I.J. Intelligent Systems and Applications, **2013, 01, 1-15**
Published Online December 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2013.01.01

Energy-Sustainable Framework and Performance Analysis of Power Scheme for Operating Systems: A Tool

P. K. Gupta
Department of Computer Science and Engineering & IT, Jaypee University of Information Technology, Waknaghat, Solan-173 234, India
Email: pradeep1976@yahoo.com

G. Singh
Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Waknaghat, Solan-173 234, India
Email: ghanshyam.singh@juit.ac.in

Abstract— Recently, an Information and Communications Technology (ICT) devices has become more user-friendly, which raised the problem of power dissipation across the globe and computer systems are one among them. This emerging issue of power dissipation has imposed a very significant issue on the system and software design. The concept of 'green computing' gaining popularity and is being considered as one of the most promising technology by the designers of Information Technology (IT) industry, which demonstrate the environmentally responsible way to reduce the power consumption and maximize the energy efficiency. In this paper, we have proposed an energy sustainable framework of the power schemes for operating systems to reduce the power consumption by computer systems and presented a Green Power tool (GP tool). This tool is designed using JAVA technology, which requires least configuration to make a decision for reducing the power consumption and proposed *Swift mode* algorithm, allows users to input the working time of their choice then after the end of time algorithm starts detection of human activity on the computer system. We also compared the *Swift mode* algorithm with existing power scheme in the operating system that provides up to 66% of the power saving. Finally, we have profiled the proposed framework to analyze the memory and Central Processing Unit (CPU) performance, which demonstrated that there is no memory leakage or CPU degradation problem and framework's behavior remain constant under various overhead scenarios of the memory as well as CPU. The proposed framework requires 3–7 MB memory space during its execution.

Index Terms— Energy Efficiency, Exhaustive Mode, Green Computing, Hibernate, Shutdown, Swift Mode

I. Introduction

With the explosive growth of Information Technology (IT), Information and Communications Technology (ICT) devices and its computer systems software can be considered as a high performance computing devices. If the system operator leaves these computer systems running without any work then the systems will not only consume the energy but also produce enough amount of heat. This is not good for the environment as energy consumption puts a lot of pressure to generate more energy production and produced heat is responsible for the greenhouse gas effect, which have various adverse effects on the human body [1]. According to the prediction from the United States Department of Energy, the energy use by the computer systems in commercial sector is expected to grow by 2.9% per annum on an average between 1998 and 2020 [2]. The green computing also known as sustainable computing [3] and focus on two important areas such as energy-efficient computing and power management [4]. Green computing also represents an intelligent and environment friendly way that transforms the existing engineering and various other disciplines to implement this concept into its processes, products and systems while considering the effects on environment as well as on human.

In 1998, Microsoft, Intel and Toshiba developed the Advanced Configuration and Power Interface (ACPI), which shifted towards the major control of power management from basic input/output system (BIOS) to the more user-friendly environment known as operating systems [5]. This power management features in the operating systems are known as power schemes and used to minimize the power consumption by computer systems. In ACPI interface the operating system checks for the application software activity. In case of no any activity is detected the operating system sends a message to the BIOS and then it starts a timer. Once the

Copyright © 2013 MECS

time-out occurs, it starts the power management by sending a message to that device and changes its state from active to idle, or sleep depending upon the configuration [6].

Though ACPI is the industry wide accepted interface but it has some major drawbacks. The time-out approach is one of them that finds the idle period of software and machine components like Central Processing Unit (CPU), Hard disk drive and other peripheral devices. If one considers the case of Windows Operating System, then these power-saving schemes exist from past two decades in it. However, the sales of computers and other ICT devices have increased tremendously during this period and Windows Operating Systems still occupies largest (90%) market share [7], but there is no change in ACPI interface, or in Windows power schemes. In most cases difficulties in properly configuring the power scheme is found and only 25% of the computer systems achieve the energy savings [8]. Therefore, there is a significant need of energy sustainable computing for the operating systems, which change the design of power schemes of operating systems to make it more interactive and environment-friendly.

In this paper, we have proposed an energy sustainable framework of power schemes for Windows Operating Systems because it is one of the preferred operating system by the users. This framework is totally different from the existing framework as it is very much human centric and continuously checks for CPU of computer system and target the human activity on the machine [9] in comparison to the interface of existing power scheme. The remainder of the paper is organized into various sections. Section II focuses on the detailed literature survey which consists of various techniques like dynamic voltage and frequency scaling (DVFS), CPU frequency and its utilization, memory optimization techniques and various designed tools. Section III focuses on proposed framework of power scheme which is very much human centric and claims more energy savings in comparison to the existing power scheme. Here, we have shown the internal view of the framework that includes the details of each package with their dependency diagram. Section IV implements the algorithm for *Swift mode* used by proposed framework and further for its performance evaluation. Section V represents the detailed comparison of both the modes of proposed framework with the existing power schemes. Section VI represents the detailed performance evaluation of the proposed framework and focuses on the CPU performance, memory performance and thread monitoring. Finally, the section VI concludes this work and recommends future directions.

II. Related Work

Recently, Li and Zhou [9] have discussed the various potential issues related to the green computing and emphasized the need of energy efficiency and energy-awareness. They have stated by having the proper awareness about the energy-awareness, a computer system can tune the power consumption of various devices to reduce the overall energy consumption and categorize the energy awareness approaches into two groups. Firstly, energy-awareness is the entire responsibility of individual application and secondly, by considering the entire responsibility of a computer system as a whole. In this second category, the authors have discussed about the time out approach by considering the inactivity period of various system components like CPU, Random Access Memory (RAM), etc. may be turned-off or the entire system may be hibernated. However, they have realized that this kind of applications could sacrifice performance and functionality of computer system. Gupta and Singh [11] have presented the detailed survey on minimizing the power consumption by computer systems and discussed various key areas where energy-awareness policy could be implemented. However, they have also presented various case scenarios based on operating system's power scheme settings and found that these power schemes are not sufficient to minimize the power consumption and focused on the need of developing some intelligent algorithms for reducing the energy consumption. Gupta and Singh [12] focused on the adverse effect of heat dissipation on environment and human health by the computer system and realized the need to minimize this heat dissipation by minimizing the unwanted processing of computer systems. Here, the authors have calculated the power consumption of each running processes on the machine and proposed a algorithm for the state when all the processes becomes idle and there is no power consumption to switch off the computer system. Chen *et al.* [13] have investigated the adverse effects of DVFS 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 to the resource parameters. They used these CPU gradients for performance-aware energy conservation by deploying energy controllers. Chen *et al.* [14] have also focused on the problem of power dissipation by computer systems and observed this as a major problem for the modern computer systems. In their experiments, they have measured the idle and busy states of various components like CPU, Memory, and disk. for their power dissipation using various benchmarks and find out that cache memory of the computer system provides the more accurate results compared to CPU, etc., and expressed that there should be some easy-to-use indicator that could more accurately reflect the CPU power. Zhong *et al.* [15] proposed an energy consumption model based on CPU utilization. This model utilized system-level information in the form of average CPU utilization

