

**ENERGY SUSTAINABLE USER CENTRIC
FRAMEWORK AND ALGORITHMS TO MINIMIZE
POWER CONSUMPTION BY PERSONAL
COMPUTERS**

*Thesis submitted in partial fulfillment of the requirements
for the degree of*

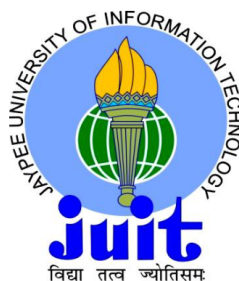
DOCTOR OF PHILOSOPHY

in

Computer Science and Engineering

by

PRADEEP KUMAR GUPTA



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY
WAKNAGHAT, SOLAN-173 234
INDIA**

ROLL NO. 086202

DECEMBER 2012

©Jaypee University of Information Technology

Waknaghat, Solan – 173234

India

December 2012

All rights reserved.

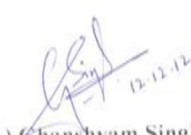


JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY

(Established by H.P. State Legislature vide Act No. 14 of 2002)
P.O. Wagnaghat, Teh. Kandaghat, Distt. Solan - 173234 (H.P.) INDIA
Website : www.juit.ac.in
Phone No. +91-01792-257999 (30 Lines).
Fax: +91-01792-245362

CERTIFICATE

This is to certify that the thesis entitled ***"ENERGY SUSTAINABLE USER CENTRIC FRAMEWORK AND ALGORITHMS TO MINIMIZE POWER CONSUMPTION BY PERSONAL COMPUTERS"*** which is being submitted by ***Mr. PRADEEP KUMAR GUPTA*** in the partial fulfilment for the award of degree of Doctor of Philosophy in **Department of Computer Science and Engineering by the Jaypee University of Information Technology, Wagnaghat, Solan, India** is the record of candidate's own work carried out by him under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma..


Prof. (Dr.) Ghanshyam Singh
Associate Professor
Department of Electronics and Communication Engineering
Jaypee University of Information Technology (JUIT)
Wagnaghat, Solan-173 234, India.
(Adviser)

ACKNOWLEDGEMENT

It would not have been possible to write this doctoral thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

This thesis would not have been possible without the help, support and patience of my adviser, **Prof. (Dr.) G. Singh**. It was really an experience to work with and to learn from him. I salute him for his unbroken faith in experimentation and deep quest for scientific knowledge. I thank him not only for the guidance he rendered in the field of research but also unflinching encouragement and support for enlightening the path of my life with deep love for the environment. His truly scientist intuition has made him as a constant oasis of ideas and passion in research field, which exceptionally inspired and enriched my growth as a student and a researcher.

Words are inadequate to express my deepest appreciation and respect to **Prof. Y. Medury**, Chief Operating Officer of Jaypee Education System. I salute his vision and mission which makes me to compete with the world. I would like to express my sincere thanks to Honorable Vice Chancellor **Prof. Ravi Prakash** for the inspiration and support. My Sincere thanks to our Director, **Brig. (Retd.) Balbir Singh** for his constant support and encouragement. He is really an excellent gentleman who is always ready to give full support for education.

I am greatly thankful to **Prof. (Dr.) S.P. Ghrrera**, (Head of the department, Computer Science and Engineering) for their valuable discussions and kind support during my research work. My sincere thanks also goes to **Prof. Tejinderbir S. Lamba**, (Dean, Academic and Research), who always challenged me to explore new ideas and never quit learning.

I sincerely acknowledge the help and cooperation rendered by library staff for arranging the books and papers whenever required, and also by computer lab staff specially **Mr. Ranvijai Singh** and **Mr. Vijay Kumar** for making my tenure in lab happy and memorable with joyful discussions.

I bow my head towards in the memory of **(Late) Shri. Ram Niwas Varsheny** (Father-in-law) whom I lost in the early stage of my work and I greatly acknowledge him for his big moral support and encouragement towards start of my research work.

Last, but not the least, my parents **Er. D. P. Gupta** and **Smt. Usha Gupta** deserve special mention for their inseparable support and prayers. Their love, support and constant patience taught me so much about sacrifice, discipline and persistent confidence in me has made me able to bear the load on my shoulder. My brother **Shri Raj Kumar Gupta** continuously supported and encouraged me to complete my research. Finally, my special thanks goes to my daughter **Prisha** and son **Raedik**, and to my cute nephew **Akshraj** and niece **Priyal** as without their sweet memories and happiness it was not possible for me. They have been a constant source of inspiration to me.

(Pradeep Kumar Gupta)
December 12, 2012

TABLE OF CONTENTS

Acknowledgement	
List of Figures	i-ii
List of Tables	iii
List of Publications	iv
Abstract	v-viii
CHAPTER 1 INTRODUCTION	
1.1 Introduction	1
1.2 Facts related to use of electricity by personal computers	4
1.3 Power consumption by personal computers	5
1.4 Eco-labeling	7
1.5 Power management in personal computers	8
1.5.1 Advanced power management	9
1.6 Power saving modes and problems with power management	12
1.7 Energy-sustainable computing	14
1.7.1 Energy sustainability	15
1.8 Organization of the thesis	16
CHAPTER 2 MINIMIZING ENERGY CONSUMPTION	
2.1 Introduction	18
2.2 Drawbacks of available power scheme in operating system	20
2.3 Power aware system design	21
2.4 Power measuring and profiling	22
2.4.1 Power model	23
2.5 Simulation based power management approaches	25
2.6 Hardware-based power management approaches	26
2.6.1 Power measurement with power meters	26
2.6.2 Power measurement by specially designed devices	27

2.6.3	Power measurement by integrating sensors	28
2.6.4	Power management by using benchmarks	28
2.7	Software-based power management approaches	29
2.7.1	Dynamic power management scheme	29
2.8	Dynamic voltage and frequency scaling	34
2.8.1	CPU utilization algorithms	35
2.9	Designing power-efficient architectures	38
2.9.1	Programming approaches for power efficient architecture	39
2.10	Conclusion	40
 CHAPTER 3 ENERGY SUSTAINABLE FRAMEWORK		
3.1	Introduction	41
3.2	Energy and power consumption modeling	42
3.3	User centric energy management	47
3.4	Proposed energy sustainable framework	49
3.4.1	Internal view of the framework	52
3.5	Comparison of proposed framework with existing power schemes	55
3.5.1	Existing scenario of power scheme in windows operating system	56
3.5.2	Comparison with swift mode	56
3.5.3	Comparison with exhaustive mode	57
3.6	Conclusion	58
 CHAPTER 4 ENERGY SUSTAINABLE ALGORITHMS		
4.1	Introduction	60
4.2	Proposed swift mode algorithm	61
4.3	Proposed energy sustainable snapshot technique	63
4.3.1	ESSA framework	64
4.3.2	ESSA algorithm	66
4.4	Experimental methodology	68

4.4.1	Experiment setup	69
4.4.2	Evaluation	70
4.5	Results	70
4.5.1	Usage scenario 1	71
4.5.2	Usage scenario 2	72
4.5.3	Internal scenario	73
4.6	Conclusion	80
 CHAPTER 5 PERFORMANCE EVALUATION OF FRAMEWORK		
5.1	Introduction	82
5.2	Thread monitoring	83
5.2.1	Thread monitoring in swift mode	85
5.2.2	Thread monitoring in exhaustive mode	85
5.3	Analyze memory performance	87
5.3.1	Heap analysis	87
5.3.2	Memory leakage	88
5.3.3	Thread analysis	90
5.4	Analyze cpu performance	92
5.4.1	CPU performance in swift mode	92
5.4.2	CPU performance in exhaustive mode	93
5.5	Conclusion	96
 CHAPTER 6 CONCLUSIONS AND FUTURE SCOPE		
6.1	Conclusions	98
6.2	Future scope	101
REFERENCES		102

LIST OF FIGURES

Figure No.	Title of Figure	Page No.
Figure 1.1	Worldwide market segments of the Server, Desktop and Mobile PC.	2
Figure 1.2	Percentage of power usage by ICT devices.	3
Figure 1.3	Power consumption of various devices during the year 1988 to 2001.	6
Figure 1.4	Personal computer power management and communication paths.	11
Figure 2.1	Power saving schemes in operating systems (a) Windows 95 to XP and (b) Windows Vista & Windows 7.	19
Figure 3.1	Relationship between the workload intensity and power consumption.	43
Figure 3.2	Power consumption as a function of utilization such as: a) System utilization and b) CPU utilization.	43
Figure 3.3	Power consumption (%) with CPU utilization.	44
Figure 3.4	Energy management schemes for (a) existing systems and (b) proposed systems.	48
Figure 3.5	Framework of the proposed power saving scheme.	49
Figure 3.6	GUI implementation of the proposed algorithms.	50
Figure 3.7	Repository of losable and non-losable software's.	51
Figure 3.8	User prompt to input login duration time.	52
Figure 3.9	Package dependency diagram of the proposed framework.	54
Figure 4.1	Framework of the energy-sustainable snapshot technique.	64
Figure 4.2	Snapshots of total CPU usage (%): (a) below threshold value and (b) above threshold value.	66
Figure 4.3	Total CPU usage (%) for idle computer systems in the cluster.	72

Figure 4.4	Total CPU usage (%) for active computer systems in the cluster.	73
Figure 4.5	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.	76-79
Figure 5.1	Various active threads during the framework monitoring in swift mode.	85
Figure 5.2	Various active threads during the framework monitoring in exhaustive mode.	86
Figure 5.3	Analyze memory performances for allocated heap size vs. used-heap. For each graph, x-axis denotes the time in (HH:MM) and y-axis shows the used-heap size in (MB) (a) Swift Mode and (b) Exhaustive Mode.	88
Figure 5.4	Analysis of memory performance for Surviving generations vs. Relative time spent in GC. 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 (%). (a) Swift Mode and (b) Exhaustive Mode.	89
Figure 5.5	Analysis of memory performance using ESSA algorithm for threads versus loaded classes. 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. (a) Swift mode, (b1) and (b2) Exhaustive mode. (b2 is an extension of b1).	91
Figure 5.6	Analysis of CPU performance in Swift mode (a1) call tree methods for AWT-EventQueue-0, Thread-8 and main (a2) call tree methods for Thread-7, Thread-10 and Thread-9.	92, 93
Figure 5.7	Analysis of CPU performance in exhaustive mode (a1) call tree methods for AWT-EventQueue-0, Thread-8, main, and Thread-7 (a2) various user threads to monitor user activity (a3) few expanded user threads with methods.	94, 95

LIST OF TABLES

Table No.	Title of Table	Page No.
Table 3.1	Detailed view of each package and JAVA program files.	55
Table 4.1	Cluster configuration.	69
Table 4.2	Usage scenario 1.	71
Table 4.3	Usage scenario 2.	72
	Maximum CPU usage (%) by each cluster machines (M1	
Table 4.4	– M15) in active (A) and in idle (I) mode up to 20 minutes.	75
Table 5.1	Thread details.	84

LIST OF PUBLICATIONS

- [1] **P. K. Gupta** and G Singh, “Energy-Sustainable Framework and Performance Analysis of Power Scheme for Operating Systems: A Tool,” International Journal of Intelligent Systems and Applications, vol. 5, no. 1, pp. 1 – 15, 2013.
- [2] **P. K. Gupta** and G Singh, “Energy-sustainable snapshot algorithm for operating systems to minimize power consumption,” Elsevier journal of Sustainable Computing: Informatics and Systems, 2012. (Under revision)
- [3] **P. K. Gupta** and G Singh, “Minimizing power consumption by personal computers: A Technical Survey,” International Journal of Information Technology and Computer Science, vol. 4, no. 10, pp. 57- 66, 2012.
- [4] **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, vol. 1, no. 2, pp. 145 – 163, 2011.
- [5] **P. K. Gupta** and G Singh, “User centric framework of power schemes for minimizing energy consumption by computer systems,” IEEE International Conference on Radar, Communication and Computing, (ICRCC-12), Jan. 2013.

ABSTRACT

With the explosive development of the Information and Communication Technology (ICT) devices, the increase in energy consumption and heat dissipation by these devices poses the problem of an energy crisis and exacerbation of the greenhouse gas problem with global warming. Recently, ongoing focus on the environmental sustainability proliferating in various domains and sustainable computing/green computing has been getting increased attention. The principle aspects of sustainable computing is the reduction of energy requirement for running any computing infrastructure, energy longevity of computing equipment to reduce need for their replacement and ensuring energy consumption within the energy available from the renewable energy sources in the environment. Therefore, these factors are getting more attention and a lot of work has been carried out on “*Green computing*”, which represents an environmentally responsible way for the above discussed scenario by reducing the energy consumption, and also address to various environmental related issues such as waste management, greenhouse gases etc. An emerging issue of the power dissipation has imposed a very significant question on the system and software design and it is well understood that in the future, there will be a great demand for energy-sustainable software. Therefore, the vision of a sustainable planet and minimization of the energy consumed by computer systems motivated us to find more energy-sustainable computing methods for personal computers.

Earlier, the computer systems were extremely inefficient with low computing power and high energy consumption. In newer computer systems, the average power requirements are decreased by almost 50% (nearly 100 watts to 50 watts) and standby power consumption stayed relatively constant at approximately 25 watts. The overall computer system power consumption is increasing however, the standby power consumption is decreasing.

In this thesis, we have emphasized on the need of minimizing this standby power consumption of 25 watts, which can be used for other purposes and discussed various

techniques used to minimize the energy consumption of computer systems. Currently, Advanced Configuration and Power Interface (ACPI) is used by various designed operating systems for personal computer systems to reduce the energy consumption, which sets the display to low power-modes after specified periods of inactivity on the mouse and keyboard and its efficiency strongly depends upon the inactivity intervals set by the users. There are various techniques which are categorized into different categories such as hardware based approaches, simulation based approaches, software based approaches, and energy-sustainable approaches. In this thesis, we have used the various case scenarios of power schemes available in Windows operating systems to evaluate the working of proposed user centric framework and algorithms. As we know that these available power saving schemes in Windows operating system are used to minimize the power consumption of computer system but in case, if these power saving schemes are not configured properly they do not provide power saving and most of the time computer systems remain switched on and this not only consumes the energy but also not good for the environment due to its continuous heat dissipation.

We have designed an energy sustainable user centric framework for understanding of user-application behavior and their interactions with machine sub-systems. We have presented a power model of a system that demonstrates the relationship between the serviced workload and power consumed for it. This is a common power model of the computing equipment, which provides the information about power consumed over the system utilization. To represent the system utilization, we have used the CPU utilization as a scalar value for this model, which demonstrates that the CPU's power consumption increases linearly with its utilization. We have developed the graphical user interface (GUI), based on this power model for user centric energy management. In contrast to the existing energy management policies, which are device centric either ignore the user by assuming unchangeable operational environment of the device or rely on very simplified policies that causes to large energy losses. The proposed and developed energy sustainable user centric framework implements the two different working modes known as Swift mode and Exhaustive mode, for power saving. In the comparison scenario, we have compared the existing power schemes of Windows operating systems with the proposed two modes of operation. In discussed comparison scenarios we found that Swift mode provides 66% of energy saving and Exhaustive mode provides 93% more energy savings over the existing power schemes.

We have also proposed the algorithms for aforementioned energy sustainable user centric framework. To design an energy efficient computer system ultimately require the development of fundamental frameworks, algorithmic techniques, and principles that can be used to guide practical solutions. However, several dynamic voltage and frequency scaling (DVFS) techniques has been evolved, which work around CPU utilization. These techniques explore the opportunities for minimizing the energy consumption by computer systems. Here, we proposed the algorithms like Swift algorithm and Energy Sustainable Snapshot Algorithm (ESSA) for the two aforementioned modes, respectively and both the algorithms are user centric that check the user activity on the computer system continuously and switches the system into hibernate or shutdown whenever no user activity is found on the computer system. The proposed ESSA is very much effective and considers for total CPU usage of the system. We have been focused on user activity and proposed a concept of the repository for the installed programs on the computer system. The main role of this repository is to provide zero loss to user's data while switching the computer system to power saving mode.

We have also evaluated the performance of proposed energy sustainable user centric framework under both the modes and obtained the various real time results. As the main objective of the proposed framework is to reduce the energy consumption while maintain the satisfactory level of performance. This also becomes important as there are several more problems associated with the modern software's, which include the underutilization of client resources, installation of additional hardware equipments and computer system congestion either because of complete memory or CPU utilization. We have used the profilers to find the overall appropriate level and bottlenecks in the proposed and implemented energy sustainable user centric framework. The performance measures include the various system components like thread monitoring, analyzing memory and CPU performance under specific workload and obtained various results from this profiled session, which are very much satisfactory for both the aforementioned modes without degradation on the performance of CPU, memory, or overall system performance.

The proposed energy sustainable user centric framework is very much effective for minimizing the energy consumption by the computer systems without compromising the performance and also finds the user's activity efficiently, which is missing in all the

user centric devices. The contributions and implications of this work for future research are also discussed in this thesis. This work can be extending by introducing the concept of image processing that will be more user's centric. As we know, that modern mobile computers and personal computers are equipped with webcam facility where we can use it to know the presence of the user on the machine. This future work will not only strengthen proposed ESSA algorithm but also will be able to make more appropriate decision to minimize energy consumption.

Chapter 1

INTRODUCTION

1.1 INTRODUCTION

Currently, globe is facing major three kinds of crisis known as Energy crisis, Environmental crisis and Economic crisis and it is predicted if these crisis are not controlled then the scenario will become worst in future. In this regard Government of India has approved the National Mission on Enhanced Energy Efficiency (NMEEE) [1] which is one of the eight missions planned under the National Action Plan for Climate Change (NAPCC). Explosive growth in the technological sector is also one of the major reasons behind these crises. The information and communications technologies are one of them as in this sector various technological breakthroughs have been taken place and more are yet to come. The rate at which information and communication technology (ICT) devices are being produced is proportional to the increase in the energy consumed and heat dissipated by these devices, which poses the problem of an energy crisis and the exacerbation of the greenhouse gas problem and global warming. We cannot escape the fact that the world is becoming more and more dependent upon the use of ICT. All over the world, personal computers are being increasingly used right from kids to professionals in the course of their everyday lives. In the late 1990s, power, energy consumption, and power density had become the limiting factors not only for the system design of portable and mobile devices, but also for high-end systems [2, 3]. The design of computer systems has been changed from the performance-centric stage to power-aware stage [4]. The predicted worldwide growth rate [5] of sales of Servers, Desktops, and Mobile computers up to year 2015, is shown in Fig.1.1, which reveals that the demand of PCs from 1990s onwards have been driven by the evolution of PC from command line-driven machines with floppy disk drives and capable of limited tasks, to user friendly, powerful PCs with Pentium processors and add-ons capable of doing anything. The decreasing cost of personal computers also allowed more people access to personal computers and added to their increasing

popularity. During the 1990s, personal computers continued to increase in popularity and this was the time when internet users had started increasing. Romm et al. [6] have estimated that the number of Internet users in the United States soared from 5 million in 1993 to 62 million in 1997, to over 100 million by mid-1999. Kawamoto et al. [7] have estimated that annual shipments of the computers increased by a factor of five in the 1990s. Matthews et al. [8, 9] estimated that in 1998 about one in four personal computers sold were laptops. There could be basically two major reasons behind this growth, first the technology is becoming cheaper with each passing day, and secondly people are getting more and more addicted to these ICT products and use of Internet is one of them. However, the offshoot in getting used to these technologies are the innumerable other adverse effects caused to our environment, health and economy [4]. Explosive growth rate of the sales of personal computer is proportional to the energy consumption, which can be observed from Fig 1.2 that personal computers occupy the largest share among the several available ICT products in the market and making them also responsible for the high quantum of power consumption [10].

