Towards Generalizing "Big Little" for Energy Proportional HPC and Cloud Infrastructures

Violaine Villebonnet, Georges da Costa, Laurent Lefevre, Jean-Marc Pierson, Patricia Stolf

Towards Generalizing "Big.Little" for Energy Proportional HPC and Cloud Infrastructures

Violaine Villebonnet*†, Georges Da Costa*, Laurent Lefevre†, Jean-Marc Pierson* and Patricia Stolf*
*IRIT, University of Toulouse, France
†Inria Avalon Team - LIP Laboratory - Ecole Normale Superieure of Lyon, University of Lyon, France

***Abstract*—Reducing energy consumption is part of the main concerns in cloud and HPC environments. Today servers energy consumption is far from ideal, mostly because it remains very high even with low usage state. An energy consumption proportional to the server load would bring important savings in terms of electricity consumption and then financial costs for a datacenter infrastructure. In this paper, we propose a platform composed of heterogeneous architectures to achieve proportional computing goal. We select low power ARM processor for a light load, and a range of regular x86 servers when performance is required. We propose a comparative study of benchmark execution in order to find the best configuration depending on the current load and show the effective results in terms of energy proportionality.**

***Keywords*—heterogeneous architectures, energy proportionality, virtualization, emulation, ARM processor**

I. Introduction

With the explosion of data in last few years and its evolution tendency for the future, the need for datacenters is growing faster and faster. Whether they are directed towards cloud or HPC applications, their number is exploding, and so is their energy consumption. In [1], authors estimate that worldwide datacenters consumed up to 270 TWh in 2012, which accounts for almost 2% of global energy consumption. Therefore this represents the main limitation for building such an infrastructure because electricity is the most important cost. In addition, inside a datacenter not all the consumed energy goes to computing. Despite the fact that there are some loses due to power conversion, and that a subsequent fraction of the total energy is needed for the cooling infrastructure, servers are in most cases always powered on even if they are idle. The problem is that when a server is up but idle (powered on but without activity), its energy consumption is already significant, and wasted. Some idle servers can consume as high as 50% of its maximum power consumption when fully loaded.

The need to achieve proportional energy consumption in virtualized large-scale clusters, grids and clouds, has been raised for the first time by Luiz Andre Barroso and Urs Holzle in 2007 [2]. They conducted experiments in a Google datacenter, and noticed that servers are mostly used at a load between 10 and 50%. This means they are rarely completely unused, and therefore in a state where they could be shut down, and also rarely at full performance, where they are the most energy efficient. The energy consumed when a machine is idle is called the static consumption and this is the issue we want to tackle in our work. For example on Fig.1 from Barroso and Holzle paper, the static consumption represents 50% of the peak. Our goal is to reduce this static cost as much as possible, and to extend the dynamic consumption to 100%. Graphically, we want to replace the green curve by a straight line starting from 0 and going to the peak in a fully proportional way. An architecture with such a consumption pattern would bring significant energy savings.

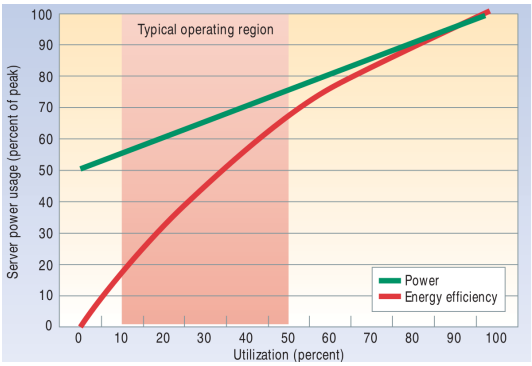


Fig. 1. Server power consumption and energy efficiency from 0 to 100% utilization (Figure from [2])

We start this paper by an overview of some related works on energy efficiency and proportionality in next section. Then we detail our different contributions in section III, and describe our experimental platform in section IV. Along section V we comment the obtained experimental results. To wrap up our article we draw some conclusions and propose some perspectives in section VI.

II. Energy efficiency and proportionality

Energy savings in clouds and HPC infrastructures is a popular and quite recent research field. Many works have been done in this area to find solutions to optimize the utilization of resources inside datacenters. A famous approach is consolidation which consists in gathering the working load on the fewest number of servers to be able to switch off the unused ones. This can be achieved thanks to virtualization which allows several independent operating systems to coexist on a single physical machine. Live migration [3] is the mechanism used to dynamically move virtual machines through physical servers without impacting applications running inside. Performing consolidation aims at saving energy by freeing lightly loaded machines. The goal is to switch unused servers off, or put them in a low power mode, and only turn them on when they are needed. This idea is not as simple as it seems because switching off and on a server takes time and consumes extra power. Hence these actions must be well decided to actually save energy. Most consolidation approaches are based

on heuristics algorithms, which are variants of the bin packing problem, but other alternatives have been proposed and tested such as constraint programing [4], genetic algorithms or Ant-Colony metaheuristics [5]. Another green leverage is DVFS, which stands for Dynamic Voltage and Frequency Scaling. The principle is to adapt the frequency of the processor to the current server needs, because the energy consumption decreases when the frequency is reduced. However like other leverages, important energy savings can only be reached if those actions are performed wisely, and this will rely on a good knowledge or prediction model of the workload. In [6], the authors propose to monitor performance counters in order to get current profiles of running systems and predict their evolutions. This system allows to take decisions according to the predictions and thus make more effective energy savings.

