Computing (2013) 95:269–287
DOI 10.1007/s00607-012-0218-8

Study on energy-consumption regularities of cloud computing systems by a novel evaluation model

Jie Song · Tiantian Li · Zhi Wang · Zhiliang Zhu

Received: 26 May 2012 / Accepted: 21 September 2012 / Published online: 17 October 2012

Abstract Due to the limited energy, environmental problem and fast growth of computer power consumption, energy-efficient computing has now become a critical and urgent issue. Therefore, an energy-consumption (energy consumed per task) evaluation model is necessary to be established. However, the existing models are almost qualitative rather than quantitative and with poor practicability because the EC measurement of cloud computing systems is rather difficult. In this paper, we propose an EC model and the corresponding calculation approach for cloud computing systems and prove its correctness and accuracy through a series of experiments. Based on this model, we also analyze the EC features of cloud computing systems, summarize some EC regularities and propose some EC optimization suggestions. With this model, one can calculate EC only by the values that are easily measured without special hardware, and can explore and evaluate new optimization methods.

Keywords Cloud computing · Energy consumption · Energy optimization · MapReduce

Mathematics Subject Classification (2000) 68M14

J. Song · T. Li · Z. Wang · Z. Zhu (✉)
Software College, Northeastern University,
349 Mailbox, No.11, Lane 3, WenHua Road, HePing District,
Shenyang 110819, Liaoning, China
e-mail: zzl@mail.neu.edu.cn

J. Song · T. Li · Z. Wang · Z. Zhu
Cloud Computing Lab, Software College, Northeastern University,
Shenyang, People's Republic of China

1 Introduction

With the coming of cloud computing era, people's demand for IT resources and servers connected to the Internet is increasing constantly, thus leading to huge energy costs and carbon dioxide emissions. In 2007, the annual energy consumption of IT resources is as much as the annual power supply of overall 14 large power stations and its indirect carbon dioxide emissions are equal to that of the global aviation companies [1]. From the year 2000 to 2005, the number of servers has been doubled. In a word, the limited energy, environmental problem and rapidly climbing energy consumption have made energy-efficient computing a critical and urgent issue.

To achieve real energy-efficient computing, it is necessary to find an appropriate energy-consumption (EC) evaluation model, through which we can evaluate EC and explore potential optimization methods. Here, EC refers to the consumed energy (i.e. energy consumption) of a certain task, which should be atomic and general. The existing models are almost qualitative rather than quantitative and with poor practicability for the following two reasons. One is that the power or power consumption should be constrained by task, otherwise it is meaningless to evaluate the consumed energy of different tasks; the other one is that the active power (or real power) is different from the rated power, which is dynamic, and the EC cannot be obtained just through a simple calculation like $Power \times Time$.

On the other hand, to evaluate EC, the real power should be measured in real-time. Most power-meters have a certain measurement range, unable to adapt to cloud computing systems with a large number of nodes; and the distributed environment in cloud is not suitable for setting up power-meters, if we set up one power-meter for the whole system, we cannot distinguish the power of each node; the measurement granularity is too coarse to contribute to the EC optimization research. If we set up one power-meter for each node, their communication and synchronization could be a problem.

To solve the problems mentioned above, in this paper we propose an EC model through which one can calculate the real-time power and EC by some easily measured values. We take three steps to do the research. First of all, we define the mathematical expression of EC which considers "atomic task" as a constraint. Then, we describe EC's measurement and calculation approaches and deduce its extremum conditions. In addition, to facilitate calculating EC, we improve the calculation formula of power based on previous efforts [2–5]; the formula also appeared in [6] in Chinese. Finally, we design a series of experiments to prove the correctness and accuracy of the proposed model. Based on this model, we also evaluate and analyze the EC features of CPU-intensive, I/O intensive and interactive tasks [7–11] under a cloud computing environment. According to the analysis results, we propose the Waiting Energy Consumption (WEC) which means computers of a cluster or components of a computer could be in "passive idle" or "busy idle" state because they are waiting for other resources; the EC could be optimized if such energy consumption could be reduced. In a word, with this model one can calculate EC only by the values that are easily measured without special hardware, and can explore and evaluate new optimization methods.

The rest of this paper is organized as follows. Following the introduction, Sect. 2 introduces the related work. Section 3 gives the definition of atomic task and presents the EC model. Section 4 describes the measurement and calculation approaches of EC, and deduces its extremum conditions. In Sect. 5, the correctness of the proposed model is validated, the EC regularities of three typical tasks are analyzed, and the EC optimization approaches by reducing WEC are proposed. Section 6 summarizes this paper and presents the further works.

2 Related works

Energy-efficient computing is a recently proposed topic, and to our best knowledge, few works focus on Energy-Consumption (EC) evaluation and calculation models. Literature [12] summarized the challenges of achieving self-optimization framework that considered both energy and reliability in green cloud environment. It pointed out that "QoS/energy" should be adopted as the new measurement, but it did not propose any applicable solutions. Literature [13] defined EE as the ratio of the performed work to the consumed energy, and deduced it as the ratio of performance to power. However, the model they proposed is used to evaluate the EE of a database server, unable to be applied in cloud computing systems for the reasons mentioned in Sect. 1.

Besides, there are many researches about energy consumption optimization. Literature [14] considered energy consumption as $Power \times Time$, without considering the time-varying computer power. Literature [15] illustrated EC characteristics of cloud computing with plenty of experiments, and proposed an energy-saving framework based on Virtualization and Live Migration technologies. In [15], the computer power under a fixed task is used for measuring the effects of energy optimization, without considering the effects of task type on energy consumption and the relationship between power and CPU working state.

Literature [5] showed the relationship between computer power and CPU working state: $P = P_{fix} + P_f f^3$; P_{fix} is a constant indicating the power of all non-CPU components, P_f is the CPU power coefficient and f is CPU frequency. This relationship has been quoted and proved by many literatures [16–18]. However, it stands under two important assumptions: One is that the power of all non-CPU components is constant and the other is that CPU is fully utilized. Actually, the power of all non-CPU components is not exactly constant and CPU usage also changes with the workload size and the features of performed tasks. Sometimes, CPU cannot be fully utilized for I/O or network bottlenecks. Moreover, literature [19] proved that there is no absolute linear relationship between power and the cube of CPU frequency through an example of I/O-intensive task. We think it is right because I/O-intensive tasks cause CPU waiting for I/O operations, so the CPU usage is relatively low, not satisfying the second assumption in literature [5]. However, literature [19] did not propose a new equation. In addition, literature [2] proposed a linear and an empirical power model: $P_1 = C_0 + C_1 u$, $P_2 = C_0 + C_1 u + C_2 u^r$; Literature [3] proposed a power model: $P_3 = C_0 + C_1 u_{CPU} + C_2 u_{disk}$; Literature [4] proposed a power model: $P_4 = C_0 + C_1 u_{CPU} + C_2 u_{disk} + \sum C_i P_i$. Models like P_1 and P_2 are too simple to be accurate enough in some scenarios and models like P_3 and P_4 are comprehensive

but hard to use in practice because some of the variables in the model are difficult to measure in cloud computing systems. Furthermore, these models have a common and essential deficiency: not taking CPU frequency, which has a significant influence on power, into consideration.