Copyright © 2013 MECS

model and transistor-level information in the form of calculating the energy consumed by a single transistor. Minartz *et al.* [16] evaluated the power-saving mechanism in high-performance computing. They have designed a power-aware cluster that includes the hardware's from different manufacturers. They have measured the P-states and C-states for the set of processors of Intel and AMD and power consumption of Hard disk drives in different modes. Spiliopoulos *et al.* [17] created a framework to optimize the power efficiency of real system and implemented Linux DVFS governors based on their analytical DVFS models. They have considered the activity of CPU when it is not lightly loaded and executes program. For memory-bound cases, they have achieved 55% of power efficiency. Ye *et al.* [18] stated that as the size of main memory is growing over the period of time so the energy consumption by them and proposed a dynamic approach to reduce the memory energy consumption. This approach clearly provides the energy optimization for memory by using various mechanisms like energy-aware memory allocation and static-power state management. In their results, they observed that proposed mechanism provides the best performance and energy savings. Duan *et al.* [19] evaluated several energy management mechanisms for main memory proposed for desktops and servers. They have considered the RAM optimization technique for smart-phone hardware and proposed a hybrid mechanism by applying optimization techniques to the newly emerging phase change memory with their energy efficiency and performance. In their results, they achieved more than 98% of energy saving as compared to the standard smart phone. Wang and Wang [20] have presented and analyzed the data on the electricity consumptions in the IT industry and by the household computers in five major Chinese cities up to five consecutive years from 2005 to 2009. Based on the data, they predicted that there is a great importance of green computing to reduce the energy consumption of IT industry as well as household computers. Weng *et al.* [21] utilizes the concept of simultaneous multithreading for today's microprocessors and proposed a power aware fetch policy for evaluating the power consumption of computation resources and memory-accessing resources. They have evaluated the power consumption for every thread and they found that proposed power aware fetch policy improves the power efficiency on an average by 27% over others policies. Li *et al.* [22] proposed a novel solution that considers the dependence of power consumption on temperature and provided a new power model for temperature aware power allocation (TAPA). They have considered the cluster size of 13 machines to implement their optimization algorithm and obtained that with the rise in CPU temperature power consumption increases. In their solutions they achieved higher computational efficiency over static solutions and DVFS solutions. Gupta *et al.* [23] designed a Green Data Center Simulator (GDCSim) tool to minimize the energy consumption in data centers. This tool study the energy efficiency of data centers under various data center geometries like its workload, power management schemes and scheduling algorithms. This tool basically captures the inter-dependencies between various resource-management techniques available online and the physical behavior of data center. Do *et al.* [24] 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.* [25] 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 them 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 [26] 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. Naumann *et al.* [27] 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.

III. Proposed Framework of Power Scheme

This section discusses the proposed framework of power schemes for Windows Operating Systems, which is developed as a tool known as Green Power tool (GP tool). This framework is shown in Fig. 1 and starts its functioning with the execution of power saver module.

This power saver module starts as a local services of the Windows Operating System and prompts the user to input the approximate time also known as login duration of working on the computer system. As soon as the user inputs the value of login duration this value is utilized by three major modules of proposed framework that is power-saver main window, duration and calculate CPU usage. Here, power saver-main window represents the Graphical User Interface (GUI) of proposed GP tool whereas further two modules are very much concerned about implementation of the proposed algorithms to minimize the power consumption by the computer systems. These algorithms are implemented as two different modes known as: 1) *Swift Mode* and 2) *Exhaustive Mode* and are part of GUI as shown in Fig. 2. These two modes are very different in their functioning.

Copyright © 2013 MECS

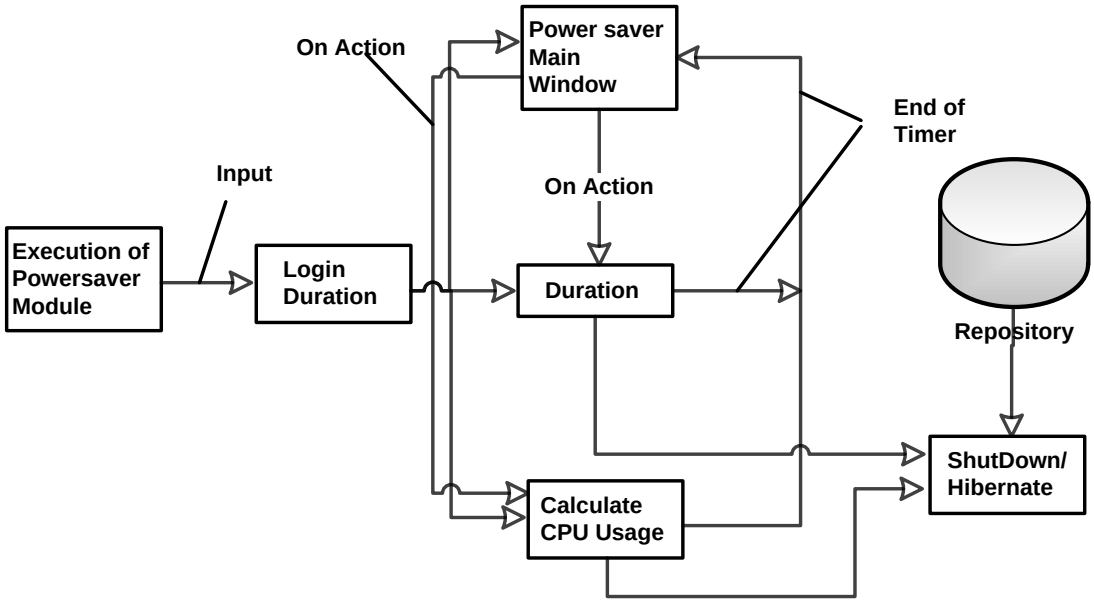


Fig. 1: Framework of the proposed power scheme

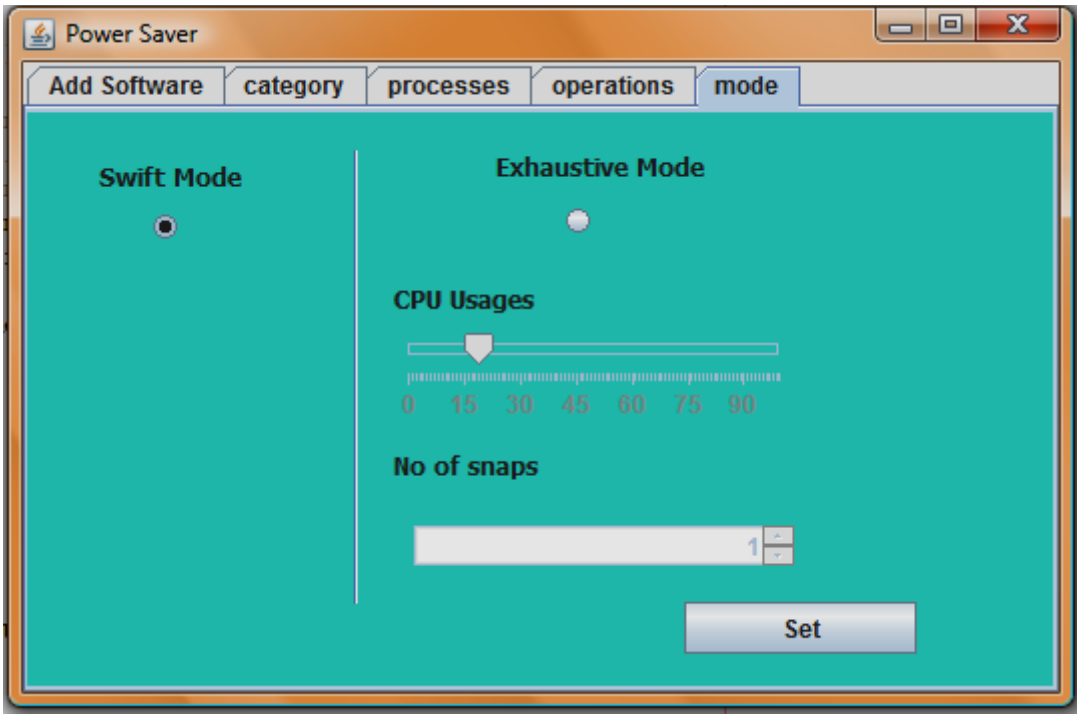


Fig. 2: GUI Implementation of proposed algorithms in GP tool

Swift Mode – considers, the time of user login-duration and computer system continues its working till the time of login-duration comes to its end. Once the time login-duration reached to its end, user of computer system notifies about that by prompting a window asking, *do you want to continue your work?* By doing so, one can found the availability of user on the computer system. If user is there and wants to proceed their working, they have to enter a new time value of login duration otherwise proposed energy sustainable framework will switch the computer system to shutdown, or hibernate mode based on the various running application software's and repository configuration.