But these approaches have some limitations, they only try to reduce the overall energy consumption. Consolidation enables the servers to be fully loaded, where they are the most efficient, but the problem of high static consumption remains. With our work we want to bring a solution which eliminates static costs by trying to reach energy proportionality. The goal is to approach a nearly null consumption at idle state, and then a linear consumption proportional to the load. If such a proportional hardware, or system, could exist, then consolidation would not necessarily be needed because the energy efficiency of the system would be constant.

Regarding proportional computing, some works [7] [8], propose metrics to evaluate the proportionality of an architecture. The first one compares the consumption curve as a function of load, to the ideal proportional linear curve. While the second one defines two separate metrics: one to measure the difference between idle and maximum consumption, and another to measure the linearity of this consumption. These metrics are then applied to existing architectures to study the evolution of the hardware from this point of view through recent years. The architectures are in general more and more proportional, but it is noted that meanwhile the gap between maximum and idle consumption is reduced, the linearity is degraded. Perfect proportionality is then not reached yet and still seems far away.

Consequently, one solution to achieve the goal of energy proportionality is to use several architectures with different performance and consumption characteristics. This is the concept used in heterogeneous multicore processors, which are also called hybrid processors. Different companies have proposed their implementation of this concept like ARM with their big.LITTLE processor [9] which combines a low-power processor with a high-performance one, or Nvidia with their new Tegra K1 processor [10] that couples ARM processors with GPU accelerators. Inside those systems, the applications are chosen to run on the processor that best suit their computing needs. Moreover, they feature a shared cache memory thanks to a cache coherence interconnect system that eases the migration of tasks between the processors.

Fig.2 depicts the architecture of big.LITTLE hybrid processor according to ARM itself. On the left, Cortex-A15 plays as the 'big' processor and on the right Cortex-A7 is the 'little' one. The particularity of this proposition resides in the CCI module, functioning with interruptions, which brings full coherency between the two processors. This concept allows a nearly transparent task migration from one processor to another, and this enables to better fit to the evolutions of application resource needs. ARM proposes different forms of utilization of this architecture : CPU Migration and Global Task Scheduling. In the first one, each big core is paired with a little one, and only one core of each couple can be active at a time. Whereas in the second form, all cores are viewed in a global way and any core can be active or shut down independently. The last option offers more flexibility but also more complexity and brings many challenges for the system management.

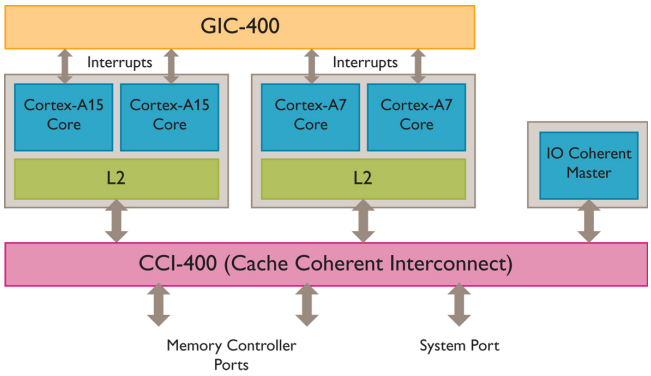


Fig. 2. big.LITTLE system architecture (figure from ARM white paper [9])

Hybrid multiprocessor is an innovative approach, but for the moment they are dedicated to mobile devices. The idea behind this concept is to extend battery life duration for mobile devices, so to consume as little as possible during the idle periods, while delivering good performances when needed, for instance for game playing or video watching. Our goal is to extend this concept at a datacenter scale. We want to exploit ARM processors with low power consumption when the load is low and keep the traditional servers for the performance. Although our idea is inspired from this concept, the two approaches differ on some points. Hybrid processors involve threads migrations while at machines level it implies migrations of virtual machines or applications containers. This kind of migrations represents a challenge because it is not easy to have a shared memory among heterogeneous machines. Furthermore, as the heterogeneity resides at the architecture level, the design of applications or virtual machines are different. Next challenge is to find the most effective and appropriate way to take advantage of the different architectures characteristics by performing migrations at the right time.

In [11] they consider having heterogeneous architectures with stateless web servers. A set of hardware composed of Raspberry Pi, Intel Atom and Intel i7 is considered. It shows that for a load up to 50 requests per second, it is more interesting to use a combination of low power processors (Pi and Atom) than using one i7. Doing so, the power consumption curve gets closer to the ideal proportional. However, this work only applies to this specific application type, and for more complex applications such as statefull web servers, additional mechanisms are implied. Our work targets a vaster range of applications that is why we focus on virtual machines to be the most general possible.

III. Contributions

A. Objectives

Our goal is to always execute applications on the most suitable architecture at any time. The most suitable architecture is the one that consumes the least for the current performance needed by the application. When the performance requirements of the applications evolve, the architecture where it is currently executing may no longer be the most suitable, and we may need to move the application to a different architecture. For instance if the CPU load decreases, the application should be migrated to a less powerful and less consuming architecture in order to save energy, but if it increases, the application must be transferred to a more powerful architecture in order to satisfy its needs and not impact negatively on its execution. To do this, we use live migration which is widely used in today's datacenters, but the architecture heterogeneity that we bring makes it more difficult and we have to find a way to enable live migration between heterogeneous machines.

