Wireless Pers Commun (2015) 81:661–683
DOI 10.1007/s11277-014-2151-y

A Novel Human Computer Interaction Aware Algorithm to Minimize Energy Consumption

P. K. Gupta · G. Singh

Published online: 1 November 2014

Abstract In this paper, we have developed a novel algorithm to minimize the energy consumption of the computer system. We have discussed various scenarios to understand the human interaction with computer system like, when a computer system is in idle mode or the user of the system has left it inactive, however as both of the cases are not very significant with reference to the energy consumption as well as heat dissipation. In another scenario, we have utilized the central processing unit of the computer system to its full extent and evaluated its performance in idle and active mode. In addition to this, we have also evaluated the memory performance using the proposed algorithm.

Keywords Algorithm · Energy consumption · Human computer interaction · Performance evaluation · Sustainable computing

1 Introduction

Recently, there has been an obvious explosion of technological growth, particularly in the Information and Communication Technology (ICT) industry and several technological breakthroughs have taken place, and more are yet to come. Earlier, people used Central Processing Unit (CPU) speeds of 386, 486 or 900 MHz for their work and currently, the minimum CPU speed available is in the GHz range. However, the rate at which ICT devices are being produced is proportional to increase in the energy consumed and heat dissipated by these devices, which poses the problem of power crisis and the exacerbation of the greenhouse gas problem

P. K. Gupta (✉)
Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat, Solan 173 215, India
e-mail: pradeep1976@yahoo.com

G. Singh
Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Waknaghat, Solan 173 215, India
e-mail: ghanshyam.singh@juit.ac.in

and global warming. It has been proposed that by the year 2020 [1] there will be an enormous demand for electricity by developing countries, such as India and China, and if a sufficient number of energy-generation sources are not developed to meet this demand, the situation will become worse. Therefore, the vision of a sustainable planet and the minimization of the energy consumed by the computer systems motivated us to examine the energy-sustainable Human Computer Interaction (HCI) methods [2]. These methods are useful for solving the energy-consumption issue by the computer systems and being environmentally friendly. It is believed that in the future there will be a great demand for energy-sustainable software.

The computer systems are supposed to use a variable amount of power when they are switched on, which depends on their configuration and various software processes running on them [3,4]. In computer systems, the operating system acts like a manager and controls a computer's hardware and software, therefore it is considered as a major source of energy consumption. To manage the energy consumption, there are several predefined power schemes at one's disposal. These power-saving options are responsible for switching a computer system to different states, such as Standby mode, Sleep mode, and monitor and hard disk drive (HDD) shutdown, depending on the inactivity period defined by the power scheme of the operating system. However, most of the time, the users rely on the default settings of power schemes, which allow for only up to 20 % in power savings [5]. However, two major issues that arise from this situation are as follows:

(i) Are these available power schemes sufficient for the energy-sustainable computing?
(ii) What type of energy-sustainable HCI methods could be applied for sustainable development?

In response to these issues, this paper contributes the following:

- We have proposed a novel HCI algorithm that estimates the percentage of total CPU usage by the various processes running on the machine over a given time interval. The experimental results reveal that this technique is more effective in handling computer system fully, and minimizing the energy consumption using sustainable development.
- Furthermore, we have implemented this technique on a Windows platform as it is considered one of the most preferred operating system. However, the use of JAVA has made the implementation of this technique very user-friendly and begins as soon as the user logs in to the system, therefore users need not configure it all the time.
- We have introduced the concept of a repository that keeps a record of all installed applications on the computer system and is maintained by the user. This repository requires a one-time configuration by the user to classify the various running applications into two groups, *Hibernate* and *Shutdown*, which helps the operating system to adopt the mode when it is turned-off.
- Finally, we have evaluated the performance of proposed HCI algorithm using CPU as well as memory performance analysis and find that the performance of the proposed HCI algorithm is much better over existing techniques in the power schemes, with no memory leakage or degradation in CPU speed.

The remainder of the paper is organized into various sections and subsections that address the energy-sustainable methods to minimize the power consumption of computer systems. In Sect. 2, we have summarized various reported literature related to energy consumption using approaches like hardware, software as well as energy sustainable techniques. In Sect. 3, we present the proposed HCI framework and algorithm, which addresses the issue of human inactivity and idleness of the computer system. The proposed HCI algorithm continuously monitors the total CPU usage (%) of various running processes on a system as well as human

activity on the system. The Sect. 4 presents the experimental setup and evaluation of the proposed HCI algorithm. Section 5 discusses the results obtained thoroughly and considers the various scenarios under which the proposed HCI algorithm is executed. Furthermore, Sect. 6 discusses the performance evaluation of the proposed HCI algorithm and analyzes the various results obtained after CPU and memory performance analysis. Finally, Sect. 7 concludes the work and recommends future directions.

2 Related Work

To date, much work has been performed with respect to the issue of energy consumption by the computer systems. In this section, we describe only those areas that are related to our proposed work and emphasize only software-based approaches. These software-based approaches can be classified into various categories as per the available literature.

2.1 Dynamic Power Management-Based Approaches

Dynamic power management (DPM) is an approach by which one can reduce power consumption by placing system components into different states. DPM also refers to the selective shutting or slowing down of computer system components that are idle for a long time or rarely used. These approaches can be classified into three subcategories: *stochastic, predictive* and *time-out* approaches. Benini et al. [6,7] explored several approaches to system-level DPM and modeled a power-managed system as a set of interacting power-manageable components controlled by a power manager and then analyzed the DPM implementation issue in electronic systems. Later, they used the stochastic approach to power-managed systems and categorized the set of components into different states based on their performance and power-consumption levels. They have created a power-management policy to decide when to perform component state transitions and which transitions should be performed, depending on system history, work-load, and performance constraints. In [8], Li has investigated the multiprocessor environment with dynamically variable voltage and speed and analyzed the problem of minimizing schedule length with energy-consumption constraints and the problem of minimizing energy consumption with schedule-length constraints; moreover, they compared the performance of algorithms with optimal solutions analytically and validated their results experimentally. Wang et al. [9] investigated the smart power-saving scheme *PowerSleep* for servers with the aim of reducing static power consumption using DPM. To minimize the mean power consumption, they have chosen the execution speed for servers with dynamic voltage scaling and sleep periods when placing the servers into sleep mode with DPM. Huang et al. [10] focused on implementing the DPM in hard real-time systems and proposed online algorithms to change the mode of the system. They considered three modes for the system, active mode, standby mode and sleep mode, and stated that based on the controller's decision the device can be switched to any mode to reduce energy consumption. Using these algorithms, they predicted the next moment for mode switching. Abbasian et al. [11] introduced an adaptive method for DPM that is based on wavelet forecasting theory, which allows for very accurate modeling of system components with non-stationary behavior. They also stated that this model can be used to capture the local information of a system very accurately and achieved 95 % accuracy in their results when predicting the state of a HDD. Hwang and Wu [12] focused on the need to switch off a computer system running in idle or sleep mode and presented a predictive system-shutdown method to avoid sleep mode operations and thereby save energy when running event-driven applications. They used static

power management and DPM techniques to define and detect the sleep modes and idle period. Srivastava et al. [13] conducted an extensive analysis of various system shutdown approaches and proposed a predictive system shutdown strategy for event-driven applications in portable devices. They developed two predictive formulas: one based on general regression-analysis techniques to compare the length of an upcoming off period with that of a previous one and the other obtained by the observation of on-off activity. Jiang et al. [14] investigated the timeout policy for DPM and formulated a semi-Markov control process model to optimize and analyze the performance of the timeout DPM policy. They also stated that the timeout policy is equivalent to a stochastic policy in terms of power performance tradeoffs, and this relationship is expressed as a mathematical formula.

2.2 Energy-Sustainable Approaches

In recent years, sustainability-based approaches have received a very good response when designing any energy-saving approach. These types of approaches directly focus on the environment and are categorized under sustainable computing. This section lists the various energy-sustainable approaches developed by various researchers over time. Wang et al. [15] investigated the existing power models by re-evaluating them on multi-core computer systems (MCSs). They proposed a two-level power model that estimates the power consumption for each core on MCSs. Furthermore, based on this model, they designed and implemented a software power analyzer by using only one performance-monitoring counter and frequency information from the CPUs to identify the power behavior of MCSs. Chen et al. [16] investigated the adverse effects of dynamic voltage and frequency scaling and running a virtual machine on system performance using methods used for energy conservation in server consolidation. They proposed a new application-aware approach by introducing a new set of metrics: CPU gradients that predict the impact of changes in CPU frequency. These gradients are simple models and represent the local point derivatives of the end-to-end response time with respect to the resource parameters. They later used these CPU gradients for performance-aware energy conservation by deploying energy controllers. Naumann et al. [17] addressed the consumption of power and resources by ICT and presented a software-based model of *GREENSOFT*. This model addresses the issue of energy reduction and resource consumption in ICT and the use of ICT to contribute to sustainable development. Chen et al. [3] relied on operating-system-level power-saving strategies to minimize the energy consumption of computer systems and introduced the concept of process-level power management in their tool pTopW. This tool captures real-time power-consumption data at the process level to make critical power-saving decisions. They then introduced a power-aware system module called Energy Guard, which is used to terminate the abnormal behavior of an application to curb energy consumption.

