2010

The Impact of MIS Software on IT Energy Consumption

Eugenio Capra
Politecnico di Milano, eugenio.capra@polimi.it

Giulia Formenti
Politecnico di Milano, giuliaformenti@gmail.com

Chiara Francalanci
Politecnico di Milano, francala@elet.polimi.it

Stefano Gallazzi
Politecnico di Milano, sgallazzi@gmail.com

THE IMPACT OF MIS SOFTWARE ON IT ENERGY CONSUMPTION

Journal:	*18th European Conference on Information Systems*
Manuscript ID:	ECIS2010-0073.R1
Submission Type:	Research-in-Progress Paper
Keyword:	Green IS, Green IT, Software engineering/development, Software quality

THE IMPACT OF MIS SOFTWARE ON IT ENERGY CONSUMPTION

Capra, Eugenio, Politecnico di Milano, Dipartimento di Elettronica e Informazione, via Ponzio, 34/5, 20133 Milano, Italy, eugenio.capra@polimi.it

Formenti, Giulia, Politecnico di Milano, Dipartimento di Elettronica e Informazione, via Ponzio, 34/5, 20133 Milano, Italy, giuliaformenti@gmail.com

Francalanci, Chiara, Politecnico di Milano, Dipartimento di Elettronica e Informazione, via Ponzio, 34/5, 20133 Milano, Italy, francala@elet.polimi.it

Gallazzi, Stefano, Politecnico di Milano, Dipartimento di Elettronica e Informazione, via Ponzio, 34/5, 20133 Milano, Italy, sgallazzi@gmail.com

Abstract

The energy consumption of IT has a great impact on operational costs, in addition to being important for social responsibility and system scalability issues. Research on IT energy efficiency has always focused on hardware, whereas within the software domain it has mainly focused on embedded systems. In this paper we present the preliminary results of some experiments that we conducted to evaluate MIS applications from an energy efficiency point of view. We analyze in details some selected case studies, including 2 ERPs, 2 CRMs and 4 DBMS. Our evidence suggests i) that not only the infrastructural layers, but also the MIS applications layer does impact on the energy consumption; ii) that different MIS applications satisfying the same functional requirements consume significantly different amounts of energy; and iii) that in some scenarios energy efficiency cannot be increased simply by improving time performance.

Keywords: Green Software; energy efficiency; software quality; software development.

1 INTRODUCTION

Green IT, i.e. the study of the energy consumption of IT, is attracting more and more attention from both the academic and the industrial point of view. It is important for a ethical reasons, cost reasons, and scalability reasons (Murugesan, 2008). First of all, IT infrastructures are responsible for 2% of the CO_2 world emissions and for the greenhouse effect, which is the first reason of the global warming. Second, energy costs have dramatically increased and their impact on the overall IT infrastructural costs is becoming even more significant (e.g., according to Kumar, 2007, nowadays yearly power and cooling costs for servers are almost 60% of the initial purchasing cost). Moreover, energy requirements represent one of the data center scalability issues, since providers often have difficulties in supplying data centers with all the required energy (Lee and Brown, 2007). IT energy consumption sustainability is important from an economic, societal and environmental perspective for organizations. These three dimensions are overlapping factors for sustainability, but very often the economic and societal are ultimately constrained by the environment. Energy efficient software can play an important role in these three overlapping spheres of sustainability.

Research has always focused on *hardware* energy efficiency, and only marginally on software. In particular, energy efficiency has been investigated mainly for embedded systems and low-level software (Sivasubramaniam at al., 2002; Fornaciari et al., 2001), and not at Management Information Systems (MIS) level. Accordingly, hardware energy efficiency has significantly improved in the last years, with particularly high gains in the energy efficiency of mobile devices, as a response to battery autonomy issues. Over the past 30 years, the value of MIPS/W of mainframe systems has increased of a factor of 28.000 (ACEEE, 2008), which represents an improvement much higher than those achieved by production machines in other industrial sectors, such as steel production or automotive. The starting theoretical foundation of our work (*citation omitted in this version of the paper for the sake of anonymity of the authors*) is that software is the main driver of power consumption as it indirectly causes all the commutations performed by the processor and thus induces all the consumption of the above infrastructural layers (e.g., cooling, UPS, etc.) By analogy, in order to reduce car pollution it is important to increase the mileage per liter of gasoline, but also to optimize the trips in order to reduce the overall number of driven miles. Similarly, it is important to reduce the energy required by hardware to perform elementary computations, but also to optimize the number of computations required to satisfy a given set of functional requirements and workloads.