Exhaustive Mode – the functioning of this mode is very much different with the previous *Swift* mode, the authors considered the time of login-duration as well as monitored the user activity continuously on the computer system by proposing Energy Sustainable Snapshot Algorithm (ESSA) [28].

Copyright © 2013 MECS

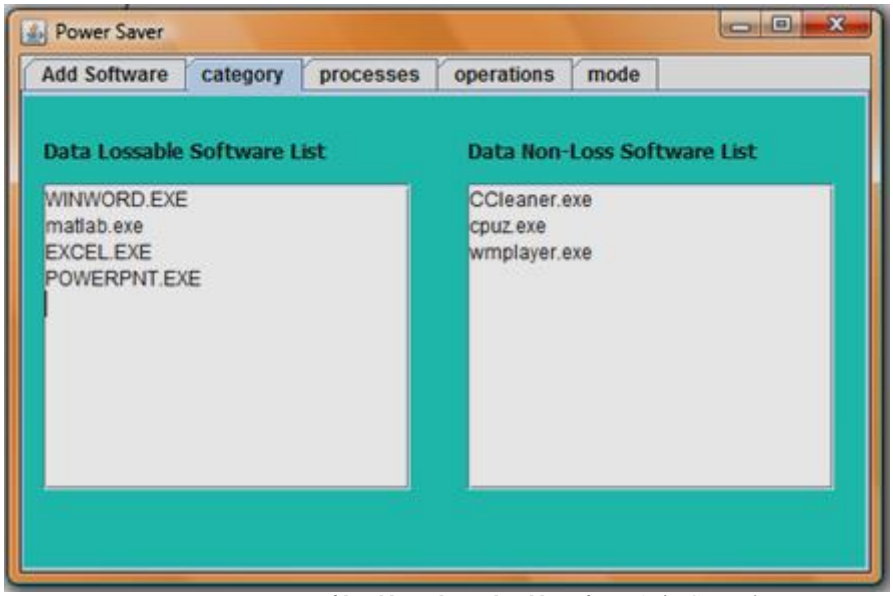


Fig. 3: Repository of losable and non-losable software's in GP tool

This algorithm puts another check for minimizing the power consumption of the computer system. Therefore, this mode provides better way of energy sustainability by the proposed framework.

In this framework, we have used the concept of repository that provide the uniqueness to proposed GP tool and keep the record of various installed application software's on the computer system into two categories, losable and non-losable, as shown in Fig. 3.

This repository is used by the shutdown/hibernate module to switch-off the computer system while making the decision by the proposed framework. Here, if it is found that any software running from the lossy category means that there is every chance to lose the user data while making energy-sustainable decision then the proposed framework switch the computer system to hibernate mode otherwise it gets shutdown.

3.1 Internal View of the Framework

This section describes about the internal view of proposed framework that includes the details on various packages, JAVA files, major classes, methods and inter-dependency of packages. This internal view of the framework focuses on the five different packages connected with each other, details of each package, major classes, and methods defined are given below:

A. *PowerSaver* package

This is the main package of proposed framework and various other packages are dependent on this. This package consists of single JAVA code file that calculates the screen size and popup the window of login-duration in the middle of the screen as shown in Fig. 4.

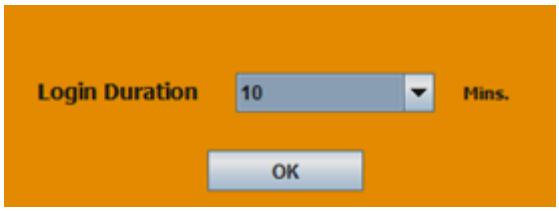


Fig. 4: User prompt to input login duration time

B. *org.juitw.visual* package

This is the second next invoked package that consists of number of JAVA program files with number of classes defined in it. These program files are used to capture the login-duration from the user of the computer system to pop-up the main power saver window and an inner timer thread starts in sleep mode which checks for the expiry of the login-duration time. This package is also responsible to measure the percentage of total CPU usage that is utilized by the exhaustive mode.

C. *org.juitw.timer* package

This package consists only JAVA program file that stores the login-duration time entered by the user and when it get expires a pop-up message get invoked and also starts an inner-timer thread of fixed duration of one minute to get response from the user. When no user activity is found on the computer system, a process that collects the details of all running applications on the system gets started and performs a check on repository with the collected details for making a decision whether to shutdown, or hibernate the computer system.

D. *org.juitw.process.collector* package

This package consists of the number of JAVA program files that implements some important methods for the proposed framework of power scheme. Here,

Copyright © 2013 MECS

CPUInfo.java is used for fetching and displaying the CPU information, for example, in this framework we have used its percentage, *CalculateCPUUsage.java* declares the number of methods, out of which there is one method **createTimer( )** that takes input from the following configuration files:

- ***mode.config*** which is responsible for checking the running mode of proposed framework for power scheme. There are only two modes, swift and exhaustive, for the proposed framework of power scheme.
- ***custom.config*** this file is basically used to store the various input values of ESSA and is used by the exhaustive mode. These are the user-defined values and can be redefined according to the user's requirement. For better energy efficiency, power-saving and management of computer system, it is suggested that the CPU usage value should be kept always less than or equal to 20.

Whereas, *ProcessCollector.java* program file is used to store the value of the various running application programs on the computer system.

E. *org.juitw.process.bean* package

This package declares the only class about the ProcessBean and its various attributes like pid, processname, memory, on behalf, and status. The defined class is just a collection of various getter and setter methods for various declared attributes in the proposed framework.

F. *org.juitw.actions* package

The role of this package is to include various execution files to the repository of the proposed framework. This package includes two JAVA program files, firstly, *SimpleFileFilter.java*, which is used to create a filter for various exiting files inside the folders. Secondly, program files *SimpleFileView.java*, which is used to show the various filtered executable files with an extension .exe from various folders to the user of the computer system and adds it to the repository.

This internal view of the proposed framework of power scheme also represents the package dependency diagram as shown in Fig. 5.

Fig. 5, shows the dependency among various packages in which they are used in the proposed framework as well as connectivity of these packages with various other packages used as import line in the respective program file.