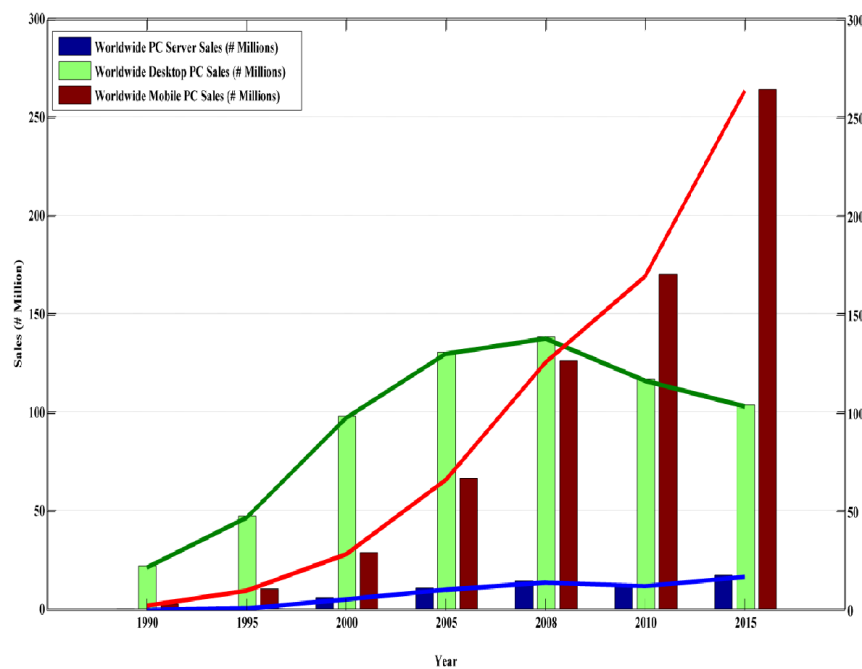


Figure 1.1: Worldwide market segments of Server, Desktop and Mobile PC.

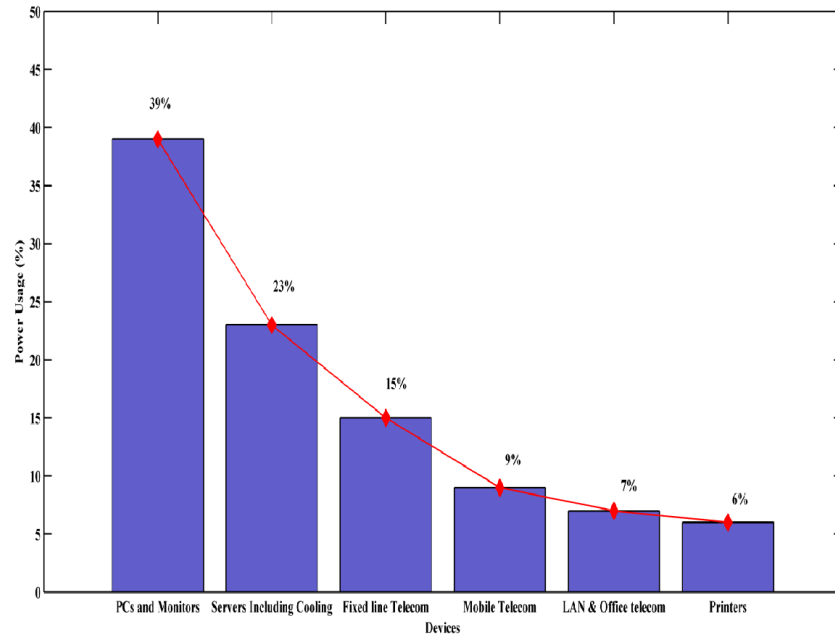


Figure 1.2: Percentage of power usage by ICT devices.

This emerging issue of power dissipation has imposed a very significant question on the system and software design and it is believed that in the future there will be a great demand for the energy-sustainable software. Therefore, the vision of a sustainable planet and the minimization of the energy consumed by computer systems motivated us examine energy-sustainable computing methods [11].

There are vast majority of the users, which leaves the computer systems running on all the time. There are various myths among the users related to powering off personal computer. These myths are listed as follows [4, 12]:

- a) One of the main myths is that turning off the computer system and then back on uses more energy than leaving it on. Whereas, the reality is the power used by a computer to boot up is far less than the energy your computer uses when left on for more than three minutes.
- b) The second major myth is that computer system is designed to handle 40,000 on/off cycles. If you are an average user, significantly more cycles than you will

initiate in the computer's five to seven year life. Whereas, the reality is when you turn your computer off, you not only reduce energy use, also lower the heat stress as well as wear on the system.

- c) Screen savers save energy. This is a common misconception among the users of the personal computer whereas, the reality is totally different as the screen savers were originally designed to help prolong the life of monochrome monitors and nowadays these kinds of monitors are technologically obsolete. This is one of feature of the Windows operating system, which exists till now from its initial version-3.1. These Screen savers save energy only if they actually turn off the monitor's screen.
- d) LCD monitors use less energy than CRTs, so we can leave it on all the time is another common myth. Whereas, in reality these LCD monitors are considered to be "vampire energy users," meaning the display will still be drawing power, even in Standby mode. Moreover, if we consider the case of business organization where hundreds to thousands of LCDs are in use simultaneously, this adds up in cost.
- e) Network connections are lost when computers go into low-power or Standby mode. Whereas in reality this was true with the older computer system. Newer computer systems are designed to Standby on networks without loss of data or connection. Central Processing Units (CPUs) of these computer systems are designed with Wake on LAN technology and can be left in Standby mode.

1.2FACTS RELATED TO USE OF ELECTRICITY BY PERSONAL COMPUTERS

In this section, we have listed some interesting facts about the electricity used by personal computers that focuses on the need of proper use of these systems [12]. As proper management of these systems not only save energy but also good for environment [4].

- An average desktop computer system requires 85 Watts just to idle, even with the monitor off. If that system were in use or idling for only 40 hours a week instead of a full 168, over \$50 in energy costs would be saved annually.

- One computer system left on 24 hours a day costs you between \$120 and \$175 in electricity costs annually while dumping 1,500 pounds of CO₂ into the atmosphere.
- A tree absorbs between 3 and 15 pounds of CO₂ each year that means up to 500 trees are needed to offset the annual emissions of one computer left on all the time.
- If each household in a metro city turned off its computer for just one additional hour per day, it would save \$3.2 million in electricity costs and prevent 19,000 tons of CO₂ from heating the atmosphere.
- The added heat from inefficient computers can increase the demand on air conditioners and cooling systems, making your computing equipment even more expensive to run.
- Even though, presently most of the today's desktop computers are capable of automatically transitioning to a Standby or Hibernate mode when inactive, but about 90% of the systems have this function disabled.
- Some 25% of the electricity used to power home electronics like computers, DVD players, stereos, and televisions is consumed while the products are turned off because anything that uses a remote continues to consume power even when they are turned off. This phenomenon is called "*vampire energy use*" or "*phantom energy use*" where a device draws Standby power in home.
- This vampire energy loss represents between 5 to 8% of a single-family home's total electricity use per year. This is on average equals one month's electricity bill and adds up to at least 68 billion kilowatt-hours of electricity annually.
- On a global scale, standby energy accounts for 1% of the world's carbon emissions.
- Electricity production is the largest source of greenhouse gas emissions in most of the countries like United States, India and China, ahead of transportation.

1.3 POWER CONSUMPTION BY PERSONAL COMPUTERS

Very early, the computers were extremely inefficient with low computing power and high energy consumption. The energy efficiency of computers increased from the mid-1980s until the mid-1990s, as demonstrated in [13-16] and also depicted in Fig. 1.3. In early 1990s, personal computer's energy consumption first entered the literature of the

energy conservation communities [17, 18]. The power requirements of computer system have changed considerably since the 1980s and are indicated in two modes, active mode is when the device is in operation and Standby mode refers to a mode which attempts to conserve power with instant recovery. In 1988, Norford and Dandridge [17] have reported that newer models of computers with equivalent performance were often more energy efficient. In newer systems, the average power requirements is decreased by almost 50%, from nearly 100 watts (W) in the mid-1980s to 50 W in the mid-to late 1990s, and standby power consumption stayed relatively constant at approximately 25 W. The computer power consumption is on the increase, however, while standby power consumption is decreasing. The Pentium 4 computer systems consume more power than its predecessors at 67 W in active mode, while consuming only 3 W in Standby mode [19, 20]. There is much more variation in the power requirements of the modern computer systems, due to the addition of consumer-specified features, such as increase in speed of Hard disk drives (HDDs) capacity and add-on cards, which vary power requirements of similar models [21-24]. In Fig. 1.3, the power consumption of average computer system was compared [4]. We can also find that the modern HDDs require significantly less power than earlier models (10W compared with 35W), as does the motherboard (25W compared with 52W in 1988) [4]. The modern CPU is one of the few computer components that use more energy than earlier models. The CPU was not recorded in the study because it consumed minimum power – compared with an average of 34W in 2001 [17, 18].

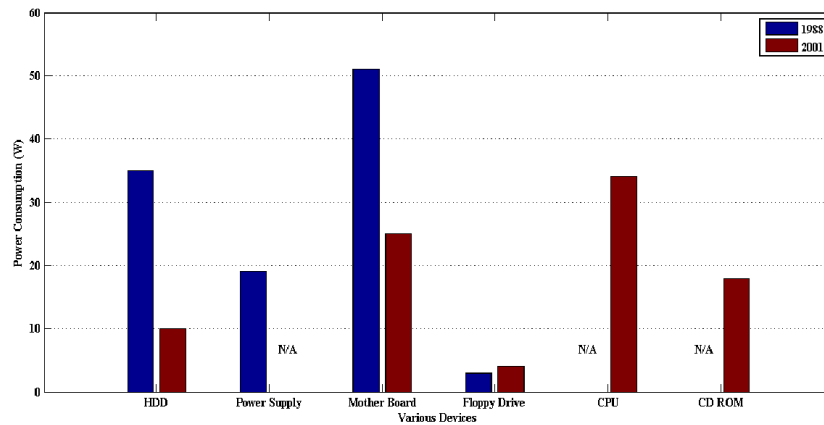


Figure1.3: Power consumption of various devices during the year 1988 to 2001.

The power required by the CPU can be expressed as “the product of the processing speed, the number of transistors being switched, and the energy required to switch each transistor, which in turn varies with the capacitance and the square of the voltage” [17]. The number of transistor in a CPU has increased faster than the transistor size has decreased, as indicated by the growing overall dimensions of CPUs [17]. Power consumption of an electronic chip depends mainly on the type of transistor used like NMOS and CMOS. Both types of transistors drew similar amounts of power when switched, but CMOS requires almost no power when in a quiescent state. Use of CMOS transistors has reduced power consumption by around 30 to 40% [17, 18].

1.4 ECO-LABELING

Eco-labels are a primary tool available to inform consumers about the environmental characteristics of the products [4]. They appear as a label or logo that gives consumers actual data on the product and/or lets them know that it meets a fixed set of environmental criteria. Eco-labels are often characterized in three categories: type-I, type-II and type-III. Type-I, Eco-label is essentially a “certificate of approval” for the product and given by the third party organization like government agency. Type-II Eco label, a company declares that its product meets independent standards such as for recyclability or energy-efficiency [4]. A type-III, Eco-label is designed to provide a set of quantitative environmental data to consumers so that the consumers can use this information themselves to evaluate the product. Eco-labeling of personal computers poses a number of difficulties, however, one major difficulty is that personal computers are complex and rapidly changing products and in many cases new models are introduced every six months means that the environmental issues associated with personal computers can change more rapidly. The criteria for an Eco-label for a personal computer should address the major environmental issues like energy consumption, effect of heat dissipation, the impacts of hazardous substances, and possible exposure to chemicals during production processes. A great variety of eco-labels for computer systems have been introduced over the years and most existing labels are of type-I. On this way, the Energy Star program [25-27] launched by the United States Environmental Protection Agency (EPA) in June 1992 is one of them, which is designed to promote energy efficiency via a voluntary, EPA-certified, Eco-label on a wide variety of equipment. This label has been widely adopted by number of

organizations and computers and monitors were addressed from the program's inception. Swedish Confederation of Professional Employees introduced the certification and labeling system developed and managed by the TCO which focuses on the workplace environment issues, such as limiting the electromagnetic radiation from cathode ray tube (CRT) monitors, sound emitted by devices, ergonomics, and electrical safety. Currently, there are two types of TCO labels: TCO' 95 and TCO' 99 [4, 28, 29]. The Japan Electronics and Information Technology Industries Association (JEITA) developed the PC Green label [4, 30]. Its criteria cover energy use (must satisfy Energy Star), content of hazardous material (similar to TCO' 99), and ease of recycling issue. Nordic Council of Ministers have introduced Nordic Swan certification label which is used in Finland, Iceland, Norway, and Sweden [4, 31]. At last, India's ecomark is an earthen pot [32], which is chosen as the logo for the Ecomark scheme in India. Government of India also set up Bureau of Energy Efficiency (BEE), a statutory body under Ministry of Power, on 1st March 2002, under the provisions of the Energy Conservation Act 2001 [33]. One of the regulatory functions of BEE, under this Act, is to develop minimum energy performance standards & labeling, for equipment/appliances and buildings, starting from one star for the least energy-efficient, and going up to five stars, for the most energy efficient. These star labels have been created to standardize the energy efficiency ratings of different electrical appliances and indicate energy consumption under standard test conditions. BEE star label is now mandatory for equipments from January 7, 2010 onwards. Similarly, there are various others eco-labels exist like Blue Angel from Germany [34], E.U. Flower from European Union [35], and Eco Mark from Japan [36] are roughly similar in the issues they cover.

1.5 POWER MANAGEMENT IN PERSONAL COMPUTERS

Power management technology has been developed for personal computer to reduce the energy consumption when they are not in active use. This not only provides the environmental benefits of reduced energy consumption, power management but also can improve the equipment reliability by reducing the waste heat. The heat produced by these devices has the adverse effect on the environment as well as on human health [37, 38]. First developed for laptop computers, power management is now common in desktop computers. As of early 1996, the EPA estimates that upwards of 70% of all new personal computers and nearly 100 % of all personal computer monitors sold have

power management capability [39]. Computer systems are supposed to use a variable amount of power when they are switched on and this depends on their configuration, add-on devices, and the various software processes running on them [11, 12, 40]. Though power management interact with every part of computer system but still there is the potential for unexpected interactions between power management and computing environment. Computer manufacturers have addressed this problem by making power management more flexible and more compatible with current personal computer networks. As the technology has matured, power management has emerged as an effective tool for saving energy. Early power management systems had long recovery times, awkward configuration methods, and low energy savings. However, the power management has improved rapidly, becoming more powerful, reliable, and easier to use. It also now delivers considerably more energy savings. In 1992, U.S EPA has introduced the Energy Star [25, 26] program which is among the one of Eco-label that provides guideline for power saving by the computer system. In 1993, Intel and Microsoft introduced Advanced Power Management (APM) [41, 42], which is becoming an industry standard. The APM protocol supports power management by defining how power management commands are communicated within the personal computer system. Power-management does not reduce the performance of a computer, but simply adds features to reduce their power consumption when not in use. These energy-efficient machines save money on electricity bills and reduce pollution from power plants. Most power management savings come from reducing power when the machine is not fully active by adding low-power or “sleep” modes that kick in when idle.

1.5.1 ADVANCED POWER MANAGEMENT

APM saves energy by putting the computer and monitor into a low power mode during periods of inactivity by temporarily reducing their speed or functionality. In response to Energy Star program in 1993, Intel and Microsoft first introduced APM [41, 42], which defines how power management commands are communicated within the personal computer system and established an industry standard in power management. Before the release of Microsoft Windows 95 operating system software was only minimally involved in desktop personal computer power management. At that time application software was only used to monitor power management not for controlling the personal computer itself. Thus the basic Input/output system (BIOS) was, and remains, a critical

component. Later, in 1998, Microsoft and Intel have developed the Advanced Configuration and Power Interface (ACPI) [43-45] and primary control of power management shifted from BIOS to the operating system [4] in the form of power schemes. ACPI has allowed manufacturers to produce computers that automatically power up as soon as the keyboard is touched [23, 24]. 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 power consumption. Advanced power management interacts with every part of the computer – the operating system, software CPU, and various other peripheral devices. To manage the power 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. As shown in Fig. 1.4, computers are logically organized as a hierarchy of layers [4, 46]. Those at the top are the software that the user directly interacts with, those closer to the bottom direct the physical control of electrical signals. Power management can involve the application software and the operating system, and always requires an action by the firmware (BIOS), processor and peripheral hardware. However, the control signals must still pass through each intermediate layer for action to occur. The working of these power management schemes is also shown in Fig. 1.4, which accomplishes four different levels. First is to monitor activity levels of the processors, input devices like keyboard mouse and other devices. Second, component is to utilize timers to decide when to switch the computer system to low power mode. Third component, changes in the power management status need to be communicated to the correct device and must occur actually. Finally, power management must find out when the activity gets resume and return to a full power mode. In Fig. 1.4, The BIOS monitors the keyboard, mouse and other input devices activity (as per number 1) and sends periodic signals to the operating system to begin power management (as per number 2). Here the operating system will only pass the signal through if it detects no activity from the user application software (as per number 3) and triggers the start of the power management timers in the BIOS. In case, no activity is detected, the operating system passes the signal back to the BIOS (as per number 4) and once the time-out occurs the BIOS will initiate power management by sending a message to various connected devices like CPU, HDD etc. (as per number 5). After initiating a change in mode, the BIOS begins

another timer which indicates when to initiate the next lower power management mode. The BIOS continues to monitor keyboard, mouse, and network activity. If activity is detected, the BIOS will send the appropriate messages to return the personal computer to an active state. The timing of the power management modes is determined by the settings (usually in BIOS or in Power Scheme), specifying the delay between each power management mode and the next. Each successive power management mode represents a decrease in energy consumption and CPU function, and therefore more time is required to bring the computer back to active mode [4, 46].

In addition to the direct electricity savings, power-managed computers generate less heat, and since most offices have to cool the air more than they heat it, for every four kWh of energy saved by the computer, an additional kWh is saved in the cooling and ventilation system [47, 48]. Power management in personal computer relies on the fact that for most of the time a typical personal computers is on, it is not doing anything productive. As long as the computer is idle, energy use can be reduced without interfering with work.

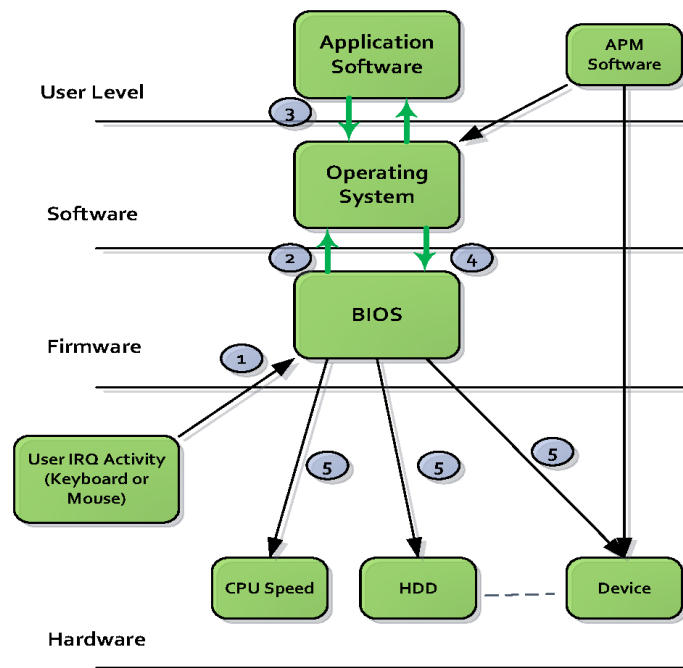


Figure 1.4: Personal computer power management and communication paths.