To summarize, now there are few EC evaluation models suitable for cloud computing systems and the existing power models did not take CPU frequency into consideration. In this paper, we propose a specialized EC model considering both CPU frequency and usage for cloud computing systems, and also evaluate and analyze the EC features of CPU-intensive, I/O intensive and interactive tasks, which are selected based on our study and analysis of GridMix, Hive Performance Benchmark, Yahoo Sorting Algorithm, HiBeach and DFSIO [7–11], in a cloud computing environment. According to the analysis results, we also propose some EE optimization suggestions.

The proposed EC model refers to the consumed energy per task; it contributes to a better study of energy-efficient computing and provides a benchmark for EC optimization. Some existing studies are about power reduction [15,16] or task immigration [17,20–23], while in our opinions reducing computer power does not necessarily reduce EC because lower-power may lead to lower-performance too, and turning off idle nodes by task immigration may save energy but also delay the task execution, in which case EC may not be reduced. In this paper, we have not yet proposed a complete optimization method, but we do have proved the existence of the minimum EC and deduced its occurrence conditions mathematically. We also have verified the extremum conditions by experiments, and propose the idea of optimizing EC by reducing Waiting Energy Consumption (WEC). These works contribute to the study of EC optimization.

3 Energy consumption model

Energy consumption of a cloud computing system includes three parts: computers, network and other auxiliary facilities such as cooling systems. This paper focuses only on the energy-consumption (EC) of computers. Literally, EC is the consumed energy of computers during the time that a certain task is performed. EC is different from power consumption because it is task related. The most common model of EC is power multiplying the executed time of an atomic task. We can define an atomic task as sorting-1,000-records in a sorting system, or the response of a HTTP request in a Web system, or a transaction in a database system, but such definition is not general and it depends on the task's complexity. The cloud computing system provides all kinds of services, so it is difficult to find a general task. Therefore, without losing generality we define EC as the consumed energy per task.

FLoating-point Operations Per Second (FLOPS) or Million Instructions per Second (MIPS) are generally used to measure computer efficiency, and floating-point-operation instructions could be the atomic task of a computer. Watt is used to measure computer power. Therefore, Watt/FLOPS is equal to the ratio between the consumed energy and performed floating-point operations in a second, it can be used as computer EC measurement, expressing the consumed energy (joule) of a floating-point operation. The value of FLOPS depends on not only CPU frequency but also the number

of pipelines processing the floating-point operations and the average clock ticks consumed to execute a single-precision floating-point operation. Pentium IV processor uses Prescott core which has two floating-point pipelines, and performing a single-precision floating-point operation consumes six clock ticks averagely. If it works at frequency of 3 GHz, the theoretical value of single-precision floating-point operations completed in 1 s is $3 \times 10^9 \times 2 \div 6 = 10^9$, power is about 80 W, and EC is about 8×10^{-8} W/FLOPS. On the contrary, early computer UNIVAC I (1951) can process 1,905 floating-point-operations 1 s and its power is 125 KW, EE is as lower as about 65.6 W/FLOPS.

The above example shows FLOPS is the result of CPU frequency multiplying the number of pipelines and dividing the average clock ticks consumed per single-precision floating-point operation, which is complicated to be measured. As we know, the CPU frequency is dynamic, while the rest parameters are static and only related to the CPU type. If we highlight CPU frequency and let FLOPS $=Fp(t) = \theta \times f(t)\ (\theta > 0)$, where $Fp(t)$ is FLOPS at time t, $f(t)$ is CPU frequency at time t and θ is CPU feature like "number of pipelines dividing the average clock ticks consumed per single-precision floating-point operation". Therefore, FLOPS has linear relationship with CPU frequency. The situation above is under the condition that CPU is full-loaded; if CPU usage is $\omega(\%)$ at time t, which means only ω percent of maximum floating-point-operations is performed, then

$$Fp(t) = \theta_{CPU} \times f(t) \times \omega(t)$$

To simplify the definition of the task, we design a new atomic task which is similar with $Fp(t)$ but much simpler in both unit and calculation. We define the atomic task as the floating-point-operations performed by a full-loaded 1 GHz processor in 1 s, denoted as $1K$. Assume that during time T (s), CPU works at frequency f (GHz) and usage $\omega(\%)$ regularly, and then the performed tasks could be expressed as follows:

$$K = f\ (\text{GHz}) \times \omega(\%) \times T\ (\text{s}) \times \lambda_{CPU}\ (\text{unit of } K\ is\ \text{GHz} \times \text{s})$$

Different types of CPUs have different performance, so $1K$ may stand for different amount of computation. To address this problem, we adopt the product of CPU frequency, CPU usage and a coefficient λ to measure the computation performed by a specific CPU. However, in terms of EC optimization, we can study EC regularities and evaluate the effects of optimization methods as long as we use a unified atomic task. Moreover, the main purpose of defining the EC model is to guide and evaluate EC optimization, and the definition of task K has no impact on achieving such a purpose. Therefore, without loss of generality, in this research, we set λ as 1 to simplify the model; while in actual measurement, we can adjust the value of λ according to the types of CPUs in cloud computing systems.

Next, we will derive the mathematical expression of EC. Let $K(T)$ be the performed tasks during time T and $E(T)$ be the corresponding consumed energy, energy consumption $\varepsilon(T)$ can be expressed as:

$$\varepsilon(T) = E(T)K(T)^{-1} \tag{1}$$

When ω=0, that means CPU is totally idle, then $\varepsilon = 0$. The unit of ε is *Joule/*GHz s.

For a cloud computing system, assume that there are N nodes, each denoted as $c_i (1 \le i \le N)$. For node c_i, let $f_i\ (t)$ be its CPU frequency, $\omega_i\ (t)$ be its CPU usage and $p_i\ (t)$ be its power at moment t, then

$$K_i(T) = \sum_{i=1}^{N} \int_0^T f_i(t)\omega_i(t)dt \tag{2}$$

$$E_i(T) = \sum_{i=1}^{N} \int_0^T p_i(t)dt \tag{3}$$

$$\varepsilon_i(t) = E_i(t)K_i(t)^{-1} = \frac{p_i(t)}{f_i(t)\omega_i(t)} \quad \varepsilon(T) = E(T)K(T)^{-1} = \frac{\sum_{i=1}^{N}\int_0^T p_i(t)dt}{\sum_{i=1}^{N}\int_0^T f_i(t)\omega_i(t)dt} \tag{4}$$

Each computer has its rated power and CPU frequency. If CPU is fully utilized, the theoretical value of EC is the ratio between its power and rated frequency. While in fact, CPU usage depends on the features of the performed tasks and CPU frequency also scales with the workload size, so does the computer power. Therefore, EC should be measured by real power and frequency instead of rated ones.