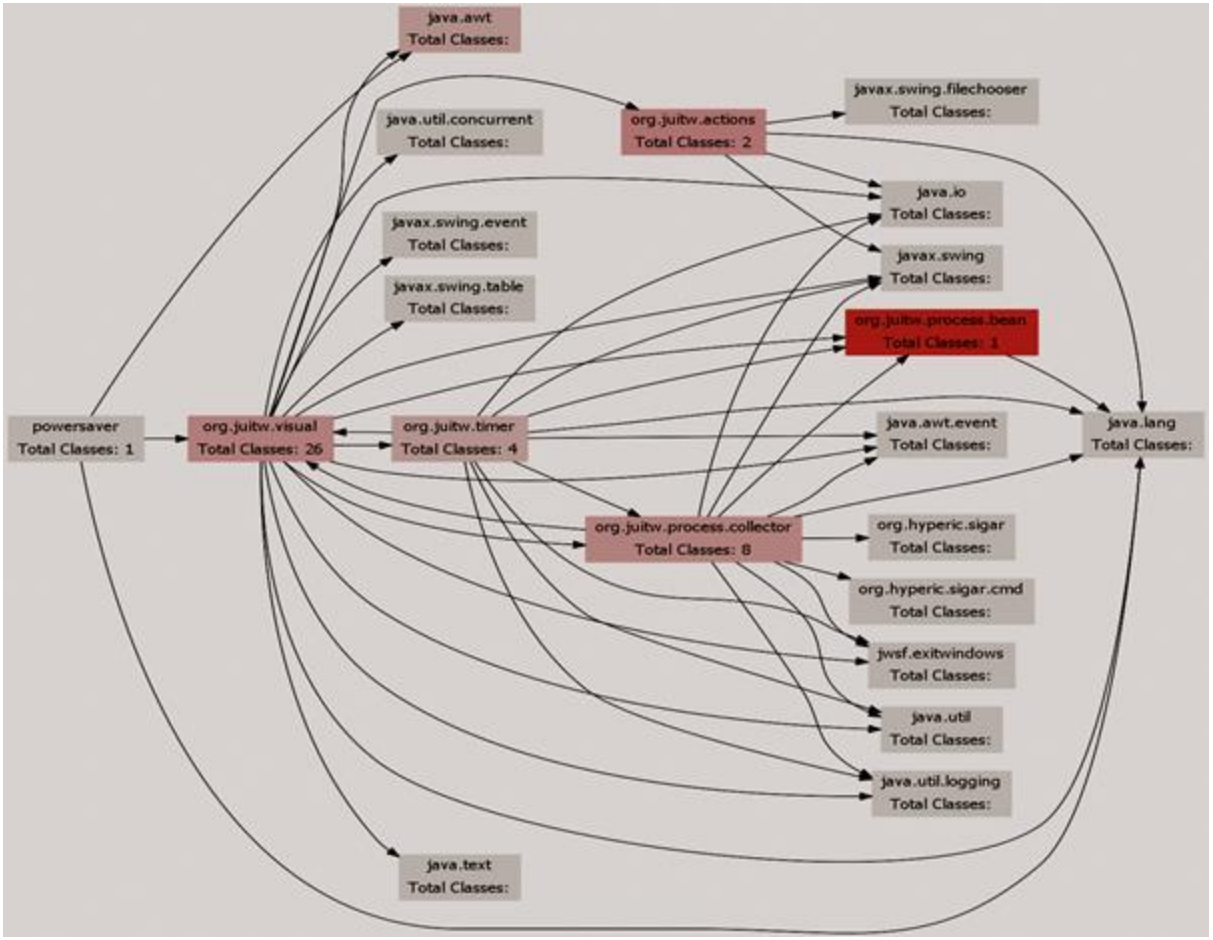


Fig. 5: Package dependency diagram of proposed framework for GP tool

Table 1: Detailed view of each package and JAVA program files

S. No.	Source Packages	Files	Line of Code (LOC)	Lines with Import Ts
1	Power Saver	PowerSaver.java	35	4
2	org.juitw.visual	Category.java	157	3
		CurrentProcessPanel.java	67	2
		Customization.java	309	4
		LoginDuration.java	195	13
		PowerSaverMainWindow. java	821	18
		ProcessTableModel.java	74	4
3	org.juitw.actions	SimpleFIleFilter.java	54	2
		SimpleFileView.java	34	4
4	org.juitw.timer	Duration.java	173	17
5	org.juitw.process.collector	CPUInfo.java	62	6
		CalculateCPUUsage.java	173	17
		ProcessCollector.java	116	7
6	org.juitw.process.bean	ProcessBean.java	72	0

Table 1 gives the detailed overview of discussed packages and details about JAVA program files in them.

IV. Proposed *Swift mode* Algorithm

This section represents algorithm implemented for the previously discussed scenario of Swift mode and used for collecting the data for analysis and result purpose. This algorithm initializes the variables L, M and X in step 1 and starts a two timer threads T1 and T2 into sleep mode in step 2. These timer threads represents that the user has logged into the system and input the value of login duration into the proposed framework of power scheme. In step 3, the working mode M from the two available modes Swift and Exhaustive is selected. In this case, the mode is Swift. Selection of mode is part of user configurations and selected or changed when user wants to switch the mode. Next step 4, is a major step and used for switching the computer system into hibernate or shutdown mode once the time of login duration gets expired.

Algorithm: Swift mode

Symbols used in this algorithm:

i) *L* – The total login duration time
ii) M – The mode of operation
iii) *X* – The sleeping time
iv) η – Extra time required if any
v) T1 and T2 – Timers

Step – 1: ***Initialize variables***

Initialize *L, M and X*

L ← Take the input from the user during login to the system.

M ← Take the value from the configuration file set by the user.

X ← The sleeping time of a thread.

Step – 2: ***Start of Timers***

Starts two timer threads T_1 and T_2;
T_1: expires after *L* x 60 x 1000 ms
T_2: expires after each 1 x 60 x 1000 ms and sleeps for *X* time.

Step – 3: ***Mode Selection***

Selection of Mode (*M*):
IF *M* = = *Swift* then
X ←*L* x 60 x 1000
ELSE
X ← 1 x 60 x 1000

Step – 4: ***Functioning in Swift mode***

When T_1 expires

a) Cancel T_2
b) η ← input extra time.
c) Starts a thread T_3. T_3 will expires after 1 x 60 x 1000 ms.
d) After 1 x 60 x 1000 ms
IF η = = 0 **THEN** check any losable program is running (check with repository)
IF yes
HIBERNATE the computer system
ELSE
SHUT DOWN the computer system
END
ELSE
GO TO STEP (1) with
L← η
END

END

V. Comparison of Proposed Framework with Existing Power Schemes

For detailed comparison of working of existing power-scheme in Windows Operating Systems with proposed framework of power-scheme, we have compared the working function of both proposed Swift mode and exhaustive mode with the functioning of existing scenario that available in windows power scheme.

5.1 Existing Scenario of Power Scheme in Windows Operating System

This scenario represents the configuration of existing power scheme in Windows Operating Systems. In this existing scenario of power scheme the user has performed various settings, as given below, to minimize the power consumption by the computer systems.

Turn-off monitor/display	*= after 30 min.*
Turn-off hard disks	*= after 30 min.*
System standby/sleep	*= after 30 min.*

User works for 2 min. and leave the computer system inactive.

Using the above scenario as defined by the user, system will start power-saving only after the 30 min. of inactivity of computer system. The drawbacks of this setting are as follows:

a) The settings are very much system-oriented and are based on time-out approach defined for various devices. There is no provision that detects for human activity on the system.
b) The existing power scheme starts its functioning only after 30 min. of inactivity of keyboard and mouse, whereas user works for only 2 min.
c) After the inactivity of 30 min. the computer system will be switched to sleep mode. In sleep-mode computer system also consumes small amount of power and consumption varies from computer system to computer system based on their configuration.

5.2 Comparison with Swift Mode

The authors compared the scenario of existing power scheme as stated above with the Swift mode of our proposed framework of power scheme. The only setting implemented by the proposed framework for this mode is given below:

Login-duration = 10 min.

User works for 2 min. and leave the computer system inactive.