As previously said, proportional hardware does not exist yet, hence we decide to choose different pieces of hardware to approach a proportional one. Nowadays, datacenters are mostly composed of x86 processor based servers. Since the 2000s, almost all these processors, built by Intel and AMD, have 64 bits memory addressing. They have a good performance/price ratio, and are the most widespread. However, their main drawback is their high power consumption in idle state. We consequently focus on very low consumption processors to see if we can counteract these static costs. It appears that ARM processors offer the best compromise between performance and power consumption. Indeed, ARM processors are historically designed for embedded systems so the low power consumption was the main constraint. But now they are more and more designed for mobile devices such as smartphones and tablets, thus they are becoming more and more powerful. In addition, some of them recently include virtualization extensions. This last point has strengthened our idea to bring those processors into a datacenter.

B. Virtualization challenge

Our first concern is to study the existing virtualization solutions and find if some of them are compatible with both ARM and x86 architectures, and if they can be used to perform live migration, or have a mechanism of checkpoint/restart. We also want to study other specifications such as operating systems, kernel versions, to see which solution is the least restrictive. Our objective is to select a technology upon these criteria, which will be a good basis to develop an extended migration that works between heterogeneous architectures.

We consider two main categories : virtual machines and application containers. We focus on open source solutions, that is why we selected KVM and Xen hypervisors for the virtual machine approach, and LXC and OpenVZ for containers.

Although application containers seem to be a promising technology with a very light virtualization process and then a very low overhead, it implies many constraints. Linux containers only work with Linux based OS, and the guest shares the same operating system as well as the same kernel version as the host. Moreover we observe that checkpointing for containers is still a feature in development whereas live migration is well implemented in hypervisors like Xen or KVM. OpenVZ has a functional live migration but it works only on x86 hosts. As far as LXC is concerned, developers are not planning to implement any kind of live migration, but some work is done about checkpoint and restart of LXC containers inside the CRIU project - which stands for Checkpoint/Restore In Userspace.

TABLE I. Comparizon of virtualization solutions

	Virtual machines		Linux containers	
	Xen	KVM	LXC	OpenVZ
On x86	yes	yes	since 2.6.29	patched kernel
On ARM	since 3.7	since 3.9	since 2.6.29	patched kernel
Live migration	yes	yes	not yet (CRIU project)	yes, but not on ARM
Guest OS	any	any	only Linux based	only Linux based

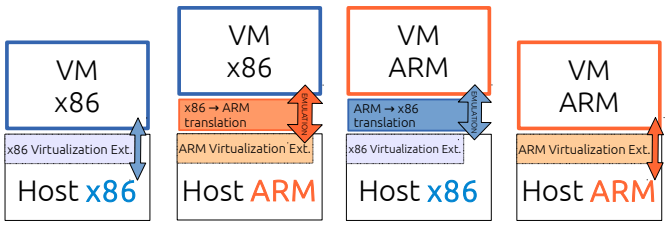
This comparison leads us to select the virtual machine solution as it is the most common approach in datacenters and also the most general solution as it does not impose any restriction on application type. The two propositions KVM and Xen are quite equivalent, we have chosen the first one because of previous work experience with it.

C. Virtual machine and emulation

As we propose to gather two different physical architectures, ARM and x86, in the same datacenter, it means we also have to make a choice for the virtual machine architecture. Our idea is to select one architecture for all the virtual machines. When the virtual and the physical machines share the same architecture then we benefit from the virtualization extensions. On the contrary, if the two architectures are different, we have to use emulation. Emulation is a concept which allows to execute programs compiled for an architecture different from the host we have. It consists in an hardware abstraction and the program will be executed through dynamic translation of the binary instructions.

For this purpose we have chosen QEMU emulator because it is closely related to KVM. In fact QEMU can detect if the virtual machine and the host have the same architecture, in this case emulation is not needed and it automatically uses virtualization extensions of the hardware. Hence our idea is based on the assumption that it could be possible to migrate one virtual machine of fixed architecture between two different hosts. During the migration, the system should just have to switch from emulation to virtualization extensions, or the opposite, according to the architecture of the source and destination hosts. Status of our work about migration is detailed in section V-E.

Figure 3 pictures the two alternatives for the virtual machine and their underlying functioning. First and last cases have low overhead thanks to the virtualization extensions while the two cases needing software translation suffer from a high performance impact. The resulting overhead of emulation is discussed in section V-A.



(a) x86 VM (b) ARM VM

Fig. 3. Emulation / Virtualization extensions

IV. Experimental platform

A. Hardware and power monitoring

We have selected the ARM Cortex-A15 processor for its low power, its good performances and its virtualization extensions. It is a quite recent processor, its first implementation was done by Samsung with the Exynos5250 SoC. The first device powered by this chip is the Samsung Chromebook released in 2012. One year later, HP released the HP Chromebook 11 which has this same SoC and similar technical characteristics. As their code names suggest, these notebooks come with Google's Chrome Operating System, but to be able to use KVM software and virtualization extensions we need a Linux distribution. Moreover as mentioned in Table I, the Linux kernel version should be equal or posterior to 3.9. We have managed to make the Samsung Chromebook boot an Ubuntu 12.04 with a Linux kernel 3.13, and installed QEMU 2.0 and KVM for virtualization extensions. We use a Plogg, an external watt-meter to get instantaneous power consumption of the notebook. Resulting experiments with an ARM virtual machine are detailed in section V-D. Concerning the experiments with binaries execution which do not require virtualization experiments we use the HP Chromebook. We made it run an Ubuntu distribution based on the ChromeOS kernel already installed. To get power consumption information we use powerstat Ubuntu package that gets monitoring data from the battery via ACPI.