2.3 Tool-Based Approaches

This section presents the various tool-based approaches proposed by the researchers to estimate and minimize the energy consumed by computer systems. Here, we address a few tools in addition to tools discussed in the previous section. Do et al. [18] developed a tool, *pTop*, to estimate the amount of energy consumed by each application in a system. This is basically a process-level profiling tool that runs parallel to services of the operating system at the kernel level and provides energy-consumption data. Gurumurthi et al. [19] investigated the existing power simulators for their design and found that they are manly targeted for particular hardware such as CPU and memory and do not capture the interaction between

other components. The *SoftWatt* tool developed by Gurumurthi et al. considers the disk drives in addition to the CPU and memory and quantifies the power behavior of applications and operating systems. This tool also locates the power hot spots in system components and identifies the power-hungry processes in operating systems. Banerjee and Agu [20] introduced the tool *PowerSpy*, which tracks the battery power consumed by different running threads and various I/O devices attached to the device. This tool requires no additional hardware to monitor the power consumption of a device. Chen et al. [21] investigated a user-level simulator at the micro-architectural and memory level and found that operating system activity is not modeled in them. They introduced the tool *SimWattch*—a system-level simulation tool and a flexible user-level simulation tool for predicting performance and power dissipation.

2.4 Other Useful Approaches

This section describes various useful approaches used to minimize the energy consumption of a computer system. These approaches are based on different methods and algorithms proposed by various researchers. Ramanathan and Gupta [22] investigated the problem of power management in an embedded system and introduced online algorithms to manage the power of these systems by shutting off parts of the system when they are not being used and turning them back on when required. Li and John [4] characterized the power behavior of an operating system with respect to a number of applications to understand the operating system energy profile and to estimate its runtime energy dissipation. All estimates were based around the operating system because it is considered a major software application and dissipates a significant amount of power in executing and running applications. Bircher and John [23] introduced the microprocessor performance counter for measuring the power consumption of computer system. They considered the trickle-down effect of performance events in a microprocessor. The results obtained after implementing the developed tool show an accurate estimate of total system power, with an average error of $<9\,\%$ per subsystem across the considered workload. Cameron [24] investigated various power-management issues and their threatening impacts. He stated that a number of techniques and tools are available to calculate and optimize the power consumption of hardware components. Thus, he introduced and demon strated formal approach to modeling how software affects power dissipation. Cho et al. [25] took advantage of the dynamic-voltage and frequency-scaling technique used to minimize the power consumption of a system and introduced an algorithm for embedded devices that uses system scaling and sets the deadline of a task as per the value provided by the user. Their study resulted in an algorithm that can reduce energy consumption by $45\,\%$.

3 Proposed HCI Framework and Algorithm

As illustrated in the preceding section, there are various approaches that have been used to minimize the energy consumption of the computer systems. The proposed HCI algorithm focuses over CPU utilization, which is based on snapshots of total CPU utilization. If there is any work/processing being performed on a computer system, the CPU of the system must be in use and the percentage of total CPU usage should be greater than zero, otherwise, it should be equal to zero. In [2], Gupta et al. also suggested that a processor not performing any operation can be kept in sleep or hibernation mode to reduce the energy consumption of the system. The proposed framework is shown in Fig. 1.

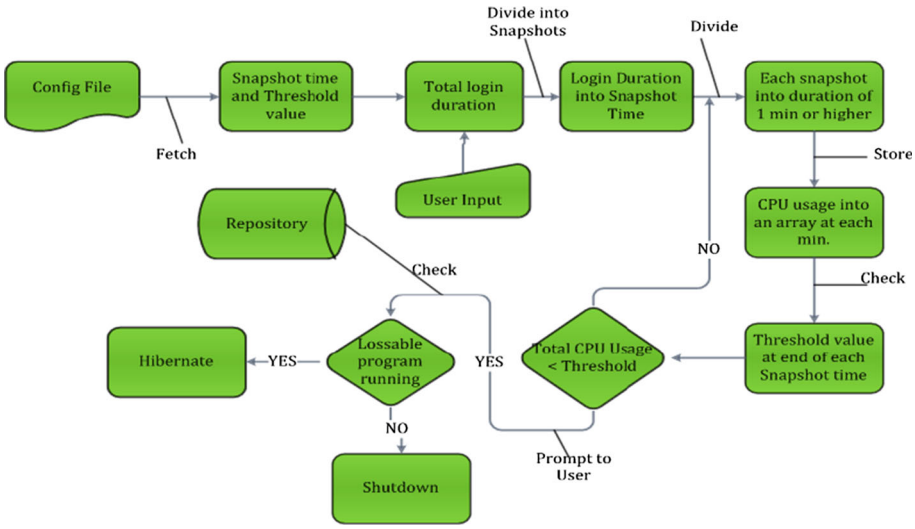

Fig. 1 Framework of proposed HCI technique

This proposed HCI framework takes input from two sources:

- First, in the form of a configuration file where a user has to define the value of the snapshot time (S) and its threshold limit (τ). Here, the threshold limit represents the percentage of total CPU usage, which should be kept at its minimum. In the proposed HCI framework, we have considered the range $20 \leq \tau \leq 10$ because during this time very few application processes in addition to system processes are running on the computer system and CPU remains in idle state.
- Second, the input is also taken from the user when logging in to the computer system, the user must select the roughly estimated time (L) he/she will be working on the computer system. Thus, it follows that:

$$\text{Snapshot time } (S) < \text{Total Login duration } (L) \tag{1}$$

If any difference is found in the values of S and L, then the proposed HCI algorithm as given in Sect. 3.1 will consider the total login duration (L) as the snapshot time (S). In the next step, the total login duration (L) will be divided equally into the small chunks of snapshot times $S, 2S, 3S, \ldots, nS$. This value is determined as follows:

$$\textit{Total number of chunks of snapshot times} = \frac{\text{Total login duration } (L)}{\text{Snapshot time } (S)} \tag{2}$$

For example, if a user inputs a total login duration $L = 15$ min and the value of the snapshot time defined in configuration file $S = 5$ min, according to Eq. (2) there will be only three chunks of snapshot times, $S = 5$ min, $2S = 10$ min and $3S = 15$ min. This shows that after each interval of 5 min. the total CPU usage will be checked. In the next step, to increase the accuracy of the system's decision, we have defined the concept of snapshots for obtaining the percentage of total CPU usage after each interval of time (X), which is also known as the sleeping time within a given chunk of snapshot time (S).

$$X = S/n \tag{3}$$

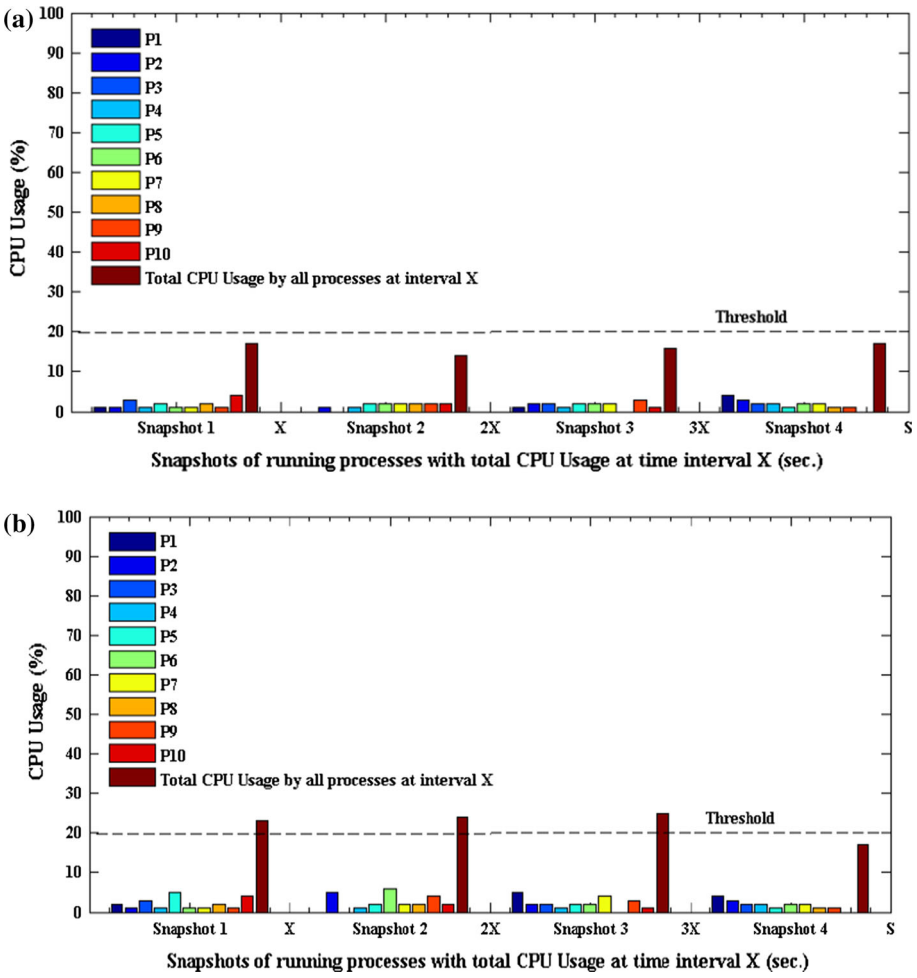

Fig. 2 Snapshots of total CPU usage (%): **a** below threshold value and **b** above threshold value

