2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing

Profiling energy consumption of VMs for green cloud computing

Qingwen Chen, Paola Grosso, Karel van der Veldt, Cees de Laat
System and Network Engineering research group
University of Amsterdam
Amsterdam, The Netherlands
Email: qingwen@science.uva.nl, p.grosso, karel.vd.veldt, delaat (@uva.nl)

Rutger Hofman, Henri Bal
High Performance Distributed Computing group
VU University
Amsterdam, The Netherlands
Email: rutger, bal (@vu.nl)

***Abstract*—The GreenClouds project in the Netherlands investigates a system-level approach towards greening High-Performance Computing (HPC) infrastructures and clouds. In this paper we present our initial results in profiling virtual machines with respect to three power metrics, *i.e.* power, power efficiency and energy, under different high performance computing workloads. We built a linear power model that represents the behavior of a single work node and includes the contribution from individual components, *i.e.* CPU, memory and HDD, to the total power consumption of a single work node. Our results could be part of a power characterization module integrated into clusters' monitoring systems; future GreenClouds energy-savvy scheduler would use this monitoring system to support system-level optimization.**

***Keywords*-Green Clouds, Energy-efficient computing, Power benchmark, Virtualization, Cloud Computing, Kernel-based Virtual Machine (KVM)**

I. Introduction

Cloud computing is beginning to be seen as an alternative to owned HPC clusters. There are two main appealing incentives: firstly, the clouds' utility-based usage model allows users to pay per use, in a manner similar to other public utilities services such as electricity; secondly, there is only the relatively low investment needed for the end devices that access the cloud resources.

Besides economic incentives, the cloud model provides also benefits from the environmental perspective, since the computing resources are managed by cloud service providers but shared among all users, which increases their overall utilization [1]. The US EPS's report on server and data center energy efficiency [2] predicted that about 10% of annual savings in energy consumption could be achieved by 2011 through state-of-the-art hardware and virtualization technologies.

We agree with the above conclusion, but we also are convinced that even bigger saving can be booked in cloud systems devoted to HPC if they are augmented with global system-level optimizations. Such optimizations focus on providing energy-efficient optimizations for the entire data center while taking each individual's power behaviors as input parameters. Essentially they are not limited to local optimizations (*e.g.,* reducing clock speeds), but look at the overall cloud system's behavior. We believe this approach should also allow to better balance the required performance of demanding HPC applications with the needs for energy consciousness and energy efficiency.

The challenges of such system-level approach are to identify:

- the power behavior of each individual, *i.e.* virtual machine (VM) in virtualized environment, and how would it be affected by different patterns of application running on it and/or different hardware architectures hosting it; this is the focus of our contribution in this article;
- how power characteristic information be incorporated into the system-level optimization.

The GreenClouds project [3], where we perform our research, leverages three promising ideas as basic components for a system-level approach in a distributed setting:

- hardware diversity: computations should run on those architectures (*e.g.* GPUs, multicores) that perform them in the most energy-efficient way;
- elastic scalability: the number of resources should dynamically adapt to the application needs, taking both computational and energy efficiency into account;
- hybrid networks: optical and photonic networks can transport data and computations in a more energy-efficient way.

This article presents comprehensive power benchmarks that we performed on the state-of-the-art infrastructure provided by DAS-4 clusters [4]. We also present heuristic recommendations on how these results can be included in the GreenClouds energy-conscious elastic scheduler.

II. Related Work

Several simple techniques provide basic power management for servers operating in a cloud, *i.e.* turning on and off servers, putting them to sleep or using Dynamic Voltage/Frequency Scaling (DVFS) to adjust servers' power states. These three methods cannot solve the power consumption optimization problem in the presence of virtual machines, unless one combines them with forced migration of VMs to concentrate them in fewer servers. This migration can clearly be undesirable to guarantee applications' performance.

978-0-7695-4612-4/11 $26.00 © 2011 IEEE
DOI 10.1109/DASC.2011.131

One therefore needs to look at the behavior of the individual VMs, and consequently at the availability of correct power models and energy profiles. These can be obtained by actively using power benchmarks or by closely monitoring the energy profile of individual system components such as CPU, cache, disk, and memory at run time. We have been looking at previous work to determine if and which power model we could use in GreenClouds.