For x86 architecture the choice is much larger. In order to benefit from servers with power monitoring, we have run our experiments on the Grid'5000 testbed. Grid'5000 is a French experimental platform, geographically distributed over 11 sites in France and Luxembourg, dedicated to scientific research concerning large scale infrastructures. We have chosen an Intel Xeon and an AMD Opteron from monitored clusters located respectively in the cluster named Taurus of Lyon and the Parapluie cluster of Rennes. We find relevant to select two kinds of x86 servers because it allows to highlight the possible differences between two generations and two constructors of quite similar servers. Both servers run a Debian Wheezy operating system with QEMU 1.7 installed. In Lyon, electrical consumption is acquired thanks to watt-meters from Omegawatt and accessible on Grid'5000 intranet, whereas in Rennes monitored PDU from EATON are used and power data is fetched via SNMP requests.

Characteristics of the hardware are detailed in the following table, as well as a comparison of power consumption and native performances.

TABLE II. Summary of selected hardware

Codename	Chromebook	Taurus	Parapluie
Fullname	Samsung // HP 11 Chromebook	Dell PowerEdge R720	HP Proliant DL165 G7
Architecture	ARMv7 32 bits	x86 64 bits	x86 64 bits
CPU	2 x Cortex-A15	2 x Intel Xeon E5-2630	2 x AMD Opteron 6164
Total cores	2	12	24
Power consumption	5 – 25 W	96 – 227 W	180 – 280 W
Release year	2012 // 2013	2012	2010

One striking point here is the huge difference between idle consumptions. Parapluie idle power is more than 20 times greater than the Chromebook, and 2 times greater than Taurus. The upper bound corresponds to the maximum measured power consumption when the processor is fully loaded.

B. Benchmarks

As x86 virtual machine on ARM host is not fully functional at the time we are writing, we made experiments with QEMU User Emulation which allows to execute binaries compiled for a different architecture by dynamically translating the instructions during the execution. Not all programs can be executed with this type of dynamic translation, we need an application compiled with statically linked libraries. For this purpose, we have chosen the nbench [12] benchmark program. It is a simple application written in C, which is composed of several subprograms designed to test CPU capabilities of a machine. We have chosen the Idea encryption program for integer computation and the Fourier algorithm regarding float computation.

Because QEMU User Emulation is restrictive and does not allow to execute cloud applications, we also made some experiments with actual virtual machines. We have selected a database based benchmark in order to illustrate the cloud aspect of our work. TPCC-UVA benchmark [13] is a free and open-source implementation of the standardized TPC-C benchmark from Transaction Processing Performance Council. It consists in a set of requests against a PostgreSQL database containing 9 data tables whose size differ according to input specifications.

V. Experiments results

Throughout the following subsections we present both HPC and cloud scenario with different benchmarks and experiments. In the goal of evaluating if emulation is a good perspective, we measure the emulation overhead, mainly in terms of performance but also concerning energy consumption. Next, to show the benefits of having heterogeneous architectures inside one datacenter, we execute the same program on each selected hardware and observe its behavior. This is a necessary step to be able to know which hardware is the most interesting in terms of power consumption for each level of performance.

A. Emulation overhead

TABLE III shows the overhead of emulation, by dynamic translation, for each selected hardware and for two types of computing benchmark. Column 'Int' refers to Idea encryption

TABLE III. OVERHEAD OF EMULATION FOR EACH HARDWARE

	Native		Emulation		Overhead	
	Int	Float	Int	Float	Int	Float
Chromebook (ARM)	8233,9	27251	932,46	604,07	8,83	45,11
Taurus (x86)	102893,9	380437	11479,22	11153,06	8,96	34,11
Parapluie (x86)	113569,8	320823	15239,46	12599,76	7,45	25,46

program, and 'Float' to Fourier algorithm of nbench benchmark. The first column is the maximum number of iterations per second for a native execution, and the second one is for an execution of the 'opposite' architecture binary via QEMU user emulation program.

The last column whose title is 'Overhead' represents the ratio between emulation performances and native performances. We realize that the order of magnitude of the overhead is the same no matter the underlying physical architecture. For integer computation the emulation is around 7 to 9 times slower, while for float computation the overhead is much important, from 25 to 45 times, the largest being for the Chromebook.

Even if x86 processors are natively more powerful than ARM ones, (about 12 to 13 times in our examples) the important overhead causes the emulation to slow down a lot all the processors. We can even notice for float computing that the ARM native execution is in fact more powerful than the emulated execution on x86 servers. Indeed, if we consider an ARM compiled float computation, the Chromebook reaches 27251 iterations per second whereas Taurus and Parapluie reach respectively 11153,03 and 12599,76 iterations per second. In the following sections, we discuss the performances together with the corresponding energy consumption of each hardware.