Common methods used to reduce the energy use are slowing down or stopping the processor clock, spinning down the hard disk, and turning off entire system components such as video or sound cards or disk controllers.

1.6 POWER SAVING MODES AND PROBLEMS WITH POWER MANAGEMENT

In computer systems, for minimizing the energy consumption various power saving modes are defined by the power scheme of operating system [2, 4]. The following section lists about the various power saving modes implemented either by APM or by operating systems in personal computer to save the energy.

Full-Power-on: In this mode all the connected devices and components with the computer systems are fully powered with no power management activated. This mode supports no energy savings.

APM-Enabled: In this mode, power management features are activated and based on the BIOS settings CPU is slowed or stopped whereas other connected devices and components draw the power similar to full-power-on mode. This mode supports up to 25% of energy savings.

Standby mode: In this mode, power management features works actively and CPU may stopped its operations depending upon the activity detected on the personal computer as well as connected devices and components also gets switches to low power saving mode. This mode claims up to 30% of energy savings. In case of any activity is detected may trigger the personal computer back to enabled or full-power-on mode.

Suspend mode: In this mode, CPU is stopped and most power managed devices or components are not powered except network card and provides the maximum power savings under APM. This mode claims up to 45% of energy savings. In this mode, any activity can trigger the change of state of computer system from suspend to standby, enabled or full-power-on mode.

HDD-off mode: this mode is not a part of APM; hence it is not a system mode and normally managed by the operating system. In this mode, HDD spin is stopped to save energy used by HDDs. This mode is independent of other power saving modes and hence other connected devices or components may remain at full-power-on mode or in

enabled, standby and suspend mode. This mode claims up to 10% of energy savings only and any activity on the personal computer can facilitate quick reactivation of HDD to operational mode.

Hibernate mode: This mode is also not a part of APM mode and normally managed by the operating systems. In this mode, all available memory contents and current state of the PC saved to the HDD and further PC gets switch off. When user wants to work again on hibernated PC, then he/she has to power on the PC and system takes 15 to 60 seconds to recover the user's state. This mode claims 90 to 100% of energy savings. This mode must be properly configured to take its advantage.

Shutdown mode (Off mode): In this mode, the computer system, various connected devices and components gets switch off. As compared to previous hibernate mode no operational parameters are saved to the HDD. This mode supports 96 to 100% of energy savings and whenever user wants to perform some operations on the computer system then he/she has to switch the system to full-power-on mode.

Though, with the change of time, the power management feature has been shifted from BIOS to Operating system and more power saving modes are being designed to save power but a small amount of power ranges from 3W to 15W is consumed in these modes [4]. However, most of the time, these power saving modes are not properly configured or most of time users rely on the default settings of power saving modes, which allow for only up to 20% in energy savings [4, 12]. The difficulties in properly configuring power management in computers and monitors are the largest barrier for achieving energy savings from automatic energy management. Earlier, many power management systems for both monitors and computers had long recovery time, awkward configuration methods, and low energy savings [46, 47]. In [49], Webber et al. estimated that 80 percent of monitors and 50 percent of computers are Energy Star-activated. A study of power management features and configuration of Energy Star-compliant machines found only 11 percent of CPUs fully enabled and about two-thirds of monitor's power managed [23, 24] whereas Kawamoto et al. [7] have obtained in their study that only 25 percent are correctly power-managed to achieve maximum power savings [7]. It is very difficult to determine whether power management is properly operating in machines. The only indication to the user that power management is occurring in their computer is when the HDD audibly spins down or delays in the

appearance of keystrokes when spinning up. Other than this, it is difficult to know whether the computer is accomplishing any further power management [23, 24]. There is an illusion among the users of personal computer that their system is power-managed because the Energy Star logo appears during start up. Many users do not realise that they must first activate the power management features to save energy [15, 50].

Once enabled, the power management may present some further challenges. Power management interacts with every part of the computer, and therefore there is potential for unexpected interactions, which may cause problems [51, 52]. In most cases, however, automatic power management is not a substitute for switching the machine off when not in use for extended periods of time. However, substantial energy savings can also be made by switching the personal computer off, since it has been found that more than 50 percent of computers are left on at night [4, 12, 17, 18].

1.7ENERGY-SUSTAINABLE COMPUTING

To handle the issue of power management in efficient manner in the computer systems, the concept of ‘Sustainable Computing’ is gaining a lot of popularity presently and is being considered as one of the most promising technology by the designers of Information Technology (IT) industry. Sustainable computing is also known as ‘Green computing’ of which computing methods provide the benefits of solving the energy-consumption issue by computer systems and being environmentally friendly [11, 53]. There are basically three major aspects of sustainable computing: i) reduction of energy consumption from any running computing process on the system ii) to ensure the longer life cycle of any computing equipment, and iii) ensuring the need of energy consumption within the energy available from all resources in the environment [11]. Therefore, the sustainable computing performed from energy perspective is known as energy-sustainable computing. A major issue in addressing the different aspects of sustainable computing is the need for awareness of the non computing processes in the physical environment like dependency of the equipment life cycle on the environmental factors and the availability of energy from the available resources in the environment. On this way, energy sustainable computing can also be defined as the balance between the power required for computation and power available from sources like renewable sources, green sources etc. However, the power required and available power may vary according to the time as solar power will not be available in the night and computing

operations will become unsustainable if the required power is higher than the power available. This varied nature of energy sustainability can be defined as follows:

1.7.1 Energy Sustainability

Energy-sustainable computing needs to ensure minimum energy requirement from the grid or battery and emphasizes a lot on green sources. In case, if there is no or less energy available from green sources then energy sustainability brings down the average energy required by reducing the computing operations. There are different directions in achieving energy-sustainable computing [2] such that need for grid and battery power is minimized.

a) Energy storage

Here, energy storage devices are used to store the energy available from green sources with the help of various available techniques like ultra-capacitors, compressed air storage, batteries, fuel-cells, and flywheels [16-18 paper]. This storage of energy is constrained by the energy capacity limit of the storage device.

b) Reducing energy requirement

Another major direction for energy sustainability is to reduce the energy requirement to avoid unsustainable operations or to reduce the energy need from grid or batteries which, can be achieved in variety of ways: 1) by using spatio-temporal distribution of operations, where computing operations are distributed among multiple computing units and no machines gets overloaded. These types of spatio-temporal distributions are used in data centers [12, 24, 25, paper]. 2) by using computing power management methods to reduce the power requirements. This can be achieved by switching the computing units in to different power saving modes for example, processor not performing any operation can be switched to sleep mode or hibernate mode to reduce the power requirement [paper, 27, 28, 29]. 3) by managing non-computing systems where power requirement by the computing units is followed with the requirement from some associated non computing processes like for cooling, for server longevity etc.

c) Scavenging energy from various sources: This is another complimentary option for energy-sustainable computing which focuses on need of energy

harvesting and requires identification of different energy sources to scavenge energy from them [2,7,30,31 - paper].

1.8 ORGANIZATION OF THE THESIS

This thesis aims to understand the limitations of existing power schemes available in the operating systems and to investigate new ways of designing energy sustainable framework and algorithms for minimizing the power consumption by personal computers. The framework discussed in this thesis is a user centric energy sustainable framework. The remainder of the thesis is organized as follows:

Chapter 2 concerns with the various ways of the minimizing energy consumption in the computer systems. Here, we have represented various simulation based, hardware based, and software based scenarios to minimize the power consumption and focused on the software based sustainable techniques. We have also discussed the need of enhancement of the dynamic power management and dynamic voltage and frequency scaling methods to achieve this goal.

Chapter 3 presents the user centric energy sustainable framework. In this chapter, we have modelled the energy and power consumption, which is build around the CPU utilization and presented a user centric energy sustainable framework. This framework also considers the total CPU utilization and implements two different modes known as Swift mode and Exhaustive mode for the power saving. We have also discussed about the various advantages of the proposed framework over traditional power schemes available in the operating systems.

Chapter 4 focuses on the algorithmic implementation of the proposed energy sustainable framework. This framework implements two algorithms known as Swift algorithm and Energy Sustainable Snapshot algorithm (ESSA) for the proposed modes, respectively. The various results are obtained using these algorithms for the respective modes using scenario when there is no load on the machine and when there is processing going on which shows that the CPU of the machine is utilized. Here, only the proposed ESSA algorithm is designed around the percentage of total CPU usage, which is recorded for each minute whereas the proposed swift algorithm focused on its smooth functioning for the supplied login-duration time.

Chapter 5 evaluates the performance of proposed energy sustainable framework for its two different modes in the real time environment. By using profiling, we have evaluated the framework for thread monitoring, memory performance like heap analysis, thread analysis, memory leakage, and garbage collection is done. Finally, we have also analyzed the CPU performance by using three methods for finding invocations of each method during login-duration. Using profiler, various results have been obtained in real time environment and are presented in this chapter.

Finally, we conclude the thesis and recommend the future scope of the work in Chapter 6.

Chapter 2

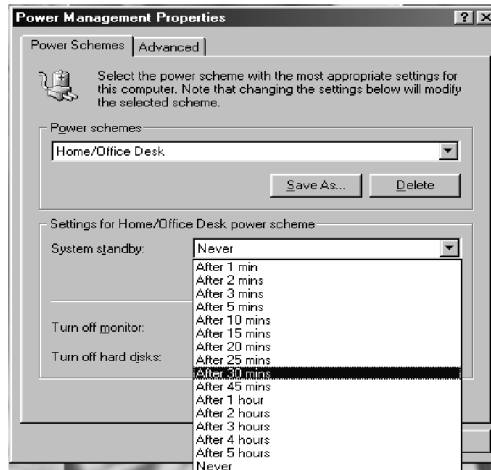
MINIMIZING ENERGY CONSUMPTION

2.1 INTRODUCTION

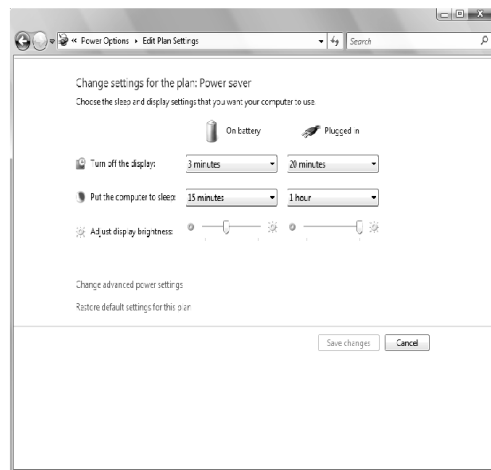
As we have seen in the previous chapter that most of the power saving schemes if not configured, are unable to save the power and most of the time computer systems remain switched on and this not only consumes the energy but also not good for the environment because of its continuous heat dissipation, and sadly, the power management tools are not active on 90% of desktop personal computers [4]. Though, these power saving modes are designed to save power but a small amount of power ranges from 3W to 15W is consumed in these modes [13]. In this chapter, we have discussed about the techniques used for minimizing the energy consumption by the computer systems. Therefore, the vision of a sustainable planet and minimization of the energy consumed by computer systems motivated us to examine the energy-sustainable computing methods [10].

Over a period of time, various techniques have been proposed by the researchers/scientists for minimizing the power consumption by the computer systems. These techniques can be categorized into various classes such as hardware based approaches, software based approaches, tool based approaches, and energy-sustainable approaches. Here, in this thesis, we have used the example of Microsoft Windows power schemes [54], as it is one of the most widely used operating system since last two decades and will remain the favorite to many users. According to the various quarterly results [55], it is found that Microsoft Windows still leads the market with a market share of around 88–90% among the other key players like Linux, Mac OS, and Android. Thus, we can conclude that the Windows OS is one of the most popular and people still prefer to work with it. Though it is most widely used operating system and has become energy efficient with the introduction of energy star program [15, 16] and Advanced Power Management [41, 42]. In Microsoft Windows power management

features are implemented with the help of various power schemes. These power saving schemes are shown in Fig. 2.1 (a) and Fig. 2.1 (b).



(a)



(b)

Figure 2.1 Power saving schemes in operating systems (a) Windows 95 to XP and (b) Windows Vista & Windows 7.

Fig. 2.1 (a) shows the available power options from Windows 95 to XP, and Fig. 2.1(b) shows the available power options for Windows Vista and 7. If we compare both the scenarios, then in these power saving schemes, we see that various options are available

for power management that address the issue of human inactivity, using that a personal computer can enter automatically into the sleep mode, standby mode, hard disk drive turn-off, monitor turn-off, and switching the personal computer to hibernate. Another interesting fact that we can find from Fig. 2.1(a) and Fig. 2.1(b) are the option to turn-off the hard disk drive after a certain period of time, which is available in Fig. 2.1(a), is no longer available in Fig. 2.1(b). This reveals that the organization has started thinking seriously about the issue of power consumption by the computer systems. In one of the recent blog of Microsoft Developers Network (MSDN) [56] from the Windows engineering team of Microsoft, have emphasized in the need to minimize the power consumption of your personal computer.

2.2 DRAWBACKS OF AVAILABLE POWER SCHEME IN OPERATING SYSTEM

As we have seen in Fig. 2.1(a) and Fig. 2.1(b), number of power-saving options are available in these power saving schemes, but a significant question arises is: “Are they sufficient enough to save power?” The answer to this question is very clear and direct: maybe not because all these available options as discussed in previous section are totally based on the time-out approaches, they consider the keyboard and mouse event for switching back on the system’s state in any of the options available in the power schemes. Presently, the available power schemes in Microsoft Windows 95/XP/Vista or in any other operating system are not sufficient for effective power management or to save power completely and reduce the heat dissipation by the personal computers. This situation is described in the following two cases [54]:

Case A:

The user of a personal computer selects the following settings in the power scheme available in the operating system to save power:

Turn-off monitor/display = After 45 min.

Turn-off hard disks = After 45 min.

System standby/sleep = After 1 hour

The user of this system has to leave the system for some reason just after login into the system, and he/she came back to the system after 30 minute or more and continues his work.

Case B:

The user of a personal computer selects the following settings in the power scheme available in the operating system to save power:

Turn-off monitor/display = After 10 min.

Turn-off hard disks = After 10 min.

System standby/sleep = After 30 min.

The user of this system has to leave the system for some reason just after login into the system, and he/she came back to his system after 30 minute or more and continues his work.

In the above scenarios, though the users have configured the power schemes differently in both the cases, the power consumption of the computer system is very low in Case B as compared to Case A even though the inactivity period of the user is the same, that is, 30 minute in both the cases. So, in spite of having the power scheme options in the operating system, we could not minimize the power consumption in Case A. The one major reason for this could be that these power schemes are very much personal computer and peripheral oriented and have nothing to do with the operator's behavior, which means that they are unable to say something about the operator's inactivity [57] period on the computer system. Therefore from the above cases, we can easily find out that inactivity is the enemy of the power consumption and one cannot minimize it without predicting the inactivity period of the operator.

2.3POWER AWARE SYSTEM DESIGN

In the early age, researchers/scientist tried to minimize the energy consumption during the architecture design stage, because the power problem obstructs the development of computer system. Most of the worked out techniques by the researchers/scientist are designed to decrease the energy consumption of the processor, which is considered as a dominant energy consumer in a typical computer system. Some of these techniques

include multi-core on-chip processor, dynamic voltage and frequency scaling, clock gating, phase-change memory and solid-state disk drive are few of them used to decrease the power consumption in computer systems. Although these hardware based approaches have been proven useful for reducing the energy consumption of the computer system and they will be of continuing importance in the future and it is also found that these techniques alone are not enough. However, some higher-level strategies are to be used for reducing the energy consumption [58, 59]. From previous section, we can see that current operating systems are not by considering the power problem, thus, even if the system is running in idle mode it consumes a large amount of energy and considered as waste, which is not used for any kind of computing. Vahadat et al.[60] have proposed the traditional operating system design, which should be revisited for the energy efficiency. H. Zeng et al. [61] have presented Ecosystem, which tries to manage power as one kind of system resource, except for designing new operating systems, some other high-level power aware strategies like power aware scheduling [62] is also providing a better option for minimizing the energy consumption.

2.4 POWER MEASURING AND PROFILING

Power measuring and profiling is emerging as a new area with a rising concern over the energy consumption when designing computer systems. In the early work [58, 59, 63-65] researchers/scientist have used simulation based hardware based design to minimize the energy consumption and most of them estimate the power consumption of the hardware circuits based on the classical power model [66-68]. The major drawbacks of these circuit-level power models is that they cannot be used to estimate the power consumption of real products and researchers/scientist have used the instruments to measure the power consumption of these real products directly. To cope with the limitations of direct measurement, in this thesis, we have tried to use software methods to estimate the power of the online computer systems. These works are performed in different levels and categorized as component level [69], core level [70], CPU functional unit level [71], process level [72], and virtual machine level [73]. In these techniques either use operating system events or hardware performance counter events to build their power model. Basically, these power measuring and profiling techniques are divided into three categories: simulation based approaches, hardware-based

approaches, and software power profiling. These approaches are discussed in detail in following sections.

2.4.1 Power model

The software-based approach usually builds power models to estimate the power dissipation of different levels like instruction level, program block level, process level, hardware component level, system level, etc. These methods first try to find the power indicators that could reflect the power of these software or hardware units and then they build the power model with these power indicators and fine tune the parameters of the power model. Based on the difference of the power indicators, we categorize these methods into two categories known as: a) system profile-based method and b) hardware performance counter (PMC)-based methods.

A) System Profile-Based Power Model: As we know that system profile or systems events are a set of performance statistical information supplied by the operating system and reflect the current state of hardware and software. For example, CPU utilization is the performance metric that can reflect the current workload of the processor. Nearly all operating systems support these system events. Microsoft Windows support this event by supplying a set of Application Programming Interfaces, called the performance counter to access various values. The operating system, which constituted with a set of system processes, consumes the large amount of power even when the system is idle, and it is the main cause of disproportional system usage and power dissipation. In [60, 61, 74], researchers/scientist have presented the review of traditional operating system design with energy as one of the foremost important considerations. Thus, understanding the power dissipation of these system processes is very important for designing a significant energy sustainable operating system. In [75], Li and John have estimated the power dissipation caused by the operating system and find the power behavior of three types of operating system routines like interrupts, process and inter-process control, and file system. These operating system routines have different power behavior. However, author's find the power of these operating system routines has a linear relationship with instructions executed per cycle (IPC) and build the power model based on IPC. In [72], Do et al. have built the process-level energy models for three main components CPU, disk, and wireless network interface card (WNIC). Here, the CPU energy model is based on system events like active time of the CPU, the time that

the CPU worked on each frequency and frequency transition time. The energy consumed by the disk and WNIC is computed with the amount of data operated by these devices. In [73], Kansal et al. have also built the energy model for three main components, CPU, memory and disk and proposed CPU energy model which is based on CPU utilization, memory energy model is based on number of LLC (last level cache) misses, and the disk energy model based on the bytes of data that the disk reads and writes. Similar to Kansal et al. [73], Dhiman et al. [76] also proposed an online power prediction model for virtualized environment.

B) Hardware Performance Counter Based Power Model: The hardware performance based counter is a group of special registers, which are designed to store the counts of hardware-related activities within computer system. These hardware performance counters provide low overhead access to the detailed performance information related to the CPU's functional units, main memory, and other components. Bellosa et al. [77] first propose the idea of using hardware performance counter and analyzed the various hardware events like integer operations, floating point operations, second-level address strobos, and memory transactions with the power of the component and observed that these events are tightly related to the functional units. Similar to Bellosa et al. [77], in [78] Joseph et al. validated the correlation of more than 10 CPU functional units and used several performance events that are most related to CPU power. In [79], Contreras et al. build power models for the Intel PXA255 processor and memory. Their processor power model is based on for hardware events like instructions execute, data dependencies, instruction cache miss, and translation look aside buffer (TLB) whereas, memory power model was based on three counters like instruction cache miss, data cache miss, and number of data dependencies. In [80], Bircher et al. discussed the events that have higher correlation with power are all IPC related. In [70], Bertran et al. proposed a core level power model and categorized the processor components into three categories, the in-order engine, the memory subsystem, and the out-of-order engine, based on the extent that the performance monitoring counters can monitor their activities.