In Swift mode, user has to input the login duration just after the login to the Windows Operating System. This framework gets the user consensus during the login to the system about the user's working on the computer system. This nature provides the interactive and dynamic environment to proposed framework. Here, the user has consensus that he/she will work only for 10 min., but due to some reasons he/she leaves the computer system inactive after 2 min. of time. The comparison observations with the existing power schemes are as follows:

a) The existing power scheme scenario, which is more static in nature, each time user has to change the power scheme settings if he/she wants to work less than the defined period in power scheme. Though it is possible but irritating too, as most of the time users are not concerned about the configurations in these power schemes.
b) For users, it is difficult to memorize the various time interval values defined in the existing power scheme as these values are defined once and used forever.
c) Proposed Swift mode overcome from the disadvantages discussed in (a) and (b), one does not need to worry about the various time intervals values.
d) This mode offers more than 66% of power-saving over existing power scheme scenario.

5.3 Comparison with Exhaustive Mode

The authors have compared the scenario of existing power scheme with the exhaustive mode of proposed framework of power scheme. The various settings of this mode are given as below:

Login-duration	=	*10 min.*
CPU usage	=	*≤ 20*
Number of snapshots	=	*2*

User works for 2 min and leave the computer system inactive.

When compared to the Swift mode, the exhaustive mode needs to be configured once from user side with minimal setting parameters like the value of total CPU usage and number of snapshots to be compared before taking any decision of hibernate/shutdown the computer system. The comparison observations of exhaustive mode with existing power schemes and Swift mode are given below:

a) The scenarios of exiting power scheme, where there is no power-saving up to 28 min. and for Swift mode this time reduces to 8 min. by knowing the user consensus during the start up of the computer system.

Copyright © 2013 MECS

b) In exhaustive mode, we have tried to minimize the outstanding time of Swift mode for more energy saving.

c) The exhaustive mode will switch the computer system to hibernate/shutdown mode after 4 min. of time interval and offers more than 93% of power-saving over existing power scheme.

It is assumed that the user is not available on the computer system after the working of 2 min., therefore various prompts like *"Are you available on the system..."* or *"Your login duration is expiring very soon, want to enter new time value..."* invoked by the proposed framework of power schemes to check the user activity on the computer system gets expired after a certain period of time.

VI. Performance Evaluation of Proposed Framework

The performance measures for the proposed framework of power scheme include the thread monitoring, analyzing memory and CPU performance under various overheads. We used the profiler available in NetBeans-IDE [29] for evaluating the performance of proposed framework under Swift mode by giving an algorithm. By using profiler, one can easily determine the performance of system's memory and CPU under various performance measures like memory leakage [30], memory heap, and memory garbage collection [31], thread monitoring, CPU timestamps for each invoked methods etc. The following are the results obtained after rigorous performance analysis of CPU and memory.

6.1 Thread Monitoring

The authors have monitored various active threads for the proposed framework. Fig. 6 provides the details of each active thread during profiling of proposed framework. These threads are divided into two categories, system threads and user threads. The descriptions about these threads are given in Table 2.

Table 2: Thread details

S. No.	Thread name	Uses class	Type of Thread	Description
1	**Reference Handler**	java.lang.ref.Reference$ReferenceHandler	System	High prior thread that enqueue pending references.
2	**Finalizer**	java.lang.ref.Finalizer$FinalizerThread	System	Performs finalization of objects before their garbage collection.
3	**Attach Listener**	java.lang.Thread	User	User Thread
4	**Java 2D Disposer**	java.lang.Thread	System	Handles disposal of native data associated with java objects in Java 2D.
5	**AWT-Shutdown**	java.lang.Thread	System	AWT system thread, handles shutdown of AWT (Event Queues) when no GUI is displayed.
6	**AWT-EventQueue-0**	java.awt.EventDispatchThread	System	AWT thread, which is the main thread executing a GUI java applications.
7	**DestroyJavaVM**	java.lang.Thread	User	User Thread
8	**Timer Queue**	java.lang.Thread	System	Used to manage all javax.swing.Timer instances in one thread.
9	**Thread-7**	java.lang.Thread	User	User Thread
10	**Thread-8**	java.lang.Thread	User	User Thread
11	**Thread-10**	java.lang.Thread	User	User Thread

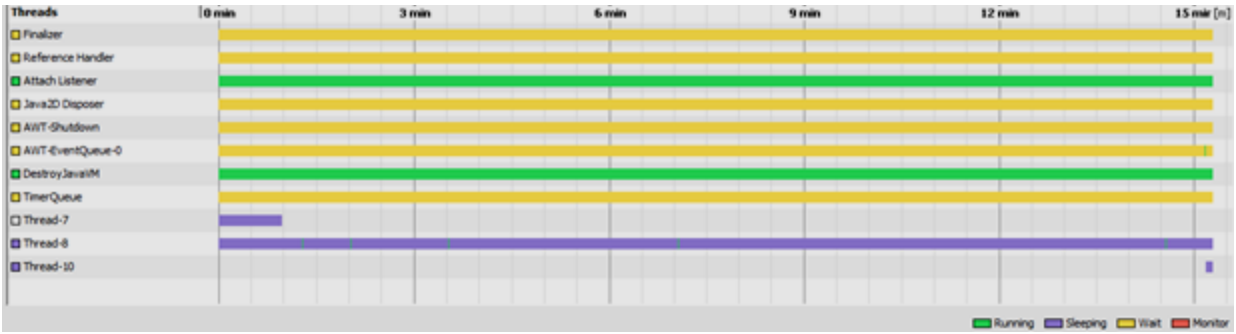


Fig. 6: Various active threads during framework monitoring

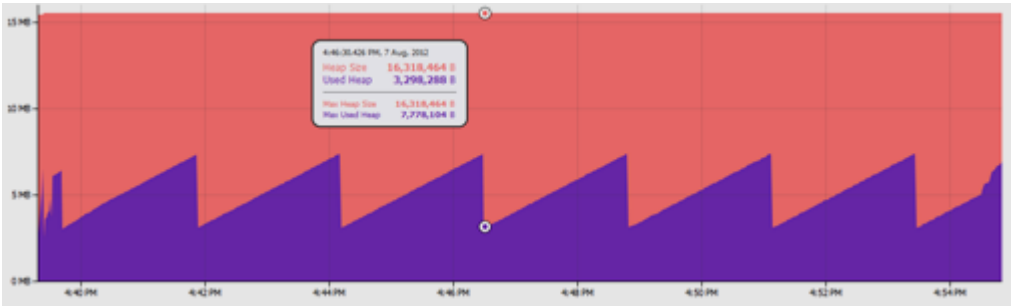
From Fig. 6, one can easily found that the user threads 7 and 8 gets started into sleep mode by providing the login-duration time to the framework and after a certain period of time inner-timer which corresponds to thread 7 gets expired and thread 8 continues till the end of login-duration. Once the login-duration ends then the AWT-EventQueue-0 system thread which is in wait state, gets activated and pop-up the message window to the user that "your time is finished" do you want to continue your working, if user reply for yes, then another message window gets pop-up to enter the extended time duration and the user thread 10 gets started into sleep mode for the new login-duration.