B. Comparison study of binaries executions

In these experiments, the aim is to make a comparative study of the two solutions for the virtual machine architecture as depicted in Figure 3. Except here we are not dealing with full virtual machines but only with binaries execution, natively or through dynamic translation.

The following two figures, 4 and 5, show the average power consumption for an evolving number of iterations per second, from 0 to maximum, of the IDEA benchmark from nbench. The curve starting point is the average power consumption at idle state, and the ending point of each curve corresponds to the average power during a complete execution of the benchmark. We have slightly modified the nbench benchmark by introducing 'nanosleep' calls in order to reduce the maximum performance and then get more data points. We have chosen to run the benchmark five times with five different duration of sleep for which we get the maximum number of iterations per second reached and the average power consumption during the execution. We have in total 5 data points for each hardware curve and we approach these points with a linear fitting.

Each graph plots three curves corresponding to our three selected hardware presented in TABLE II. The most powerful is the server from Rennes cluster (Parapluie), and it defines the maximum scale of our graphic. The two left curves are also endless because we reproduce the power consumption scheme we obtain for one experiment as if we can have several servers of each type and cumulate their performance. The least powerful hardware is the Chromebook plotted in green, but because of its very low consumption it can be repeated several times and still fit in the graph. The maximum performance of one single Chromebook is symbolized by the vertical purple dashed line. On the opposite, when we repeat server Taurus from Lyon, it shortly becomes out of scale because its static idle consumption is too important.

Figure 4 corresponds to the case depicted in Figure 3(b) where the executed program is compiled for ARM architecture. The program is executed natively on Chromebook, green curve, and through dynamic translation on Taurus and Parapluie, red and black curves. On the opposite, Figure 5 represents what happens in the case of Figure 3(a) where the target architecture is x86 and the emulation only concerns Chromebook. When we compare the two graphs, and especially when we observe the maximum number of iterations per second, we can find the overhead of emulation introduced in section V-A. The overall total performance is reduced by 7.45 times when we use an ARM binary.

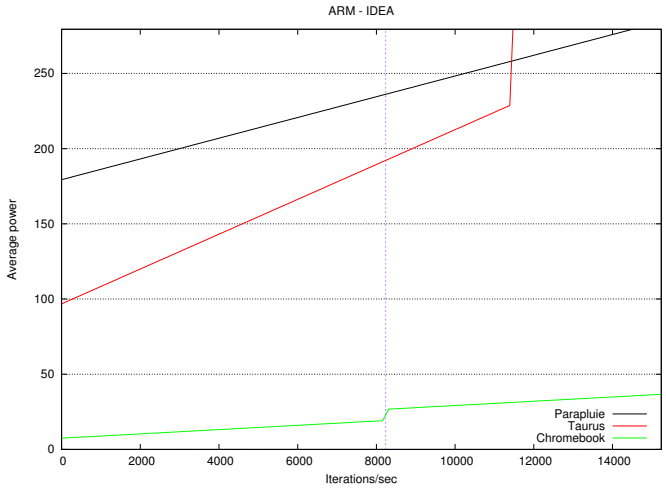


Fig. 4. Average power consumption (watts) according to number of iterations per second of the same ARM program (IDEA benchmark) on 3 different hardware devices

For the ARM program on Fig. 4, we see that x86 architectures perform quite poorly, and if the execution of the program could be done on two ARM platforms, then it would always be the most relevant configuration concerning energy consumption. If we cannot consider parallel design of the program, then Taurus would be the chosen platform from approximately 8000 to 110000 iterations per second, and Parapluie would be elected passed this threshold of performance. This create a result not far from energy proportionality, and this aspect is discussed in next section V-C.

On the other hand, for x86 program on Fig.5, the performance of ARM platform is very low because it is reduced due to dynamic translation. Consequently, what we can observe on the zoom area is that the Chromebook would be chosen until about 900 iterations per second if no parallelization, and until approximately 3600 iterations per second, which

represents 4 Chromebook nodes, if possible. Considering the last perspective, the energy consumption is thus reduced for the first 1/30th of the total performance, and the global shape is still far from proportionality.

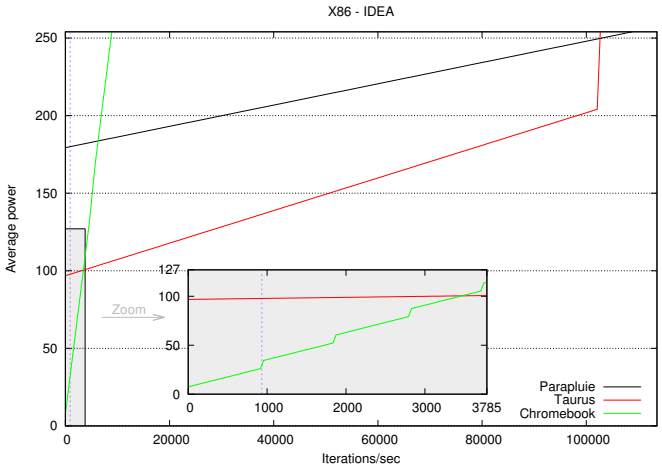


Fig. 5. Average power consumption (watts) according to number of iterations per second of the same x86 program (IDEA benchmark) on 3 different hardware devices

C. Towards energy proportionality

In this section we want to picture how far from proportionality our solution could be. According to the results from last section, we have selected the least consuming hardware for each level of performance. On Fig.6, we plot only these selected parts of curves, as well as the ideal proportional consumption we aim at, represented as a blue line. The ideal curve is indeed a strict proportional line starting from 0 and reaching the maximum point. This maximum refers to the average power consumption needed to reach the maximum number of iterations of the most powerful hardware of our platform. We have only considered a 1-1 relation by selecting only one hardware at a time and not consider a multiple number of Chromebook nodes as explain in previous section.