In the next section, we will describe the measurement and calculation approaches, improve the relationship between computer power and CPU working state and analyze the EC extremum conditions.

4 Measurement and calculation approaches

Based on the energy-consumption (EC) model proposed above, this section introduces the measurement and calculation approaches of EC. Limited by instruments, the measurement of cloud computing system's EC is relatively complex. Therefore, we propose a calculation approach, which calculates EC by the easily measured values, such as CPU usage and frequency. We also deduce the minimum EC and its occurrence conditions according to the calculation formula.

4.1 Measurement approaches

Measurement of EC includes two parts: $K\ (T)$ and $E\ (T)$. According to Eq. (2), $K\ (T)$ is the integral of $f\ (t)$ and $\omega\ (t)$ on time. The value of $f\ (t)$ depends on the workload size and frequency scaling algorithm of operation system (OS). For example, CPU can work under 5 different modes in Linux OS (*Performance*, *Powersave*, *Userspace*, *Ondemand* and *Conservative* mode). The value of $\omega\ (t)$ depends on the task types. For example, $\omega\ (t)$ is higher when performing CPU-intensive task and is lower when

performing I/O intensive task. In a word, it is hard to find the mathematical expression of $f(t)$ and $\omega(t)$. Hence, $K(T)$ cannot be obtained by a simple calculation.

Fortunately, most OSs provide programming interfaces for retrieving CPU frequency and usage. We can deploy a CPU monitor (software) on each node in cloud computing environment. Moreover, it is easy to synchronize CPU monitors and collect data in cloud computing environment. Let the monitor collects CPU frequency f and usage ω every Δt time, and we get M records in T time, then:

$$K(T) = \sum_{i=1}^{N} \int_{0}^{T} f_i(t)\omega_i(t)dt \approx \sum_{i=1}^{N}\sum_{j=1}^{M} f_i(\Delta t \cdot j)\omega_i(\Delta t \cdot j)\Delta t (M = T/\Delta t) \quad (5)$$

Right-hand side of Eq. (5) is reasonably close to $K(T)$ when Δt is small enough.

As to $E(T)$, there are many measurement approaches. Each computer has a rated power while the real power is time-varying. Therefore, it is also hard to find the mathematical expression of $p(t)$ so that we can calculate $E(T)$ according to Eq. (3). Actually, we can also collect power every Δt time. Then, similar to Eq. (5):

$$E(T) = \sum_{i=1}^{N} \int_{0}^{T} p_i(t)dt \approx \sum_{i=1}^{N}\sum_{j=1}^{M} p_i(\Delta t \cdot j)\Delta t (M = T/\Delta t) \quad (6)$$

Right-hand side of Eq. (6) is reasonably close to $E(T)$ when Δt is small enough. However, it is hard to measure $p(\Delta t \cdot j)$ because most computers or OSs do not provide interfaces to get the real-time power. If we install a power-meter for each node in the cloud computing system, we need lots of power-meters, and meanwhile, the communication between them (such as time synchronization and data summarization) is complex. A better solution is measuring the real-time power of the whole cloud computing system using one power-meter rather than that of each node.

Unfortunately, there still exist some deficiencies using such a method: ① Each power-meter has a certain measurement range, 0∼2,500 W for example; For a cloud computing system, the total power would be far beyond the measurement range. ② By this method, we cannot distinguish each node's power. The measurement granularity is too coarse to contribute to the EC optimization research. In a word, the measurement approach of $E(T)$ in cloud environment requires additional equipments; it is much more complex than the measurement of CPU frequency and usage.

If we can calculate the real-time power according to the real-time CPU frequency and usage, and further calculate the energy consumption according to Eq. (6), it will greatly simplify the measurement of EC. We will describe such calculation approach in Sect. 4.2.

4.2 Calculation approach

In this section, we briefly introduce the power calculation approach that has appeared in our [6] in Chinese. We assume the cloud computing system as homogeneous. For

a heterogeneous cloud, it can be treated as a number of homogeneous sub-clouds and we can calculate $E(T)$ in each sub-cloud and then sum them up as the whole $E(T)$.

According to the definition in [5], when CPU runs at frequency f with full usage, the relationship between computer power and CPU frequency is: $p_c = X_1 + Y_1 f^3$ (X_1 and Y_1 are system-related coefficients); according to the definition in [2], when CPU runs at usage ω under a fixed frequency f, the relationship between computer power and CPU usage is: $p_c = X_2 + Y_2\omega$ (X_2 and Y_2 are system-related coefficients). However, literature [5] assumes that the CPU usage is 100%, while in fact CPU usage changes with the workload size and the task types. CPU frequency can also be scaled according to the workload size. So it is unreasonable to consider a simple relationship between computer power and CPU frequency or usage. We believe that computer power p is associated with not only CPU frequency f but also CPU usage ω, i.e. $p = P(f, \omega)$ .To simplify the deduction of $P(f, \omega)$, we consider f^3 as an independent variable, and then deduce the relationship as follows:

$$\left.\begin{array}{l} \frac{\partial P(f^3,\omega)}{\partial \omega} = X_1 + Y_1 f^3 \\ \frac{\partial P(f^3,\omega)}{\partial f^3} = X_2 + Y_2\omega \end{array}\right\} \Rightarrow P(f^3, \omega) = Af^3 + B\omega f^3 + C\omega + D \qquad (7)$$
$$p = P(f, \omega) = f^3 + B\omega f^3 + C\omega + D$$

In Eq. (7), A, B, C and D are system-related coefficients, whose values can be determined by experiments in stand-alone environment (see Sect. 5). As we consider the nodes in cloud system homogeneous, the coefficient values of all nodes are the same. Combining Eqs. (3) and (7), we can calculate the energy consumption of the whole system during T time as follows:

$$E(T) = \sum_{i=1}^{N} \int_0^T p_i(t)dt = \sum_{i=1}^{N} \int_0^T [Af_i(t)^3 + B\omega_i(t) f_i(t)^3 + C\omega_i(t) + D]dt \qquad (8)$$

Combining Eqs. (6) and (8):

$$E(T) = \sum_{i=1}^{N} \sum_{j=1}^{M} [Af_i(\Delta t \cdot j)^3 + B\omega_i(\Delta t \cdot j) f_i(\Delta t \cdot j)^3 + C\omega_i(\Delta t \cdot j) + D]\Delta t \qquad (9)$$

$$\begin{aligned} \varepsilon(T) &= E(T)K(T)^{-1} \\ &= \frac{\sum_{i=1}^{N} \sum_{j=1}^{M} [Af_i(\Delta t \cdot j)^3 + B\omega_i(\Delta t \cdot j) f_i(\Delta t \cdot j)^3 + C\omega_i(\Delta t \cdot j) + D]\Delta t}{\sum_{i=1}^{N} \sum_{j=1}^{M} f_i(\Delta t \cdot j)\omega_i(\Delta t \cdot j)\Delta t} \end{aligned} \qquad (10)$$

According to our previous analysis, $f(\Delta t \cdot j)$ and $\omega(\Delta t \cdot j)$ can be easily measured by CPU monitors (software). EC of the cloud computing system during time T can be calculated according to Eq. (10). The calculation approach can calculate not only the EC of the whole cloud computing system, but also that of each node independently. It has a fine granularity and is easy to be implemented.