Nevertheless, whereas hardware has been constantly improved to be energy efficient, *software* has not recorded a comparable track. The software development life cycle and related process management methodologies rarely consider this parameter. Not surprisingly, the over 50 ISO software quality parameters do not include energy efficiency (cf. ISO 9126:2003). The prompt availability of increasingly efficient and cheaper hardware components has lead designers up to now to neglect the energy efficiency of end-user software, which remains largely unexplored. In the last decades research has focused on optimizing the energy consumption of operating systems, infrastructural component and embedded systems (see Section 2), for example by striving to develop power-efficient compilers (Daud, Ahmad and Murthy, 2009), but very little research has been made on the energy efficiency of end-user applications, and in particular of Management Information Systems (MIS) software. A paramount difference between embedded systems and MIS is that in MIS contexts hardware and low-level architectures are usually imposed and cannot be easily influenced. For example, from a low-level programming point of view an emerging technique for reducing energy consumption is the dynamic configuration of clock frequency (Huang, Li and Li, 2009), but if we assume an MIS perspective it would be quite difficult for the CIO of a manufacturing company that wants to improve the energy efficiency of the ERP to apply such a technique.

As discussed in our position paper (*citation omitted in this version of the paper for the sake of anonymity of the authors*) we have elaborated a research roadmap to i) provide MIS users with metrics and tools to assess applications energy efficiency, and ii) provide developers with guidelines for developing more energy efficient code. This paper presents the preliminary results of our work and proposes empirical evidence answering the following questions:

1) A common claim is that MIS software does not significantly impact on the overall consumption, which is thought to be led by the operating systems and the infrastructural layers. To what extent does MIS software energy consumption matter?
2) How different is the energy consumption of structurally different MIS applications performing the same functional workload? i.e., to what extent managers that select or coordinate the development of MIS applications may influence the green performance of the system?
3) Is energy efficiency always equivalent to performance or it exists a trade-off to evaluate?

Our research is based on the empirical analysis of some case studies: we selected 3 Enterprise Resource Planning systems (ERP), 2 Customer Relationship Management systems (CRM), and 4 Database Management Systems (DBMS), which are widely used MIS applications.

The presentation is structured as follows. Section 2 discusses the state of the art of research on software energy efficiency; Section 3 presents our empirical methodology; Section 4 presents our results; finally, Section 5 discusses limitations and future works.

2 STATE OF THE ART

The energy consumed by a data center is absorbed by different components. Figure 1 (based on data from Renzi, 2007, and Stanford, 2008) shows that approximately 40% of the power entering a data center is used for cooling, distribution devices and batteries, and that an additional significant amount is used by auxiliary components of the servers (e.g., AD/DC converters, fans), whereas only 18% reaches the processor.

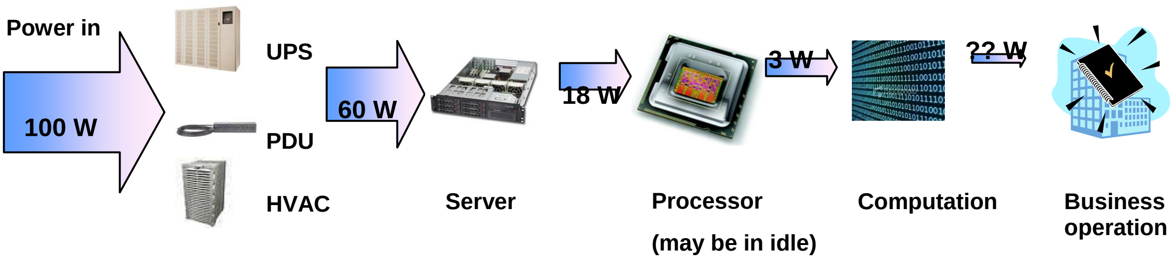


Figure 1. Power consumption break down in a data center.

The processor may stay in idle for some time, consequently only a minor part of energy is used for real computation (on average 3%). In addition to that, it is not yet clear what is the energy efficiency of the operations performed by the processor with respect with the final business operations, which are the goal of the whole data center working.

A lot of research has been conducted to optimize the power consumption of all the infrastructural layers of a data center. Vendors are improving the efficiency of UPS and HVAC systems (Avelar, 2007), and data center designers are striving to conceive innovative layouts to maximize cooling efficiency. A lot of researches, both by academies and hardware vendors, have focused on the power performances of hardware devices (e.g., APC, 2009). The inefficiency caused by idle time can be brilliantly counteracted by implementing virtualization, which keeps physical processors usage high so to reduce the impact of the overhead caused by infrastructural components. Research on virtualization techniques has already reached brilliant results (e.g., Uhlig, Neiger, Rodgers, Santoni, Martins, Anderson, Bennett, Kagi, Leung and Smith, 2005).

Even though most of the power absorbed by a data center is absorbed by the infrastructural layer, we believe that it is also important to investigate the role of software, which is the first cause of consumption, as it guides the operations performed by the processor and thus influences the consumption of all the layers above.