We can see that the ARM hardware leads to huge energy savings, in fact the green curve is way under the ideal, except for the very beginning because its idle power consumption is not equal to zero. Moreover, having these two different x86 servers is also a good leverage and allows to better stick to the ideal proportionality line. This confirms the assumption we made when selecting two different kinds of x86 hardware, and we can interpolate and imagine that even more recent servers would add more proportionality.

Nevertheless, even if ARM solution seems not far from proportionality, it is not optimal at all because it wastes the potential performance of x86 servers. We have to look at the proportionality comparison of x86 solution in Figure 7. In this case, the gains from ARM hardware are only profitable for a reduced part of low performance, that we can only see on the zoomed part of the graph. The most predominant hardware is Taurus, we realize that Parapluie only brings a small improvement in performance but consumes a lot more than Taurus most of the time. This can be justified by the fact that the Dell PowerEdge R720 is the most recent server of

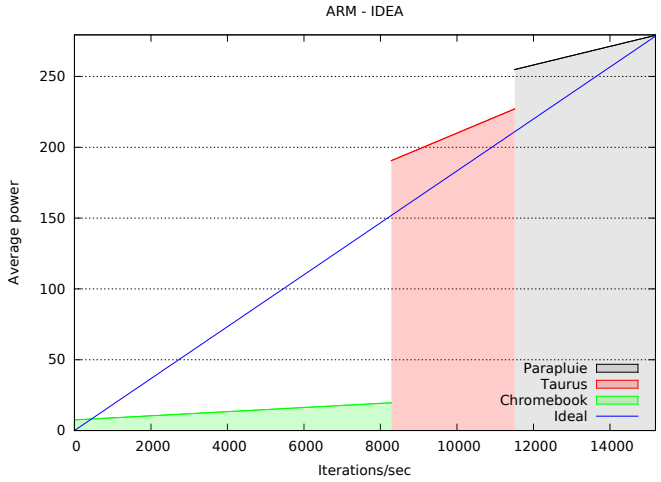


Fig. 6. ARM solution compared to ideal proportionality (IDEA benchmark)

our selection, and the energy efficiency aspect must have been better considered during its design.

We conclude on these results that it is necessary to find other pieces of hardware that would fit between Chromebook and Taurus, and would help to approach the ideal proportional consumption. Indeed the ARM Cortex-A15 is a great low power processor that brings promising energy savings, but the performance gap between itself and x86 servers is too large.

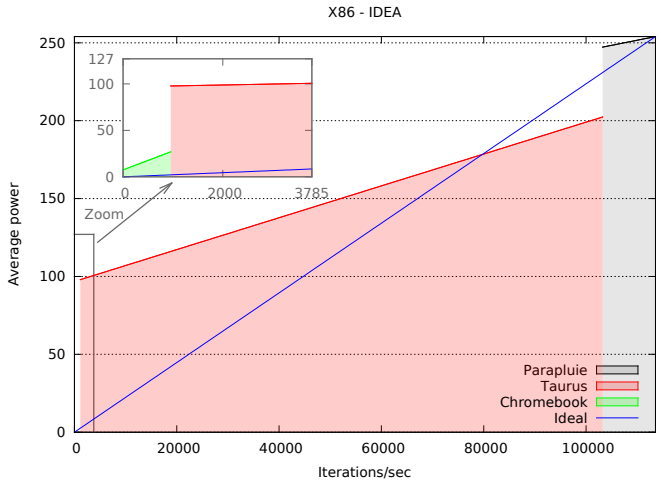


Fig. 7. x86 solution compared to ideal proportionality (IDEA benchmark)

D. ARM virtual machine and Database benchmark

Measuring the impact of emulation on cloud applications requires virtual machines. We could only do this part with an ARM virtual machine because x86 emulation in QEMU from ARM devices is not fully effective for the moment. We have built a Debian Wheezy virtual machine of ARM architecture that emulates the board Versatile Express A15 containing one virtual ARM Cortex-A15 processor, with a 4 Go virtual disk. As it has one virtual CPU, it only executes itself on one core of each host machine.

For the purpose of the TPCC-UVA benchmark the VM includes a PostgreSQL 9.3.4 database. Tables of the database represent the activities of a wholesale supplier including transactions for entering, delivering orders, recording payments, monitoring the stock level in the warehouses. The main input data of the benchmark is the number of warehouses, and the other tables are populated accordingly. The principal output result is a number called 'tpmC-uva' which is the number of performed transactions per minute. Other outputs are also given such as the percentage of well-done transactions, response time and think time, for each type of transactions. The number tmpC only concerns transactions of type 'New Order'.

We have performed the same execution of the benchmark over our three selected machines. We have set the following inputs : database is populated with 1 warehouse, ramp-up period is set to 20 minutes and measurement period is 2 hours. Figure 8 shows the evolution of the throughput over time of the benchmark execution. The throughput is represented by the number of transactions per minute. We can notice the ramp-up period which is the first period until time t = 1200 seconds. After this period is the measurement period when data is collected to compute benchmarks outputs.