Stoess et al. [5] were among the first to present methods to determine the energy usage of VMs. They relied on the availability of models for each hardware component to create a framework for power optimization and the development of energy-aware OS.

Kansal et al. [6] proposed *Joulemeter*, a power meter for virtual machines. Also *Joulemeter* makes use of power models of individual hardware resources; at runtime software components monitor the resource usage of VMs and they convert it to energy usage using the available model.

The power modeling technique *vMeter*, proposed by Bohra et al. [7] is most relevant for us. They observed a correlation between the total system's power consumption and component utilization. They created a four-dimensional linear weighted power model for the total power consumed $P(total)$:

$$P_{total} = c_0 + c_1\mathcal{P}_{CPU} + c_2\mathcal{P}_{cache} + c_3\mathcal{P}_{DRAM} + c_4\mathcal{P}_{disk} \quad (1)$$

where $p_{CPU}, p_{cache}, p_{DRAM}$ and p_{disk} are specific performance parameters for CPU, cache, DRAM and disk, and c_0, c_1, c_2, c_3, c_4 are weights. The weights are calculated per workflow. They refined the power model by separating the contribution of each active domain in a node, either a VM or *dom0*:

$$P_{total} = P_{baseline} + \sum_{k=1}^{N} P_{domain(k)} \quad (2)$$

where $P_{domain(k)}$ is the power consumed by an active domain k, and N is the number of active domains (including dom0).

The work done by Liu et al. [8] for the GreenCloud architecture and by Dhiman et al. [9] in *vGreen* are also relevant for us. The GreenCloud architecture utilizes live migration of VMs based on power information of the physical nodes. With this technique Liu and his colleagues show a significant energy reduction for applications running in clouds, specifically for online gaming. They define an integrated approach similar to the one we are setting out ourselves to follow. *vGreen* also consists of a multi-tiered software system, where policies are used to schedule across the available machines.

Our efforts differ from the above as we aim to explore the benefits of using very heterogenous hardware in creating, managing, and (when needed) migrating application-tailored VMs.

III. Experimental setup

GreenClouds will experiment with the Distributed ASCI Supercomputer 4 (DAS-4 [4]), a six-cluster wide-area distributed system that provides a common computational infrastructure to Dutch researchers. Initially we will use the two clusters in Amsterdam: the 74 node cluster at the VU University (VU) and the 16 node cluster University of Amsterdam (UvA). All cluster nodes have dual-quad-core CPU (Intel E5620),24 GB memory, 30TB HDDs and roughly 1 TB of storage. The only difference between the two clusters is that the VU nodes are equipped with 2 Nvidia Tesla C2050 accelerators.

Our power measurement setup is built adapting onto DAS-4 the approach of the Green500 list [10], and described in [11]. In this approach there are three basic entities: a System Under Test (SUT), a power meter and a data logger.

SUT: Our SUT is one of the DAS-4 cluster node. Table I lists part of Intel E5620's specifications. The operating system running on the SUT is a fresh CentOS 5.6 (Kernel version: 2.6.18-238.9.1.el5) with Dynamic Voltage-Frequency Scaling (DVFS) enabled.

Table I: Specifications of Intel E5620

Basic specifications		Advanced features
# of Cores	4	Intel Turbo Boost Technology
# of Threads	8	Intel Hyper-Threading Technology
Clock Speed	2.4GHz	Intel Virtualization Technology (VT-x)
Max Turbo Frequency	2.66GHz	Idle State
Max TDP	80W	Enhanced Intel SpeedStep Technology

Our virtualization is done with Kernel-based Virtual Machine (KVM) [12] [13] which is a full virtualization for Linux on supported x86 hardware. A guest VM running with KVM, in principal, lives as a regular linux process on its host.

In our experiments all of the VMs are configured with the same amount of virtual memory but with different numbers of virtual CPUs (vCPUs) and this corresponds to the number of physical cores assigned to the VM. For each VM we allocate 20GB memory out of 24GB available physical memory in order to get the best performance from Linpack. The number of vCPUs of a VM varies from 1, 2, 4, 8, to 16, where the last case is an overcommitted VM since it has more vCPUs than available physical cores.