6.2 Analysis of Memory Performance

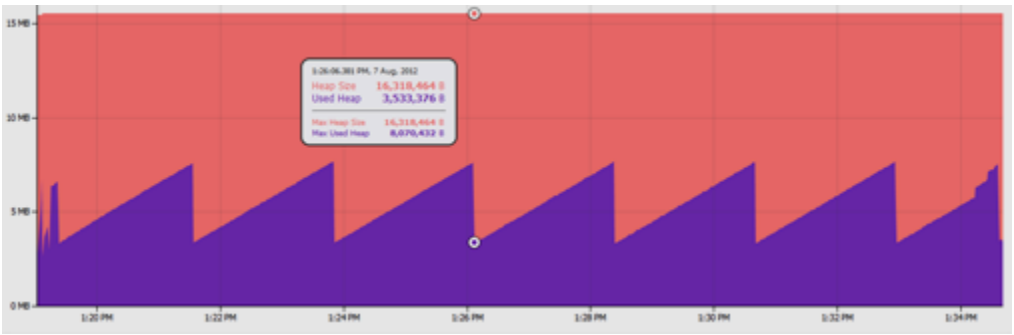
The analysis of memory performance of the proposed framework under different overheads includes the various results like memory heap, garbage collection and threads and loaded classes. We have profiled the proposed framework under two different scenarios, firstly, performance overheads, where the memory performance have recorded for object creation only, secondly, both object creation and garbage collection have recorded. In second overhead scenario the default value of stack sampling interval has fixed at 10. This value means that the 10th object allocation of every class will be recorded. Using VM telemetry monitor various real time results have obtained as shown in the figures of subsections. In all our real time results of memory performance, we have profiled the proposed framework up to 15 min.

a) Heap analysis

In Fig. 7(a) and Fig. 7(b) the memory heap size over the period of time for both the overhead scenarios have been analyzed as stated above. It is easy to find out the details of maximum available heap size versus used-heap by the profiled framework.



(a)



(b)

Fig. 7: Analysis of memory performance for allocated heap size versus used-heap (a) object creation only and (b) both object creation and garbage collection. For each graph, *x*-axis denotes the time in (HH:MM) and *y*-axis shows the used-heap size in (MB)

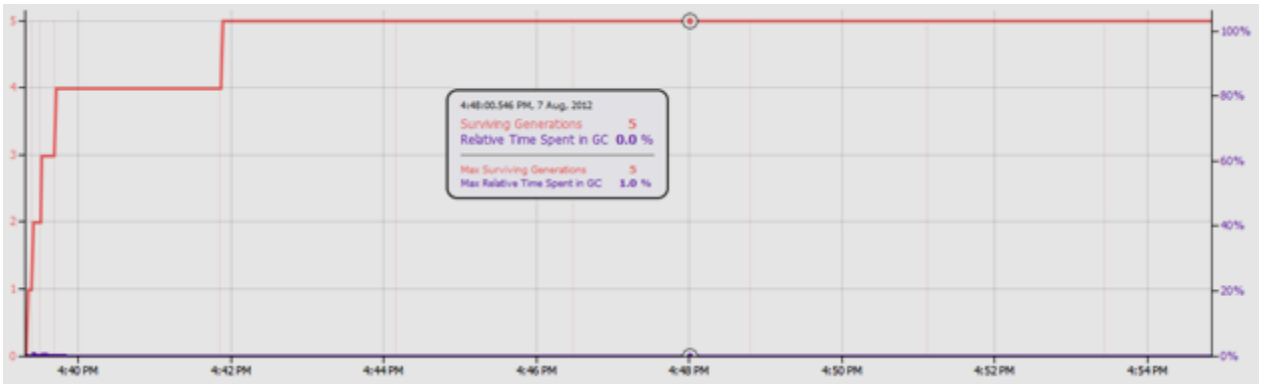
For both of our scenarios, maximum available heap size is the same with little variation in used-heap size in Fig. 7(b). Garbage collection is done after a certain interval of time, which minimizes the used-heap size. These intervals are easily noticeable in Fig. 7(a) and Fig. 7(b). Throughout the login-duration framework continues its functioning smoothly, which can be noticed with the sharp edges of Fig. 7(a) and Fig. 7(b), but when the login-duration time comes to at end, there is some deviations in the edge that refers to the activation and creation of threads in the memory. To know more about the threads, one can refer the thread monitoring section discussed above.

b) Memory Leakage

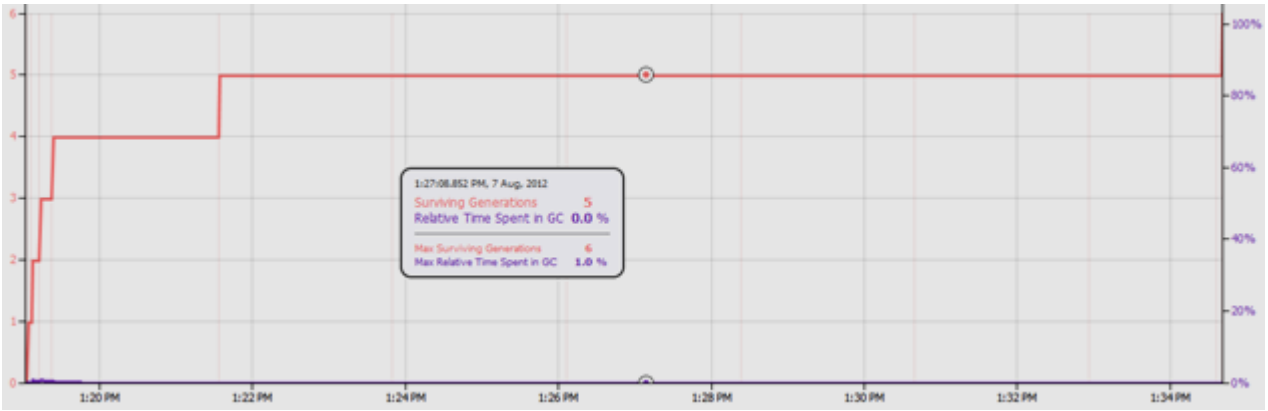
The problem of memory leakage for the proposed framework have analyzed by finding various surviving generations and relative time spent in garbage collections. From Fig. 8(a) and Fig. 8(b), one can find that once the framework gets initialized total number of surviving generations becomes constant and remains at

Copyright © 2013 MECS

5 till the login-duration ends, whereas in Fig. 8(b) this number reaches up to 6 at the last which shows the removal of finished threads from the memory that profiled the framework for both object creation and garbage collection. So, there is no problem of memory leakage in proposed framework. Maximum relative time spent in garbage collection is 1% as shown in Fig. 8(a) and Fig. 8(b).



(a)

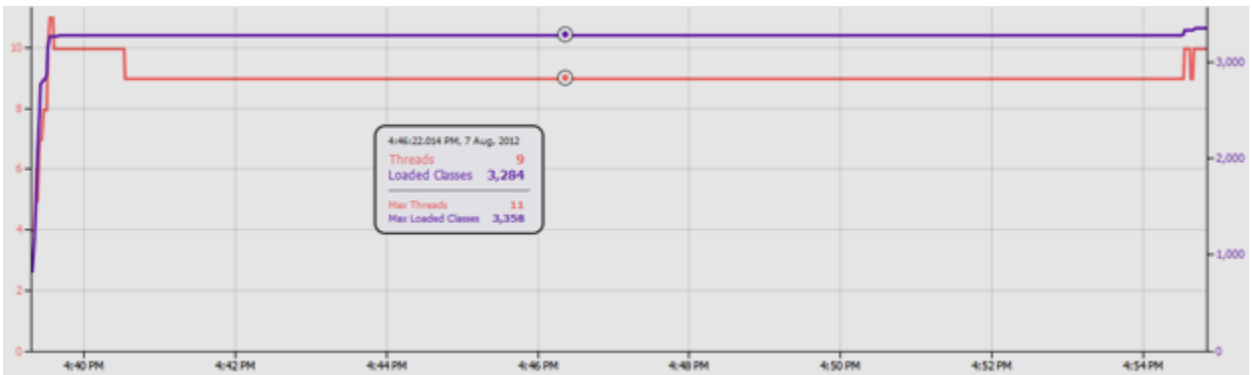


(b)