2.5 SIMULATION BASED POWER MANAGEMENT APPROACHES

The complexity of modern computer systems have increased dramatically and software simulation has become the only method of testing, evaluating, and prototyping the system. Simulation based approach has become the key step for the system designer for evaluation of the research ideas, to verification of hardware design, and once the system is ready then for its performance tuning. In all kinds of simulations designers face the problem between its speed and accuracy and they have to reduce the accuracy to make the simulation run in an acceptable amount of time. In [81, 82], the authors have proposed a simulation-based power model using FAST simulator and stated that traditional power models that is used with software simulators can directly benefit from this simulator. This simulator is able to tolerate the overhead communication between functional model and Timing Model. Rosenblum et al. [83] have presented a SimOS simulation environment for complete computer system modeling, which also support for full operating system and application programs that run on it. It provides speed, accuracy, and flexibility during simulation. It simulates the CPU by using the process abstraction provided by the host operating system. Ye et al. [65] presented the design of an execution driven power estimation tool SimplePower. The SimplePower energy simulator uses transition sensitive energy models and simulates the Simple scalar instruction sets which are a suite of publicly available tools for simulation of modern CPUs. It also provides the energy consumed in the memory system and on-chip buses using analytical energy models. Gurumurthi et al. [82] investigated the existing power simulators for their proposed design and obtained that they are mainly targeted for particular hardware such as CPU and memory and do not capture the interaction between other components. The SoftWatt tool proposed and 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. Brooks et al. [58] presented an architectural simulator for estimating the CPU power consumption using Wattch. This simulator is very much faster than any existing lower level power tools. They have quantified the power consumption of all the major units of CPU and show how these power estimates can be integrated into a high-level simulator. Chen et al. [84] investigated a user-level simulator at the micro-architectural

and memory level, and observed that the 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. Do et al. [72] 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.

2.6 HARDWARE-BASED POWER MANAGEMENT APPROACHES

Hardware-based power management approaches uses instruments to measure the current or voltage of the hardware devices and further uses these measured values to compute the power of the measured object. This measurement instrument includes different types of meters, special hardware devices that can be embedded into hardware platforms and power sensors that are designed within hardware. Normally, these methods make the use of micro-benchmarks [66, 71] and can only measure the component level power. Direct measurement of the energy consumed by the infrastructure is used to validate the models, but these measures suffer from two flaws: if one wants to evaluate very precisely one component of one host, sensitive power meters and oscilloscopes are used. These measures are very costly in maintenance and shipping. Other less-expensive and easy-to-use solutions measure the power directly at the electrical plug. This hardware measurement will be simplified as more and more hardware components start to include some measurement system designed especially for processors attributes like temperature, energy consumption, etc. However, the heterogeneity of such hardware and the size of large distributed systems put back the complexity on the measurement, interoperability, and management software infrastructure.

2.6.1 Power measurement with power meters

Direct power management with power meters is a common method to understand the power dissipation of devices as well as whole system. Various research [72, 85] focus on the power meters to measure the real power and use it to validate the research work or to do analysis. Moreover, some works [86, 87] focus on measuring the hardware components power and break it into lower levels based on some indicators that could

reflect the activity of these lower-levels units. The only difference in these methods are which type of power meter is used to do the measurement and at which place the measurement is performed.

A globally available power meter is digital multimeter which is easy to use and able to sample the measured object each second. In [66] Joseph et al. have discussed the use of multi-meters to measure the power while executing different benchmarks and make power trade-offs. By using the same method Feng et al.[88] measured the node level and component level power for node of the distributed system. In another type of meter, which is also used by many researchers/scientist, is the clamp meter, which measures the current without disconnecting the wire. In general, these kinds of meters has the larger measurement range than the digital multi-meter and thus can be used to measure the power of systems in which the current is quite more. In [67] Kamil et al. presented the direct power measurement method with clamp-meters to measure the power on a CrayXT4 supercomputer under several high performance computer workloads. Here, multi-meter and clamp meters are used to measure the DC power by connecting them between the power supply and the measured component. In addition, there is one more kind of power meter, such as 'Watts-up' [89] to measure the AC power which is used to measure the system level power because only power supplies are powered by AC that is good to understand the overall system behavior but it is not suitable for lower-level power analysis.

2.6.2 Power measurement by specially designed devices

As we have seen in the previous section that direct measuring with the help of various power meters is simple, however it is unable to control the measurement process. Therefore, to overcome from this problem some specially designed power measurement devices are designed for the power measurement. Viredaz and Wallach [90] has worked towards the development of flexible research platforms for pocket computing and developed a project with name ITSY. This project is used to measure the power of mobile devices and battery lifetime under different loads, while continuously monitoring the power consumption. Snowdon et al. [91] represented the structure of PLEB which is a single board on computer and designed with a set of current sensors on board. The microcontroller of this device is integrated with an analog-to-digital converter to read the sensors. PLEB platform can also be used to isolate the power of

processor, memory, and other input/output devices. In various other approaches used by the researchers/scientist [92-96], used a National Instruments (NI) data acquisition card [92, 93] as it can measure 16 components simultaneously, or by using basic laboratory equipment [94-96] that measures the voltage and current of the system.

2.6.3 Power measurement by integrating sensors

This type of approach is basically used by the high-performance servers. In the last decades, number of designed servers contains a service processor [97, 98], which is hardware and software integrated platform that works independently of the main processor and server's operating system. In this architecture, most of the hardware of service processor includes power sensors to monitor the power supplied to the administrator for power management. For example, Intel's service processor Intelligent Platform Management Interface (IPMI) [99] supports APIs to read the power information monitored by the sensors. There are few other techniques have been discussed in detail in [100, 101] to improve the energy efficiency of data centers that are used on servers integrate power sensors at a deeper level. Though online power-aware applications can use this method but it is difficult to gain lower-level power information, in which case hardware circuits are too complicated to distinguish the originality of the power dissipation. Moreover, there power monitoring circuits also dissipate a large amount of power as well.

2.6.4 Power management by using benchmarks

Several standards have been proposed for evaluating the energy consumption of nodes in cluster, of multitier servers and of supported applications and later on these standards were evaluated on specific hardware like the benchmark [102], which evaluates both the application and the infrastructure. The Green500 [103] initiative is a challenging for the most powerful machines in terms of flops/watts. SpecPower [104] is a widely used industry standard that evaluates the power and performance of servers and multimode computers. It exercises the CPUs, caches, memory hierarchy, and scalability of shared memory processors as well as the implementations of Java Virtual Machine (JVM), Just in Time (JIT) compiler, garbage collection, threads, and some aspects of operating systems and computes the overall server-side-Java operations per watt, including the idle time on specific workloads. The Transaction Processing Performance Council (TPC) proposes the TPC-Energy benchmark [105] for transactional applications like

Web application services, decision support, online transaction processing etc., TPC measures Watts/operations on the TPC benchmarks. The Embedded Microprocessor Benchmark Consortium has similar approach and provides a benchmark for energy consumption of processors [106]. Additionally, manufacturers provide information about the consumption of their components by using average loads. For example, Advanced Micro Devices (AMD) describes the average CPU power [107], which characterizes power consumption under average loads. In the best cases, all these benchmarks provide information about standard applications on specific hardware. They limit their purpose on ranking between different architectures for specific workload.

2.7 SOFTWARE-BASED POWER MANAGEMENT APPROACHES

Although the hardware approach can supply very accurate power information but due to some technical problems as we listed in the preceding sections limit its application range. However, software-based approach can be used to supply more fine grained online power information, which could be used by power aware strategies. The live power information of systems is highly needed for designing high-level energy-efficient strategies. For example, Eco-system [61, 74] proposes the concept of managing system energy as a type of resource and requires the support of real-time power information at different levels. Moreover, as a part of the energy-centric operating system, energy profiles are also needed by new power aware scheduling algorithms [108, 109]. Furthermore, these software-based power management approaches provide more flexibility in comparison to the hardware approaches and can be applied in different platforms very easily.

2.7.1 Dynamic power management scheme

Recently, the dynamic power management (DPM) scheme has become essential for the modern computer systems as well as for battery driven embedded systems. This is an approach by which one can reduce power consumption by switching system components into different states [110-118]. We consider a device with active, full-on, standby, and sleep modes with different power consumptions and controller decides when to change the power mode of the device. DPM also refers to the selective shutting or slowing down of computer system components that are idle for a long time or rarely

used. DPM schemes at the operating system level refer to the adjustment of supply voltage and clock frequency while tasks are running. These approaches can be classified into three subcategories: a) predictive, b) stochastic, and c) time-out approaches.

A) Predictive approach

In various practical systems, it is very difficult to predict the future input event and DPM decisions have to be taken based on some uncertain predictions. The rationale in all predictive techniques is that of exploiting the correlation between the past history of the workload and near future in order to make reliable predictions about future events [110]. The screen saver adopts the predictive approach for the energy conservation and when there is no keyboard activity after the screen display will be turned-off, and the same approach can be applied to disk drives. Predictive policies for the hard disk drives [119-123] and for interactive terminals [116, 117, 124] force the transition to low power state as soon as the component becomes idle if the predictor estimates that the idle period will last long enough whereas, an incorrect prediction can cause both performance and energy penalties. The regression model is used for predicting the duration of future idle periods. A simplified power management policy predicts the duration of an idle period based on the duration of the last activity period. Predictive approaches are categorized into two categories: 1) Static techniques where predictive shutdown [117] and predictive wakeup methods [116] are used. Srivastava et al. [117] conducted an extensive analysis of various system predictive approaches and proposed a predictive system shutdown strategy for event-driven applications in portable devices. They developed two predictive formulas: one based on the 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. They have claimed that the simple policy performs almost as well as the complex regression model and it is much easier to implement. In [116], Hwang and Wu have 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. Authors have used static power management and DPM techniques to define and detect the sleep modes and idle

period. 2) Another approach is an adaptive techniques where optimality of DPM strategies depends on the workload statistics, static predictive technique are all ineffective when the workload is either unknown a priori, or non-stationary. Several adaptive techniques have been proposed to deal with non-stationary workloads [117]. Douglass et al.[120] surveyed and classified several predictive policies and introduced an approach to keep only one timeout value and to increase it when it is causing many shutdowns. Krishnan et al. [125] maintained the set of time out values and each timeout is associated with an index indicating how successful it would have been. The proposed policy selects the timeout for each idle time that performs best among the available ones. In an another policy presented by Helmbold et al.[121], also keeps a list of candidate timeouts and assigns a weight to each timeout based on how well it would have performed relatively to an optimum offline strategy for past requests. The actual timeout is obtained as a weighted average of all candidates with their weights. Abbasian et al. [115] 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. In [115] it is also predicted 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.

B) Stochastic approach

These approaches are based on stochastic models and provide more optimal results in comparison to predictive techniques. Stochastic control based on Markov models has several advantages over predictive techniques. 1) It captures the global view of the system, thus allowing the designer to search for a global optimum that possibly exploits multiple inactive states of multiple interacting resources. 2) It enables the exact solution of the performance-constrained power optimization problem. 3) It exploits the strength and optimality of randomized policies. Here, the models use distributions to describe the times between arrivals of user requests, the length of time it takes for a device to service a user's request, and the time it takes for the device to transition between its power states. Simunic et al. [126-129] have discussed the system model for stochastic optimization, which can be described with the memory-less distribution [130-132] or with general distributions [126-129]. Similarly, the power management policies can also be classified into two categories by the manner in which decisions are made

like discrete time and event driven. Qiu et al.[133] proposed a abstract model of the power managed electronic system and formulated the problem of system-level power management as a controlled optimization problem based on the theories of continuous-time Markov decision processes and stochastic networks and solved this problem by using linear programming. Huang et al. [114] focused on implementing the DPM in hard real-time systems and proposed online algorithms to change the mode of the system. Here, they have considered three different modes for the system known as 1) active mode, 2) standby mode, and 3) sleep mode. However, based on the controller's decision the device can be switched to any mode to reduce the energy consumption. Using these algorithms, they predicted the next moment for mode switching. Jiang et al. [118] 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. It is also predicted that the timeout policy is equivalent to a stochastic policy in terms of the power performance tradeoffs, and this relationship is expressed as a mathematical formula.

C) Time-out approach

The time-out approach is a widely used approach in the industry, and thus it is easy and safe to implement. This policy is widely supported by number of Operating Systems, like Linux, Microsoft Windows, and so on. In this kind of DPM approach, whenever the time last at idle state reaches an assigned time-out value, this approach switches the system component to low power mode. In previous works on timeout policy, the timeout value [42, 134] was exploited to identify the elapsed idle time as observed event used to predict the total duration of the current idle period, and tested with simulations as well as measurements to assess the effectiveness. Moreover, some researchers [113, 135,136] have implemented the DPM approaches in different ways, that is, either they have considered the architecture and algorithm to shut down the HDD irrespective of the value declared by the “timeout after idleness” algorithm in most of the Windows-based computer systems, or by considering the parameter of Dynamic Voltage Scaling by proposing a smart Sleep [113, 136] power-saving scheme with minimal performance impact. This scheme adopts the sleep mode and forces in the system into sleep mode, though this may not be a good choice, as it is difficult to find such a state where server utilization is low. Recently, several analytical

models have been introduced for power-management systems controlled by timeout policy. Benini et al. [110, 111] explored several approaches to system-level dynamic power management and modeled a power-management system as a set of the interacting power-manageable components controlled by the power manager and then analyzed the DPM implementation issue in the 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 the system history, work-load, and performance constraints. Zheng et al. [137] modeled the timeout policy driven power management systems with multiple vacations and an attention span. They have presented their closed form solution based on queue theory. This analysis revealed a traffic load threshold based ‘best’ policy that is obviously equivalent to the optimal deterministic policy and therefore is inefficient to trade-off power for performance with tight constraints. Rong et al. [138] extended the continuous-time Markov decision processes (CTMDP) model for designing a timeout policy driven power management system and presented offline and online gradient-based algorithms for determining the optimal timeout value based on perturbation analysis theory. This model suffers from modelling inaccuracy and brings extra computational burden. Li [112] 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 the algorithms with optimal solutions analytically and validated their results experimentally. Wang et al. [113] 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, authors have chosen the execution speed for servers with dynamic voltage scaling and sleep periods when placing the servers into the sleep mode with DPM.

2.8 DYNAMIC VOLTAGE AND FREQUENCY SCALING

The energy awareness of high performance computing systems has been achieved by several ways which varies from electrical material to various circuit designs to system integration and system software. The main goal of each technique is same that is to reduce the overall system power consumption without compromising the performance of the system. This section presents the study on dynamic voltage and frequency scaling (DVFS) from system-software's perspective. From this perspective, the strategy of supplying the minimal power to various system components when they are not in use is achieved by powering down them. DVFS is one of the most promising technology by which one can allow the systems software to reduce the CPUs voltage and frequency. This is because when there is no processing at the computer system then the CPU is underutilized for the machine. The dynamic power of CPU depends upon the CPUs voltage (V) and its operating frequency (f) [2]. This relation is shown as:

$$P_{dynamic} \propto V^2 \cdot f \quad (2.1)$$

The dynamic power is used for switching between 0 and 1 in the CPU registers when it is not in use. Therefore from the above relation, we can find that by lowering the voltage or frequency, can minimize the CPU power consumption, which shows that the operating frequency of CPU is dependent on its supply voltage as:

$$Ef \propto V \quad (2.2)$$

The equation (2.2) reveals that we can decrease both the voltage and frequency simultaneously to reduce the CPU power consumption. DVFS allows the system software to supply high power when it is required only. With the intent of bringing power management into operating system's control DVFS is introduced into ACPI [43-45, 136, 139]. ACPI defines the various performance states to the CPU which is one of the part of operating system power management [43-45]. From ACPI point of view, this prediction model utilizes the concept of a variety of C-states [140]. These C-states are C₀, C₁, C₂ and C₃. Where C₀ refers to active state, and rest states are the idle state where power consumption of the CPU decreases as we move from C₁ to C₂ to C₃.

Though DVFS is a highly promising technology but system's software needs to find the correct voltage and frequency to execute the application process on the computer system and if it is not correct then the overall performance of the systems gets down

and this results into more energy consumption by the non CPU components like memory and HDDs [141, 142]. The task of finding the appropriate voltage and frequency is performed with the help of algorithms and these algorithms are generally implemented as running systems. In this chapter, we have investigated several ubiquitous DVFS algorithms to facilitate an in-depth study about their behavior. DVFS algorithms are ubiquitous if it merely looks into the performance status of the system every time it makes a decision on which voltage and frequency to be used. Weiser et al. [143] presented one of the earliest collections of DVFS interval-based scheduling algorithms for general purpose operating system and the time has been divided into fixed-length intervals and determine the CPU voltage as well as frequency at the start of each interval for finding whether the CPU is underutilized and used the CPU utilization ratio. Here, low CPU utilization ratio implies that CPU requires low amount of voltage and frequency. Yao et al. [144] also presented the first task-based DVFS scheduling algorithms for real time systems. These two works [145, 146] have motivated lot of researchers/scientists to work in this domain and several papers [145-154] targeting embedded and mobile computing platforms using DVFS have been published like Pering et al. [151, 152], Grunwald et al. [148] have implemented the modified version of original DVFS algorithms. There are various other scheduling algorithms [155-157], which constantly monitor the memory access rate to know when a program is entered into memory or exited from memory. On the basis of various design issues DVFS scheduling algorithms can be characterized into three categories like the 1) Abstraction of CPU utilization, 2) Prediction of the trend in CPU utilization, and 3) The association of performance states with CPU utilization. These various design issues have been discussed in the following sections.

2.8.1 CPU Utilization Algorithms

The CPU utilization ratio is the fraction of time which CPU spends non-idle in an interval therefore its value is considered either 0 or 1. These CPU utilization algorithms can be used to estimate future values of utilization based on the previous utilization sample and broadly classified into two categories like utilization prediction and state selection.