Power meter: Our power meter is a 32A PDU gateway from Schleifenbauer. It can provide power data in public APIs [14] in PHP, Perl, and SNMP with the precision of 1 V in voltage and 0.01 A in current. The instant power consumption is calculated by multiplying the voltage, the current and the power factor together.

Data logger: We chose Ganglia [15] as our monitoring tool to collect and analyze the power data. Though Ganglia has plenty of built-in monitoring metrics, the power metric is not included due to its dependency on the APIs provided by the manufacturer of the power meter. In our experiments, we integrated the power metric into Ganglia with the Perl APIs provided by Schleifenbauer. For stress tests which normally will run for more than 10 minutes and where power consumption of the work node will not vary much, we collect the power data every 5 seconds in order to avoid adding too much overhead on the work node; for instant workload tests where both the resource usage and power consumption of the work node vary dramatically with time, power data is collected every second to get accurate and reliable data.

IV. Metrics and workload generators

A. Power metrics

We utilize five metrics within our power benchmarks, classified into two categories: measurement metrics and derived metrics. Measurement metrics are directly read from the hardware monitoring tools or software applications without advanced calculations; the derived metrics are derived from measurement metrics. Table II describes each metric in detail.

Table II: Definition of benchmark metrics

Metric	Definition	Value
Performance	Evaluation of how well the SUT performs benchmarks	Value as reported by benchmark tools themselves
Power consumption	Average power consumed by the SUT during benchmarks	Average of values reported by the power meter
Execution time	Time duration of benchmarks	Wall clock time reported by the data logger
Power-efficiency	Evaluation of how efficient the power is used	Performance divided by power
Energy consumption	Cumulative power consumption over execution time	Power consumption multiplied by execution time

Resgard to the two derived metrics, power efficiency and energy consumption, we believe the energy consumption metric to be the appropriate metric for applications with definite execution time; while it is better to use the power-efficiency metric for applications with unlimited execution time (e.g. hosting a web server).

B. Workload generator

Our workload generators are carefully selected to profile the power consumption of the SUT in generic and extreme use cases. The generic use case corresponds to applications that keep running but do not fully utilize the resource, *i.e.* a database server or a webserver. The extreme case can be an application that stresses resource usage to its limits.

We broadly identified CPU, Memory, Hard disk drive (HDD), and GPUs (if present) as the major components which consume the most power in high-performance computing environments. We did not yet perform any measurements on GPUs; currently, our work focuses on CPU and Memory energy profiles. Our goal is to profile separately the impact of each component with respect to the total power consumption. We choose a few major workload generators as benchmarks: *Stress* [16], the Intel optimized *Linpack* [17] benchmark and the Dhrystones and Fhourstones benchmarks. We also wrote our own scripts to perform divisions on random numbers to mimic generic non-stressful workloads. The time interval between operations is automatically adjusted so that the workload, *i.e.* CPU usage, will be increased gradually and then decreased in a similar way after it reaches 100%.

V. Component tests results

figure We performed two types of tests: component tests and overall tests summarized in Table III and Table IV respectively. Each benchmark runs on the SUT itself and on virtual machines with different number of vCPUs as we had explained in Section III. A *guest VM* is the virtual machine configured with the same number of vCPUs as the number of available physical cores, *i.e.* 8 cores. We chose this configuration for the *guest VM* to investigate the impact of virtualization on energy consumption and performance.

Table III: Summary of component benchmarks

Component	Test type	Workload	Metric
CPU	CPU usage	Script	Power consumption
	Freq. scaling	Linpack	Performance, power efficiency, energy consumption
	Different VMs	Linpack	Performance, power efficiency, energy consumption
Memory	# of workers	Stress	Power consumption
	Memory usage	Stress	Power consumption
HDD	Timeline	Stress	CPU usage, Power consumption

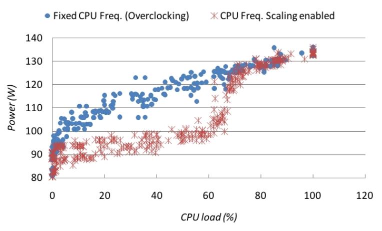
Table IV: Summary of overall benchmarks

Test type	Workload	Metric
Floating-point operation	Linpack	Performance, power efficiency, energy consumption
Integer operation	Dhrystone Fhourstones	Performance, power efficiency, energy consumption
Hyper-Threading	Linpack	Performance, power efficiency, energy consumption