- As to a heterogeneous cloud, the calculation approach could also be applied for the following reasons and advantages:
- A heterogeneous cloud can be divided into several homogeneous sub-clouds and such sub-clouds should not be many, thus the determination of A, B, C and $D$for all sub-clouds is feasible.
- For a homogeneous sub-cloud, the coefficients can be determined by measuring only one node.
- The coefficients will not change once determined, while measuring power using power-meter is a long-term iterative process. The advantage of the proposed calculation approach is changing the "power monitoring" to "CPU monitoring" by "determining the coefficients once".
- Some optimization approaches are equipments-independent. Therefore, some EC optimization approaches concluded in a homogeneous experimental cloud computing system can also be used in a heterogeneous practical cloud computing system.

Certainly, the EC regularities happened in a homogeneous cloud may not happen or keep consistent in a heterogeneous cloud. The EC regularities of a heterogeneous cloud remain to be well studied in future.

4.3 Extremum analysis

In this section, we will explore the EC optimization approaches from a theoretical point of view. Equation (10) shows that EC is associated with $f\ (t)$ and $\omega\ (t)$. Reducing CPU frequency or usage can decrease power, but may not reduce EC. In our opinion, we only need to consider the EC of a single node because the EC of the whole system is the best when EC of each node reaches the minimum. And within time period T, if at any moment t, $\varepsilon\ (t)$ reaches the minimum, then $\varepsilon\ (T)$ will reach the minimum too. Therefore:

$$\varepsilon(t) = E(t)K(t)^{-1} = \frac{Af(t)^2}{\omega(t)} + Bf(t)^2 + \frac{C}{f(t)} + \frac{D}{f(t)\omega(t)} \tag{11}$$

All the coefficients and variables are positive, so it is easy to know from Eq. (11) that the larger the CPU usage is, the lower the EC is. Since the range of $\omega\ (t)$ is [0,1], EC will achieve the minimum value when $\omega\ (t) = 1$.

Now we study the relationship between $\varepsilon\ (t)$ and $f\ (t)$. The partial differential of Eq. (11) on $f\ (t)$ can be expressed as follows:

$$\begin{aligned} \frac{\partial\varepsilon(t)}{\partial f(t)} &= \frac{K(t)\frac{\partial E(t)}{\partial f(t)} - E(t)\frac{\partial K(t)}{\partial f(t)}}{K(t)^2} = 0 \\ \Rightarrow K(t)\frac{\partial E(t)}{\partial f(t)} &= E(t)\frac{\partial K(t)}{\partial f(t)} \end{aligned} \tag{12}$$

$$\Rightarrow f(t)[3Af(t)^2 + 3B\omega(t)f(t)^2] = Af(t)^3 + B\omega(t)f(t)^3 + C\omega(t) + D$$
$$\Rightarrow f(t) = \sqrt[3]{\frac{C\omega(t) + D}{2[A + B\omega(t)]}}$$

In Eq. (12), when $\omega(t)$ is fixed, there exists an extremum value, and it can be proved to be minimum value. Through the above deduction, we know that when $\omega(t) = 1$, and $f(t) = \sqrt[3]{\frac{C+D}{2(A+B)}}$, the EC achieves its minimum value, i.e., the computer reaches its minimum EC when CPU works at a certain frequency under full utilization. The correctness of this conclusion will be confirmed by the experiments in Sect. 5.

5 Experiments

According to the description in Sect. 4, measuring the energy-consumption (EC) needs to measure the real-time CPU frequency and usage of each node as well as the overall energy consumption, while calculating the EC only needs to measure the real-time CPU frequency and usage of nodes in each homogeneous sub-cloud. To verify the correctness of the proposed EC model, we select 10 PCs and build an experimental cloud computing environment based on Hadoop HDFS and MapReduce. The testbed is described in Table 1.

5.1 Validation of the model

In this section, we design a series of experiments to prove the correctness of the proposed EC model in both stand-alone and cloud environment. First of all, through the data measured during the experiments, we verify the linear relationship between $\omega(t)$ and $p(t)$ when $f(t)$ is fixed, as well as the linear relationship between $p(t)$ and the cube of $f(t)$ when $\omega(t)$ is fixed, and determine the values of A, B, C and D in Eq. (8). Then, we compare the calculated and measured EC. The comparing results show that the EC model proposed is accurate enough.

In stand-alone environment, we adopt multi-threads Pi Monte Carlo Algorithm [25] as the workload, along with the method called "forcing thread to sleep for a certain time", thus making the usage change according to the "sine curve". We can collect sufficient data of usage and corresponding computer power by such method.

The experiment results prove again the linear relationship both between $\omega(t)$ and $p(t)$ when $f(t)$ is fixed, and between $p(t)$ and the cube of $f(t)$ when $\omega(t)$ is fixed, as same as that in our previous work [6]. Therefore, the results have not been displayed here.