Contrarily from all the other layers, energy efficiency of the last layer of a data center remains largely unexplored. Researchers have not yet even agreed on a common methodology to measure and assess the energy efficiency of end-user and MIS applications, and no code-based predictive energy consumption metrics are so effective to be usable at MIS level.

Some works (e.g. Chatzigeorgiou and Spehanides, 2002) propose methodologies to estimate software energy efficiency. (Fornaciari, Gubian, Sciuto and Silvano, 2001) investigate low power embedded systems and introduce accurate and efficient power metrics to drive the hardware/software co-design. However, all these works are limited to embedded systems and cannot be extended to business applications, such as ERPs. The flow of operations performed by an embedded system is by far more predictable and less subject to change than in larger systems. In addition to that, in embedded systems software is tightly coupled with hardware and its power consumption can be more easily modeled. A typical MIS has multiple layers (hardware, operating systems, middleware, database management system, end-users applications), with multiple and concurrent operations. Accordingly, the MIS software developer has a lot of different choices and usually cannot modify the lower architectural layers.

Software engineering literature proposes several metrics for software quality. However, for none of these metrics a direct relationship with power consumption has been proved (Capra and Merlo, 2009).

Albers and Fujiwara (Albers and Fujiwara, 2007) studied scheduling problems in computer devices that operate on batteries with the aim of minimizing the energy consumption without losing a good Quality of Service. However, this work is focused on a very specific problem for battery-operated devices and is not easily extendible for general systems.

Also (Chatzigeorgiou and Stephanides, 2002) in their work address software energy efficiency and propose software metrics in terms of software energy consumption. Their metrics start from considering that the power is primarily dependent on the executing software and they derive energy measures that can be extracted from the flow graph of the program. The limitation of this work is that the metric have been validated on a certain kind of programs, drawn from matrix algebra and multimedia, whose execution flows are easily predictable, and have not been validated on every kind of program. Moreover the flow graph of a program is the representation of all the paths that might be traversed during the program execution, but it does not express the real execution of the program, since it is not possible to know in advance the number of times a cycle is executed or if a certain path will be taken.

In some cases, classic asymptotic complexity (Shaffer, 1998) is tentatively used as a proxy for energy efficiency. However, classic asymptotic complexity, which is used for measuring performance and scalability of computation algorithms, takes into consideration the total number of executed operations, but not of the consumption of each single operation. Consequently, it may be very discordant with measures of total consumption. Moreover, asymptotic complexity can be computed only through a semantic analysis of the code, and not automatically.

A number of consolidated design quality metrics, such as cyclomatic complexity (McCabe, 1976), Halstead Software Science (Halstead, 1977), and the set of metrics proposed by Chidamber and Kemerer (1994) and Brito e Abreu (1995), are easily measured automatically, but their relationship with energy efficiency has not been proved yet. The relationship between these metrics and energy efficiency is not even clear at an intuitive level. In fact, it is reasonable to suppose that in order to reach the maximum level of optimization of an application it may be necessary to renounce to its internal cohesion and clarity, thus affecting the values of classic metrics.

Other researches (Oliveira et al., 2008) have analyzed and evaluated the relationship between quality metrics and physical metrics including also measures of power consumption, but again these works are limited to embedded systems.

3 EMPIRICAL METHODOLOGY

3.1 Sample selection

We performed detailed analyses on a selected sample of case studies in order to gather empirical evidence. In particular, we focused on three categories of widely used MIS applications: ERPs, CRMs and DBMSs. For each category we selected some structurally different, but functionally comparable applications to perform our tests. We preferred open source applications so that we will be able to inspect the code for the next steps of our research roadmap and look for energy efficient design partners.

The applications selected are: *Adempiere* and *Openbravo* for the ERP category; *SugarCRM* and *vTiger CRM* for the CRM category; *MySQL*, *Ingres 2006*, *PostgreSQL*, and *Orable DB 11g* for the DBMS category.

3.2 Experimental setting

We developed a Java tool called *Workload Simulator* that for each application in our sample can simulate a given flow of operations and execute it a certain number of times for a given number of simultaneous users, thus generating a benchmark workload. *Workload Simulator* eliminates the user thinking times between subsequent operations so to allow comparisons across different applications. As all the selected applications have a client/server structure, *Workload Simulator* synchronizes multiple clients together. We measure the power consumed by the *Server Machine*, which receives the requests from the *Clients.* Figure 2 shows the overall system architecture. It can be noted how all the tools needed to monitor the power consumption and to generate the benchmark loads do not interfere with the server load.