- a) **Utilization Prediction:** For this category, two broad classes of technique are used where one class of technique is to treat $\{u_i^p\}$ as a time series and use smoothing

techniques like moving average to predict the future events. The other class is to view a DVFS scheduling algorithm as a control system and apply control-theoretical techniques for utilization prediction. The various smoothing techniques for utilization predictions are as follows:

- Simple moving average (SMA): A SMA_N is the un-weighted average of the previous N data points. When calculating successive values, a new value u_{i-1}^o always comes in and old value u_{i-N-1}^o drops out, which represents that a full summation each time is unnecessary. However, a queue is needed to hold the most recent N values [147].

$$u_i^p = \frac{1}{N} \sum_{j=i-N}^{i-1} u_j^o = u_{i-1}^p + \frac{u_{i-1}^o - u_{i-N-1}^o}{N} \quad (2.3)$$

- LongShort: It is a type of weighted moving average and averages the 12 most recent intervals of CPU utilizations, weighting the three most recent utilizations three times more than the others [147].

$$u_i^p = \frac{1}{12} \left(3 \sum_{j=i-3}^{i-1} u_j^o + \sum_{j=i-12}^{i-4} u_j^o \right) \quad (2.4)$$

- Cumulative moving average: this represents the un-weighted average of all data points up to the most current one. [147]

$$u_i^p = \frac{1}{i-1} \sum_{j=1}^{i-1} u_j^o = \left(\frac{i-2}{i-1} \right) u_{i-1}^p + \left(\frac{1}{i-1} \right) u_{i-1}^o \quad (2.5)$$

- Exponential moving average or EMA_λ : This is a weighted moving average with weights decreasing exponentially over time. Here, λ shows the degree of weight decrease, and a lower values of the λ discounts older observations faster [149, 158].

$$u_i^p = (1 - \lambda) \sum_{j=1}^{i-1} (\lambda^{(i-1)-j}) u_j^o = \lambda u_{i-1}^p + (1 - \lambda) u_{i-1}^o \quad (2.6)$$

- AVG_λ : this is similar to EMA_λ with $\lambda = N/(N+1)$ [148, 151].

$$u_i^p = \frac{1}{N+1} \sum_{j=1}^{i-1} \left(\frac{N}{N+1} \right)^{(i-1)-j} u_j^o = \left(\frac{N}{N+1} \right) u_{i-1}^p + \left(\frac{1}{N+1} \right) u_{i-1}^o \quad (2.7)$$

- PAST: it predicts that the upcoming interval's utilization will be the same as the past interval's and is equivalent to AVG_0 and SMA_1 [144].

$$u_i^p = u_{i-1}^o \quad (2.8)$$

In another class of utilization-prediction technique is to view a DVFS scheduling algorithm as a control system and applies control theoretical techniques for the prediction. The following section list about these techniques:

- not quite PID: This technique is also known as nqPID_N. PID (proportional-integral-derivative) is a classical control-systems technique, which is used to find the appropriate value of changing workload system that represents the present and past of the system. It simplifies the standard PID algorithm by eliminating feedback and the continuous-time integral and derivative with their discrete-time counterparts like summation and difference [159].

$$u_i^p = K_P \cdot u_{i-1}^o + K_I \cdot \left(\frac{1}{N} \sum_{j=i-N}^{i-1} u_j^o \right) + K_D \cdot (u_{i-1}^o - u_{i-2}^o) \quad (2.9)$$

where K_P , K_I and K_D are the weights of the corresponding terms. Here, the proportional term acts similarly to PAST [144], the integral term acts like a SMA_N and the derivative term expects the coming change in utilization to be the same as the difference between the previous two utilization samples.

- Proportional difference (PD): this is one of the old and traditional control theories, which estimate the change concerning the direction of a slope [160].

$$u_i^p = u_{i-1}^o + K_P \cdot (1 - u_{i-1}^o) + K_D \cdot ((1 - u_{i-1}^o) - (1 - u_{i-2}^o)) \quad (2.10)$$

Here, K_P and K_D are the weights of the corresponding term similar to the previous technique.

- b) State Selection:** These kinds of algorithms, uses the simple PAST [144] methods for utilization prediction and are commonly seen in the operating system's computing environment. Typically, state selection algorithms are divided into three categories as listed in the following section and differ in the formula used for state selection.
- **powernowd:** This algorithm was developed by John Clemens [161], which is a very simple algorithm used to adjust the CPU performance state based on the CPU utilization ratio. For state selection algorithm establishes the high and low thresholds to trigger a step either from a low-performance state to a high-performance state or vice versa. Therefore, the selection of threshold value

becomes important but there are no formal guidelines available on selection of threshold value and typically an end-user tune these values empirically according to their workload.

- **CPUSpeed:** This algorithm was developed by Carl Thompson [162], which can be found in many LINUX platforms like SUSE Linux 9.3 or later and Fedora Core 4. This is a part of every Dell PowerEdge server and is known as “demand-based switching.” The state selection method is very much similar to powernowd and considered as more conservative because it often uses high performance states.
- **ondemand:** This algorithm is also a part of Linux kernel [163] and in its very earlier versions algorithm reduces CPU frequency at minimum steps of 5% of peak frequency whereas in recent versions it is changed and it looks for lowest frequency that can sustain the load by keeping idle time over 30%. Here, we can say that the optimal frequency is the lowest frequency that can support the CPU usage without changing the policy.

2.9 DESIGNING POWER-EFFICIENT ARCHITECTURES

The improvement in the architecture of single-core CPU performance has been slowed down significantly because of the power wall and computer architects started targeting alternative architectures to improve the CPU performance per watt. The path to power efficiency can go through cutting transistors from performance-enhancing hardware structures that do not contribute to the performance in proportion to their power requirement, whereas on the software side, most of the performance is achieved by manual coding, auto-tuning, and to a lesser extent by automatic code generation. A power efficient architecture could provide the support for vector processing and carefully written code can directly exploit the power of these architectures.

The difficulty of writing codes to power-efficient architectures is due to the burden put on the software developer to manage all aspects of code optimizations that affect the performance. For example, the developers need to understand the essential memory operations required to manage the memory transfers and need to express the parallelism in an explicit manner in the software.

2.9.1 Programming approaches for power efficient architecture

The code development productivity, portability, and efficiency are very critical issues the success and the survival of any architecture. There are various programming approaches that can be used with accelerator based architecture but there are some difficulties and limitations associated with them. These approaches are discussed as:

A) Code generation

Recently, a lot of research work is performed on code generation for accelerators to utilize their potential performance as well as their efficiency. Code generation can be done manually [164-171] or this can be an automated process [172-175] or by searching large optimization space [176-177]. Manual code rewriting has shown best potential of accelerators for scientific computing.

B) Automatic code migration

This code migration has been successfully prototyped to the various accelerators at the functional level by multiple research projects and commercial compilers like IBM Watson [175], PGI Compiler [178], CAPS Enterprise [179] and CellSS [180]. O'Brien et al. [175] demonstrated the difficulties, which is be faced by the automated compilation process. Some compliers do not provide fully automated process like CellSS compiler relies on manual data layout. Automatic code migration of legacy code to GPUs [173, 174] is somewhat more successful because of large memory of GPUs.

C) Autotuning

This is another approach to achieve the best performance for a wide set of architecture. It usually requires an expertise beyond that is needed for code optimization, because it demands good knowledge of optimization space, good heuristics to search the optimization space and ability to produce the various variants of the same code. These complexities are addressed by auto-tuning libraries like ATLAS [181], Spiral [182], FFTW [183], and OSKI [184] but they are not targeting newly emerging power-efficient architectures.

2.10 CONCLUSION

In this chapter, we have focused on the various techniques required to minimize the energy consumption by the computer systems. Various discussed techniques, which varies from hardware centric to software centric techniques and is easy to implement as well as flexible too. We have discussed the case of Microsoft Windows power schemes as it is one of the most popular operating system presently in the market, and showed the various drawbacks of its power schemes. Power saving modes defined by these schemes consumes the 3W – 15W of energy in these modes, which is critical from energy saving point of view. Power measuring and profiling has already been studied extensively and more work is being performed to improve the power profiling techniques and various discussed techniques in this chapter, is very useful in power aware system design. From the various discussed approaches like Simulation based, hardware and software based, latter two approaches are getting more attention in this direction. In the future, the operating system should supply a module for power profiling and supply configurable accuracy. We have presented the general view of DVFS scheduling algorithms for energy-aware computing and these algorithms have been characterized with three design issues: the abstraction of CPU utilization, the prediction of the trend in CPU utilization, and the association of the voltage and frequency values with CPU utilization. We have also discussed the utilization based CPU utilization ratio as it present various algorithms to predict the CPU behavior.

We have discussed about the power efficient architecture which can be achieved using custom designs or heterogeneous architectures with specialized accelerators. The potential challenge in designing the power efficient architecture is how to program them in a productive way and the wide adoption of these architectures relies heavily on how to automate porting legacy code to these architectures.

Chapter 3

ENERGY SUSTAINABLE FRAMEWORK

3.1 INTRODUCTION

The energy consumption and sustainability of the computing systems is an highly demanding area of researchers and scientist [185, 186]. According to the IEA, the energy used by computer systems and consumer electronics will be increased threefold by 2030. Even with the improvements foreseen in the energy efficiency, consumption by electronics in the residential sector is set to increase by 250% by 2030 and most likely will become the largest end-use category before 2020, unless effective steps are taken [187]. Therefore, the fundamental challenge for successful implementation of “green computing” is to understand and then balancing of the energy efficiency with system performance as well as application performance. The development of models and interfacing of systems software like operating systems, job schedulers are highly in demand today. In a typical computer system, to reduce the energy, the operating system based ACPI [43-45, 135, 139, 140] sets the display to low power-modes after specified periods of the inactivity on the mouse and keyboard. The ACPI efficiency strongly depends upon the inactivity intervals set by the users. However, understanding of the user-application behavior and their interactions with machine subsystems and other applications continues to be under development. As a result, standardized interfaces between the machine, its subsystems, and its operating environment do not yet exist. The operating systems measurement tools like NWPerf as discussed in [188] are designed to report metrics of the interest to system developers and administrators on a global view of the system behavior.

Recently, sustainability-based approaches have received a very high response when designing any energy-saving approach. These types of approaches directly focus on the environment and categorized under sustainable computing aspect. This section lists the various energy-sustainable approaches developed by various researchers over time. Wang et al.[189] investigated the existing power models by re-evaluating on multi-core

computer systems (MCSs) and proposed a two-level power model that estimates the power consumption for each core on MCSs. Moreover, based on this model, authors 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. In [190], Chen et al. have investigated the adverse effects of dynamic voltage and frequency scaling by running a virtual machine over system performance, using energy conservation methods in server consolidation. In [190], authors also proposed a new application-aware approach by introducing a new set of metrics CPU gradients, which predict the impact of changes in CPU frequency. Proposed 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 the performance-aware energy conservation by deploying energy controllers. Naumann et al. [191] have 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 it to contribute to the sustainable development. Chen et al. [53] 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 the 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.

3.2 ENERGY AND POWER CONSUMPTION MODELING

A general way of representing a power model of a system is expressing the relationship between the serviced workload and power consumed by servicing it, which is shown in Fig. 3.1. Despite the elegance of this definition, there is the fundamental question to know about the “workload intensity.” However, the workload intensity known as different measures depending on the system context. A common power model of computing equipment is the power consumed over the system utilization. As we know that system consist of many components, like HDDs, memory, peripheral devices, and CPU, so the system utilization could be a multidimensional quantity. Hence the conventional way of representing system utilization with a scalar value is to use CPU

utilization level as shown in Fig. 3.2 about CPU utilization, we have also discussed in previous chapter in DVFS section and we have observed that various algorithms are designed to find the CPU utilization ratio. Power versus utilization is the standard way of profiling the power consumption of a system. Until recently, systems showed near-linear power consumption with respect to utilization which is also observed by several research studies [192-194].

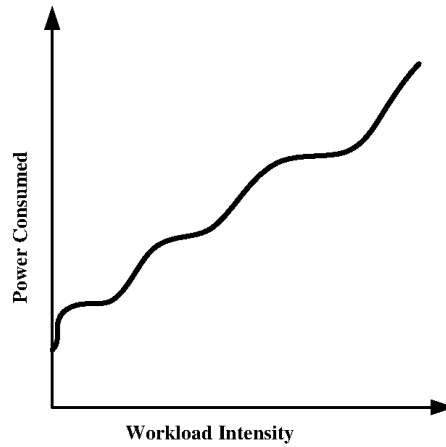


Figure 3.1: Relationship between the workload intensity and power consumption.

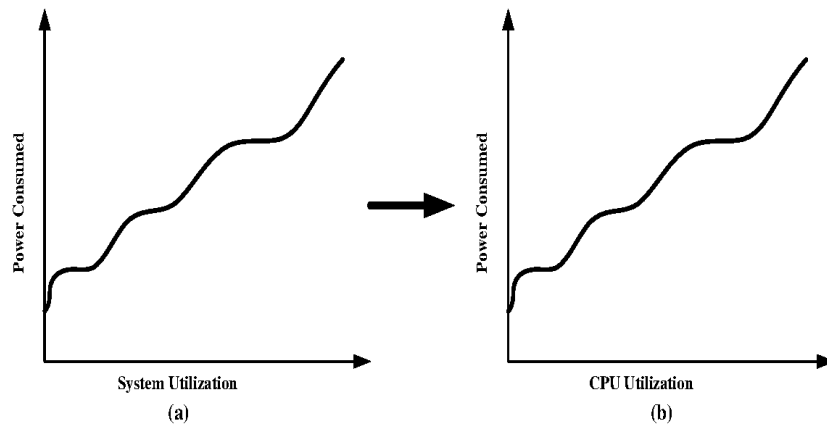


Figure 3.2: Power consumption as a function of utilization such as: a) System utilization and b) CPU utilization.

Typically, a CPU's power consumption increases linearly with its utilization. However, the idle power (starting point of this linearity) accounts for a substantial portion of the peak power. Fig. 3.3 gives a typical model of CPU power over utilization.

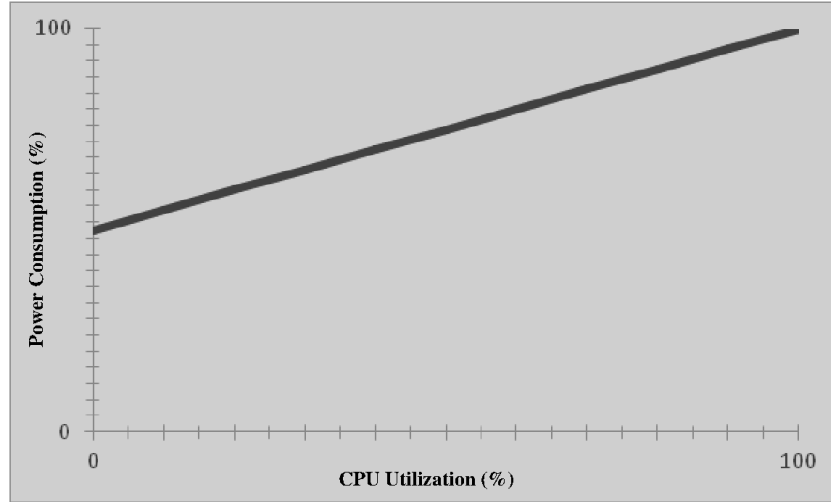


Figure 3.3: Power consumption (%) with CPU utilization.

The power consumption P_n at any specific CPU utilization in (%) can be calculated as:

$$P_n = (P_{peak} - P_{idle}) \frac{n}{100} + P_{idle} \quad (3.1)$$

where:

P_{peak} is the maximum power consumption at full utilization.

P_{idle} is the power consumption when CPU is at the idle state.

In various CPUs, the idle power may be up to 50% of the peak power, resulting in serious energy inefficiency, particularly, when the computation workload is low. To save idle power, CPU can be commanded to enter a low-power mode known as C-state. Modern CPUs have several power modes, from halt to deep sleep and even deep power down. Therefore, this advancement in the CPU power management saves more energy because the clock signal and power to idle units are cut inside CPUs. The more units are stopped, the more energy is saved, but the more time is required to wake-up a CPU to be fully operational. For example, with the new technique C6 known as 'deep power

down' state, a CPU's internal voltage can be reduced to zero [140]. The wake-up time from C6 state is much longer because it does not preserve the CPU context.

The power performance function gives a simple relationship between CPU power and supply voltage or frequency. However, CPUs are not the sole component consuming energy. A substantial part of the power is also used by memory, disk, and other peripheral devices, which does not change with supply voltage or frequency of the CPU. Therefore, the total load of a system can be modeled in two ways: 1) the power of CPU that can be adjusted by supply voltage and, 2) constant power consumed by other components, which is show as follows:

$$P_{load} = (P_{cpu} + P_{const}) \quad (3.2)$$

This is the amount of power consumed when the system is under load. Here, P_{cpu} is the capacitive power of CPU and known as power consumption of a CMOS-based CPU, defined as the summation of capacitive, short-circuit, and leakage power. This can be defined as:

$$P_{cpu} = ACV^2f \quad (3.3)$$

and

$$f = \left(\frac{k(V - V_t)^2}{V} \right)$$

where:

A is the number of switches per clock cycle,

C is the total capacitance load,

V is the supply voltage,

f is the frequency that is roughly in proportion with **V**,

k is the constant of circuit, and

V_t is the threshold voltage.

Therefore, the amount of power consumed is:

$$P_{load} = (P_{cpu} + P_{const}) = ACV^2 f + P_{const} \quad (3.5)$$

For the power at idle time, we assume the idle power of CPU can be ignored because of the effective power-saving C-state, and the power decreasing in other components like memory and HDDs is also ignorable compared to the total idle power on the system board. So, the idle power of the system can be estimated as follows:

$$P_{idle} = P_{const}$$

Based on the aforementioned functions of load power and idle power, we can calculate the energy efficiency of the servers under different CPU supply voltages or frequencies. We can consider two resource usage scenarios for the proposed model.

- 1) When shutdown of system is not feasible to save power in idle time. For a given computation work W , we assume

$$T_{computation} = \frac{W}{f}$$

Then, for certain time duration T under consideration, the idle time

$$T_{idle} = T - T_{computation}$$

The energy consumed during time T is:

$$\begin{aligned} E &= P_{load} T_{computation} + P_{idle} T_{idle} \\ &= (P_{cpu} + P_{const}) T_{computation} + P_{const} (T - T_{computation}) \\ &= P_{cpu} T_{computation} + P_{const} T \\ &= ACV^2 W + P_{const} T \end{aligned} \quad (3.5)$$

From the Equation (3.5), it is seen that by reducing the CPU supply voltage, the energy consumption can be minimized.

- 2) We assume that the systems can be put into a deep sleep or even shutdown when idling, especially when the cost of the wake-up is relatively negligible compared with the total response time. Here, the idle power consumption is saved by shutting down the energy consumption is [2, 59, 75, 101]:

$$\begin{aligned}
E &= P_{load} T_{computation} \\
&= \frac{(P_{cpu} + P_{const})W}{f} \\
&= ACV^2W + \frac{P_{const} W}{f}
\end{aligned} \tag{3.6}$$

The energy consumption in this function includes two parts: CPU power that increases with the square of supply voltage and consumed by other components during the computation time, which increases inversely with CPU frequency. This shows that when the CPU voltage is scaled down, less energy is consumed by CPU however more is wasted by the components.

3.3 USER CENTRIC ENERGY MANAGEMENT

Precise and detailed monitoring of the user activity is the basis for continuing to improve the performance and energy efficiency of computing systems. Understanding the user interaction provides valuable information about which resources is needed ahead of time. This leads to the performance optimizations such as better resource allocations for applications that can utilize a given resource more productively. The last optimization refers to the situation where the device is turned-off to reduce the more energy consumption. Energy efficient design requires systematic optimization at all levels of the design abstraction, from the process technology and logic design for architectures and algorithms. There are various static techniques applied during the design phase at the algorithmic level – strength reduction [195], algorithmic and algebraic transformation [196, 197], and retiming [198], at architectural level – pipelining [199] and parallel processing [200], and at logic level – logic minimization [201, 202], pre-computation [203], circuit [204], and technology [205]. An alternative approach is to adjust the system operation and energy consumption to application workload dynamically, that is, during the system operation and various methods like: application-driven system reconfiguration [206], dynamic voltage-frequency scaling [207], energy-quality scaling [208, 209] and network driven optimizations [210].

Despite differences, these dynamic methods exploit the same idea, namely, to keep the system in the lowest power mode whenever there is no activity input, and activate the system whenever the input signals change. To implement the idea, the system incorporates an extra unit that constantly monitors the input activity or workload and it

determines the new operational mode for the system, as shown in Fig. 3.4(a). Depending on the application, the workload can be measured by different metrics like the average rate at which events arrive at processor [209] and idling time per sample interval [207-211]. However, it is not always possible to make correct predictions due to the peculiarities of application, operational environment, and/or user demands, which are varying with time.

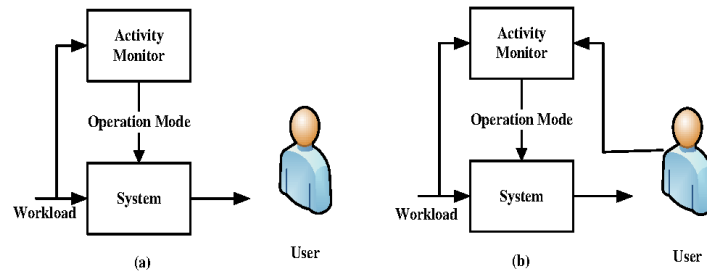


Figure 3.4: Energy management schemes for (a) existing system and (b) proposed system.