where n represents the total number of CPU usage (%) snapshots in each chunk. In the proposed technique, the value of X is 1 min, which means that after each 1-min of time interval the percentage of total CPU usage will be stored in an array. These stored values are compared with the predefined value of the threshold limit. If it is obtained, the percentage of total CPU usage consumed by all processes running on the system is below the defined threshold value, as shown in Fig. 2a, which indicates that the user is not actively working on the system or very few processes are running on the system, then the proposed HCI algorithm will make its decision accordingly to minimize the energy consumption. Furthermore, if it is found that the percentage of total CPU usage exceeds the threshold value in any interval of time for a given chunk, as shown in Fig. 2b, which means that the CPU of the machine is being utilized and therefore the user or some application processes are continuously working, the next snapshot chunk will be traced out and the percentage of total CPU usage will be stored to determine the idle period of the CPU. The decision to "Hibernate" or "Shutdown"

before the system goes off will be completely based on the configuration of the repository and the various applications running on the operating system at that time.

3.1 HCI Algorithm

On the basis of previously discussed HCI framework (in previous section), this section represents the HCI algorithm and collects the various CPU processing data for analysis and result purpose. The functioning of this algorithm is divided into three easy steps. Starting with the various used variables in the algorithm, step-1 is associated with the initialization of variables like CPU Usage (ξ), threshold limit for CPU usage (τ), snapshot duration (S), time to compare the snapshots (s), total login duration (L) and an array of CPU usage (A). In step-2, timer T1 and T2 gets started according to the value provided by the user. Here, the timer T1 represents the Total login duration given by the user whereas the timer T2 represents the duration time to compare the snapshots of total CPU percentage. The timer T2 gets started internally and its value is predefined in the configuration file. In step-3, which is a major step in this algorithm that computes the total CPU usage (%) and store this value into an array and compares the array value for taking the decision when to shut down or hibernate the computer system.

Symbols used in this algorithm

(i) ξ—For CPU usages
(ii) τ—The threshold for CPU usage
(iii) S—The snapshot duration.
(iv) s—Time to compare the snapshots.
(v) L—The total login duration time
(vi) X—The sleeping time
(vii) A—An array of CPU Usage
(viii) T1 and T2—Timers
(ix) C—Counter

Step-1

Initialize ξ, τ, s, S, L and A
$\tau \leftarrow$ Take the value from configuration file which is stored by the user while settings of power saver module.
$s \leftarrow$ Take the value from configuration file which is stored by the user for comparing the various snapshots.
$S \leftarrow$ Take the value from configuration file which is stored by the user while settings of the power saver module.
$L \leftarrow$ Take the input from the user during login to the system.
A creates an array of size s. $C \leftarrow 0$

Step-2

Starts two timer threads T_1 and T_2;
T_1: expires after $L \times 60 \times 1{,}000$ ms
T_2: expires after each $1 \times 60 \times 1{,}000$ ms and sleeps for X time.

Step-3

IF T_2 expires **THEN**
a) $X \leftarrow$ 1 x 60 x 1000
b) Calculate $\xi \leftarrow$CPU (P_1) + CPU (P_2) + ...+ CPU(P_n)
c) $A[C] \leftarrow \xi$
d) **IF** $C == s$ **THEN**
IF $\forall\, A[0], A[1], \ldots A[S-1] < \tau$ **THEN**
Invoke user prompt to find user availability on the system;
IF any loss-able program is running, (checks with repository)
HIBERNATE the computer system
ELSE
SHUT DOWN the computer system
END
ELSE
C $\leftarrow$ 0
END
e) Increment $C \leftarrow C + 1$ and T_2 should sleep for X (ms)
END
END

4 Experimental Methodology

To characterize the behavior of the HCI algorithm herein proposed, we first describe the experiment setup used to verify the proposed algorithm. Then, we will describe how we performed the experiment and compared the various states of computer systems with respect to the percentage of total CPU used by the various running processes. This section describes the experiment setup and the evaluation of the algorithm under the executed workload.

4.1 Experiment Setup

We have used a cluster of 15 machines to execute proposed HCI algorithm and thus record various snapshots of the percentage of total CPU being used during different intervals of time.

The experimental platform as stated above, we have used cluster of IBM Thinkcentre desktops. Table 1 summarizes the configurations of cluster machines that are used in this experiment to evaluate the proposed HCI algorithm. The simulated processors is Intel Core i3 2100 with core speed of 1,597.8 MHz and stock frequency 3,100 MHz. The L1 data cache is 8-way set associative, with size 2×32 KBytes, whereas the L2 cache size is 2×256 KBytes and L3 cache is 12-way set associative with 3 MByte size, for all the cluster machines. The processors operate at voltages of 0.986 V. The memory type is DDR3 with 2 GB size, single bank and 665.7 frequencies for all cluster machines. The HDDs used to store the percentage of total CPU usage data and for the performance evaluation of cluster machines are from Seagate, measuring 320 GB SATA and running at 7,200 rpm.

4.2 Evaluation

We have used the software *StressMyPC* [26] for a thorough evaluation for the proposed HCI algorithm. This software is freely available on the internet and checks the CPU and HDD of the system by executing some algorithms. We have used this software on all the cluster machines and executed it so that the CPU of the computer system could remain busy for a certain period of time to determine the accuracy of proposed algorithm. The snapshot time of the percentage of total CPU usage is recorded for each second in snapshot chunks lasting

Table 1 Cluster configuration

Component	Specification parameters	Cluster of 15 machines
Make	Manufacturer	IBM Think centre
	Type	Desktop
CPU	Name	Intel Core i3 2100
	Code name	Sandy Bridge
	No. of processor	1
	Processor specification	Intel® Core™ i3-2100 CPU @ 3.10 GHz
	Package	Socket 1155 LGA (0 × 1)
	Technology	32 nm
	Core speed	1,597.8 MHz
	Stock frequency	3,100 MHz
	Core VID	0.986 V
	Max TDP	65 W
	Multiplier × FSB	16.0 × 99.9 MHz
	Number of cores	2
	Number of threads	4
	Instruction set	MMX, SSE (1, 2, 3, 3S, 4.1, 4.2), EM64T, VT-x, AVX
	L1 Data cache	2 × 32 KBytes, 8-way set associative, 64-byte line size
	L2 Cache	2 × 256 KBytes, 8-way set associative, 64-byte line size
	L2 Cache	3 MBytes, 12-way set associative, 64-byte line size
Memory	Memory Type	DDR3
	Size	2,048 MBytes
	Number of banks	1
	Voltage	1.5 V
	Frequency	665.5 MHz
Disk	Size	320 GB SATA
	Manufacturer	Seagate
	Speed	7,200 rpm

one minute each. We also recorded the snapshots of the total CPU usage when there was no process running on the system or when there was a process running completely within the computer system's memory and there was no or very limited interaction with the CPU. We have used NetBeans [27] to implement the proposed algorithm. Then, using the profiler available in NetBeans IDE, we have evaluated the performance of the proposed algorithm by monitoring the memory and CPU. The results obtained from this profiling have been discussed in the performance evaluation chapter.

5 Results

This section describes the various results obtained after executing the proposed HCI algorithm in a real environment. We used various scenarios—usage scenario1, usage scenario 2 and an internal scenario—to evaluate the proposed HCI algorithm. Usage scenarios1 and 2 also describe an environment that includes the system and operating parameters. These scenarios were implemented on all cluster machines. Specifically, we executed commonly used software and obtained the results for the following scenarios.