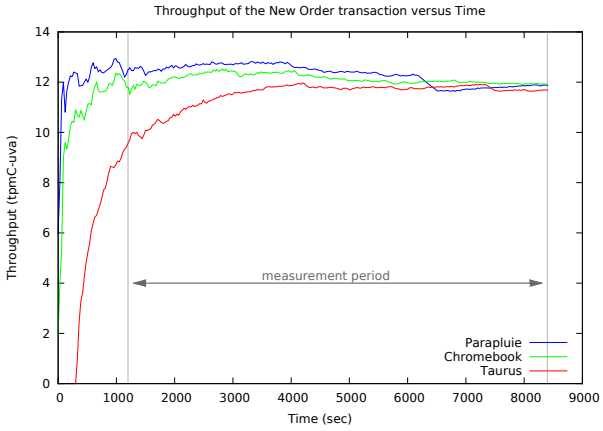


Fig. 8. Comparison of TPCC throughput over time for each machine

On the Chromebook the virtual machine is executed via virtualization extensions whereas Taurus and Parapluie use emulation software. We conclude that there is not a big performance difference between the three machines for this scenario. For all machines the throughput stabilizes around 11 to 12 transactions per minute for this execution. Parapluie server plotted in blue is more efficient than the Chromebook for most of the measurement time except for the last thirty minutes. On the opposite, Taurus server is most of the time the least powerful.

The following two graphs present the distribution of the response time of two types of transactions. Fig.9 concerns New Order transactions and Fig.10 Payments transactions. These results confirm the ones shown by Fig.8. For New Order transactions on first graph, Parapluie offers a very short response time, very close to zero, for few transactions and then for most of them the response time is around 0.2 seconds. This pattern is reproduced also for Payment transactions for both Parapluie and Taurus, while Chromebook's response time is more compacted around a single value. We can justify this result by the sometimes unstable behavior of the emulation software. We also find the same pattern we present in the section V-A about emulation overhead. Indeed here Parapluie seems to benefit from the best emulation capabilities. But to confirm this, these results should be compared with an execution of the same benchmark inside an x86 virtual machine.

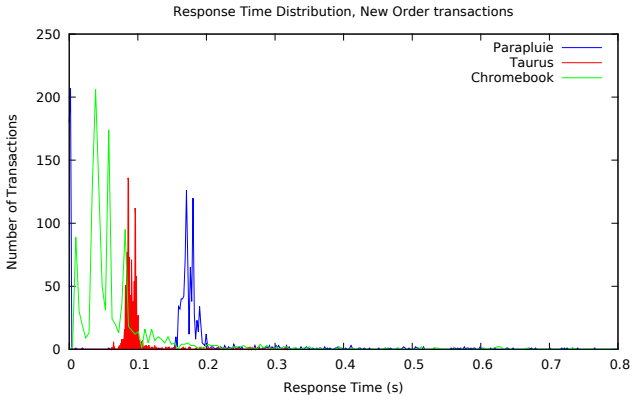


Fig. 9. Response time distribution for New Order transactions

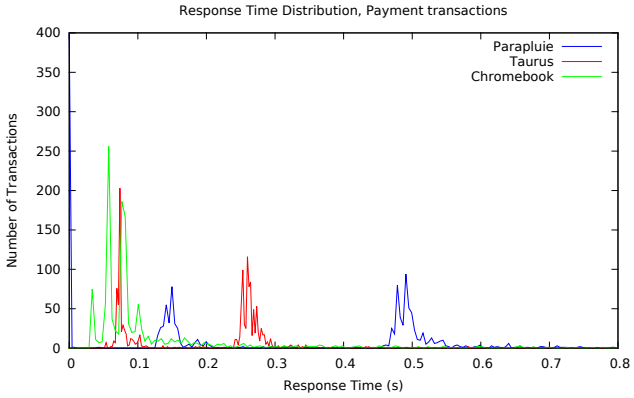


Fig. 10. Response time distribution for Payment transactions

As energy consumption is concerned, we find again the same order : Parapluie is the most consuming machine with an average power consumption of 173 watts during the execution time of the benchmark. Taurus's average power is around 128 watts and between 9 and 11 watts for the Chromebook. These values are less high than in section V-B because here only one core of each machine is executing the virtual machine and not all the processors are fully loaded.

Based on the results of this scenario, we can say that the Chromebook is the best option if we consider both energy consumption and performance. We have run the same experiment but with a database populated with 2 warehouses, and this larger scenario shows a difference in performance between the two architectures. Indeed the Chromebook cannot handle more than 12 transactions per minute, whereas Parapluie and Taurus are able to treat all the requests with success, reaching around 24 transactions per minute. This scenario revealed the weakness of the ARM processor, and the interest in switching to more powerful processors when needed.

E. Live migration impacts

We have performed some experiments of live migration with the ARM based virtual machine used in last section V-D. For this purpose we used Libvirt version 1.2.9 as VM manager. Hardware used is an HP 7800 server with an Intel Xeon E5620 CPU, and the previously described Samsung Chromebook. They are both monitored with external watt-meters Watts'upPro and power data is acquired and stored via Kwapi API [14]. At the current status of our first experiments, only migration from the server to the Chromebook works. Figure 11 presents the extra power consumption of each host during the process of virtual machine migration. In fact in order to focus only on the overhead consumption implied by the migration, we have removed the static idle consumption.