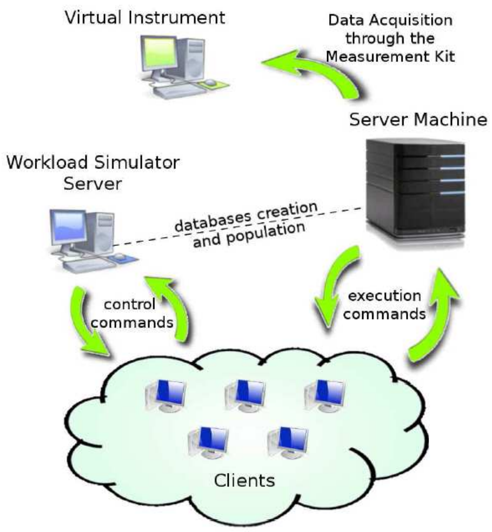


Figure 2. System architecture for workload simulation.

All the experiments have been performed on two servers with the same hardware setting, one running Microsoft Windows 2003 Server Enterprise Edition, and the other running Linux CentOS. Table 1 reports the configurations used for the Server Machine.

Parameter	
Processor	2x Intel Xeon 2.40 GHz
Cores	2 per processor
Internal Data Cache	2x 8 kb
On-board cache	2x 512kb
Motherboard	Asus PR-DLS
Total Memory	1 GB DIMM
Memory Bus Speed	4x 100 MHz (400MHz)
Chipset	Server Works CMIC-LE
Storage Device	68 GB SCSI hard disk
Operating Systems	Microsoft Windows 2003 Server Enterprise Edition Linux CentOS

Table 1. Server Machine configuration.

We measured the power absorbed by the Server Machine by an ad-hoc developed kit based on Hall effect current sensors, in order to have as accurate measures as possible. We sampled the values of power consumption at a frequency of 250 Hz by means of a NI USB-6210 DAQ (Data Acquisition Board). All the collected samples were then analyzed, aggregated and digitally stored by means of an ad-hoc tool called *Virtual Instrument* that we implemented with LabVIEW (Formenti and Gallazzi, 2009).

Our kit can measures both the total power absorbed by a system and the power absorbed by its main three subcomponents, i.e. the processor, the hard disk device and the motherboard.

3.3 Benchmark workload definition

For each category in our sample we identified some of the most typical flows of operations that are representative of that category.

For ERP systems we identified three different typical flows of operations:

1) the process of creating and inserting a business partner inside the system;
2) the process of inserting and handling products in the system, and
3) the process of creating sales and purchase orders using the ERP system.

As regards CRM systems, we selected the following scenarios:

1) the creation of a new account, and
2) the creation of a new campaign.

For DBMS we implemented an ad-hoc version of the benchmark TPC-C (Transaction Processing Performance Council, 2007), which is one of the most popular way of comparing OLTP performance on various hardware and software configurations. The four selected DBMS have been configured in order be as comparable as possible (i.e. setting the amount of usable memory to the same parameters or choosing the same DB engine). For more details refers to Formenti and Gallazzi (2009).

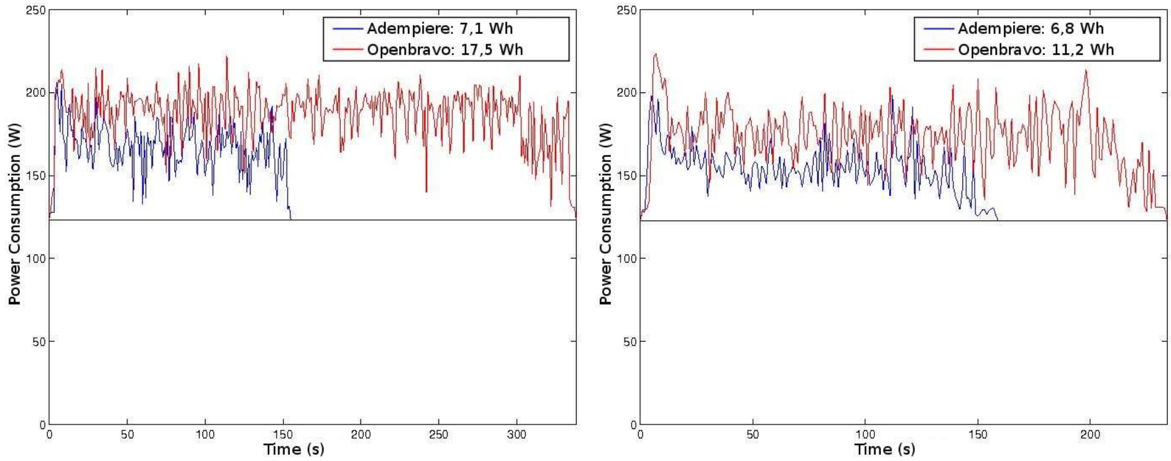
4 RESULTS

4.1 Results of the experiments

For each group of comparable applications, we executed the identified benchmark workloads by means of *Workload Simulator* and acquired power consumption data of the server machine by means of the measurement kit and the *Virtual Instrument*. Each simulation has been repeated 10 times and with a different number of clients connected to the server. We obtained a maximum variance of 5%.