5.1 Usage Scenario 1

This scenario represents the idle functioning of computer systems, during which no work is carried out. The user must simply login to the system and execute certain software, as per

Table 2 Usage scenario 1

System parameters	Cluster of 15 machines
Operating system	32-bit, Windows 7 Professional
Software executed by the user	NIL
Total number of running processes	45
Status	Idle
Operating parameters for algorithm	
Total login duration (L)	20 min
Snapshot comparison time (s)	1 min
Threshold value (τ)	20 %

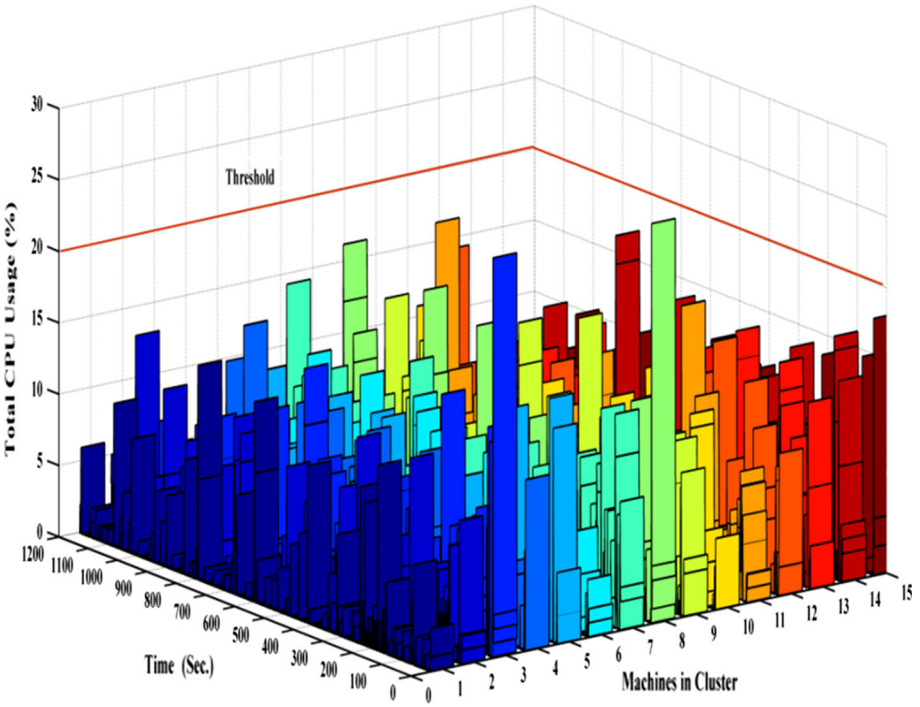

Fig. 3 Total CPU usage (%) for idle computer systems in the cluster

Table 2, and leave the system inactive due to some unknown reason. This is the most common user practice around the globe. The running computer system not only consumes energy but also completely depends on the operating system's power scheme settings to switch to idle mode or hibernate. Thus, this is when our proposed algorithm can be initialized, and the system could be powered off long before the determined power scheme is implemented.

The results obtained for the scenario described above are presented in Fig. 3 for all cluster machines. For $L = 20$, s = 1 and $\tau = 20$, Fig. 3 is equally divided and represent a total of S = 20 chunks of snapshots with lasting 60 s each for all cluster machines. We recorded the various snapshots for the percentage of total CPU usage in a file for each second. To gain greater clarity and to continue with experiment up to the last moment of total login duration, we canceled the T2 timer whenever it was invoked at the end of each snapshot. Thus, at the end of each minute, the final snapshot value of the total CPU usage percentage was recorded, which is slightly higher than the previous one because the execution of the timer

Table 3 Usage scenario 2

System parameters	Cluster of 15 machines
Operating system	Windows7
Software executed by the user	StressMyPC (Nenad Hrg)
Total number of running processes	45 + 1 (StressMyPC)
Status	Active state
Operating parameters for algorithm	
Total login duration (L)	20 min
Snapshot comparison time (s)	1 min
Threshold value (τ)	20 %

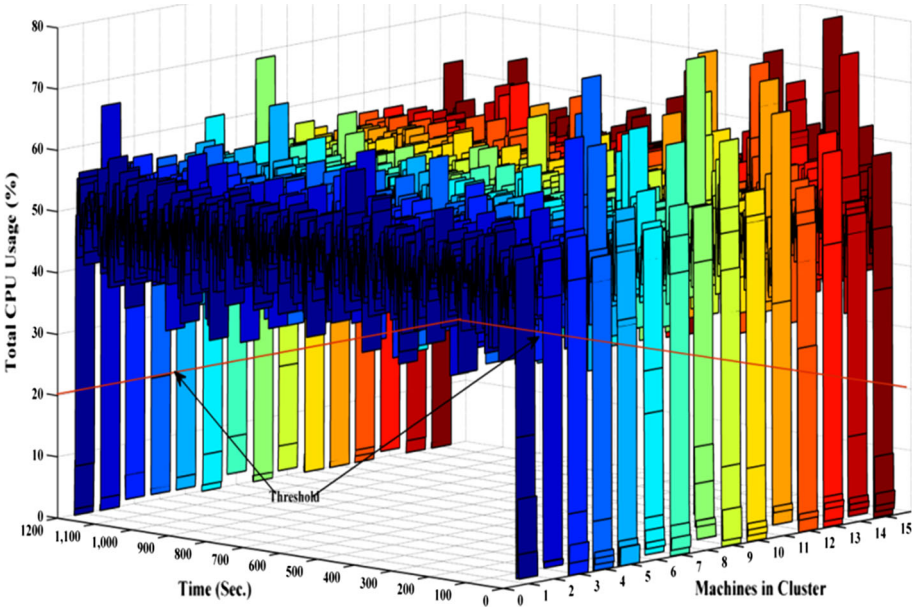

Fig. 4 Total CPU usage (%) for active computer systems in the cluster

event also increases the percentage of total CPU usage. However, the peaks in the middle of each snapshot are only due to the various processes running on the system. Therefore, if there is the percentage of total CPU usage is higher than the threshold value, then HCI algorithm will check the next snapshot and record the percentage of total CPU usage for each second in that snapshot.

5.2 Usage Scenario 2

This scenario represents the active functioning of the cluster machines, which means that there is a continuous pumping of data from the CPU side or instructions are being continuously fed from the user side to the system, which keeps the CPU busy, this is unlike the previous scenario, where user instructions were limited to the cancelation of the T2 timer. To keep the CPU busy, we used the *StressMyPC* program with other programs such as Microsoft Office and Internet Explorer. Table 3 presents the various system and operating parameters under which the user logs in to the system.

However, no change in the operating parameters observed, and the proposed algorithm is executed under the same environment as it was in the previous scenario. Figure 4 illustrates

the results obtained for all cluster machines. The processing represented by Fig. 4 is similar to that of Fig. 3; because we kept the operating parameters remain same. The only difference from the previous scenario is that the CPU of all cluster machines is actively involved in processing and we can easily notice that the percentage of the total CPU reaches up to 80% in active state. Figure 4 clearly shows that in the active state for all the cluster machines average percentage of the total CPU usage always remains above 50 % during the execution of the program *StressMyPC* as this program keeps the CPU busy all the time.

5.3 Internal Scenario

This scenario represents the various results, obtained and used by the proposed HCI algorithm make the decision whether to keep the computer system running or not. As we have seen in the previously discussed usage scenarios 1 and 2, the percentage of total CPU usage was recorded for each second in chunks lasting one minute each. In these various recorded percentage of total CPU usages for each chunk, we focus ourselves on the maximum recorded value of total CPU usage for the each cluster machines, so that it could be find whether there is any peak of running processes in each snapshot chunk that breaches the defined threshold limit or not. Figure 5 represents the maximum-recorded value of CPU usage for each cluster machines, in each chunk of snapshot respectively and compared with each other and totally based upon the value of snapshot time S.

For the previously discussed usage scenarios 1 and 2, we have considered the value of snapshot comparison time s = 1 min, which means each chunk of snapshot is treated separately and there will be no comparison. Furthermore, based on recorded maximum values for CPU usage, the proposed algorithm decides when to activate or cancel the T2 timer. The following Table 4 represents the recorded value of maximum CPU usage (%) for all cluster machines (M1–M15) in both the modes active (A) and idle (I) at a given time interval of up to 20 min.

From Table 4, we can derive the various graphs for both the states of each cluster machines. As shown in Fig. 5a when the machine is in idle state then the snapshots of total percentage of maximum CPU usage by cluster machine M1 always remains below the defined threshold value, whereas in active state percentage of maximum CPU usage for each interval of time remains above the defined threshold and represents that the continuous processing is being done on the cluster machine M1.

In Fig. 5b, the snapshots taken for total percentage of CPU usage for cluster machine M2 at each interval of time are very much similar to the snapshots presented in Fig. 5a and remains always below the defined threshold in its idle state and above the threshold in its active state during the supplied login duration time.

In, Fig. 5c for cluster machine M3, there is one snapshot of time interval at the start for idle state that breaches the defined threshold value and no user prompt will be invoked by proposed HCI snapshot algorithm and the next interval will be checked whereas in active state percentage of total CPU usage remains always above the defined threshold.

Further, the snapshots of percentage of total CPU usage for cluster machines M4, M5, M6, and M7 in Fig. 5d–g as shown above, always remains below the defined threshold in their idle state and above the defined threshold in their active state. Here, in the idle states, user prompt is invoked after each interval of time to check whether user is available or not on the machine.

