/supermicro-ipmi-what-can-it-do-for-you.aspx. Accessed: 2015-10-27.

[10] Odroid-xu3 specs. http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127&tab_idx=1. Accessed: 2015-10-27.

[11] Odroid xu3 website. `http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127`. Accessed: 2015-2-27.

[12] Viridis boston website. `http://www.boston.co.uk/solutions/viridis/introducing-the-viridis-2.aspx`. Accessed: 2015-2-27.

[13] Abdurachmanov, D., Elmer, P., Eulisse, G., Knight, R., Niemi, T., Nurminen, J. K., Nyback, F., Pestana, G., Ou, Z., and Khan, K. N. Techniques and tools for measuring energy efficiency of scientific software applications. *CoRR abs/1410.3440* (2014).

[14] et al, S. A. Geant4 a simulation toolkit. In *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, pp. 250–303.

[15] Wang, Y., Li, K., Chen, H., He, L., and Li, K. Energy-aware data allocation and task scheduling on heterogeneous multiprocessor systems with time constraints. *Emerging Topics in Computing, IEEE Transactions on 2*, 2 (June 2014), 134–148.

[16] Yang, X., Zhou, Z., Wallace, S., Lan, Z., Tang, W., Coghlan, S., and Papka, M. Integrating dynamic pricing of electricity into energy aware scheduling for hpc systems. In *High Performance Computing, Networking, Storage and Analysis (SC), 2013 International Conference for* (Nov 2013), pp. 1–11.

[17] Zeng, G., Yu, L., and Ding, C. An executing method for time and energy optimization in heterogeneous computing. In *Green Computing and Communications (GreenCom), 2011 IEEE/ACM International Conference on* (Aug 2011), pp. 69–74.

Appendix A

Techniques and tools for measuring energy efficiency of scientific software applications

The article in appendix is called *Techniques and tools for measuring energy efficiency of scientific software applications* and it is the result of the research about tools and techniques for energy consumption done during this study. The article was published in the conference proceedings of the 16th International workshop on Advanced Computing and Analysis Techniques in physics research (ACAT'2014).

PDF append article