For each simulation we plotted the power consumption values in graphs power vs. time. We also plotted the value of the power absorbed by the system in idle. We computed the integral of the power absorbed by the system performing each workload minus the idle power over the time (expressed in Wh in the upper right box in the following figures). This measure represents the effective energy consumption of each benchmark workload.

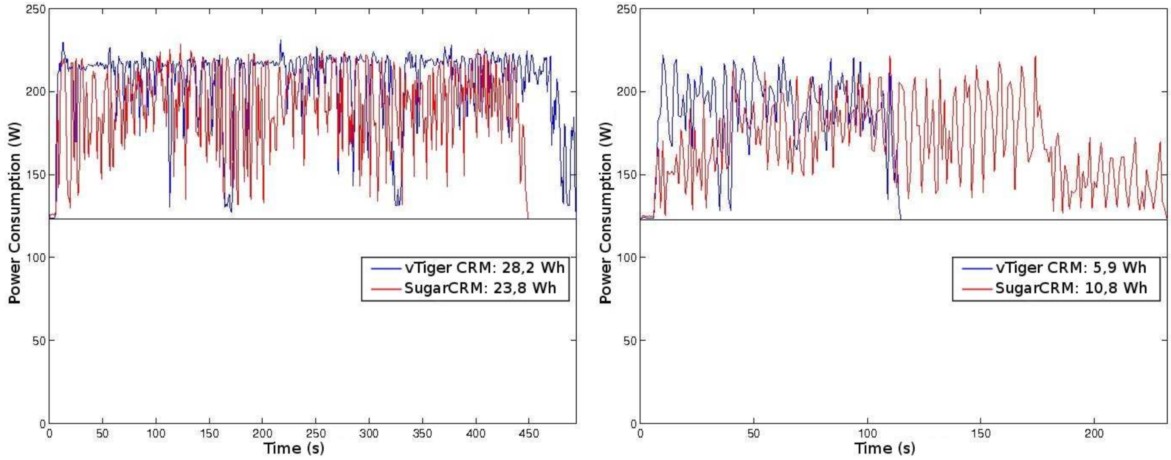
Figure 5 shows power consumption vs. time plots for *Adempiere* and *Openbravo*, both on Windows and on Linux, referred to the execution of the creation of a new business partner benchmark workload with 3 clients.



a) Windows **b) Linux**

Figure 3. ERP simulation with 3 clients.

Figure 6 shows the result of the experiment conducted on *SugarCRM* and *vTiger CRM* related to the creation of a new account, executed with 3 clients.

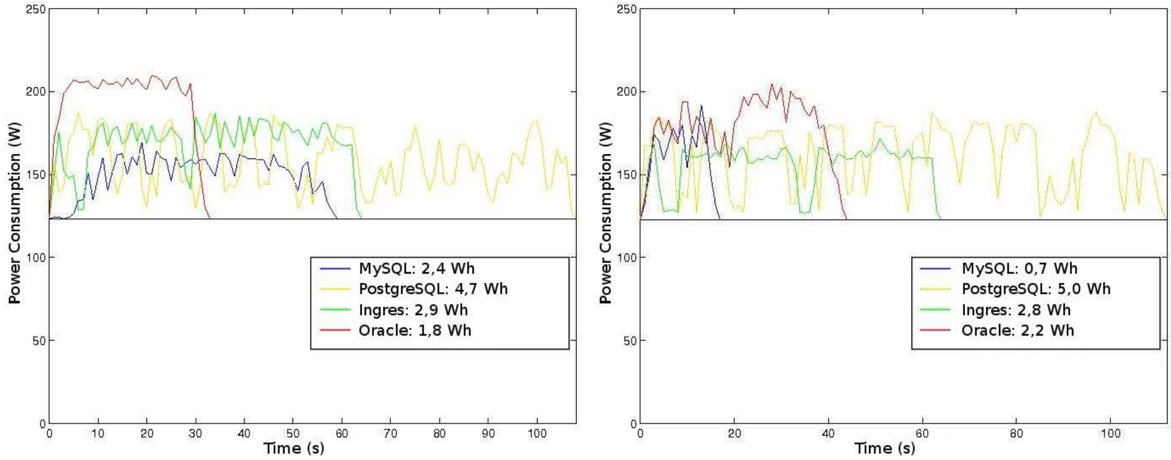


a) Windows **b) Linux**

Figure 4. CRM simulation with 3 clients.

In this case it is remarkable how different are the behaviors under the two operating systems. On Windows systems *SugarCRM* is always more energy efficient than *vTiger CRM*, whereas on Linux systems it is the opposite. Note that in the previous case both the ERP applications had the same trend. This is probably due to the fact that both the ERPs are written in Java, which makes their behaviors more platform -independent. This result also underlines how important it is to analyze the relationship between the operating system and the application running on top of it when investigating energy efficiency. This specific topic will be object of future work within our research program.