In Fig. 5h, for cluster machine M8, utilization of CPU is very much similar to the machine M3 as there is one snapshot of percentage of total CPU usage in its idle state which breaches the defined threshold value and the next time interval is checked by the algorithm whereas in active state percentage of total CPU usage always remains above the threshold.

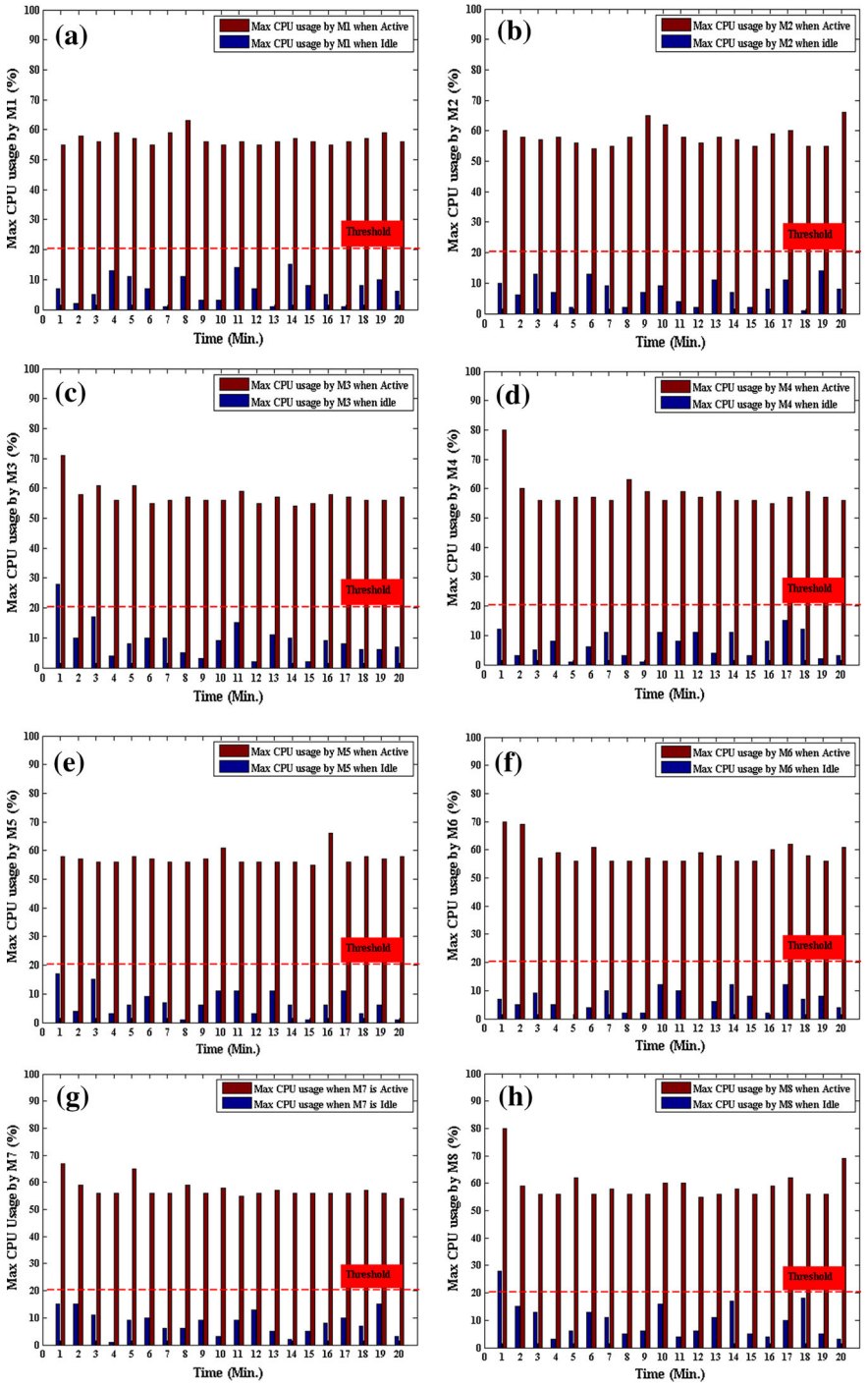

Fig. 5 Maximum CPU usage (%) by cluster machines for each snapshot when active and idle. **a** M1, **b** M2, **c** M3, **d** M4, **e** M5, **f** M6, **g** M7, **h** M8, **i** M9, **j** M10, **k** M11, **l** M12, **m** M13, **n** M14, **o** M15

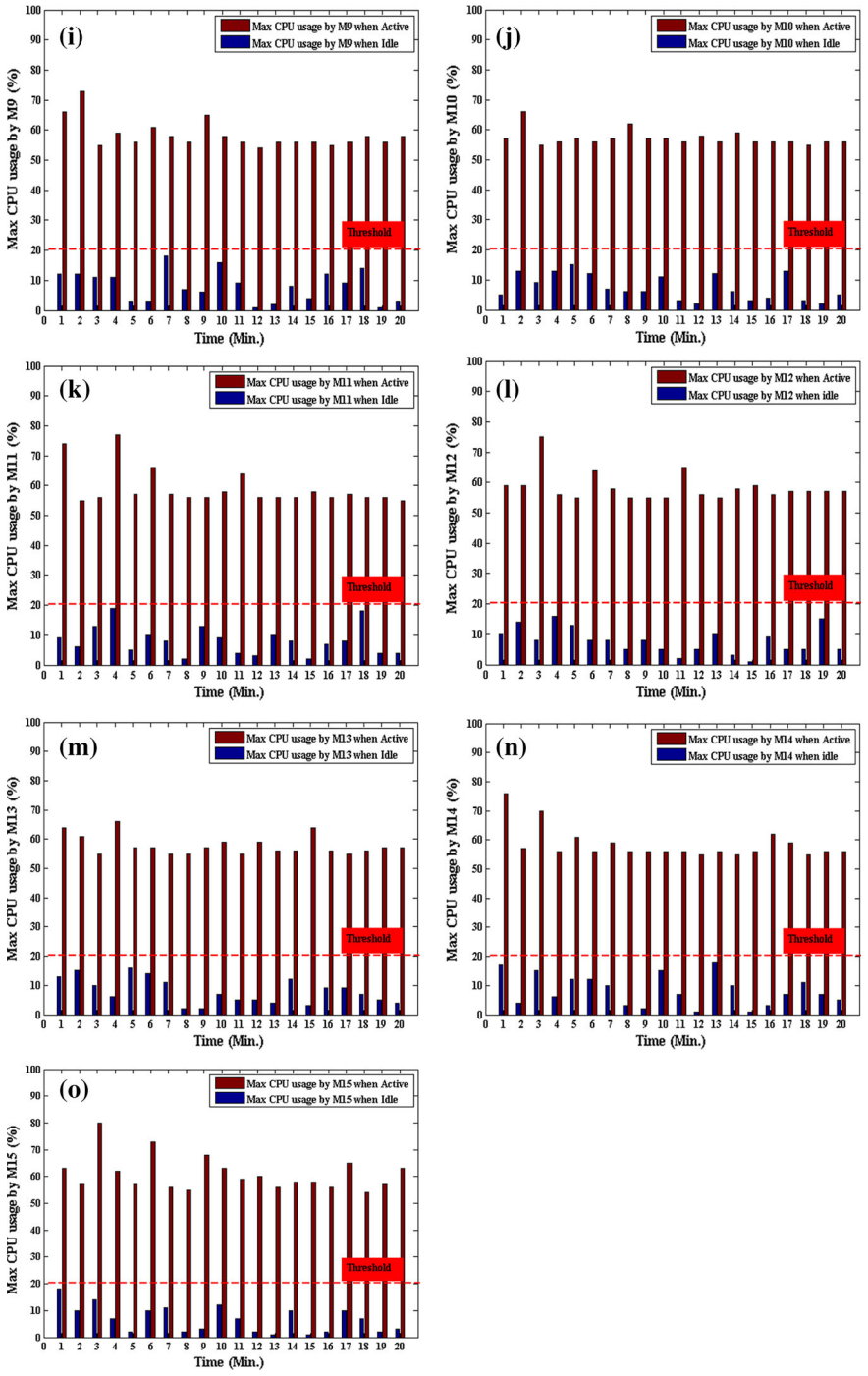

Fig. 5 continued

Table 4 Maximum CPU usage (%) by each cluster machines (M1–M15) in active (A) and in idle (I) mode up to 20 min