In general, there are two sources of energy losses in a device: intrinsic losses and user-related losses. The intrinsic energy losses are normally caused by the engineering design, technology, and materials used in construction of the device. However, the user related losses are associated with varying and inefficient usage of devices. For example, keeping a television set on when nobody watches it or leaving a computer system running cause to energy losses associated with improper use of the device. Existing energy management policies are device centric that is either ignores the user, assuming unchangeable operational environment for the device, or rely on very simplified policies, which causes to large energy losses. Therefore, it is clear if one want to reduce the energy losses then the device energy management must be user centric, means adaptable to varying user behavior [212, 213] as shown in Fig. 3.4(b). In the next section we have proposed the user centric approach to energy management, which monitors not only the system workload but also the user behavior and the environment. The main idea of the proposed approach is to extend the controller functionality to monitor the demands on system operation imposed by the user and adjust the system performance for the variation in these demands.

3.4 PROPOSED ENERGY SUSTAINABLE FRAMEWORK

This section discusses the proposed energy sustainable 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. 3.5 and starts its functioning with the execution of power saver module [214]. 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, it is utilized by three major modules of proposed framework that is power-saver main window, duration, and calculate CPU usage.

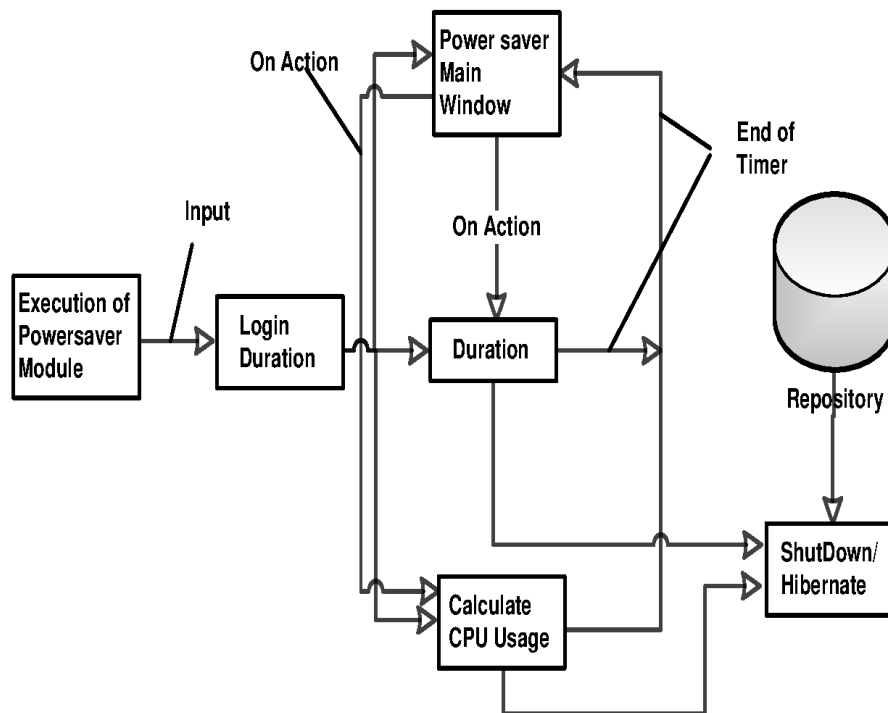


Figure 3.5: Framework of the proposed power saving scheme.

Here, the power saver-main window represents the Graphical User Interface (GUI) of proposed GP tool whereas further two modules are very much concerned about the

implementation of the proposed algorithms to minimize the power consumption by the computer systems. Proposed algorithms are implemented as two different modes known as: 1) Swift Mode and 2) Exhaustive Mode, which is part of GUI as shown in Fig. 3.6. These two modes are different in their functioning.

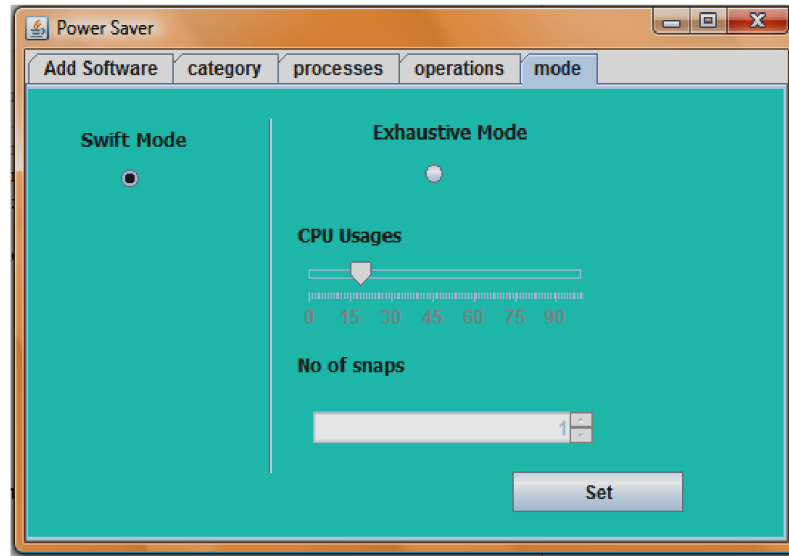


Figure. 3.6: GUI implementation of the proposed algorithms.

In Swift Mode, let us consider 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 find the availability of the 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.

The functioning of Exhaustive Mode is very much different with the Swift mode. In this mode, we considered the time of login-duration as well as monitor the user activity continuously on the computer system by calculating the total CPU usage of that

machine, which puts another check for minimizing the power consumption of the computer system. If it is found that the total CPU usage of the computer system is zero or below the user-defined threshold value and compares the defined number of snapshots for ensuring that in each consecutive snapshot of total CPU usage is below the defined threshold value, then only the computer system automatically switches to shutdown, or hibernate mode, irrespective of what time is input by the user during the login-duration. The snapshot of the total CPU usage is taken at each minute. 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. 3.7.

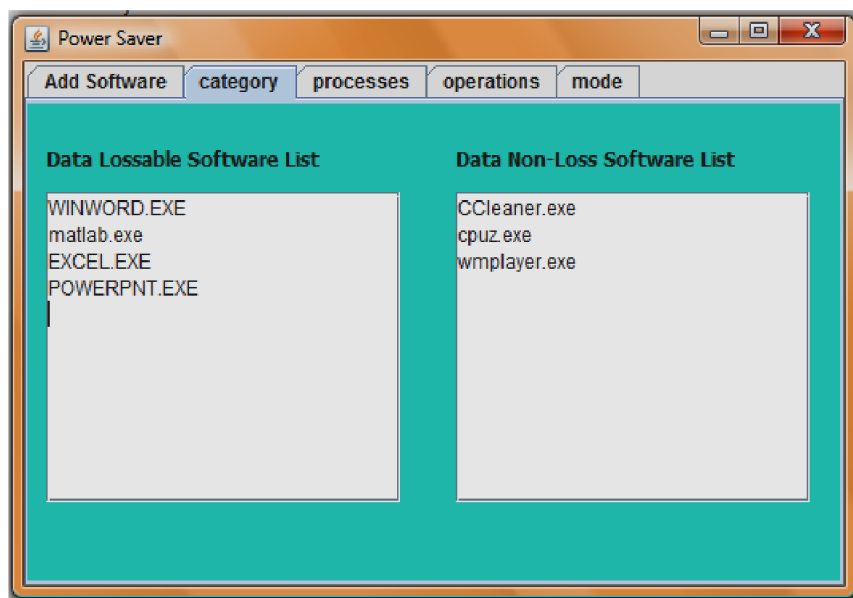


Figure 3.7: Repository of losable and non-losable software's.

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 the hibernate mode otherwise it gets shutdown.

3.4.1 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 the 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:

1) 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 pop-up the window of login-duration in the middle of the screen as shown in Fig. 3.8.

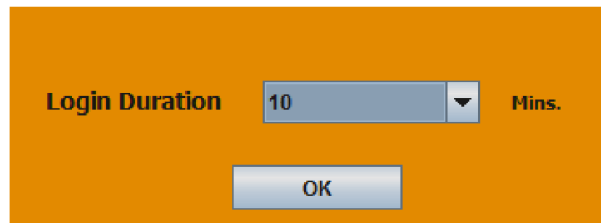


Figure.3.8: User prompt to input login duration time.

2) 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.

3) 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.

4) 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, 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 value of threshold and snapshot timing in minutes in the form of {20, 1} and is used by the exhaustive mode. Here, 20 refer to the threshold value for total CPU usage and 1 refers to the interval of snapshots in minute. These are the user-defined values, as shown in Fig. 3.6, and can be defined according to the user's requirement. For better energy efficiency, power-saving and management of computer system, it is suggested that the threshold value should be kept always less than or equal to 20, and number of snapshots must be collected for time interval greater than 1 minute whereas, ProcessCollector.java program file is used to store the value of the various running application programs on the computer system.

5) org.juitw.process.bean package

This package declares only the 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.

6) 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 and 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.

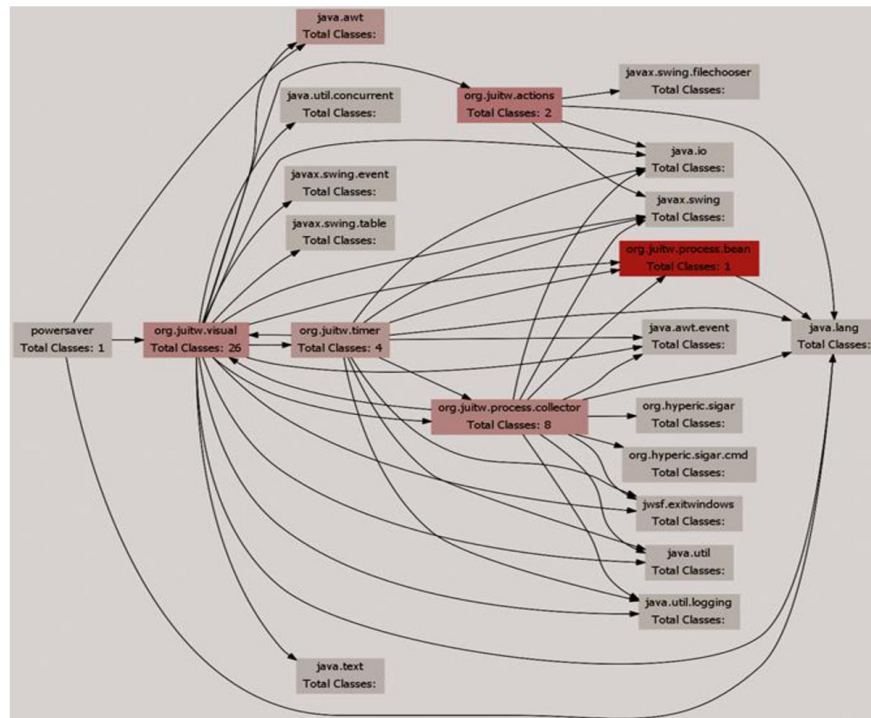


Figure.3.9: Package dependency diagram of the proposed framework.

This internal view of the proposed framework of the power scheme also represents the package dependency diagram of 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 as shown in Fig. 3.9.

Table 3. 1: Detailed view of each package and JAVA program files.

S. No	Source Packages	Files	Line of Code	Lines with Import
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 3.1 gives the detailed overview of the discussed packages and details about JAVA program files in them.

3.5 COMPARISON OF PROPOSED FRAMEWORK WITH EXISTING POWER SCHEMES

For detailed comparison of working of existing power-scheme in Windows Operating Systems with proposed user centric energy sustainable framework of the power-scheme, we have compared the working function of both above discussed Swift mode

and exhaustive mode with the functioning of existing scenario that available in the windows power scheme [214].

3.5.1 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 minutes	
Turn-off hard disks	= after 30 minutes
System standby/sleep	= after 30 minutes
User works for 2 minutes and leave the computer system inactive.	

Using the above scenario as defined by the user, system will start power-saving only after the 30 minutes 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 the human activity on the system.
- b) The existing power scheme starts its functioning only after 30 minutes of inactivity of keyboard and mouse, whereas user works for only 2 minutes.
- c) After the inactivity of 30 minutes 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.

3.5.2 Comparison with Swift Mode

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

Login-duration	= 10 minutes
User works for 2 minutes 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 minutes, 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:

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.
- 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.
- Proposed Swift mode overcome from the disadvantages discussed in (a) and (b), one does not need to worry about the various time intervals values.
- This mode offers more than 66% of power-saving over existing power scheme scenario.

3.5.3 Comparison with Exhaustive Mode

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

Login-duration =	10 minutes
CPU usage (threshold) =	≤ 20
Number of snapshots=	2
User works for 2 minutes 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 threshold 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 minutes and for Swift mode this time reduces to 8 minutes by knowing the user consensus during the start up of the computer system.
- 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 minutes 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 minutes, 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

3.6 CONCLUSION

To reduce energy consumption of the electronic systems as well as personal computers, however, represents the biggest challenge that designers will have to face in the next decade according to the SMART2020 report [215]. Therefore, new technologies that would transform energy used by the electronic systems such as personal computers, televisions, and personal media gadgets etc. all these systems are user centric as they receive inputs from the user and deliver services to them. For doing so, various works are being done by the researchers/scientists to find the new techniques for the energy management. In this chapter, we have discussed the discrepancies of ACPI as it is mostly used in the computer systems for energy management and presented the energy and power consumption model build on CPU utilization of the computer system.

We have also focused on the need of user centric framework for energy management of the computer system which monitors not only the system workload but also the user

behavior and presented an energy sustainable framework for the power schemes of Windows operating system. This framework is very much user centric as during login to the system it tries to know the user consensus by knowing the approximate user's work time on the machine and implements the two different modes of working for this framework. The main objective of this framework 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.

In this chapter, we have compared the functioning of existing power profile with the proposed user centric energy sustainable framework and that the proposed modes, Swift mode and Exhaustive mode detects the human activity on the computer system in an effective manner. It is based on the time value supplied by the user during login to the system. We have also obtained that Swift mode provides more than 66% of energy saving and exhaustive mode provides more than 93% of energy saving over the existing power scheme in operating system.

ENERGY SUSTAINABLE ALGORITHMS

4.1 INTRODUCTION

The main objective of power management systems is to reduce the energy consumed by electrical devices while maintaining a satisfactory level of the performance. Research on effective power management methods has intensified in recent years and energy consumption is now emerging as a dominant performance measure in the computer systems, rivalling the running time. To design an energy efficient computer system will ultimately require the development of fundamental frameworks, algorithmic techniques, and principles that can be used to guide practical solutions. Most of the algorithms-related research on the energy-efficient computing has been focused on task scheduling. These energy conservation techniques explore the opportunities for tuning the energy delay trade-off [216]. In one of the pioneering paper, Weiser et al. [143] first proposed the approach to energy saving by using fine grain control of CPU speed by an operating system scheduler. The main idea is to monitor the CPU idle time and to reduce the energy consumption by reducing clock-speed and idle time to a minimum. In another work, Yao et al. [144] analyzed offline and online algorithms for scheduling tasks with arrival times and deadlines on a single processor computer with minimum energy consumption. These research studies have been extended in [217-221], most of them focused on real time applications namely, adjusting the supply voltage and clock frequency to minimize the CPU energy consumption while still meeting the deadlines for task execution. In other works [149, 222-232], authors have addressed the problem of scheduling independent or precedence-constrained tasks on single processor or multi processor computers where the actual execution time of the task is less than the estimated worst case execution time. In computer systems, low power and energy efficient design techniques, and algorithms aim to minimize the energy consumption while still meeting certain performance goals. In [233], Barnett has studied the problems of minimizing the expected execution time given a hard energy budget and

minimizing the expected energy expenditure given a hard execution deadline for a single task with randomized execution requirements. In [234] Bunde, has considered scheduling jobs with equal requirements on multi processors. In [235] Cho and Melhem, studied the relationship among parallelization, performance, and energy consumption, and the problem of minimizing energy-delay product. In [109, 236], authors have attempted joint minimization of energy consumption and task execution time. In [112], authors have addressed the energy- and time-constrained power allocation and task scheduling on the multiprocessor computers with dynamically variable voltage and frequency, speed, and power as combinatorial optimization problems. On the other hand, number of studies in [112, 220, 221] have added a new dimension to the voltage scheduling problem, by considering energy-saving opportunities within the task boundaries. Here, the operating voltage of the task is dynamically adjusted according to the execution behaviour to accurately reflect the changes in the required number of cycles to finish the task before the deadline and also referred as intra task voltage scheduling. Some communities of the researchers/scientists have worked on various algorithms to minimize the power consumption by a computer system and its peripheral devices like CPU, HDD. These algorithms are Back-off algorithm [237], PowerNap algorithm [238], Power-scheduling algorithms [239-242], SoftWatt [82], and Power-aware algorithms [112, 243, 244].

4.2 PROPOSED SWIFT MODE ALGORITHM

This section represents an algorithm implemented for the discussed scenario in chapter 3 of Swift mode and used for collecting the data for analysis and result purpose [214]. The proposed algorithms initialize 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. The 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: Pseudo code for 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) T_1 and T_2 - Timers

Step - 1: Initialize variables

Initialize L , M and X

$L \leftarrow$ Take the input from the user during login to the system.

$M \leftarrow$ Take the value from the configuration file set by the user.

$X \leftarrow$ The sleeping time of a thread.

Step - 2: Start of Timers

Starts two timer threads T_1 and T_2 ;

T_1 : expires after $L \times 60 \times 1000$ ms

T_2 : expires after each $1 \times 60 \times 1000$ ms and sleeps for X time.

Step - 3: Mode Selection

Selection of Mode (M):

IF $M = \text{Swift}$ then

$X \leftarrow L \times 60 \times 1000$

ELSE

$X \leftarrow 1 \times 60 \times 1000$

Step - 4: Functioning in Swift mode

When T_1 expires

```
a) Cancel  $T_2$ 
b) Invoke user prompt to find user availability on the
   system
c)  $\eta \leftarrow$  input extra time.
d) Starts a thread  $T_3$ .  $T_3$  will expires after  $1 \times 60 \times$ 
   1000 ms.
e) After  $1 \times 60 \times 1000$  ms
   IF  $\eta == 0$  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
        $L \leftarrow \eta$ 
   END
END
```

4.3 PROPOSED ENERGY SUSTAINABLE SNAPSHOT TECHNIQUE

As discussed in the chapter 2 to minimize the energy consumption, there are various approaches, which is used to minimize the energy consumption of computer systems. In the proposed energy-sustainable snapshot algorithm (ESSA) [245], if there is any work or processing being performed on a computer system, the CPU of the system must be in use, in which case the percentage of total CPU usage should be greater than zero, otherwise, it should be equal to zero. In [11], 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. 4.1 and is a part of the GP tool.