Figure 7 shows the power consumption comparison of the four DBMS in our sample when executing the *TPC-C* benchmark with 10 concurrent clients.



a) Windows **b) Linux**

Figure 5. DBMS simulation with 10 clients.

Table 2, Table 3, and Table 4 shows the results of the empirical experiments that we conducted. We found that percentage differences of the values that we monitored are not affected by the number of clients. Accordingly, we report the results only for some significant experiments.

Benchmark workload 1 (3 clients)	**Adempiere**	**Openbravo**	**Δ (%)**	**OS**
Time (s)	158	233	47	Linux
Average power (W)	154	173	12	Linux
Total consumption (Wh)	6,76	11,2	66	Linux
Total consumption minus idle (Wh)	1,38	3,26	136	Linux
Time (s)	154	337	119	Windows
Average power (W)	166	186	12	Windows
Total consumption (Wh)	7,1	17,5	145	Windows
Total consumption minus idle (Wh)	1,8	5,9	222	Windows
Benchmark workload 2 (3 clients)				
Time (s)	114	184	61	Linux
Average power (W)	154	171	11	Linux
Total consumption (Wh)	4,9	8,7	78	Linux
Total consumption minus idle (Wh)	1,0	2,5	142	Linux
Time (s)	122	242	98	Windows
Average power (W)	152	186	22	Windows
Total consumption (Wh)	5,2	12,5	142	Windows
Total consumption minus idle (Wh)	1,0	4,2	330	Windows

Table 2. Results of the experiments on ERPs.

Benchmark workload 1 (3 clients)	**vTiger CRM**	**SugarCRM**	**Δ (%)**	**OS**
Time (s)	234	312	33	Linux
Average power (W)	187	179	-4	Linux
Total consumption (Wh)	12,1	15,5	28	Linux
Total consumption minus idle (Wh)	4,1	4,9	17	Linux
Time (s)	1120	722	-55	Windows
Average power (W)	212	184	-15	Windows
Total consumption (Wh)	66,1	37,0	-79	Windows
Total consumption minus idle (Wh)	27,7	12,2	-126	Windows
Benchmark workload 2 (3 clients)				
Time (s)	114	231	103	Linux
Average power (W)	187	169	-11	Linux
Total consumption (Wh)	5,9	10,8	82	Linux
Total consumption minus idle (Wh)	2,0	2,9	44	Linux

Benchmark workload 1 (3 clients)	vTiger CRM	SugarCRM	Δ (%)	OS
Time (s)	493	228	-116	Windows
Average power (W)	206	191	-8	Windows
Total consumption (Wh)	28,2	23,8	-18	Windows
Total consumption minus idle (Wh)	11,3	8,5	-33	Windows

Table 3. Results of the experiments on CRMs.

TPC-C (10 clients)	MySQL	PostgreSQL	Ingres	Oracle	OS
Time (s)	16	111	63	32	Linux
Average power (W)	165	162	157	179	Linux
Total consumption (Wh)	0,7	5,5	2,7	2,2	Linux
Total consumption minus idle (Wh)	0,2	1,2	0,6	0,7	Linux
Time (s)	58	107	63	43	Windows
Average power (W)	151	158	170	198	Windows
Total consumption (Wh)	2,4	4,7	3,0	1,8	Windows
Total consumption minus idle (Wh)	0,4	1,1	0,8	0,7	Windows
TPC-C (20 clients)					
Time (s)	21	208	119	66	Linux
Average power (W)	183	165	160	196	Linux
Total consumption (Wh)	1,1	9,6	5,3	3,6	Linux
Total consumption minus idle (Wh)	0,4	2,5	1,2	1,4	Linux
Time (s)	66	243	116	61	Windows
Average power (W)	166	161	173	202	Windows
Total consumption (Wh)	3,1	10,9	5,6	3,4	Windows
Total consumption minus idle (Wh)	0,8	2,6	1,6	1,3	Windows

Table 4. Results of the experiments on DBMS.

4.2 Discussion of the empirical results

Our empirical results allow us to provide some preliminary answers to the research questions proposed in Section 1.

1) MIS applications do impact on IT energy consumption. In fact, according to our experiment the application layer can increase the system's consumption up to 72% with respect to the system in idle state.
2) Different MIS applications that satisfy the same functional requirements and run on the same hardware and operating system have significantly different consumptions. In particular, our empirical results show that these differences can be up to 145% (mean value 79%, minimum

18%). [1] This means that, given a specific infrastructural setting, choosing different MIS applications completely comparable from a functional point of view may have a significant impact on the total energy consumption of the system, and consequently on the operative costs.