Snapshot time interval (in min)	M1		M2		M3		M4		M5		M6		M7		M8		M9		M10		M11		M12		M13		M14		M15	
	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I
1	55	7	60	10	71	28	80	12	66	12	57	5	74	9	59	10	64	13	76	17	63	18	58	17	70	7	67	15	80	28
2	58	2	58	6	58	10	60	3	73	12	66	13	55	6	59	14	61	15	57	4	57	10	57	4	69	5	59	15	59	15
3	56	5	57	13	61	17	56	5	55	11	55	9	56	13	75	8	55	10	70	15	80	14	56	15	57	9	56	11	56	13
4	59	13	58	7	56	4	56	8	59	11	56	13	77	19	56	16	66	6	56	6	62	7	56	3	59	5	56	1	56	3
5	57	11	56	2	61	8	57	1	56	3	57	15	57	5	55	13	57	16	61	12	57	2	58	6	56	0	65	9	62	6
6	55	7	54	13	55	10	57	6	61	3	56	12	66	10	64	8	57	14	56	12	73	10	57	9	61	4	56	10	56	13
7	59	1	55	9	56	10	56	11	58	18	57	7	57	8	58	8	55	11	59	10	56	11	56	7	56	10	56	6	58	11
8	63	11	58	2	57	5	63	3	56	7	62	6	56	2	55	5	55	2	56	3	55	2	56	1	56	2	59	6	56	5
9	56	3	65	7	56	3	59	1	65	6	57	6	56	13	55	8	57	2	56	2	68	3	57	6	57	2	56	9	56	6
10	55	3	62	9	56	9	56	11	58	16	57	11	58	9	55	5	59	7	56	15	63	12	61	11	56	12	58	3	60	16
11	56	14	58	4	59	15	59	8	56	9	56	3	64	4	65	2	55	5	56	7	59	7	56	11	56	10	55	9	60	4
12	55	7	56	2	55	2	57	11	54	1	58	2	56	3	56	5	59	5	55	1	60	2	56	3	59	0	56	13	55	6
13	56	1	58	11	57	11	59	4	56	2	56	12	56	10	55	10	56	4	56	18	56	1	56	11	58	6	57	5	56	11
14	57	15	57	7	54	10	56	11	56	8	59	6	56	8	58	3	56	12	55	10	58	10	56	6	56	12	56	2	58	17
15	56	8	55	2	55	2	56	3	56	4	56	3	58	2	59	1	64	3	56	1	58	1	55	1	56	8	56	5	56	5
16	55	5	59	8	58	9	55	8	55	12	56	4	56	7	56	9	56	9	62	3	56	2	66	6	60	2	56	8	59	4
17	56	1	60	11	57	8	57	15	56	9	56	13	57	8	57	5	55	9	59	7	65	10	56	11	62	12	56	10	62	10
18	57	8	55	1	56	6	59	12	58	14	55	3	56	18	57	5	56	7	55	11	54	7	58	3	58	7	57	7	56	18
19	59	10	55	14	56	6	57	2	56	1	56	2	56	4	57	15	57	5	56	7	57	2	57	6	56	8	56	15	56	5
20	56	6	66	8	57	7	56	3	58	3	56	5	55	4	57	5	57	4	56	5	63	3	58	1	61	4	54	3	69	3

In Fig. 5i–o as shown below for cluster machines M9, M10, M11, M12, M13, M14, and M15 represents the percentage of total CPU usage that remains always below the defined threshold in the idle state and above the threshold in active state for all the cluster machines.

From the above figures, we can easily find out that when there is no processing on the machines or machines are in idle state than the total CPU usage always remains below the defined threshold and the same condition is detected. The proposed HCI algorithm which tries to find the users availability by invoking the prompt after each defined interval of time period whenever it is found that the percentage of total CPU usage is below the threshold.

6 Performance Evaluation

This section describes the various performance measures used to evaluate our proposed algorithm in a real environment. We used the profiler available in NetBeans to examine our algorithm with respect to performance-related issues. By using the profiler, one can easily analyze the performance of a system's CPU and memory, monitor application, object allocation, garbage collection and memory leakage, etc. We profiled the proposed algorithm to analyze the CPU and memory performance using the basic settings with default overhead.

6.1 Analyze CPU Performance

By using this performance measurement, we were able to analyze our proposed HCI algorithm and obtain data related to its performance, including the time required to execute a code fragment within a method and the number of times that particular method was invoked. This analysis is shown in Fig. 6a–c, which show the call tree class of total CPU usage, login duration and various threads created to record the total CPU usage for each minute.

In the obtained results, we have analyzed the CPU performance up to 20 min, using only all project related classes. In Fig. 6a various call tree methods are shown. Here, User Thread-8 continues till the end of login-duration and User Thread-7 stops its working as the framework settle down. In Fig. 6b, the user threads are created for monitoring the user activity on the computer system for each minute and whenever the percentage of total CPU usage is found below the threshold a popup window get activated to know the user status on the machine. This pop-up window also created some user thread for a smaller duration of time as in this performance evaluation we have always given our consent in "YES" whenever the pop-up window was invoked and executed the process till the end of login-duration. In Fig. 6c some user threads are expanded to show their detailed functioning. In this figure, we have expanded the user Thread-10, 12, 18 and 14, all these threads are invoked 60 times and recorded the percentage of total CPU Usage for each minute in a file.

For all the created user threads in each minute, we found no thread which over utilizes the CPU. Here, all the methods are executed for their assigned time limit and proposed HCI algorithm invokes the popup window whenever the percentage of total CPU usage found below the threshold for that interval as we have considered the snapshots timing for a minute.

6.2 Analyze Memory Performance

To verify the performance of our algorithm, memory must also be considered. By using this measurement, we can analyze memory usage according to the various objects that have been allocated space in it, memory leakage, threads and loaded class. Using VM telemetry, we analyzed these memory measures and obtained the results shown in Fig. 7a–c.

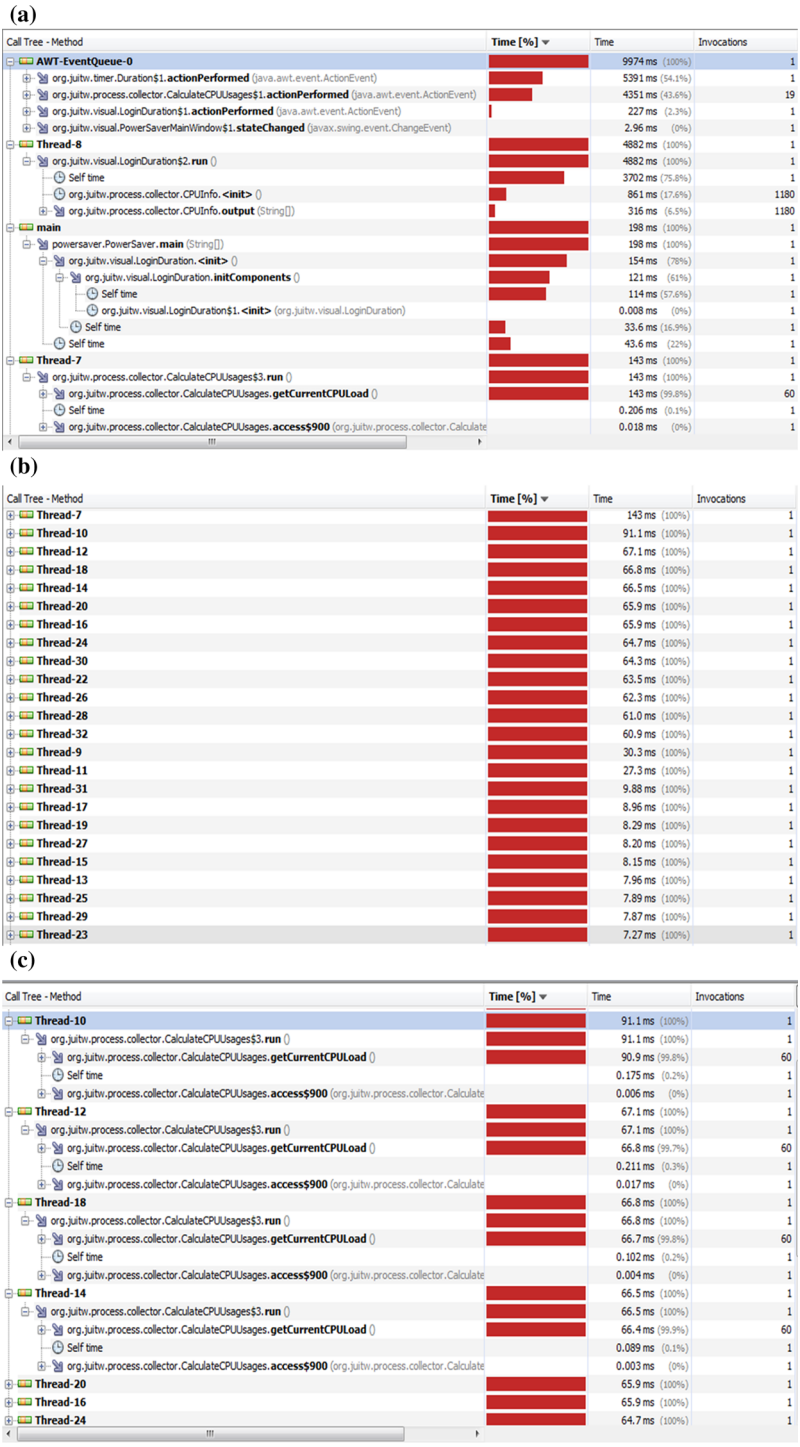

Fig. 6 Analysis of CPU performance. **a** Call tree methods for AWT-EventQueue-0, Thread-8, main, and Thread-7. **b** Various user threads to monitor user activity. **c** Few expanded user threads with methods