4.3.1 ESSA framework

This proposed energy-sustainable snapshot technique takes input from various sources. First, in the form of a configuration file that is created just after the installation of the GP tool, the user has to define the value of the snapshot time and its threshold limit. These values are stored in a configuration file and overwritten in the file in case the user makes any changes in the future. The threshold value represents the percentage of total CPU usage, which should be kept at its minimum—that is, either less than 20 or less than 10, this increases the accuracy of the technique, which occurs when very few application processes in addition to system processes are running on the computer system. Second, input is also taken from the user when logging in to the computer system, the user must select the roughly estimated time he/she will be working on the computer system. Thus, it follows:

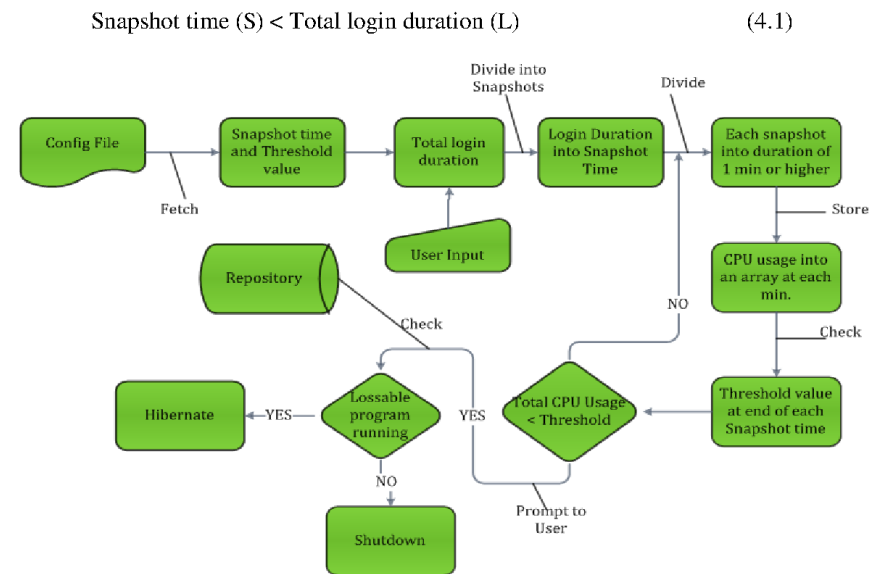


Figure 4.1 Framework of the energy-sustainable snapshot technique.

If any difference is found, then the proposed technique will consider the total login duration (L) as the snapshot time (S). In the next step, the total login duration (L) will be

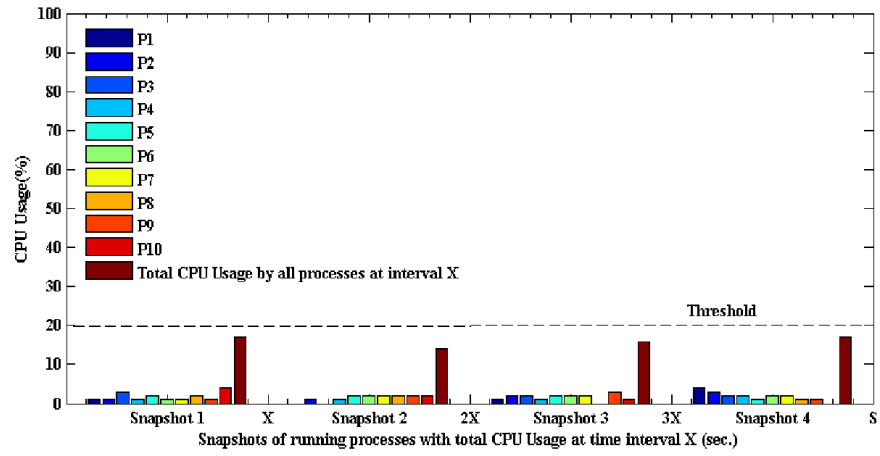
divided equally into small chunks of snapshot times $S, 2S, 3S, \dots, nS$. This value is determined as follows:

$$\text{Total number of chunks of snapshot times} = \lceil \text{Total login duration (L)} / \text{Snapshot time (S)} \rceil \quad (4.2)$$

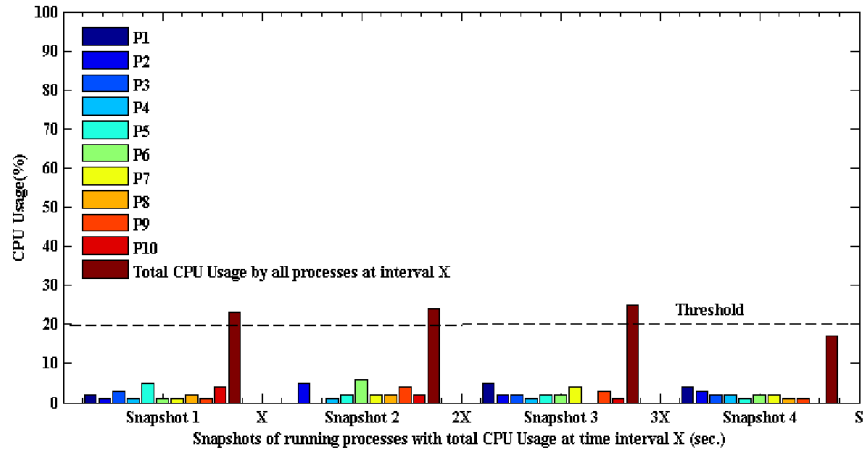
For example, if a user inputs a total login duration $L = 15$ minute and the value of the snapshot time defined in configuration file $S = 5$ minute, according to Eq. (2) there will be only 3 chunks of snapshot times, $S = 5$ minute, $2S = 10$ minute and $3S = 15$ minute. This shows that after each interval of 5 minute the total CPU usage will be checked. In the next step, to increase the accuracy of the system's decision, we have included the concept of energy-sustainable snapshots for 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).

$$X = S/n \quad (4.3)$$

Here n represents the total number of CPU usage (%) snapshots in each chunk. In the proposed technique, the value of X is 1 minute, which means that after each 1-minute 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. If it is found that the percentage of total CPU usage consumed by all processes running on the system is below the defined threshold value, as shown in Fig. 4.2(a), which indicates that the user is not actively working on the system or very few processes are running on the system, then the proposed technique 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. 4.2(b), which means that the CPU of the machine is being utilized and therefore, the user or some application processes are continuously working, the next 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.



(a)



(b)

Figure 4.2 Snapshots of total CPU usage (%): (a) below threshold value and (b) above threshold value.

4.3.2 ESSA algorithm

This section represents the ESSA algorithm [245] implemented for the previously discussed framework and used for collecting the 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 ζ ,

τ , S , L and A . In step-2, timer T_1 and T_2 gets started according to the value provided by the user. Whereas step-3 is major step in this algorithm, which computes the total CPU usage (%), store this value into an array and compares the array value for taking the decision when to shut down or hibernate the computer system.

Algorithm: Pseudo code for ESSA

Symbols used in this algorithm:

- vi) ξ - For CPU usages
- vii) τ - The threshold for CPU usage
- viii) s - The snapshot duration.
- ix) L - The total login duration time
- x) X - The sleeping time
- xi) A - An array of CPU Usage
- xii) T_1 and T_2 - Timers
- xiii) 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 $L \times 60 \times 1000$ ms

T_2 : expires after each $1 \times 60 \times 1000$ ms and sleeps for X time.

Step - 3

```

IF  $T_2$  expires THEN
a)  $X \leftarrow 1 \times 60 \times 1000$ 
b) Calculate  $\xi \leftarrow CPU(P_1) + CPU(P_2) + \dots + CPU(P_n)$ 
c)  $A[C] \leftarrow \xi$ 
d) IF  $C = s$  THEN
    IF  $\forall A[0], A[1], \dots, A[s-1] < \tau$  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

```

4.4 EXPERIMENTAL METHODOLOGY

To characterize the behavior of the proposed energy-sustainable snapshot algorithm, we first describe the experiment setup used to verify the proposed algorithm. Then, we have described about the experiment procedure 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.

4.4.1 Experiment Setup

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

Table 4.1 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 (0x1)
	Technology	32 nm
	Core Speed	1597.8 MHz
	Stock Frequency	3100 MHz
	Core VID	0.986V
	Max TDP	65W
	Multiplier x FSB	16.0 x 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	2x32KBytes, 8-way set associative, 64-byte line size
	L2 Cache	2x256 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	2048 Mbytes
	Number of banks	1
	Voltage	1.5V
	Frequency	665.5 MHz
Disk	Size	320 GB SATA
	Manufacturer	Seagate
	Speed	7200 rpm

The experimental platform as stated above, we have used cluster of IBM Thinkcentre desktops. Table 4.1 summarizes the configurations of cluster machines that is used in

this experiment to evaluate the algorithms. The simulated processors is Intel Core i3 2100 with core speed of 1597.8 MHz and stock frequency 3100MHz. The L1 data cache is 8-way set associative, with size 2x32 KBytes, whereas the L2 cache size is 2x256 KBytes and L3 cache is 12-way set associative with 3 MByte size, for all the cluster machines. The processors operate at voltages of 0.986V. The memory type is DDR3 with 2GB 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 320GB SATA and running at 7200 rpm.

4.4.2 Evaluation

We have used the software StressMyPC [246] for a thorough evaluation for the proposed algorithms. 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 could remain busy for a certain period of time to determine the accuracy of proposed algorithms. The snapshot time of the percentage of total CPU usage is recorded for each second in chunks lasting 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. NetBeans IDE 7.1.2 [247] was used to implement the proposed algorithms. 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.

4.5 RESULTS

This section describes the various results obtained after executing the proposed ESSA algorithm in a real environment. We have used various scenarios—usage scenario-1, usage scenario-2 and an internal scenario—to evaluate the ESSA algorithm. Usage scenarios-1 and 2 also describe an environment that includes the system and operating parameters. These usage scenarios have been executed on all cluster machines. Specifically, we executed commonly used software and obtained the results for the following scenarios.

4.5.1 Usage Scenario 1

This scenario represents the idle functioning of all the clusters computer systems, during which no work is carried out on the machines. The user must simply login to the system and executes certain software's, as per Table 4.2, and leave the system inactive due to some unknown reason, which is the most common user practice. The running computer systems not only consume power but also completely depend on the operating system's power scheme settings to switch the computer system into any power saving mode as defined by power scheme. Thus, when the proposed ESSA 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. 4.3 for all the cluster machines. For $L = 20$, $S = 1$ and $\tau = 20$, Fig. 4.3 is equally divided and represent a total of 20 chunks of snapshots with lasting 60 sec. 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 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 the percentage of total CPU usage is higher than the threshold value, then ESSA algorithm will check the next snapshot and record the percentage of total CPU usage for each second in that snapshot.

Table 4.2 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 time (S)	1 min.
Threshold value (τ)	20%

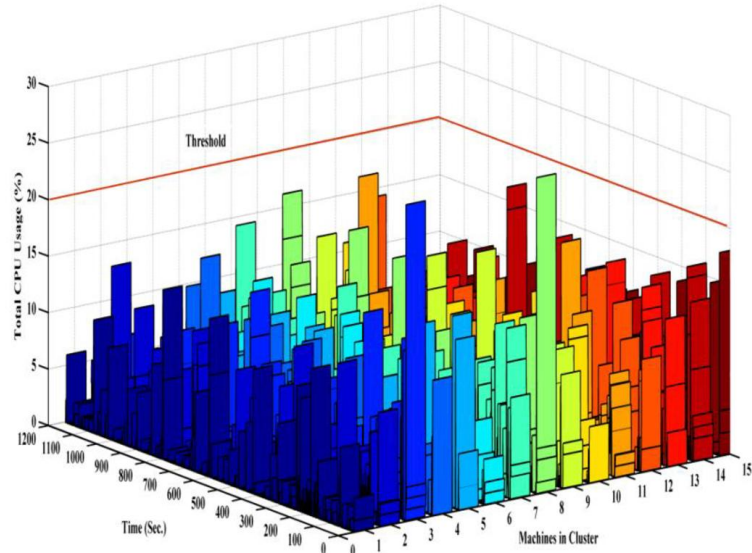


Figure 4.3 Total CPU usage (%) for idle computer systems in the cluster.

4.5.2 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 4.3 presents the various system and operating parameters under which the user logs in to the system.

Table 4.3 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 time (S)	1 min.
Threshold value (τ)	20%

However, change in the operating parameters observed, and the proposed algorithm is executed under the same environment as it was in the previous scenario. Fig. 4.4 illustrates the results obtained for all cluster machines. The processing represented by Fig. 4.4 is similar to that of Fig. 4.3, 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. Fig. 4.4 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.

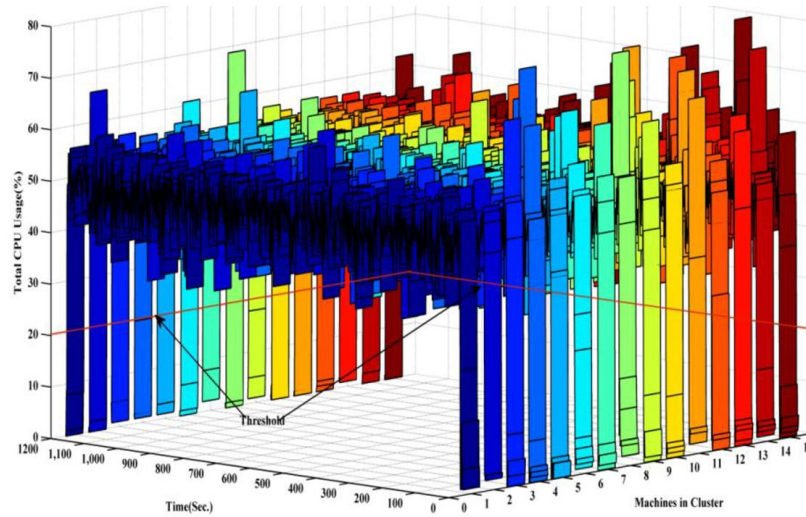


Figure 4.4: Total CPU usage (%) for active computer systems in the cluster.

4.5.3 Internal Scenario

This scenario represents the various results, obtained and used by the proposed algorithm make the decision whether to keep the computer system running or not. As we have seen in the previously discussed usage scenario-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

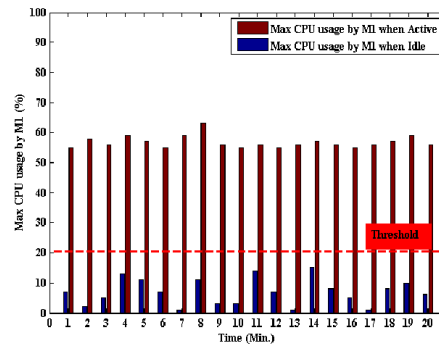
that breaches the defined threshold limit or not. Fig. 4.5 represents the maximum-recorded value of CPU usage for each cluster machines, in each chunk of snapshot respectively and compared with each other and totally based upon the value of snapshot time S.

For the previously discussed usage scenario-1 and 2, we have considered the value of snapshot time $S = 1$ minute, 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 4.4 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 minute.

Table 4.4: Maximum CPU usage (%) by each cluster machines (M1 – M15) in active (A) and in idle (I) mode up to 20 minutes.

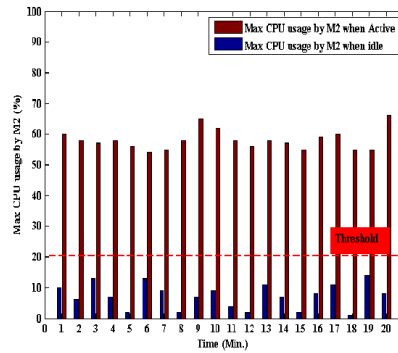
Snapshot time interval (in min.)	M1		M2		M3		M4		M5		M6		M7		M8		M9		M10		M11		M12		M13		M14		M15	
	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I	A	I
1	55	7	60	10	71	28	80	12	66	12	57	5	74	9	59	10	64	13	76	17	63	18	58	17	70	7	67	15	80	28
2	58	2	58	6	58	10	60	3	73	12	66	13	55	6	59	14	61	15	57	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

From Table 4.4, we can derive the various graphs for both the states of each cluster machines. As shown in Fig. 4.5(a) 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.

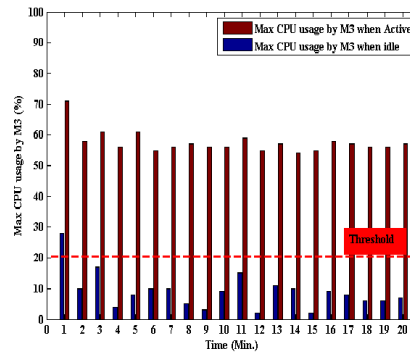


(a)

In Fig. 4.5(b), 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. 4.5(a) 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.

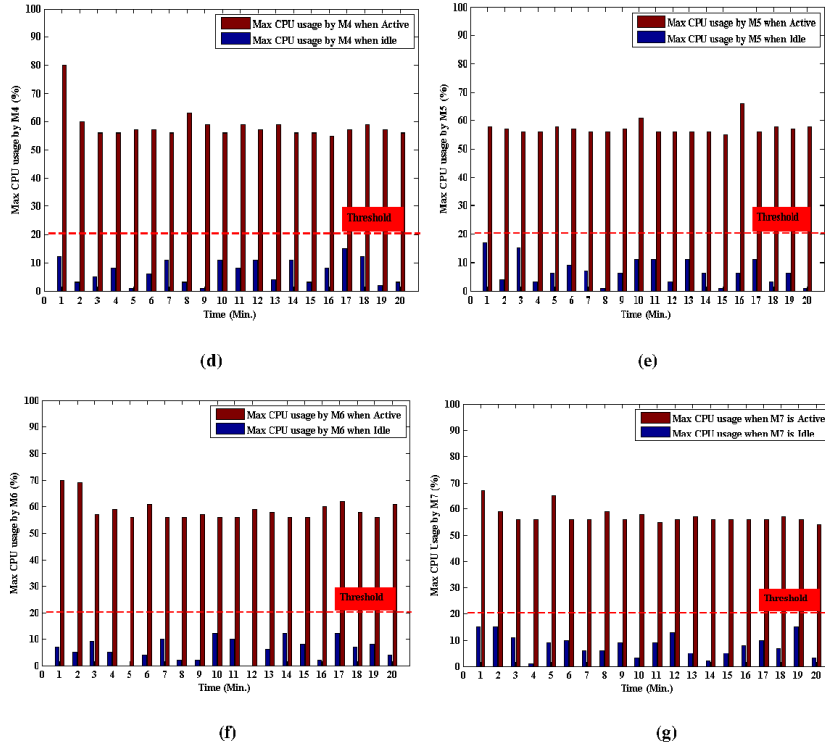


(b)

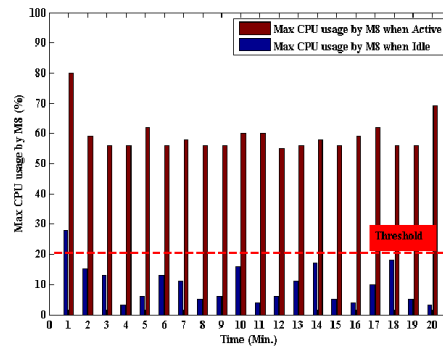


(c)

In, Fig. 4.5(c) 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 ESSA algorithm and the next interval will be checked whereas in active state percentage of total CPU usage remains always above the defined threshold.



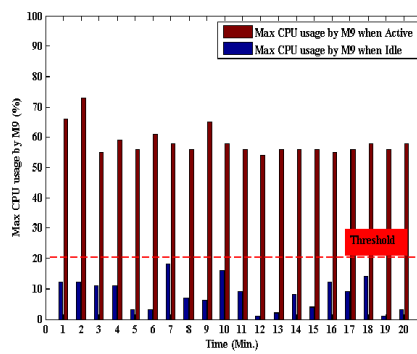
Further, the snapshots of percentage of total CPU usage for cluster machines M4, M5, M6, and M7 in Fig. 4.5(d), Fig. 4.5(e), Fig. 4.5(f), and Fig. 4.5(g) as shown above, always remains below the defined threshold in their idle state and above the defined threshold in their active state. Here, in the idle states, user prompt is invoked after each interval of time to check whether user is available or not on the machine.



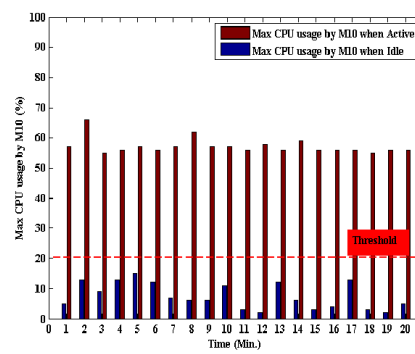
(h)

In Fig. 4.5(h), 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.