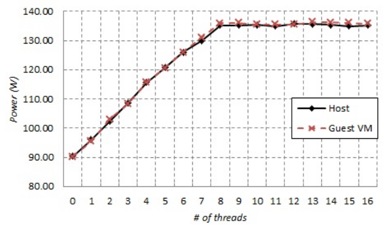
A. CPU

The CPU usage test measures the power consumption by gradually increasing:

- *Case I*: the workload on all available cores;

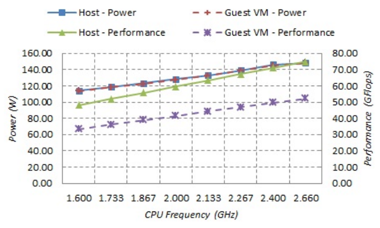


(a)

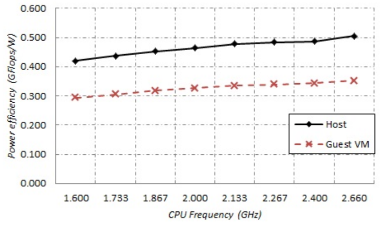


(b)

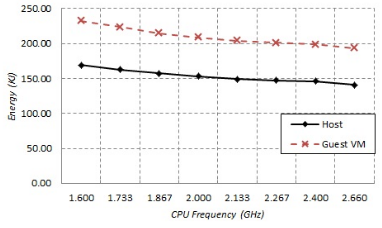
Figure 1: Power consumption versus CPU usage: gradual increase of the CPU loads on all available cores (a) and gradual increase of number of cores, where each core is at its maximum usage (b).



(a)



(b)



(c)

Figure 2: CPU frequecy scaling: Power and Performance (a), Power efficiency (b) and Total energy consumption (c).

- *Case II*: the number of cores being used and by stretching each used core to its maximum usage when it starts up.

Dynamic CPU frequency scaling in our SUT is handled by SpeedStep from Intel, and the Linux kernel will switch it to the highest frequency immediately when the load reaches the threshold. With frequency scaling enabled, we observed a clear turning point for Case I when CPU usage reached $\sim$65% in Figure 1a.

For the case with fixed CPU frequency we chose 'turbo mode' for the CPU speed where CPU frequency is fixed to be the maximum value, *i.e.* 2.66GHz for Intel E5620 in our case. Turbo mode used the TurboBoost technology from Intel [18]; TurboBoost enables the processor to run above its base operating frequency via dynamic control of the CPU's "clock rate". In this case the total power consumption is nearly linear with the CPU load except a sharp increase when the CPU load goes from idle to $\sim$5%, as shown in Fig. 1a.

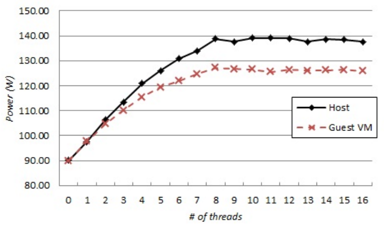
Fig. 1b shows the results for case II. The power consumption grows also linearly with the number of threads, up to the point when we start to have more than 8 threads because both the number of physical cores and vCPUs of the Guest VM are 8. We observe a negligible difference in power consumption between the host and the Guest VM since both of them stretch the usage of each used core to its limit.

B. *CPU frequency scaling test*

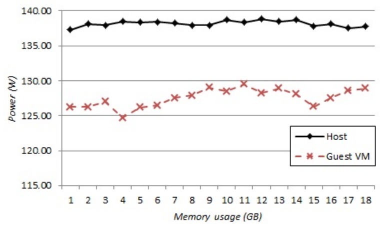
In this series of tests we disabled dynamic CPU frequency scaling and varied the CPU frequency among several available frequencies within our SUT, as shown in Fig. 2. The maximum available frequency supported by our SUT, namely 2.66GHz, corresponds to turbo mode.

Fig. 2a presents power consumption and performance of our experiments comparing the SUT with the *guest VM*. We observe that the guest VM has nearly the same power consumption as the host but with worse performance. Since both of them have a memory usage of $\sim$16GB and fully utilize the CPU, the performance degradation in the *Guest VM* comes from the virtualization. Other research [19] [20] has shown similar pattern in the case of Linpack performance on KVM VMs: the processing efficiency of KVM on floating-point operations is slow as KVM checks whether every executing instruction is an interrupt, a page fault, I/O or a common instruction in order to decide if to exit from the guest mode or stay in it.