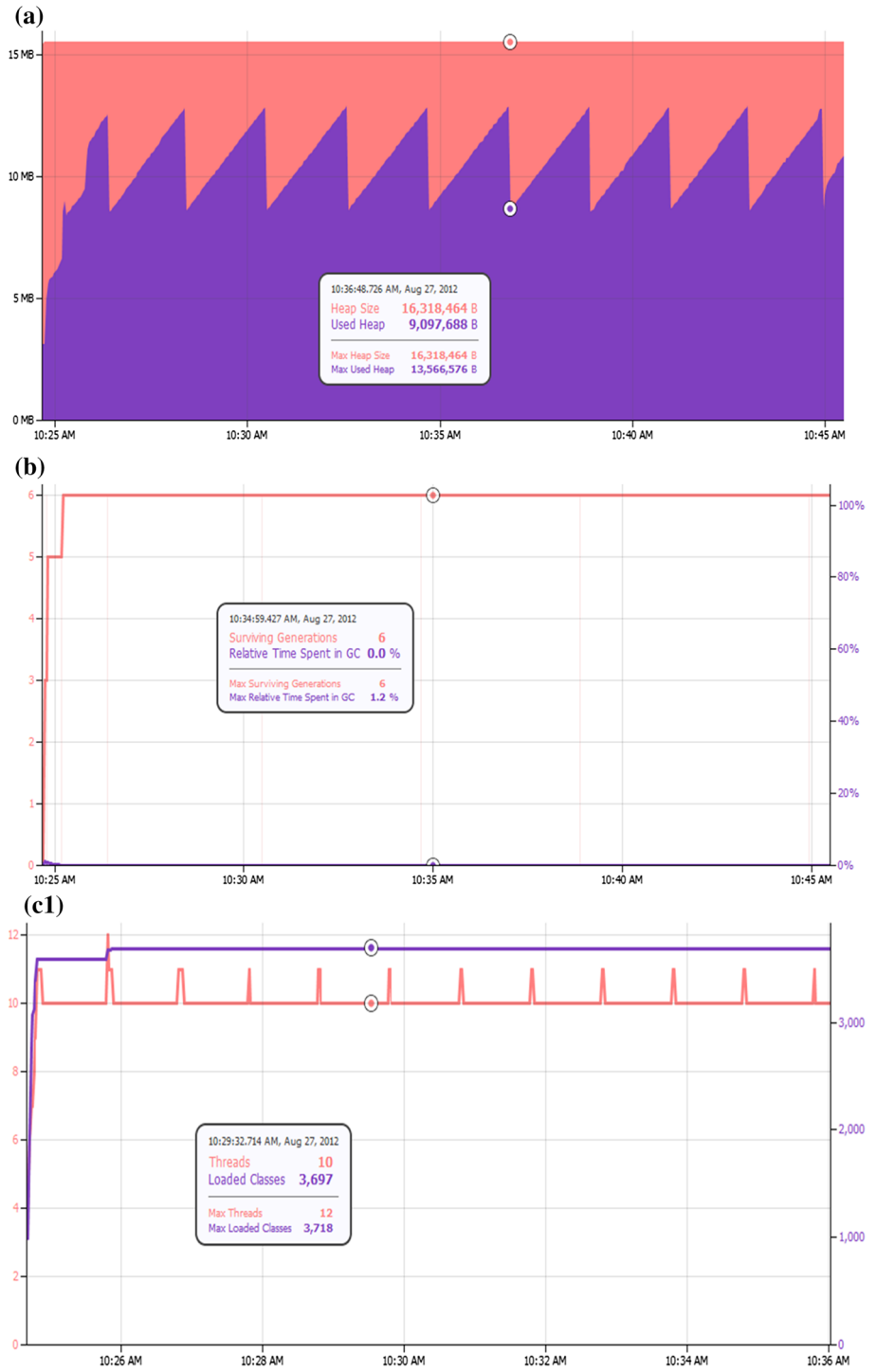

Fig. 7 Analyzing memory performance: **a** Memory–Heap. **b** Memory–GC, and **c1**, **c2** thread/loaded classes

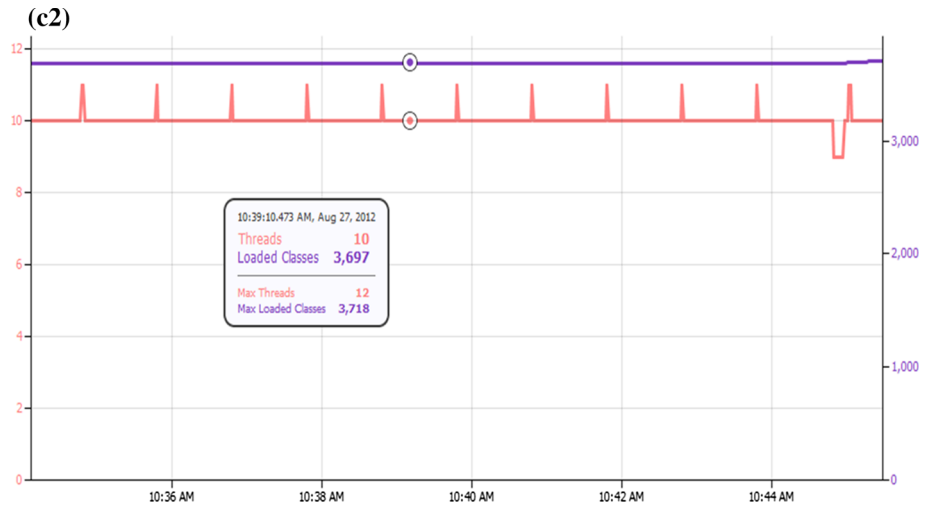

Fig. 7 continued

Here, Fig. 7a shows that our proposed HCI algorithm uses a minimum of 9.0 MB and a maximum of 13 MB of heap out of a total available heap size of 16.318 MB. This increase in the size of the heap is due to the expiration of the T2 timer, which occurs after each minute. Moreover, whenever garbage is collected by the garbage collector, we see a steep fall in the graph and heap is released; this process continues until the end of the login duration. Here, from Fig. 7b one can find that once the framework gets initialized the total number of surviving generations becomes constant and remains at 6 for the proposed HCI algorithm, till the login-duration ends. So, there is no problem of memory leakage in proposed framework. Moreover, maximum relative time spent in garbage collection is 1.2 % only.

Figure 7c1, c2 shows the various threads and loaded classes in the memory during the execution of the proposed HCI algorithm. Here, various peaks are shown after each minute of time interval, which represents that the algorithm is performing a check throughout the interval to find the percentage of total CPU usage and this utilization is found always below the threshold for each snapshots then an inner timer in the form of popup window to find the user's activity on the machine gets started otherwise this inner timer thread gets cancelled. The purpose of popup window is to know the user consensus on the machine. Here, to perform the performance evaluation till the end of login-duration we have pressed "YES" whenever the popup window was activated. These figures also indicates that immediately after the execution of the algorithm, the number of loaded classes becomes stable in the memory

7 Conclusion

In this paper, we have proposed an HCI algorithm to minimize the energy consumption by computer systems. The main objective of the proposed technique is to structure concepts, strategies, and activities to design an effective algorithm. The proposed algorithm is implemented using the various total CPU utilization snapshots, which is specially designed for desktop computer systems, though the same can also be used for laptop systems. This algorithm constantly tracks the total CPU usage of all running processes on a computer system,

and whenever it is found that the computer system is in idle mode or the user of the system has left the computer inactive, the proposed algorithm switches the state of the system from idle or inactive to hibernate or shutdown. The working principal of the proposed algorithm is based on determining whether the system is idle or in an inactive state because theoretically at that time the percentage of total CPU usage should be zero; otherwise, as indicated by our results, it should be below the threshold limit defined by the user to enable the system make the decision to hibernate or shutdown. However, through the experiment and performance evaluation, we have obtained that the proposed algorithm is very effective and for some cases which claim upto 90 % reduction in total energy consumption. To enhance the accuracy of proposed HCI algorithm, we have evaluated the CPU and memory performance. The results obtained from this evaluation are very impressive, showing no slowdown in CPU speed or memory leakage. The results show that the proposed algorithm meets the goals theoretically as well as practically for designing a complete energy-sustainable tool and suggest that the changes can be incorporated into the power schemes of operating systems. We hope this algorithm will help researchers/scientists to develop a comprehensive solution for the energy efficient HCI computing.

Acknowledgments The authors are sincerely thankful to the potential reviewers for their fruitful comments and suggestions to improve the quality of the manuscript.