According to the analysis in Sect. 4, EC reaches its minimum value when $f(t)^{max} = \sqrt[3]{\frac{C\omega(t)+D}{2[A+B\omega(t)]}}$. Based on the measured data in the above experiment, we obtain the computer-related coefficients values as: $A = 0.15, B = 1.4, C = 15.2, D = 50.8$,then

Table 1 Testbed description

Term	Description
Computer	10 homogeneous computers of Tsinghua Tongfang Z900 (http://www.tongfangpc.com/En/) with Inter Core i5-2300 2.80 GHz, 8GB memory, 1TB hard disk, Mainboard integrates sound card, video and network cards, and energy consumption of a computer exclude that of monitor, mouse and keyboard.
Node	1 admin node as *naming node* and*job tracker* of Hadoop, 9 compute nodes as *datanodes* and *tasktrackers* of Hadoop.
Operation system	CentOS 5.6, Linux 2.6.18 core
CPU working mode	Stand-alone environment: CPU frequency is fixed at user-specific value (*Userspace*) Cloud environment: CPU frequency is smoothly adjusted according to the workload size (*Conservative*)
Programming environment	Java 6
Cloud computing environment	Hadoop 0.20.2
Power-meter	PowerBay power-meter (http://www.northmeter.com/index-en.html), power precision $\pm 0.01 \sim 0.1$W, maximum 2200W, measuring frequency 1.5–3 s
Use case in stand-alone environment	Pi Monte Carlo algorithm [26], control CPU usage by adjusting the number of threads and forcing threads to sleep
MapReduce use cases	***WordCount***[27]: CPU intensive task; light load for network and disk I/O; 100MB, 250MB, 500MB, 750MB and 1,000MB data per node; ***Sort***[28]: I/O intensive task; 100MB, 250MB, 500MB, 750MB and 1,000MB data per node; ***MRBench*** [29]: Interactive task, high concurrent execution of small jobs; fix the bugs appeared during the concurrent execution of Hadoop MRBench [30], and extend sequence-requests model to concurrent–requests model; the number of concurrent requests are 1, 10, 20, 50, 60, 75, 85, 100, 110, 135, 145, 150, 200 and 300;
Δt of ω and f:	Collect data 10 times per second ($\Delta t = 0.1s$)
Measurement units	Atomic Task: K; Energy consumption: *Joule*; Power:. *Watt*, we measure the active power during the experiments; EC: ε, $\varepsilon = Joule/K$.CPU frequency: GHz CPU usage: ω, $0 \leq \omega \leq 1$, ω is sum of CPU utilization of the user and the system, i.e. $1-\text{CPU}_{idle}$

$$f^{max} = \sqrt[3]{\frac{C\omega(t) + D}{2[A + B\omega]}} \approx \begin{cases} 3.7937GHz \ (\omega = 0.25) \\ 3.2508GHz \ (\omega = 0.5) \\ 2.9593GHz \ (\omega = 0.75) \\ 2.7716GHz \ (\omega = 1) \end{cases} \tag{13}$$

The maximum CPU frequency of the experimental computer is 2.8 GHz. And theoretically, the minimum EC occurs when CPU frequency is larger than or is close to the maximum frequency (see Eq. (13). Further analyze, we find that when $\omega = 1$, Eq. (7) can be simplified as $p = P(f, \omega) = (A + B) f^3 + (C + D)$. Here, "$C + D$" is the energy consumption of non-CPU components. For a specific CPU, the values of A

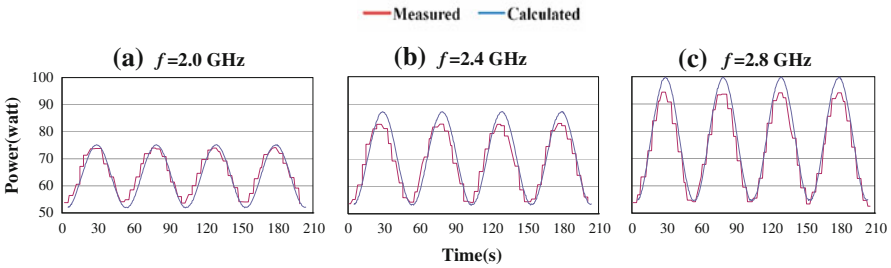

Fig. 1 Comparison of calculated power and measured power

and B are fixed. Therefore, f^{max} in Eq. (13) depends on the value of "$C + D$". That is to say, the larger the energy consumption of non-CPU components is, the faster the CPU has to perform to offset the extra energy consumption and to reduce EC. In all of our experiments, the energy consumption of monitors is excluded from the total energy consumption. If counted in, the coefficient values should be corrected as:$A = 0.15$, $B = 1.4$, $C = 16$, $D = 70$. And when $\omega = 1$, $f^{max} = 3.0272$, which is much larger than the maximum frequency.

Figure 1 shows that the calculated and measured powers are essentially consistent. In this experiment, we collect a large number of ω, therefore, the curve of the calculated power is smooth. However, influenced by the data collection frequency $\left(\Delta t_{power}\right)$ and the collection delay, the curve of the measured power is not smooth and shows step-transition. On the other hand, the peak values of the measured power are slightly lower than that of the calculated power, and the valley values of the measured power are slightly higher than that of the calculated power. It is because that measured power is averaged in Δt_{power} which is larger than Δt_{ω}.Δt_{power} is determined by the power-meter, CPU monitor measures ω in every 0.1 s ($\Delta t_{\omega} = 0.1$s) while the power-meter measures the power in every 1.5 to 3 seconds randomly $\left(\Delta t_{power} = 1.5 - 3\text{s}\right)$. On the other hand, we assume the powers of non-CPU components are constant but in fact they also float a little according to the workload and working environment.

Then, we compare the calculated and measured EC. We have proposed a calculation approach for EC in Sect. 4. If the calculation approach is accurate in stand-alone environment, we can apply it to cloud environment. We use power-meter to measure the real-time computer power, and use CPU monitor (software) to measure the real-time frequency and usage. Then according to formula 7, $P = P(f, \omega) = Af^3 + B\omega f^3 + C\omega + D\,(A = 0.15, B = 1.4, C = 15.2, D = 50.8)$, we calculate the computer power and compare it with the measured power.

Figure 2 shows the comparison result of the calculated and measured EC when ω=0.25, 0.5, 0.75,1, and f = 1.6, 1.8, 2.0, 2.2, 2.4, 2.6, 2.8 GHz. It can be seen that the calculated and measured EC are almost the same. And through calculation, we get an error rate less than 1%. Now, the correctness of the proposed EC model and its calculation methods has been proved. Next, we will continually verify their accuracy in cloud environment.

Finally, we adopt three typical types of tasks, *WordCount* [27], *Sort* [28], and *MRBench* [29] as the workload and make a comparison between the calculated and measured EC in cloud environment under the testbed described in Table 1 (see Fig. 3).