3) In some scenarios energy efficiency is not equivalent to performance. In fact we found that in some case a quicker application may overall consume more energy than a slower application, because its average power consumption is more than proportionally higher (see for example the case of Ingres and Oracle on Linux with 10 clients, Table 4). In general, the difference of energy consumption between different applications is seldom proportional to the difference in response time.

Results 2 and 3 are based on the comparison of the values of energy consumption minus power in idle computed as described in the previous section, i.e. as the integral of power over the execution time. This seems correct as long as:

1) Servers are not switched off when the processors are not in use;
2) Workloads are generated along with business processes and are not known before hand (otherwise applications with a lower processor usage should be parallelized when possible);
3) Processors are idle for a significant part of the time (otherwise applications that complete the workload in a shorter time should soon begin a new workload).

All these conditions widely apply to MIS contexts (see for example the article of Forrester et al. on McKinsey Quarterly, 2008). Whenever these conditions cease to exist, the measurement methodology should be reframed, as parallelization and queue management should be considered. The question whether energy consumption and performance are two different faces of the same issue or not, and the framing of the problem in all the possible situations, pose themselves interesting new research questions that we will address in our future work.

Our results also show how the infrastructural layer (e.g. the operating system or the Java Virtual Machine) and the interaction of this layer with the application and MIS layers have an important role in determining the energy efficiency of the overall system.

4.3 Analysis of the power consumption breakdown

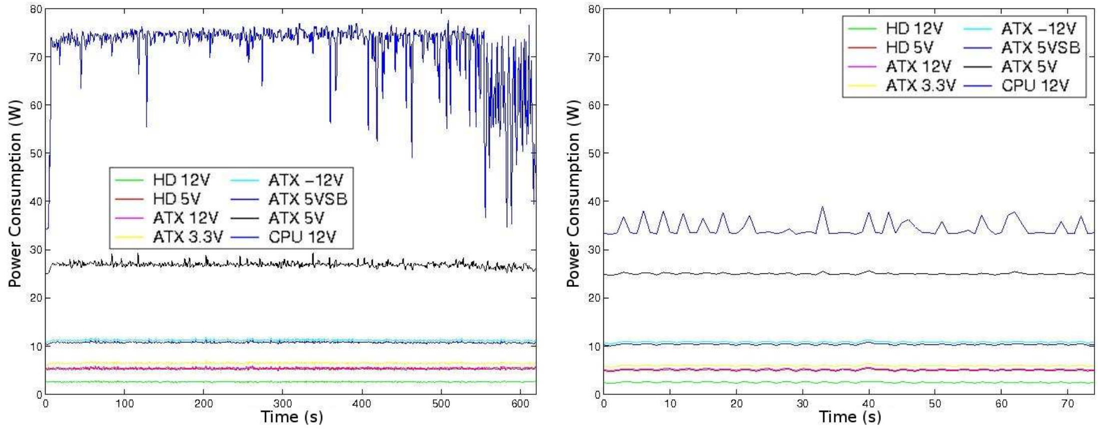
Figure 8 shows the power consumption breakdown among the different components of a server. We have collected these data by measuring the power absorption of the different power supply cables within the server: HD (2 supply cables at 5V and 12V), CPU (only one 12V supply cable), ATX (5 supply cables with different voltage).

These data clearly show that:

1) The processor is responsible for the biggest part of power absorbed (approximately 60% under load and 45% in idle);
2) The processor is the only component that is significantly affected by the load. In fact, all the other components consume more or less the same amount of power both under load and in idle. This is partially explained by the fact that most of the power of modern HD drives is used for the spindle of the disk, and not for the reading and writing operations. Similarly, dynamic RAM banks are periodically refreshed independently from effective reading and writing operations.

---

[1] These values are computed as the differences in percentage between the energy consumption minus idle power (Wh).



a) Under load power consumption. **b) Idle power consumption.**

Figure 6. Power requirements breakdown among hard-disk (HD), motherboard (ATX) and the processor (CPU).

These results suggest that energy consumption optimization research for systems adopting traditional storage devices should mainly focus on the operations performed by the processor. Power consumption of systems using Storage Area Networks or solid state drives needs further investigation.

Please note that the results presented in this section are referred to a single server systems, and are thus not comparable with the data presented in **Figure 1** and discussed in Section II, which are referred to a whole data center, including conditioning, UPS, and power distribution systems. The consumption of these components is directly proportional to the consumption of the computation units.

5 CONCLUSIONS AND FUTURE WORK

The preliminary results of our research show that MIS applications may have a deep impact on the energy consumption of an information system. Accordingly, managers should consider software energy efficiency as a new quality metric when selecting or developing software applications.