The live migration duration is 8 seconds for this example, which corresponds to a data transfer of 53 Megabytes. The two physical machines are linked with a 1GB switch and cables, but as the Chromebook does not have an Ethernet port, we use an Ethernet to USB 2.0 adapter which may reduce the network throughput. Concerning power consumption, we notice a significant overhead for the source host, about 9 watts when starting the migration. On the destination host there is an increase in power consumption when receiving the virtual machine but then the power stabilizes shortly.

These are some first steps in our work about heterogeneous migrations between Big and Little. We will continue our investigations about all the parameters which can affect the migration behavior and see how they can be enhanced.

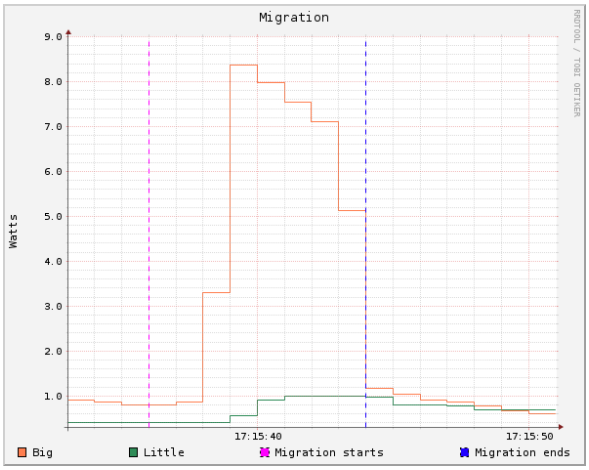


Fig. 11. Extra power consumption during live migration of ARM virtual machine from Big (HP server) to Little (Chromebook)

VI. Conclusion and Perspectives

The work presented in this paper is in early research stage, and is a proposition and a study of solutions to tackle the challenge of energy proportionality. To the best of our knowledge, this is the first approach dealing with "big.little" concept in order to reach some energy proportionality for HPC and Cloud infrastructures. In this paper, we have shown the potential interest of generalizing the "big.little" concept for the purpose of a perfect energy proportionality.

Some issues, not yet in the scope of our work, should be considered as future perspectives. For our proposition of emulation to be effective we need an operational x86 virtual machine emulation from ARM hosts, as well as an operational live migration between heterogeneous architectures. Regarding this issue, for the moment the cost induced by migrations is not taken into account. This cost may vary respecting the size of the virtual machine and even the type of applications. In a similar idea, taking decisions of migrations is very important and represent another challenge in itself. Decisions are even more crucial as the cost of the migration is high. Performing migrations in the most suitable way to reach energy proportionality require a good knowledge of the running applications, or a good system to profile and predict future application needs. Further studies on targeted applications seem to be an inevitable step to propose and build solutions for proportional computing.

Acknowledgments

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://grid5000.fr). This research is also partially supported by the ANR MOEBUS project.

References

[1] W.V. Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester. Trends in Worldwide ICT Electricity Consumption from 2007 to 2012. *Computer Communications*, 50, 2014.

[2] L.A. Barroso and U. Holzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 2007.

[3] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In *Conference on Symposium on Networked Systems Design & Implementation*, 2005.

[4] F. Hermenier, X. Lorca, J-M. Menaud, G. Muller, and J. Lawall. Entropy: A Consolidation Manager for Clusters. In *ACM International Conference on Virtual Execution Environments*, 2009.

[5] E. Feller, L. Rilling, and C. Morin. Energy-Aware Ant Colony Based Workload Placement in Clouds. In *IEEE/ACM International Conference on Grid Computing*, 2011.

[6] G.L. Tsafack Chetsa, L. Lefèvre, J-M. Pierson, P. Stolf, and G. Da Costa. Exploiting Performance Counters to Predict and Improve Energy Performance of HPC Systems. *Future Generation Computer Systems*, 2014.

[7] G. Varsamopoulos, Z. Abbasi, and S.K.S. Gupta. Trends and Effects of Energy Proportionality on Server Provisioning in Data Centers. In *International Conference on High Performance Computing*, 2010.

[8] F. Ryckbosch, S. Polfliet, and L. Eeckhout. Trends in server energy proportionality. *Computer*, 2011.

[9] ARM big.LITTLE Technology: The Future of Mobile. *ARM White Paper*, 2013.

[10] NVIDIA Tegra K1 : A New Era in Mobile Computing. *NVIDIA White Paper*, 2014.

[11] G. Da Costa. Heterogeneity: The Key to Achieve Power-Proportional Computing. *IEEE International Symposium on Cluster Computing and the Grid*, 2013.

[12] U.F. Mayer. Linux/Unix Nbench. http://www.tux.org/~mayer/linux/bmark.html.

[13] D.R. Llanos. TPCC-UVA: An Open-Source TPC-C Implementation for Global Performance Measurement of Computer Systems. *ACM SIGMOD Record*, 2006.

[14] F. Rossigneux, L. Lefevre, J.-P. Gelas, and M. Dias de Assuncao. A Generic and Extensible Framework for Monitoring Energy Consumption of OpenStack Clouds. In *The 4th IEEE International Conference on Sustainable Computing and Communications*, December 2014.