References

1. Technology outlook 2020. http://production.presstogo.com/fileroot/gallery/DNV/files/preview/9ec457bc750b4df9e040007f0100061c/9ec457bc75094df9e040007f0100061c.pdf. Accessed June 15, 2014.
2. Gupta, S. K. S., Mukherjee, T., Varsamopoulos, G., & Banerjee, A. (2011). Research directions in energy-sustainable cyber–physical systems. *SUSCOM, 1*(1), 57–74. doi:10.1016/j.suscom.2010.10.003.
3. Chen, H., Li, Y., & Shi, W. (2012). Fine-grained power management using process-level profiling. *SUSCOM, 2*(1), 33–42. doi:10.1016/j.suscom.2012.01.002.
4. Li, T., & John, L. K. (2003). Run-time modeling and estimation of operating system power consumption. *SIGMETRICS Performance Evaluation Review, 31*(1), 160–171. doi:10.1145/885651.781048.
5. Blackburn, M., & Collins, G. (2009). Why power schemes are not enough. http://www.computerworlduk.com/cmsdata/whitepapers/3208279/powerprofile.pdf. Accessed May 20, 2012.
6. Benini, L., Bogliolo, A., & De Micheli, G. (2000). A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 8*(3), 299–316. doi:10.1109/92.845896.
7. Benini, L., Bogliolo, A., Paleologo, G. A., & De Micheli, G. (1999). Policy optimization for dynamic power management. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 18*(6), 813–834. doi:10.1109/43.766730.
8. Li, K. (2008). Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed. *IEEE Transactions on Parallel and Distributed Systems, 19*(11), 1484–1497. doi:10.1109/TPDS.2008.122.
9. Wang, S., Liu, J., Chen, J.-J., & Liu, X. (2011). Power sleep: A smart power-saving scheme with sleep for servers under response time constraint. *IEEE Journal on Emerging and Selected Topics In Circuits and Systems, 1*(3), 289–298. doi:10.1109/JETCAS.2011.2167532.
10. Huang, K., Santinelli, L., Chen, J.-J., Thiele, L., & Buttazzo, G. C. (2010). Adaptive power management for real-time event streams. In *Proceedings of IEEE 15th Asia and South Pacific design automation conference* (pp. 7–12). doi:10.1109/ASPDAC.2010.5419928.
11. Abbasian, A., Hatami, S., Afzali-Kusha, A., Nourani, M., & Lucas, C. (2004). Event-driven dynamic power management based on wavelet forecasting theory. In *Proceedings of the 2004 international symposium on circuits and systems ISCAS* (pp. 325–328). doi:10.1109/ISCAS.2004.1329528.
12. Hwang, C.-H., & Wu, A. C. H. (2000). A predictive system shutdown method for energy saving of event-driven computation. *ACM Transactions on Design Automation of Electronic Systems, 5*(2), 226–241. doi:10.1109/ICCAD.1997.643266.

13. Srivastava, M. B., Chandrakasan, A. P., & Brodersen, R. W. (1996). Predictive system shutdown and other architecture techniques for energy efficient programmable computation. *IEEE Transactions on VLSI Systems, 4*(1), 42–55. doi:10.1109/92.486080.
14. Jiang, Q., Xi, H. S., & Yin, B. Q. (2010). Adaptive optimisation of timeout policy for dynamic power management based on semi-Markov control processes. *IET Control Theory and Applications, 4*(10), 1945–1958. doi:10.1049/iet-cta.2009.0467.
15. Wang, S., Chen, H., & Shi, W. (2011). SPAN: A software power analyzer for multicore computer systems. *SUSCOM, 1*(1), 23–34. doi:10.1016/j.suscom.2010.10.002.
16. Chen, S., Joshi, K. R., Hiltunen, M. A., Schlichting, R. D., & Sanders, W. H. (2011). Using CPU gradients for performance-aware energy conservation in multitier systems. *SUSCOM, 1*(2), 113–133. doi:10.1016/j.suscom.2011.02.002.
17. Naumann, S., Dick, M., Kern, E., & Johann, T. (2011). The GREENSOFT Model: A reference model for green and sustainable software and its engineering. *SUSCOM, 1*(4), 294–304. doi:10.1016/j.suscom.2011.06.004.
18. Do, T., Rawshdeh, S., & Shi, W. (2009). ptop: A process-level power profiling tool. In *Proceedings of the 2nd workshop on power aware computing and systems (HotPower'09)*.
19. Gurumurthi, S., Sivasubramaniam, A., Irwin, M. J., Vijaykrishnan, N., Kandemir, M., Li, T., & John, L. K. (2002). Using complete machine simulation for software power estimation: The SoftWatt approach. In *Proceedings of the eighth international symposium on high-performance computer architecture (HPCA.02)* (pp. 1–10). doi:10.1109/HPCA.2002.995705.
20. Banerjee, K. S., & Agu, E. (2005). PowerSpy: Fine-grained software energy profiling for mobile devices. In *Proceedings of IEEE international conference on wireless networks, communications and mobile computing* (pp. 1136–1141). doi:10.1109/WIRLES.2005.1549572.
21. Chen, J., Dubois, M., & Stenström, P. (2007). Simwattch: Integrating complete-system and user-level performance and power simulators. *IEEE Micro, 27*(4), 34–48. doi:10.1109/MM.2007.73.
22. Ramanathan, D., & Gupta, R. (2000). System level online power management algorithms. In *Proceedings Design, Automation, Test in Europe* (pp. 606–611). doi:10.1109/DATE.2000.840847.
23. Bircher, W. L., & John, L. K. (2007). Complete system power estimation: A trickle-down approach based on performance events. In *IEEE International Symposium on Performance Analysis of Systems and Software* (pp. 158–168). doi:10.1109/ISPASS.2007.363746
24. Cameron, G. (2005). Modelling software driven power consumption. In *Instrumentation and Measurement Technology Conference (IMTC), Ottawa, Canada* (pp. 2082–2087). doi:10.1109/IMTC.2005.1604540.
25. Cho, K.-M., Liang, C.-H., Huang, J.-Y., & Yang, C.-S. (2011). Design and implementation of a general purpose power-saving scheduling algorithm for embedded systems. In *IEEE conference on signal processing, communications and computing (ICSPCC)* (pp. 1–6). doi:10.1109/ICSPCC.2011.6061645.
26. StressMyPC. http://www.softwareok.com/?seite=Microsoft/StressMyPC. Accessed June 23, 2012.
27. NetBeans IDE 7.1.2. http://netbeans.org/community/news/show/1556.html. Accessed May 05, 2012.

P. K. Gupta received Ph.D. degree in Computer science and Engineering from the Jaypee University of Information Technology, Waknaghat, Solan, India in 2012. He graduated in Informatics and Computer Engineering from Vladimir State University, Vladimir, Russia, in 1999 and received his M.E. degree in Informatics and Computer Engineering in 2001 from the same university. He has been associated with academics more than twelve years in different institutions like BIT M.Nagar, RKGIT Ghaziabad in India. Currently, he is working as Assistant Professor (Sr. Grade) with the Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat, Solan, India. He has supervised a number of B.Tech/M.Tech/M.Phil. theses from various universities of India. Dr. Gupta has served as a General Co Chair for 2013 IEEE Second International Conference on Image Information Processing. He has organized more than 30 workshops on LINUX, PHP and MySQL, LaTeX, Python, SciLab and three Faculty development programs (FDPs) on LaTeX and SciLab. His research interests include Storage Networks, Green Computing, Software Testing, Cloud Computing, and Internet-of-Things. He has worked as a reviewer for several reputed Journals and Conferences. He is a Member of IEEE, Professional member of ACM, Life Member of CSI and Life member of Indian Science Congress Association.

G. Singh received Ph.D. degree in Electronics Engineering from the Indian Institute of Technology, Banaras Hindu University, Varanasi, India, in 2000. He was associated with Central Electronics Engineering Research Institute, Pilani, and Institute for Plasma Research, Gandhinagar, India, respectively, where he was Research Scientist. He had also worked as an Assistant Professor at Electronics and Communication Engineering Department, Nirma University of Science and Technology, Ahmedabad, India. He was a Visiting Researcher at the Seoul National University, Seoul, South Korea. At present, he is Professor with the Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Wakanaghat, Solan, India. He is an author/co-author of more than 180 scientific papers of the refereed Journal and International Conferences. His research and teaching interests include RF/Microwave Engineering, Millimeter/THz Wave Antennas and its Applications in Communication and Imaging, Next Generation Communication Systems (OFDM and Cognitive Radio), and Nanophotonics. He has more than 14 years of teaching and research experience in the area of Electromagnetic/Microwave Engineering, Wireless Communication and Nanophotonics. He has supervised various Ph. D. and M. Tech. theses. He has worked as a reviewer for several reputed Journals and Conferences. He is author of two books "Terahertz Planar Antennas for Next Generation Communication" and "MOSFET Technologies for Double-Pole Four-Throw Radio-Frequency Switch" published by Springer.