In addition to empirical evidence supporting our claims, in this paper we provided a scientific methodology to compare the energy consumption of different software applications that satisfy the same functional requirements. In future work we will try to identify code-based metrics to predict the energy efficiency of an application. These would be useful for selecting MIS applications without setting up complex energy consumption measurement experiments, and for controlling the quality of software development processes. Moreover, our research roadmap will also focus on identifying developing best practices and design patterns for energy efficient code. In order to achieve this, we will start by in-depth analyzing the structure and the code of the case study applications that we have so far considered and tested.

Our study suffers from some limitations. First of all, in some cases the applications that we have compared have some slight functional differences, as it is very difficult to find functionally identical applications. In addition to that, for some DBMS it is difficult to establish to what extent the configurations of two compared applications are exactly the same, as the parameters that can be set are different. However, the energy consumption that we measured were so different that we believe these errors do not change the value of our conclusion. Finally, our analyses can be further improved by separately analyzing the energy consumed by the data layer, the computation layer and the interface of a MIS software application. These analyses will be object of future work.

References

ACEEE, "A Smarter Shade of Green", ACEEE Report for the Technology CEO Council, 2008.

S. Albers, H. Fujiwara, "Energy-Efficient Algorithms for Flow Time Minimization", ACM Transactions on Algorithms (TALG), Vol. 3, Article No. 49, 2007.

F. Brito e Abreu, "The MOOD metrics set", Proceedings of ECOOP Workshop on Metrics, 1995.

E. Capra, F. Merlo, "Green IT: everything starts from the software", ECIS, 2009.

A. Chatzigeorgiou, G. Stephanides, "Energy Metric for Software Systems", Software Quality Journal, 10, pp. 355–371, 2002.

S.R. Chidamber, and C.F. Kemerer, "A metrics suite for object oriented design", IEEE Transactions on Software Engineering, vol.20, no. 6, pp. 476–493, 1994.

G. Formenti, S. Gallazzi, "A methodology to empirically evaluate software energy consumption and its impact on the Total Cost of Ownership", Master thesis, Politecnico di Milano, April 2009.

W. Fornaciari, P. Gubian, D. Sciuto, C. Silvano, "Power estimation of embedded systems: a hardware/software codesign approach", Readings in hardware/software co-design, pp. 249 – 258, 2001.

W. Forrest, J.M. Kaplan, and N. Kindler, "Data centres: how to cut carbon emissions and costs", McKinsey Quarterly, no. 14, November 2008, pp. 4-14.

M. H. Halstead, Elements of software science, Elsevier Computer Science Library, 1977.

R. Kumar, "Important power, cooling and green it concerns", Technical report, Gartner, January 2007.

C. Lee, and E. G. Brown, "Topic overview: Green it", Technical report, Forrester Research, November 2007.

T. J. McCabe, "A complexity measure", Proceedings of International Conference on Software Engineering, pp. 407, 1976.

S. Murugesan, "Harnessing Green IT: Principles and Practices", IT Professional, vol. 10, no. 1, pp. 24-33, 2008.

M. Oliveira, R. Redin, L. Carro, L. Lamb, and F. Wagner, "Software quality metrics and their impact on embedded software", MOMPES 2008, 2008.

F. Renzi, "Evolution scenarios for data centers technologies and architectures and impacts on energy consumptions", IBM, Presentation held at the conference "Green ICT", 22nd November 2007, Milan, Italy.

C.A. Shaffer, Practical introduction to data structures and algorithms, Prentice Hall, 1998.

A. Sivasubramaniam, M. Kandemir, N. Vijaykrishnan, and M.J. Irwin, "Designing Energy-Efficient Software," ipdps, International Parallel and Distributed Processing Symposium (IPDPS), vol. 2, pp. 176, 2002.

E. Stanford, "Environmental trends and opportunity for computer system power delivery", 20th Int'l Symposiumon Power Semiconductor Devices and IC's, 2008.

Transaction Processing Performance Council (TPC), "Tpc benchmark c, standard specification, revision 5.9", Technical report, 2007.

S. Daud, R. B. Ahmad, N. S. Murthy, "The effects of compiler optimization on embedded system power consumpion", International Journal of Information and Communication Technology 2009, Vol. 2, No. 1/2, pp. 73-82, 2009.

X . Huang, K. Li, R. Li, "A Energy Efficient Scheduling Base on Dynamic Voltage and Frequency Scaling for Multi-core Embedded Real-Time System ", Algorithms and Architectures for Parallel Processing, Springer, pp. 137-145, 2009.

APC, "Capping Energy Consumption In The Data Center", eBook "Data Center Facilities IT Handbook Chapter 3", 2009.

V. Avelar, "Making Large UPS Systems More Efficient", APC White Paper, 2007.

Uhlig, R. Neiger, G. Rodgers, D. Santoni, A.L. Martins, F.C.M. Anderson, A.V. Bennett, S.M. Kagi, A. Leung, F.H. Smith, Intel, "Intel virtualization technology", Computer, Vol. 38, Issue. 5, pp. 48-56, 2005.