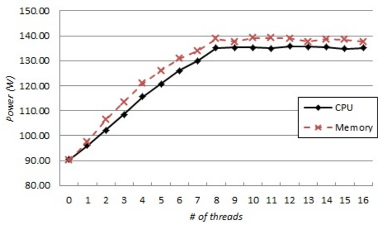
Another important result is on the total energy consumed for each experiment. As shown in Fig. 2a and 2b, performance and power efficiency improve almost linearly as the frequency scales up, but the total energy consumption decreases (see Fig. 2c). Notice that by setting the CPU to turbo mode, the SUT consumes more power than in all other cases; however, it takes less time to complete the Linpack benchmark which finally results in less energy used. Therefore we come to the general conclusion that it's



(a)



(b)



(c)

Figure 3: Memory stress tests: varying numbers of workers (a) and different memeory usage (b). Comparison of the CPU and Memory Stress tests on the host. In both cases the CPU utilization is 100% (c)

'greener' to set the CPU to turbo mode for CPU-bound workloads.

We also measured the idle power consumption of the SUT for different CPU frequencies. Through our experiments we have seen that the idle power consumption remains constant, regardless of the CPU frequency. The value for our SUT is ~90W across the whole frequency range.

C. Memory

As shown in Table III, we performed two types of tests to quantify the impact of the memory on the energy profile of the SUT and the guest VM: worker tests and memory usage tests.

Fig. 3a shows that the increase in power consumption levels off when the number of threads equals the number of vCPUs on the VM. Fig. 3b shows that less power is consumed by the VM compared to the host for the same test, and the total power consumption remains nearly constant, regardless of memory usage. Fig. 3b also indicates that the variation of total power consumption with respect to memory usage is less than ~5W.

A recent DARPA commissioned study on the challenges for ExaFLOP computing reports in [21] that the power needed for memory systems remains constant in regardless of the workloads, but that power is proportional to the number of memory chips. We also tried to separate the CPU's and memory's contribution to the total power consumed. We compared the measurements reported by the CPU usage test and the memory usage test for the host, in order to estimate the power consumed by memory. In Fig. 3c we see a separation of no more than 5W in the whole range of number of threads. Therefore, we consider the variation of memory contribution as a constant and we incorporate it with the idle power consumption in our model.

VI. Overall test results

A. Floating-point operation benchmark

In our experiments VMs are configured to have different number of vCPUs, as we explained in Section III. We have profiled the performance, the power consumed, the power efficiency and the total energy of different VMs by varying the number of threads used in Linpack benchmarks. Therefore within this series of tests, there are two variables:

- the number of threads running Linpack benchmark
- the number of vCPUs of a VM.

The actual number of physical cores involved in the benchmark is decided by their minimal value and bounded by the number of available physical cores (*i.e.* 8 in our case).

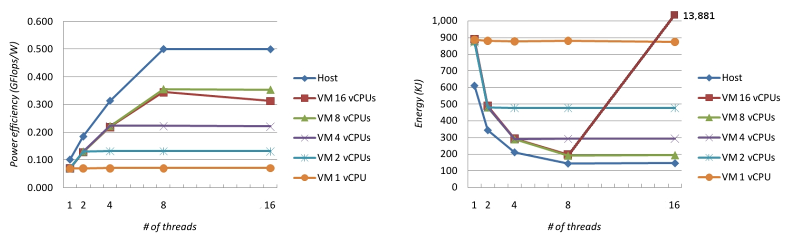
Fig. 4 shows our results. We see that all measured parameters increase until they reach a plateau when the number of threads is the same as the number of vCPUs for all non-overcommitted cases. Our outputs show that virtualization results in overall performance degradation with respect to to the Linkpack benchmark. Still this degradation is highly situation-dependent, and that there should be no impact for pure user code.