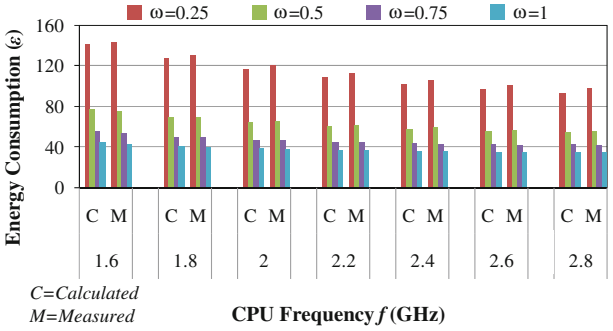

Fig. 2 Comparison of the calculated and measured EC

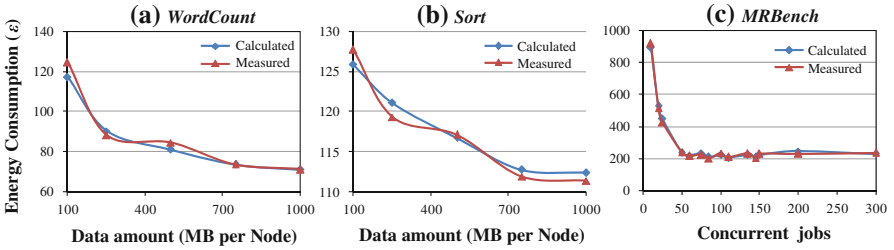

Fig. 3 Comparison of the calculated and measured EC of WordCount, Sort and MRBench under different workloads

In Fig. 3, the calculated values are obtained according to Eq. (10) based on the measured real-time CPU frequencies and usages, while the measured values are obtained according to Eq. (4) based on the measured powers. Four conclusions could be drawn from Fig. 3:

- In general, the measured and calculated values are almost the same with an acceptable statistical error rate ranging from 1% to 5%.
- The EC of CPU-intensive task (*WordCount*) is lower than that of I/O-intensive task (*Sort*), and both of them are lower than that of interactive task (*MRBench*). These results are according with the CPU usage and frequency statistical results in the stand-alone experiment.
- For CPU and I/O intensive tasks, increasing workload (data amount) reduces the EC a little. When workload increases, EC curve rises up slightly. Based on the EC extremum conditions, it is known that the optimization effect is not obvious by adjusting the workload. We must improve the task's execution method in compute node to increase the CPU usage and frequency, and finally reduce the EC.
- For interactive task, EC increases rapidly with the increase of concurrent jobs, and becomes stable when EC reaches the maximum value. For the EC optimization of interactive task, the effect is not obvious simply by increasing the number of concurrent jobs. We should reduce its EC by other approaches such as optimizing the job dispatch or execution algorithm and guarantee a certain concurrency.

5.2 Energy-consumption regularities

In this section, we further study the EC regularities of cloud computing systems under three typical types of tasks: *WordCount* [27], *Sort* [28], and *MRBench* [29]. Experiments show that both CPU usage and CPU frequency are very low whatever the type of the task is.

Firstly, we analyze the real-time f and ω of three types of tasks and calculate their occurrence frequency. For example, *WordCount* costs 771 seconds, and we collect 7716 records of f and ω. By statistics, for the admin node, there are 6195 records of ω in range of (0, 0.1], occurrence frequency is about 80%; for a compute node, there are 1613 records of f=2.8, occurrence frequency is about 20.9%. The statistical results of the admin node and one compute node are shown in Fig. 4. It shows that:

- For the compute node (Fig. 4a, c), we compare the frequency of CPU working under high usage among three tasks: *WordCount* is the highest, *Sort* is lower, and *MRBench* is the lowest. *WordCount* is a CPU-intensive task, so its CPU usage is the highest. The comparison result of the frequency of CPU working at high frequency is the same.
- For the admin node (Fig. 4b, d), all of three tasks are relatively idle. Only in *MRBench* task, CPU usage and frequency of the admin node increases a little.
- The admin node can manage a large number of compute nodes. In our experiment, one admin node only manages nine compute nodes, so its CPU is very idle, further affecting the integral EC of cloud system.
- The CPU usage and frequency statistics of the compute node under three tasks (Fig. 4a, c) are *Positively Skew*, and their values are lower most of the time, resulting in a higher EC. Based on the EC extremum condition deduced previously, what we need to do is to figure out an approach such as improving the scheduling algorithm in cloud environment to make Fig. 4a, c *Negatively Skew*, and CPU usage and frequency are higher most of the time.

We can infer that the CPU is in a waiting state rather than an idle state because the tasks on each node have not finished yet. We propose the Waiting Energy Consumption (WEC) which means computers of a cluster or components of a computer could be in "passive idle" or "busy idle" state because they are waiting for other resources, the EC could be optimized if WEC is reduced. Therefore, the testing system produces the WEC. Through analysis, we infer that there are three reasons leading to the low CPU usage and frequency: insufficient workload (waiting for jobs), the features of resource allocation of a specific task (waiting for resources), and poor parallelism among nodes (waiting for other nodes).

For the first reason, Fig. 3 in Sect. 5.1 shows that EC decreases with the increase of data amount or concurrent jobs, which proves that insufficient workload is one reason causing low CPU usage and frequency. However, Fig. 3 also shows that EC reaches stability quickly with the workload increases, which means that insufficient workload is not the essential reason of low CPU usage and frequency.

For the second reason, we further evaluate the I/O status of three tasks and compare them with CPU status respectively. Fig. 5 shows that the I/O speed (include disk and network I/O) of *Sort* is much higher than that of *WordCount*, and real-time CPU usage

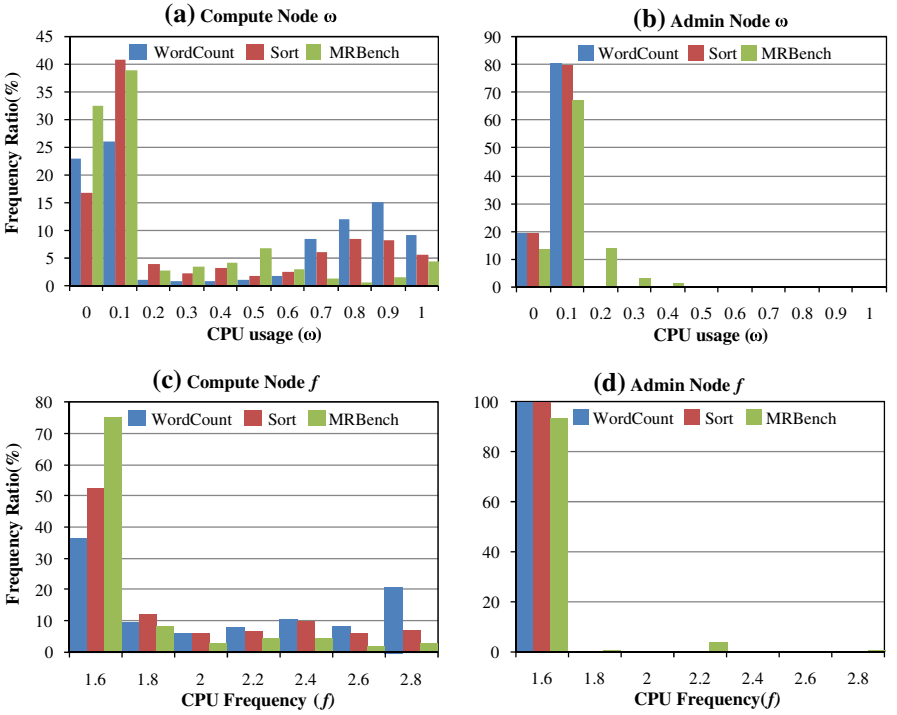

Fig. 4 Statistic comparison on $f(t)$, $\omega(t)$ of *WordCount*, *Sort* under 500MB data per node and *MRBench* under 50 concurrent jobs