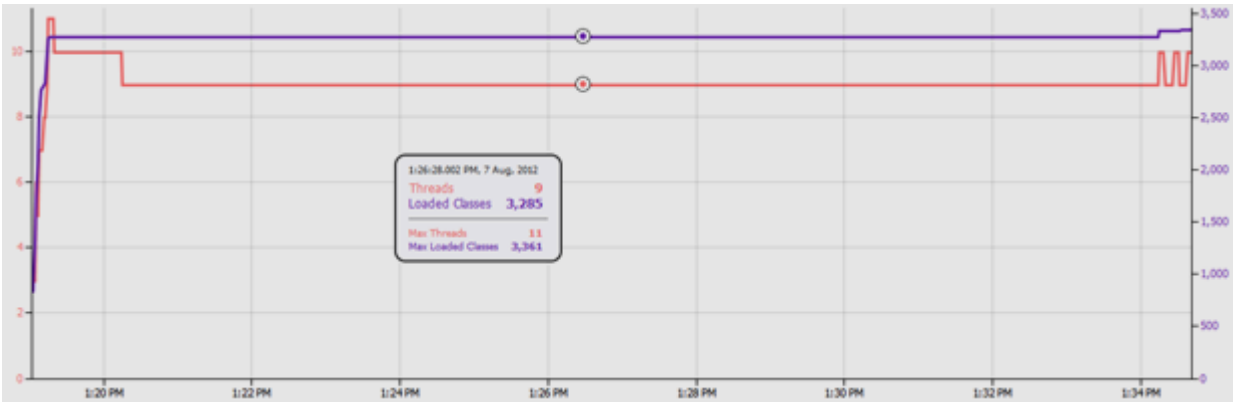
Fig. 8: Analysis of memory performance for surviving generations versus Relative time spent in GC
(a) object creation only and
(b) both object creation and garbage collection. For each graph, *x*-axis denotes the time in (HH:MM) and *y1*-axis shows the surviving generations and *y2*-axis shows the relative time spent in GC (%)

c) Thread Analysis

It is very much similar to scenario discussed in thread monitoring section. The memory performance for various running threads versus loaded classes has analyzed. From Fig. 9(a) and Fig. 9(b) one can find that the maximum number of threads remains the same in both the cases, with a little variation in maximum loaded classes in Fig. 9(b). It can also observe that during the login duration, no new thread is created and total number of threads remains constant. At the end of login-duration, some threads get activated from sleep mode and some are created newly, details of which are given in the thread monitoring section.



(a)



(b)

Fig. 9: Analysis of memory performance for threads versus loaded classes
(a) object creation only and
(b) both object creation and garbage collection. For each graph, *x*-axis denotes the time in (HH:MM) and *y1*-axis shows the running threads and *y2*-axis shows the loaded classes

6.3 Analysis of CPU Performance

By using CPU performance measurement, the proposed framework have analyzed and obtained 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. CPU under various overheads have also analyzed, firstly, by profiling the only project related classes that includes the project related core java classes only, and secondly, by profiling all classes that includes all core java classes and server classes, and imposes a very significant overhead. By using the first overhead maximum numbers of loaded classes are 2414, whereas with the second overhead, the numbers of classes are 7596, used to analyze the CPU performance. This analysis is shown in Fig. 10(a), Fig. 10(b) and Fig. 10(c) that shows the call tree class of calculate CPU usage, login-duration and various threads created to monitor the proposed framework.

(a)

(b)

(c)

Fig. 10: Analysis of CPU performance (a) using only project classes (b, c) use all classes.

Here, with both the overhead scenarios we have profiled the proposed framework around 15 min. Fig. 10(a), Fig. 10(b) and Fig. 10(c) showed that thread 7, 8, 9, and 10 corresponds to user threads, which resides into sleep state during its executions. Here, thread-8 continues till the end of login-duration and invoked only once for both the overhead scenarios, whereas the time spent by the classes of this thread is different as it is 2483 ms and 4344 ms, respectively. Thread 7, which starts with thread 8 into sleep mode and get expired after a minute of time interval. This thread is basically used for calculating the CPU usage as it is profiled the proposed framework in Swift mode and functioning of mode is independent of total CPU usage. Further, thread 9 shows that user wants to continue the work once the login- duration has reached to its expiry and enters the new time value of login duration. By entering the new time value of login-duration thread 10 gets started which is similar to thread 7.

During CPU performance analysis we found that no classes or methods that utilizes, or keeps the CPU busy all the time. From Fig. 10(a), Fig. 10(b) and Fig. 10(c) showed that by increasing the overhead in the form of load of all classes over CPU the performance of proposed framework remains the same.

VII. Conclusions

In this paper, we have proposed an energy-sustainable framework for the power schemes of operating systems to minimize the energy consumption by computer systems. The proposed framework is a part of GP Tool. The main objective of this tool is to structure concepts, strategies, and activities to design an energy-sustainable power scheme. This framework is useful for both the desktops and laptops. The unique characteristic of this tool is that it required minimal input and calculations for saving energy. The proposed algorithm for *Swift mode* detects the human activity on the computer system in an effective manner and it is based on the time value supplied by the user during login to the system. Comparison of proposed framework with the discussed scenario in existing power scheme reveals that *Swift mode* provides more than 66% of energy saving and *exhaustive mode* provides more than 93% of energy saving. To enhance the accuracy of the proposed algorithm, we have evaluated the CPU and memory performance of the proposed framework. The results obtained from this evaluation are very impressive and demonstrated that there is no slow-down in CPU performance and no memory leakage problem during its execution is found. The results revealed that the proposed algorithm achieved the proposed goals both theoretically and practically for designing a complete energy-sustainable tool and suggest that any changes can be incorporated into the power schemes of operating systems. We hope that the algorithms and GP tool helps researchers to develop a comprehensive solution for the energy-sustainable computing

Acknowledgement

The authors are sincerely thankful to the unanimous reviewers for their critical comments and suggestions to improve the quality of the manuscript.

References

[1] Ruediger Kuehr and Eric Williams, “Computers and the Environment: Understanding and Managing their Impacts,” *Kluwer Academic Publishers*, October 2010, pp. 1-285.

[2] C. A Webber, R. E Brown and J Koomey, “Savings estimate for the ENERGY STAR® voluntary labelling program,” *Energy Policy*, v28, n15, 2000, pp. 1137 – 1149.

Copyright © 2013 MECS

[3] Sandeep K.S. Gupta, Tridib Mukherjee, Georgios Varsamopoulos and Ayan Banerjee, "Research directions in energy-sustainable cyber–physical systems," *Sustainable Computing: Informatics and Systems*, v1, n1, 2011, pp. 57 – 74.

[4] S. Murugesan, "Harnessing green it: principles and practices," *IEEE IT Professional*, v10, n1, 2008, pp. 24–33.

[5] Judy A. Roberson, Gregory K. Homan, Akshay Mahajan, Bruce Nordman, Carrie A. Webber, Richard E. Brown, Marla McWhinney, and Jonathan G. Koomey, "Energy Use and Power Levels in New Monitors and Personal Computers," *LBNL-48581*, Lawrence Berkeley National Laboratory, 2002, pp. 1 – 36.

[6] B. Nordman, M. A. Piette, K. Kinney and C. Webber, "User guide to power management for PCs and monitors," *LBNL-39466*, Lawrence Berkeley National Laboratory, 1997, pp. 1 – 72.

[7] Mary Jo Foley, "Windows server still rules the server roots" IDC, June 3, 2010.

[8] Kaoru Kawamoto, Jonathan G Koomey, Bruce Nordman, Richard E Brown, Mary Ann Piette, Michael Ting and Alan K Meier, "Electricity used by office equipment and network equipment in the US," Energy, v27, n3, 2002, pp. 255 – 269.