Another interesting result in our experiments comes from the overcommitted VM with 16 vCPUs. When 16 threads are used to run the Linpack benchmark, it performs less satisfactorily than the same case on the VM with 8 vCPUs. Even though it consumes less power, its execution time is around 13 times longer, which further leads to much more energy consumed in the test, as shown in Fig. 4b. This is in line with what is known about the performance degradation of overcommitting symmetric multiprocessing guests with KVM due to dropped requests and unusable response times when overcommitting vCPUs as loads under 100% [22].

B. Integer operation benchmark

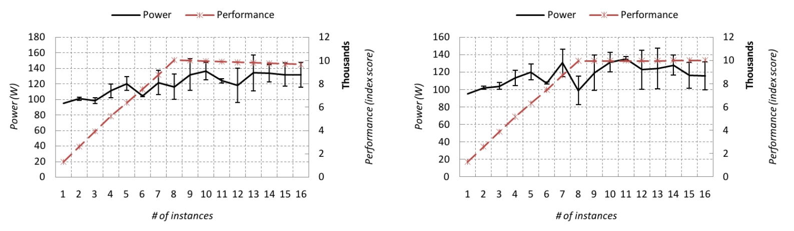
In this section we will examine the integer operation performance of our SUT as an complementary to its floating-point operation performance evaluated in the previous section.

1) Dhrystone: Specifically, the Dhrystone v2.1 from UnixBench benchmark suite [23] is used in this series of benchmarks. Even though Dhrystone v2.1 is originally designed as a single-threaded application, we varied the number of instances of Dhrystone which can run concurrently on either the host or the Guest VM.



(a) (b)

Figure 4: Linpack tests on different VMs: Power efficiency (a) and Energy (b)



(a) (b)

Figure 5: Dhrystone benchmark on the host and the Guest VM with different number of threads: on the host (a) and on the Guest VM (b)

Figure 5 presents the power consumption and performance of Dhrystone running on the host and the Guest VM respectively. As shown in Figure 5a, their power usage is unstable, which makes it less meaningful to continue calculating the power efficiency in a similar manner.

2) *Fhourstones:* The Fhourstones benchmark is a single-threaded application with small code size. Thus in this section we performs the benchmark only on the host and Guest VM. The benchmark results are presented in Table V.

Table V: Performance of the Fhourstones benchmark on the host and guest VM

	Performance (KPOS/s)	Execution time (s)	Power (W)	Power efficiency (KPOS/s/W)	Energy (KJ)
Host	8013	209.5	95.83	83.62	20.08
Guest VM	7480	224	96.42	77.58	21.60

Even though we still observed ∼7% of performance degradation in Guest VM, it is much less than the one in floating-point operation (i.e. Linpack) benchmarks. It is also ∼ 7% less energy-efficient with virtualization according to the statistics in Table V.

VII. Impact of Hyper-Threading

By enabling multiple threads to run on each core simultaneously, Hyper-Threading (HT) technology improves the overall performance of the CPU and uses it more efficiently, especially for threaded applications. Within the previous benchmarks, HT is disabled by default on our SUT. In this section, we will enable HT technology and explore its impact on the overall performance of the SUT.

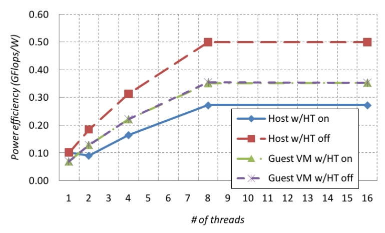
We examined its impact for both non- and overcommitted VMs and focused on their floating-point operation performance. The same VMs are used in this series of power benchmarks in order to make results comparable with our previous discoveries.

A. Non-overcommitted case

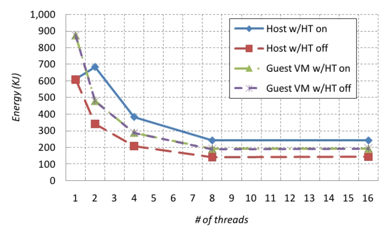
In this experiment the guest VM is used to perform the Linpack benchmark while HT is enabled on the host. Figure 6 presents the results of Linpack benchmark running on the guest VM and host.

With HT enabled on the host, there is a significant difference in performance of how the host performs Linpack benchmark, as shown in Figure 6a. It is suggested by Intel in [24] that HT is better disabled for compute-efficient applications, because there is little to be gained from HT technology if the processor's execution resources are already well used. What is even worse is that spawning a second process on the same physical core will force the physical resources such as cache to be shared. If that happens, more cache misses may occur and further degrade the performance. This issue has also been discussed in [25] reaching the same conclusion.