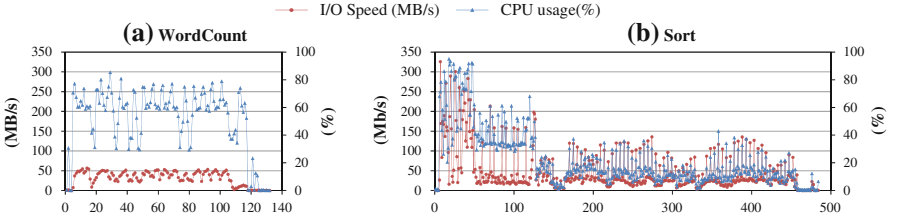

Fig. 5 I/O Speed(t), $\omega(t)$ of *WordCount*, *Sort* under 1,000MB data per node

of *Sort* is much lower than that of *WordCount*; less demand of CPU resource will cause much energy wasted during idle status. The reason why EC of *Sort* is larger than that of *WordCount* is that former allocate less CPU resource than I/O resource real-timely, not the accumulative CPU operations of the former is less than that of the latter. We believe different types of tasks have different demands for resources like CPU and I/O, the more CPU resources are allocated, the less the EC is. *WordCount* is CPU intensive and *Sort* is I/O intensive task because the CPU is relatively busy (I/O is relatively idle) of the former than that of the later, but not the accumulative CPU operations of the former in larger than that of the latter. There are two ways to improve CPU usage: on one hand, we infer that the ratio of resource allocation is

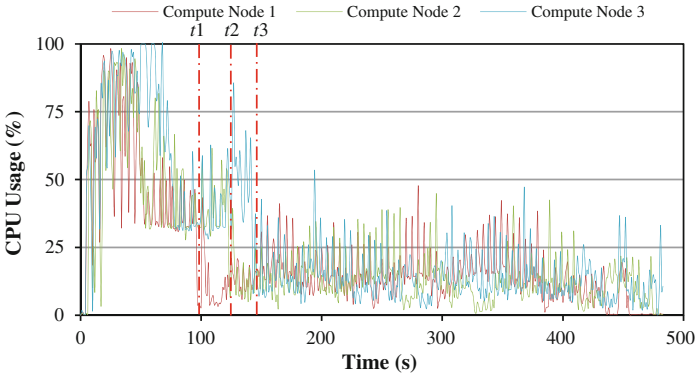

Fig. 6 Real-time CPU usage of admin node and compute node of Sort under 1,000MB data per node

algorithm specific, so we can update the algorithm to improve CPU allocation; for example, introduce index mechanism to reduce the I/O allocation. On the other hand, we can monitor the resource usage of a node real-timely and allocate the appropriate type of task in the multiple tasks environment; for example, when an I/O intensive task is performed on a node, the CPU of this node may be relatively idle, thus we could schedule an CPU intensive task to this node rather than an I/O intensive one.

The third reason is the poor parallelism among nodes. The node produces WEC because it is waiting for the completion of operations on other nodes. Though the MapReduce programming model does provide parallelism, it does not guarantee load balance, and Reduce stage is stared only when all Map stages are finished. Similar with Fig. 5b, Fig. 6 shows the real-time CPU usage of three compute nodes in *Sort* cases. The waiting for the completion of operations on other nodes could be observed by the tendency of real-time CPU usage. As we know, the *Sort* algorithm include two parts: the Hash computation in Map stage is performed on local data, so this stage allocate more CPU resource and less I/O resource, the Sorting computation in Reduce stage involve many data transferring between nodes, so this stage allocate more CPU resource and less I/O resource (refer to Fig. 5b). The node 1, 2 and 3 are in Reduce stage at t_1, t_2 and t_3 time respectively, so that the node 1 waits node 3 during $(t_3 - t_1)$ time and waits node 2 during $(t_2 - t_1)$ time. The poor parallelism is not obviously in Fig. 6 because the experimental data we used is uniformly distributed. We can decrease WEC by making the compute nodes load balanced. Besides, MapReduce deploys computation to data, which means the data distribution will also affect the computation. Therefore, we can figure out a load balancing data layout approach to achieve the load balance of compute node, and finally decrease the energy consumption per computation (ε).

6 Conclusions and future works

This paper proposes an energy-consumption (EC) evaluation model for cloud computing systems. We define the atomic task that used to measure the workload, and

give the mathematical expression of EC; then describe its measurement and calculation approaches, and analyze its extremum conditions. In order to simplify the model, we also improve the relationship between computer power and CPU working state based on previous efforts. We take both CPU frequency and usage into consideration while previous works define power models either under a fixed CPU usage or under a fixed CPU frequency. With this model, one can calculate energy consumption and EC instead of measuring them directly, which needs power meters and other specialized hardware and could be much harder for cloud computing systems.

In addition, we also study the EC regularities of a cloud computing system with the CPU-intensive, I/O-intensive and interactive tasks, and propose the optimization suggestions of reducing WEC based on the analysis results. The following three conclusions could be drawn in this paper:

- EC refers to the consumed energy per task. Low power does not necessarily lead to low EC. Proven by theoretical derivation and experimental verification, EC reaches a minimum when CPU frequency $f(t)^{max} = \sqrt[3]{\frac{C+D}{2(A+B)}}$ and CPU usage is 100%.
- It is difficult to measure the power of each node in the cloud system real-timely by power-meter. The existing models consider power either under a fixed usage or under a fixed frequency while we take both the two factors into consideration. The improved power model could be well adopted to calculate EC without setting up power-meters. Such calculation approach is easy to implement.
- Waiting Energy Consumption (WEC) means computers of a cluster or components of a computer could be in "passive idle" or "busy idle" state because they are waiting for other resources; the EC could be optimized if WEC could be reduced. WEC is caused by insufficient workload, the features of resource allocation of a specific task, and poor parallelism among nodes.
- EC of cloud computing systems is very high under three typical tasks (CPU-intensive, I/O-intensive and interactive tasks); there is a large space to improve. The key point of improving EC is reducing CPU idleness (WEC). It could be achieved to a certain extent by increasing the workload. The most effective approaches are optimizing the scheduling and execution algorithm of tasks, reducing I/O operations and network communication, or ensure the load balance among nodes.