[9] Ishfaq Ahmad and Sanjay Ranka, "Handbook of Energy-Aware and Green Computing" Chapman & Hall/CRC Computer and Information Science series, CRC Press, 2012, pp. 1 – 1196.

[10] Qilin Li and Mingtian Zhou, "The survey and future evolution of green computing," Proc. *IEEE/ACM International Conference on Green Computing and Communications*, China, 2011, pp. 230 – 233.

[11] P. K. Gupta, G. Singh, "Minimizing Power Consumption by Personal Computers: A Technical Survey", IJITCS, vol.4, no.10, pp.57-66, 2012.

[12] P. K. Gupta and G. Singh, "A framework of creating intelligent power profiles in operating systems to minimize power consumption and greenhouse effect caused by computer systems," Journal of Green Engineering, v1, n2, 2011, pp. 145 – 163.

[13] Shuyi Chen, Kaustubh R. Joshi, Matti A. Hiltunen, Richard D. Schlichting and William H. Sanders, "Using CPU gradients for performance-aware energy conservation in multitier systems," *Sustainable Computing: Informatics and Systems*, v1, n2, 2011, pp. 113 – 133.

[14] Hui Chen, Shinan Wang and Wei song Shi, "Where does the power go in a computer system: experimental analysis and implications," *Proc. IEEE International Green Computing Conference and Workshops*, Orlando, FL, 2011, pp. 1 – 6.

[15] Benjamin Zhong, Ming Feng and Chung-Horng Lung, "A green computing based architecture comparison and analysis," *Proc. IEEE/ACM International Conference on Green Computing and Communications*, Hangzhou, 2010, pp. 386 – 391.

[16] Timo Minartz, Thomas Ludwig, Michael Knobloch and Bernd Mohr, "Managing hardware power saving modes for high performance computing," *IEEE International Green Computing Conference and Workshops*, Orlando, FL, 2011, pp. 1-8.

[17] Vasileios Spiliopoulos, Stefanos Kaxiras and Georgios Keramidas, "Green governors: a framework for continuously adaptive DVFS," *Proc. IEEE International Green Computing Conference and Workshops*, Orlando, FL, 2011, pp. 1 – 8.

[18] Lei Ye, Chris Gniady and John H. Hartman, "Energy-efficient memory management in virtual machine environments," *Proc. IEEE International Green Computing Conference and Workshops*, Orlando, FL, 2011, pp. 1 - 8.

[19] Ran Duan, Mingsong Bi and Chris Gniady, "Exploring memory energy optimizations in smartphones," *Proc. IEEE International Green Computing Conference and Workshops*, Orlando, FL, 2011, pp. 1 - 8.

[20] Luyang Wang and Tao Wang, "Green computing wanted: electricity consumptions in the IT industry and by household computers in five major Chinese cities," *Proc. IEEE/ACM International Conference on Green Computing and Communications*, Sichuan, 2011, pp. 226 – 229.

[21] Lichen Weng, Gang Quan and Chen Liu, "PCOUNT: a power aware fetch policy in simultaneous multithreading processors," *Proc. IEEE International Green Computing Conference and Workshops*, Orlando, FL, 2011, pp. 1 - 6.

[22] Shen Li, and Tarek Abdelzaher and Mindi Yuan, "TAPA: temperature aware power allocation in data center with map-reduce," *Proc. IEEE International Green Computing Conference and Workshops*, Orlando, FL, 2011, pp. 1 - 8.

[23] Sandeep K.S. Gupta, Rose Robin Gilbert, Ayan Banerjee, Zahra Abbasi, Tridib Mukherjee and Georgios Varsamopoulos, "GDCSim: A tool for analyzing green data center design and resource management techniques," *Proc. IEEE International Green Computing Conference and Workshops*, Orlando, FL, 2011, pp. 1 - 8.

[24] T. Do, S. Rawshdeh and W. Shi, "PTOP: a process-level power profiling tool," Proc. *2nd Workshop on Power Aware Computing and Systems (HotPower'09)*, Big Sky, MT, 2009, pp. 1-5.

Copyright © 2013 MECS

[25] Sudhanva Gurumurthi, Anand Sivasubramaniam, Mary Jane Irwin, N. Vijaykrishnan, Mahmut Kandemir, Tao Li and Lizy Kurian John, "Using complete machine simulation for software power estimation: the Soft Watt Approach," *Proc.* 8th *International Symposium on High-Performance Computer Architecture (HPCA.02),* USA, 2002, pp. 141 – 150.

[26] Kutty S Banerjee and Emmanuel Agu, "PowerSpy: fine-grained software energy profiling for mobile devices," *Proc. IEEE International Conference on Wireless Networks, Communications and Mobile Computing,* MA, USA, 2005, pp. 1136 – 1141.

[27] Stefan Naumann, Markus Dick, Eva Kern and Timo Johann, "The GREENSOFT Model: a reference model for green and sustainable software and its engineering," *Sustainable Computing: Informatics and Systems,* v1, n4, 2011, pp. 294 – 304.

[28] P. K. Gupta and G. Singh, "Energy-sustainable snapshot algorithm for operating systems to minimize power consumption," *Sustainable Computing: Informatics and Systems,* 2012. (Under review)

[29] NetBeans IDE 7.1.2, http://netbeans.org/community/news/show/1556.html (accessed 2012-05-05)

[30] Peng Hao-lin, Liu Yi-min, You Xiang-bai, "Research on memory leakage in Java application," *Proc. IEEE International Conference on Computer Science and Information Technology,* Wuhan, China, v2, 2010, pp. 146-148.

[31] G. Chen, R. Shetty, M. Kandemir, N. Vijaykrishnan, M. J. Irwin and M. Wolczko, "Tuning garbage collection for reducing memory system energy in an embedded Java environment," *ACM Trans. on Embedded Computing Systems,* v1, n1, 2002, pp. 27–55.

Authors' Profiles

P. K. Gupta graduated in Informatics and Computer Engineering from Vladimir State University, Vladimir, Russia, in 1999 and received his M.E. degree in Informatics & Computer Engineering in 2001 from the same university. He has been associated with academics more than ten years in different institutions like BIT M.Nagar, RKGIT Ghaziabad. In India and Currently he is working as Senior Lecturer with the Department of Computer Science and Engineering & IT, Jaypee University of Information Technology, Waknaghat, Solan(HP), India. He is also pursuing his Ph.D. from JUIT Solan. He has supervised a number of B.Tech/M.Tech/M.Phil. theses from various universities of India. His research interests include Storage Networks, Green Computing, Software Testing and Cloud Computing. P.K. Gupta is a Member of IEEE, Life Member of CSI and Life member of Indian Science Congress Association.

G. Singh received his Ph.D. degree in electronics engineering from the 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 also worked as an Assistant Professor at Electronics and Communication Engineering Department, Nirma University of Science and Technology, Ahmedanad, India. He was a Visiting Researcher at the Seoul National University, Seoul, S. Korea. At present, he is an Associate Professor with the Department of Electronics and Communication Engineering, Jaypee University of Information Technology, Wakanaghat, Solan, India. He is the author and co-author of more than 120 scientific papers of refereed journals and international/national conferences. His research interests include electromagnetic and its applications, software defined radio/cognitive radio network, OFDM, nanophotonics and THz communication, imaging and sensing.

How to cite this paper: P. K. Gupta, G. Singh,"Energy-Sustainable Framework and Performance Analysis of Power Scheme for Operating Systems: A Tool", International Journal of Intelligent Systems and Applications(IJISA), vol.5, no.1, pp.1-15, 2013.DOI: 10.5815/ijisa.2013.01.01

Copyright © 2013 MECS