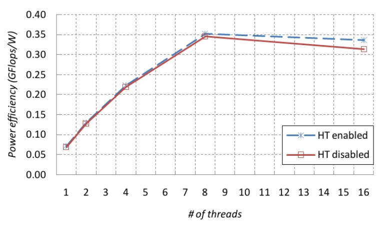
However the guest VM is, surprisingly, not affected by the HT technology according to the results presented in Figure 6 where the two data series of the guest VM almost overlap



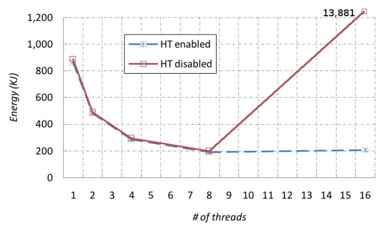
(a)



(b)



(c)



(d)

Figure 6: Impact of Hyper-Threading for Linpack tests on VM (with 16 vCPUs). Non-overcommitted case: efficiency (a) and energy (b); overcommitted case: power efficiency (c) and energy (d)

each other for all 4 metrics.

B. *Overcommitted case*

Figure 6 presents the results of both with and without HT on the host. With HT enabled, the overcommitted VM (with 16 vCPUs) has significant increment in performance, power consumption and power efficiency, compared with the case where HT is disabled. Moreover, HT technology enables much more efficient scheduling on vCPUs, which then results in great improvement in total energy consumption, as shown in Figure 6d.

Moreover, by comparing the cases where number of threads running by Linpack is less than 16, we also observed slight improvement in performance and power efficiency while HT is enabled on the host, even though they consumed almost the same power (see Figure 6c).

VIII. Discussion

Our experiments showed that the idle power consumption of the SUT remained flat with respect to the CPU frequency. However we observed a steep rise when the CPU load went from idle to 5% (see Fig. 1a). It is therefore our primary recommendation to maintain CPU frequency scaling in all systems. The effect of scaling will be significant till the CPU load reaches $\sim 65\%$. Furthermore we consider variation of clock speeds local optimizations which are not of great interest when aiming, as we do, for system-level optimization.

We observed that the total power consumption of the SUT is linear to the CPU load (see Fig. 1). The contribution from the memory to the total power consumption is ~ 5W, which is negligible since it accounts for only $\sim 5.5\%$ of the idle power consumption (90W) or $\sim 3.5\%$ of the maximum power consumption (140W). As for the HDD, we will keep it in the power model since it is difficult to quantify the HDD's impact on total power consumption as other tests (not presented here for lack of space) have shown us. Therefore by integrating the power consumption of memory into c_0 in 1 and calling it $\mathcal{P}_{idle}$, we believe that the power model proposed by Bohra et al (see eq. 1) can be modified to be:

$$P_{total} = \mathcal{P}_{idle} + c_1\mathcal{P}_{CPU} + c_3\mathcal{P}_{HDD} \quad (3)$$

A next step is to add the contribution of hardware accelerators, such as GPU to the above simplified formula. This addition is an essential component given we want to consider heterogenous environments. Fig. 4 showed that the number of virtual CPUs in a VM should be adjusted to the expected number of parallel threads. This should be one of the parameters that a green scheduler takes into account. In particular such a scheduler should avoid to create overcommitted VMs, which have very long execution times. This is due, in our opinion, to the scheduling policy within KVM which doesn't perform very well for over-committed cases, especially on a hyper-threading disabled machine.

IX. Future work: toward energy-aware schedulers

The energy profiles we have built are an initial step toward building energy-aware schedulers. We are now investigating the effects on energy consumption and performance of running multiple VMs with different workloads on the same machine. The power models shown in Eq. 3 are easily extensible into power models for clouds systems, once the contribution of individual and multiple VMs are known. By devising a scheduler's architecture that minimizes the energy profile of the entire system we plan to verify the validity of our system-level approach.

ACKNOWLEDGMENT

The authors would like to thank Vesselin Hadjitodorov for paving the way for the work presented here. We also thank Kees Verstoep for all his help and feedback. This work is sponsored by NWO - the Dutch Organization for Research.