In our future works, we will further improve this model. As to the calculation of $E(T)$, so far we do not take the memory and disk into consideration yet. Many literatures such as [23–25] show that the power of disk and memory is constant or has little influence on the workload. Thus, the calculation approach of $E(T)$ we proposed is right, for the energy consumption of disk and memory has been included in the constant part. Certainly, we will still further validate this assertion in our future works. On the other hand, the definition of $K(T)$ in our model is not comprehensive and still exists some insufficiencies. Atomic task can only partly be represented by CPU usage. For example, when the workload is too heavy for CPU and causes resources competition, CPU usage may be very high while the performance is very poor. And for an I/O insensitive computing, CPU usage is low while disk usage is high, in which case I/O operations should be included in the atomic task. Therefore, we will further

improve the definition of $K(T)$ in our future works, taking memory, disk and network into consideration.

Acknowledgments This paper is supported by the National Natural Science Foundation of China under Grant No. 61202088; the Fundamental Research Funds for the Central Universities N110417002; the Natural Science Foundation of Liaoning Province under Grant No.200102059.

References

1. Pettey C (ed) (2007) Gartner estimates ICT industry accounts for 2 percent of global CO_2 emissions. http://www.gartner.com/it/page.jsp?id=503867
2. Fan XB, Weber WD, Barroso LA (2007) Power provisioning for a warehouse-sized computer. In: Proceedings of the 34th international symposium on computer architecture, pp 13–23. ACM, San Diego. http://doi.acm.org/10.1145/1250662.1250665
3. Heath T, Diniz B, Carrera EV, Jr. WM, Bianchini R (2005) Energy conservation in heterogeneous server clusters. In: Proceedings of the ACM SIGPLAN symposium on principles and practice of parallel programming, pp 186–195. ACM, Chicago. http://doi.acm.org/10.1145/1065944.1065969
4. Economou D, Rivoire S, Kozyrakis C, Ranganathan P (2006) Full-system power analysis and modeling for server environments. In: Proceedings of the workshop on modeling, benchmarking, and simulation
5. Mudge TN (2001) Power: a first-class architectural design constraint. IEEE Comput 34(4):52–58. http://doi.ieeecomputersociety.org/10.1109/2.917539
6. Song J, Li TT, Yan ZX, Na J, Zhu ZL (2011) An energy-efficiency model and measuring approach for cloud computing. J Softw 23(2):200–214
7. GridMix program. Available in Hadoop source distribution: src/benchmarks/gridmix
8. A Benchmark for Hive, PIG and Hadoop. http://issues.apache.org/jira/browse/HIVE-396
9. Sort program. Available in Hadoop source distribution: src/examples/org/apache/hadoop/examples/sort
10. The HiBench Benchmark Suite: characterization of the MapReduce-based data analysis
11. DFSIO program. Available in Hadoop source distribution: src/test/org/apache/hadoop/fs/TestDFSIO
12. Bahsoon R (2010) A framework for dynamic self-optimization of power and dependability requirements in Green Cloud architectures. In: 4th European conference on proceedings of software architecture. Springer, Copenhagen, pp 510–514. http://dx.doi.org/10.1007/978-3-642-15114-9_52
13. Tsirogiannis D, Harizopoulos S, Shah MA (2010) Analyzing the energy efficiency of a database server. In: Proceedings of the ACM SIGMOD international conference on management of data. ACM, Indianapolis, pp 231–242. http://doi.acm.org/10.1145/1807167.1807194
14. Kumar K, Lu YH (2010) Cloud computing for mobile users: can offloading computation save energy? IEEE Comput 43(4):51–56. http://doi.ieeecomputersociety.org/10.1109/MC.2010.98
15. Orgerie AC, Lefèvre L, Gelas JP (2008) Save watts in your grid: green strategies for energy-aware framework in large scale distributed systems. In: Proceedings of the 14th international conference on parallel and distributed systems. IEEE, Melbourne, pp 171–178. http://dx.doi.org/10.1109/ICPADS.2008.97
16. Elnozahy EN, Kistler M, Rajamony R (2002) Energy-efficient server clusters. In: Proceedings of power-aware computer systems, second international workshop. Springer, Cambridge. http://dx.doi.org/10.1007/3-540-36612-1_12
17. Abdelsalam HS, Maly K, Mukkamala R, Zubair M, Kaminsky D (2009) Analysis of energy efficiency in clouds. In: Computation world, pp 416–421
18. Chen YY, Das A, Qin WB, Sivasubramaniam A, Wang Q, Gautam N (2005) Managing server energy and operational costs in hosting centers. In: Proceedings of the international conference on measurements and modeling of computer systems, pp 303–314. ACM, Banff. http://doi.acm.org/10.1145/1064212.1064253
19. Snowdon D, Ruocco S, Heiser G (2005) Power management and dynamic voltage scaling: myths and facts. In: Proceedings of the 2005 workshop on power aware real-time computing
20. Li B, Li JX, Huai JP, Wo TY, Li Q, Zhong L (2009) EnaCloud: an energy-saving application live placement approach for cloud computing environments. In: Proceedings of the IEEE international conference on cloud computing, CLOUD 2009. IEEE, Bangalore, pp 17–24. http://doi.ieeecomputersociety.org/10.1109/CLOUD.2009.72

21. Buchbinder N, Jain N, Menache I (2011) Online job-migration for reducing the electricity bill in the cloud. In: Proceedings of NETWORKING 2011–10th international IFIP TC 6 networking conference. Springer, Valencia, pp 172–185. http://dx.doi.org/10.1007/978-3-642-20757-0_14
22. Shi YX, Jiang XH, Ye KJ (2011) An energy-efficient scheme for cloud resource provisioning based on CloudSim. In: Proceedings of 2011 IEEE international conference on cluster computing (CLUSTER). IEEE, Austin, pp 595–599. http://doi.ieeecomputersociety.org/10.1109/CLUSTER.2011.63
23. Chen QW, Grosso P, Veldt KVD, Laat CD, Hofman RFH, Bal HE (2011) Profiling energy consumption of VMs for green cloud computing. In: Proceedings of IEEE ninth international conference on dependable, autonomic and secure computing. IEEE, Sydney, pp 768–775. http://doi.ieeecomputersociety.org/10.1109/DASC.2011.131
24. Kogge P, Bergman K (2008) Borkar S et al. Technology challenges in achieving exascale system, ExaScale computing study
25. Wirtz T, Ge R (2011) Improving MapReduce energy efficiency for computation intensive workloads. In: Proceedings of green computing conference and workshops, pp 1–8
26. Module for Monte Carlo Pi. http://math.fullerton.edu/mathews/n2003/montecarlopimod.html
27. WordCount program. Available in Hadoop source distribution: src/examples/org/apache/ hadoop/ examples/WordCount
28. Sort program. Available in Hadoop source distribution: src/examples/org/apache/hadoop/ examples/ sort
29. MRBench program. Available in Hadoop source distribution: src/examples/org/apache/ hadoop/ mapred/MRBench
30. https://issues.apache.org/jira/browse/MAPREDUCE-2398