REFERENCES

[1] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, and K. Pentikousis, "Energy-efficient cloud computing," *The Computer Journal*, vol. 53, no. 7, pp. 1045–1051, 2010.

[2] ENERGY STAR program, "Report to congress on server and data center energy efficiency," U.S. Environmental Protection Agency, August 2007, in response to Public Law 109-431. [Online]. Available: http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf

[3] GreenClouds project. [Online]. Available: http://wiki.cs.vu.nl/greenclouds/

[4] DAS-4 website. [Online]. Available: http://www.cs.vu.nl/das4/

[5] J. Stoess, C. Lang, and F. Bellosa, "Energy management for hypervisor-based virtual machines," in *USENIX Annual Technical Conference*, 2007, p. 114.

[6] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *SoCC'10*, 2010, pp. 39–50.

[7] A. Bohra and V. Chaudhary, "VMeter: Power modelling for virtualized clouds," in *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, april 2010, pp. 1 –8.

[8] L. Liu, H. Wang, X. Liu, X. Jin, W. B. He, Q. B. Wang, and Y. Chen, "GreenCloud: a new architecture for green data center," in *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*, ser. ICAC-INDST '09. New York, NY, USA: ACM, 2009, pp. 29–38.

[9] G. Dhiman, G. Marchetti, and T. Rosing, "vGreen: a system for energy efficient computing in virtualized environments," in *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, ser. ISLPED '09. New York, NY, USA: ACM, 2009, pp. 243–248.

[10] S. Sharma, C.-H. Hsu, and W.-C. Feng, "Making a case for a Green500 list." in *IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006)/ Workshop on High Performance - Power Aware Computing*, 2006.

[11] R. Ge and X. Feng and H. Pyla and K. Cameron and W. Feng. (2007, June) Power measurement tutorial for the Green500 list. [Online]. Available: http://www.green500.org/docs/pubs/tutorial.pdf

[12] KVM homepage. [Online]. Available: http://www.linux-kvm.org/

[13] K. Avi, "KVM: The Linux virtual machine monitor," *Proceedings of the Linux Symposium, Ottawa, Ontario, 2007*, 2007.

[14] Public APIs of Schleifenbauer PDU. [Online]. Available: http://sdc.sourceforge.net/index.htm

[15] Ganglia homepage. [Online]. Available: http://ganglia.sourceforge.net/

[16] Homepage of Stress tool. [Online]. Available: http://weather.ou.edu/~apw/projects/stress/

[17] Intel optimized LINPACK benchmark. [Online]. Available: http://software.intel.com/en-us/articles/intel-math-kernel-library-linpack-download/

[18] Intel Turbo Boost Technology 2.0. [Online]. Available: http://www.intel.com/technology/turboboost/

[19] J. Che, Q. He, Q. Gao, and D. Huang, "Performance measuring and comparing of virtual machine monitors," in *Embedded and Ubiquitous Computing, 2008. EUC '08. IEEE/IFIP International Conference on*, vol. 2, dec. 2008, pp. 381 –386.

[20] M. Fenn, M. A. Murphy, and S. Goasguen, "A study of a KVM-based cluster for grid computing," in *Proceedings of the 47th Annual Southeast Regional Conference*, ser. ACM-SE 47. New York, NY, USA: ACM, 2009, pp. 34:1–34:6.

[21] P. Kogge, K. Bergman, S. Borkar, and et al., "ExaScale computing study: Technology challenges in achieving exascale systems," September 2008.

[22] Fedora Documentation Project, *Fedora 13 Virtualization Guide*. Fultus Corporation, 2010, pp. 180–182.

[23] UnixBench project homepage. [Online]. Available: http://code.google.com/p/byte-unixbench/

[24] G. Drysdale, A. C. Valles, and M. Gillespie. Performance insights to Intel® Hyper-Threading technology. [Online]. Available: http://software.intel.com/en-us/articles/performance-insights-to-intel-hyper-threading-technology/

[25] O. Celebioglu, A. Saify, T. Leng, J. Hsieh, V. Mashayekhi, and R. Rooholamini, "The performance impact of computational efficiency on HPC clusters with Hyper-Threading technology," *Parallel and Distributed Processing Symposium, International*, vol. 15, p. 250b, 2004.