In Fig. 4.5(i), Fig. 4.5(j), Fig. 4.5(k), Fig. 4.5(l), Fig. 4.5(m), Fig. 4.5(n), and Fig. 4.5(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.



(i)



(j)

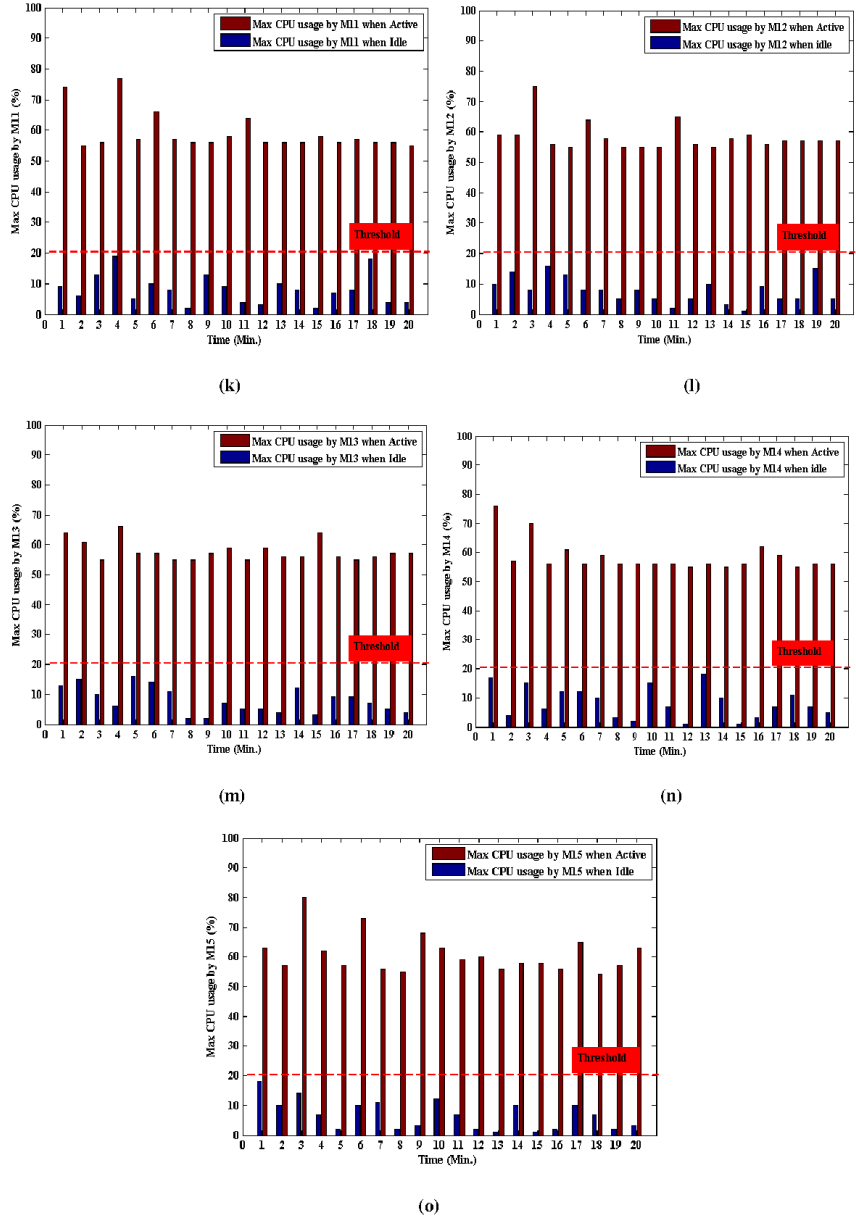


Figure 4.5: Maximum CPU usage (%) by cluster machines for each snapshot when active and idle (a) M1 (b) M2 (c) M3 (d) M4 (e) M5 (f) M6 (g) M7 (h) M8 (i) M9 (j) M10 (k) M11 (l) M12 (m) M13 (n) M14 (o) M15.

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 ESSA 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.

4.6 CONCLUSION

As we know, the research on effective power management methods has been intensified and the energy consumption is now becoming an emerging area as a dominant performance measure in the computer systems in place of considering the speed of the system. For designing an energy efficient computer system, ultimately require the development of fundamental frameworks, algorithmic techniques, and principles that can be used to guide practical solutions. The main objective of the proposed techniques is to structure concepts, strategies, and activities to design an energy-sustainable power scheme. In this chapter, we have proposed the algorithmic framework for the discussed energy sustainable framework in chapter 3. We have proposed two algorithms known as swift mode algorithm and ESSA algorithm, both the algorithms are based on user interactions with the computer system. Swift mode algorithm tries to know the user consensus before starting work on the computer system, by prompting the user to enter time for login duration and continue its working till the login duration time comes at end. Once the given login duration time expires, algorithm again tries to know the user consensus. In case, if the user is available, a new login time value may be supplied otherwise algorithm will switch the computer system to hibernate or shutdown mode to minimize energy consumption. Whereas, the proposed ESSA algorithm is very much different in its functioning with the swift mode and claims for more energy saving. 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 power saving mode. 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 the various results for cluster

machines, it should be below the threshold limit defined by the user to enable the system make the decision to hibernate or shutdown.

From the results, we can also see that in case if there is no working on the machine or machine is in idle state then maximum CPU utilization is always below the defined threshold value. In the results obtained by proposed algorithms, this value is set to 20% and also could be used as an idle value for future cases.

Chapter 5

PERFORMANCE EVALUATION OF FRAMEWORK

5.1 INTRODUCTION

Due to proliferation in the software and system's market, it has become necessary to evaluate the things for various measures like performance, reliability, security etc. There are several more problems associated with the software's which include the underutilization of client resources, installation of additional hardware equipments and the congestion of computer systems either because of complete memory or CPU utilization. Therefore, the use of software and systems with enhanced performance are in high demand. There are various analytical tools available in the software engineering for examining the behaviour of applications like function-calls, time, CPU usage and memory usage etc. These tools have dramatically changed the way of analysis and become essential for optimizing an application's performance and profilers are one of them [248, 249]. Profilers can be used to find the overall appropriate level and bottlenecks in the application. These bottlenecks are the troublesome spots, which can be hidden and occur during the execution of the application. So, with the help of profilers one can reduce the time for detecting these bottlenecks in the applications. Most modern profilers support the graphical representation of the results obtained to facilitate quick analysis of the developed application. The role of profilers becomes important when the application is a real-time system where they check for whether the real-time task meet their deadlines and matches the estimated time derived by static analysis. In this chapter, we have evaluated the performance of our proposed energy sustainable framework of the power scheme under Swift and Exhaustive modes. This study investigates on more convenient and effective methods, which can minimize computer system's load and enhance the overall performance at the same time. The proposed framework requires only configuration and no installation is required, which in turn not only maximizes the conveniences for users but also keeps the system resources free for other works.

The performance measures for the proposed energy sustainable framework of the power scheme include the various system components like thread monitoring, analyzing memory, and CPU performance under specific workload. We used the profiler available in NetBeans-IDE [247] for evaluating the performance of proposed energy sustainable framework discussed in chapter 3, under different proposed modes with the help of the proposed algorithms. By using profiler, one can easily determine the performance of system's memory and CPU under various performance measures like memory leakage [250], memory heap, and memory garbage collection [251], thread monitoring, CPU timestamps for each invoked methods etc.

5.2 THREAD MONITORING

During the thread monitoring session, the profiler monitors system and application threads activity and displays the information in the threads tab. This tab is responsible for displaying the timeline for each running threads with their states like running, sleeping, wait and monitor. We can also get the detail of time spent in each state. For the proposed energy sustainable framework, we have monitored various active threads in both the modes: swift and exhaustive. On the basis of their execution, these are typically divided into two categories known as system thread and user thread. Table 5.1 provides the overview for both types of threads with their uses class and descriptions.

Table 5.1: 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
6	AWT-EventQueue-0	java.awt.EventQueueThread	System	AWT thread, which is the main thread executing a GUI java
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

5.2.1 Thread Monitoring in Swift Mode

Here, we have executed the proposed energy sustainable framework under swift mode and performed thread monitoring. We have executed the framework for the discussed usage scenario in chapter 4 up to 20 minutes. The results obtained after thread monitoring are shown in the Figure 5.1.

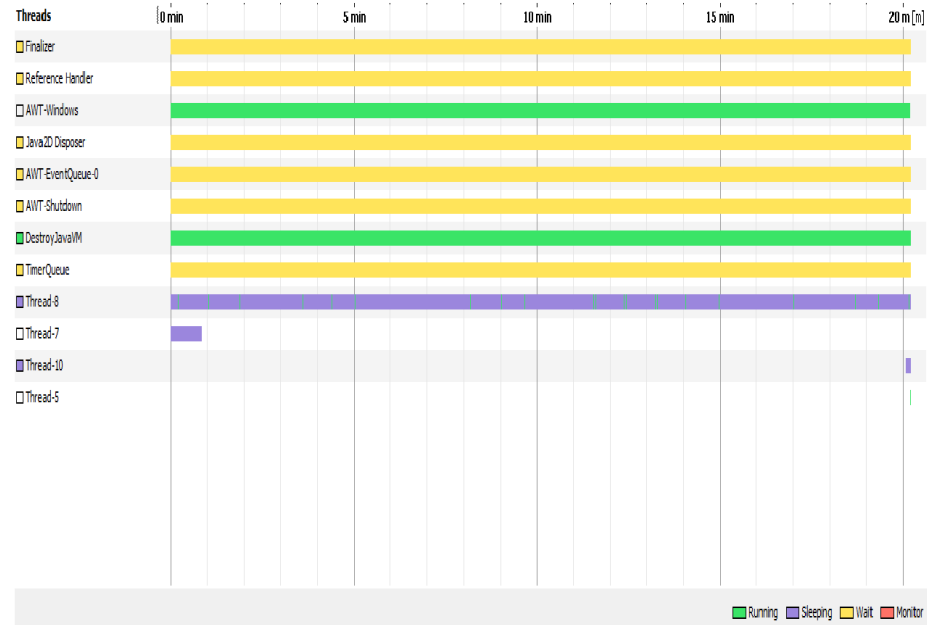


Figure 5.1: Various active threads during the framework monitoring in swift mode.

From Fig. 5.1, one can easily observe 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 the thread 7 gets expired and the thread 8 continues till the end of the 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 the sleep mode for the new login-duration.

5.2.2 Thread Monitoring in Exhaustive Mode

Here, we have executed proposed energy sustainable framework under the exhaustive mode and performed thread monitoring. We have executed the framework for discussed

usage scenario in chapter 4 up to 20 minutes and obtained results after thread monitoring are shown in the Fig. 5.2.

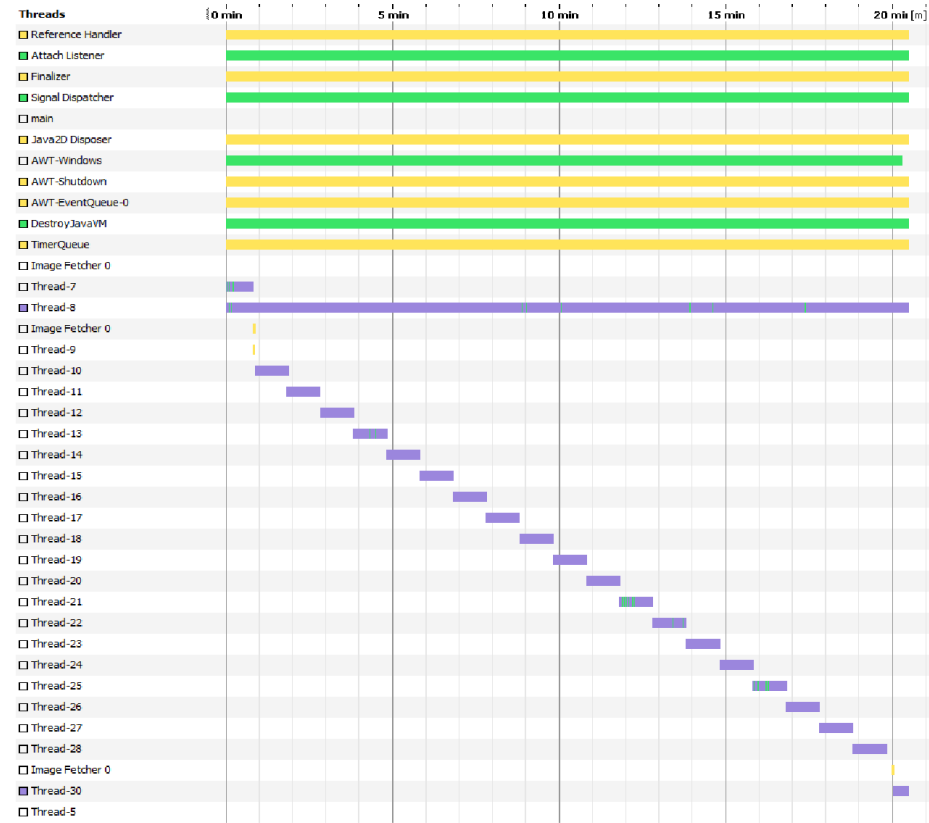


Figure 5.2: Various active threads during the framework monitoring in exhaustive mode.

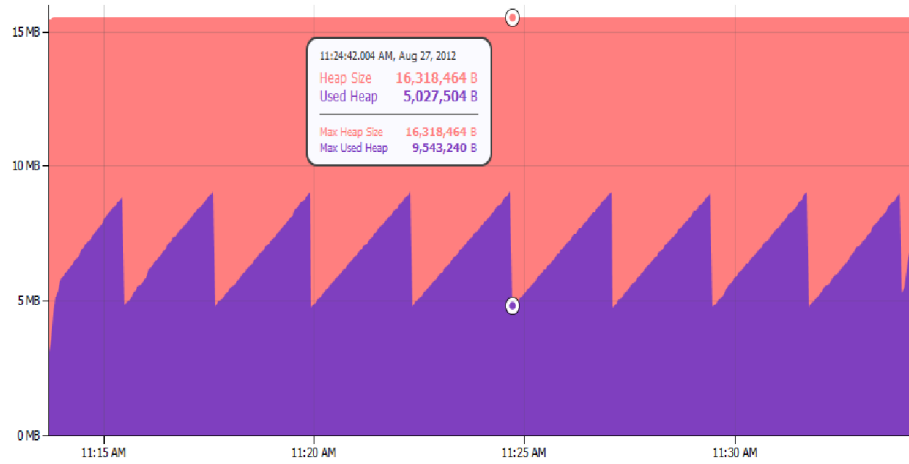
From Fig. 5.2, we can notice that all the system threads remain similar to the thread monitoring in swift mode. Whereas, various number of user threads could be observed in this exhaustive mode because proposed ESSA algorithm continuous checks the user activity on the machine and here we are observing the snapshots each with 1 minute duration and total login duration time is 20 minutes. Due to this reason, the user threads are created for each minute of login duration. In the Fig. 5.2, these threads are shown from thread-7, thread-10, and thread-11 to thread-28 each with one minute duration. As the supplied login duration time of 20 minutes gets expire proposed framework prompt a window to user for supplying new login duration time with one minute expiry and represented with user thread-30.

5.3ANALYZE MEMORY PERFORMANCE

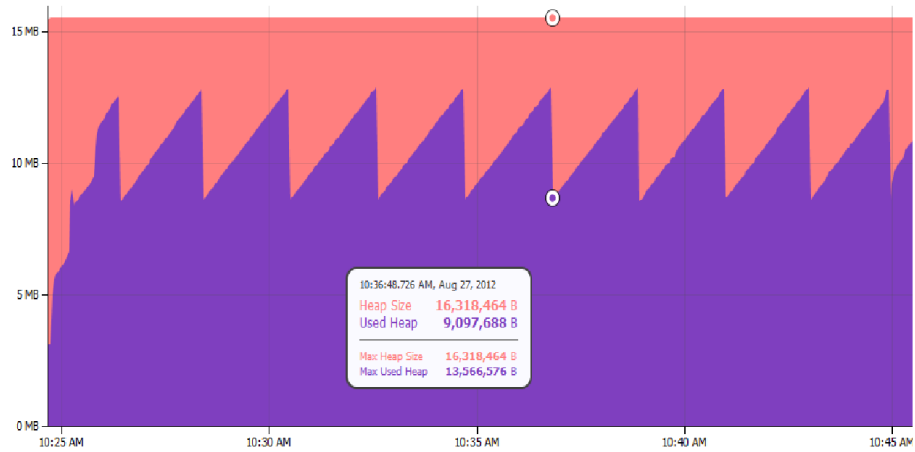
The analysis of memory performance of the proposed framework under different overheads includes the various results like memory heap, garbage collection and threads and loaded classes. Using VM telemetry, we analyzed these memory measures and obtained the various results as shown in further figures. Here, we have profiled the proposed framework for both the proposed modes: Swift mode and Exhaustive mode. In all our real time results of memory performance, we have profiled the proposed framework up to 20 minutes.

5.3.1 Heap Analysis

In Fig. 5.3(a) and Fig. 5.3(b) the memory heap size over the period of time for both the modes: Swift and Exhaustive have been analyzed, respectively. Here, we can easily find out the details of maximum available heap size versus used-heap of profiled framework for both the modes. For both scenarios, maximum available heap size is the same with variations in the maximum used-heap which is 9.5 MB to 13 MB, respectively for both the modes and the minimum used heap after garbage collection is 5.0 MB to 9.0 MB, respectively for both the modes. This is very clear as in exhaustive mode there is a continuous monitoring of the user activity for each minute is done by the framework. Here, Garbage collection is performed after a certain interval of time automatically, which minimizes the maximum used-heap sizes. These intervals are easily noticeable in Fig. 5.3(a) and Fig. 5.3(b). Throughout the login-duration framework continues its functioning smoothly, which can be noticed with the sharp edges of Fig. 5.3(a) and Fig. 5.3(b), but when the login-duration time comes to at end, there is always some deviations in the edge that refers to the activation and creation of new threads in the memory. To know more about these threads, one can refer the thread monitoring section as discussed above. We can also observe that for both of the scenarios maximum used heap never reaches up to maximum heap size and framework continues its functioning without any decrease in the performance.



(a)



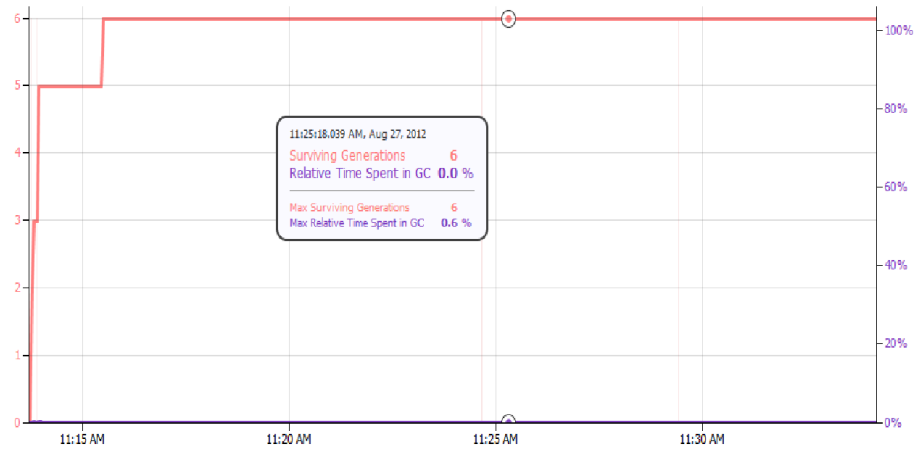
(b)

Figure 5.3: Analyze memory performances for allocated heap size vs. used-heap. For each graph, x-axis denotes the time in (HH:MM) and y-axis shows the used-heap size in (MB) (a) Swift Mode and (b) Exhaustive Mode.

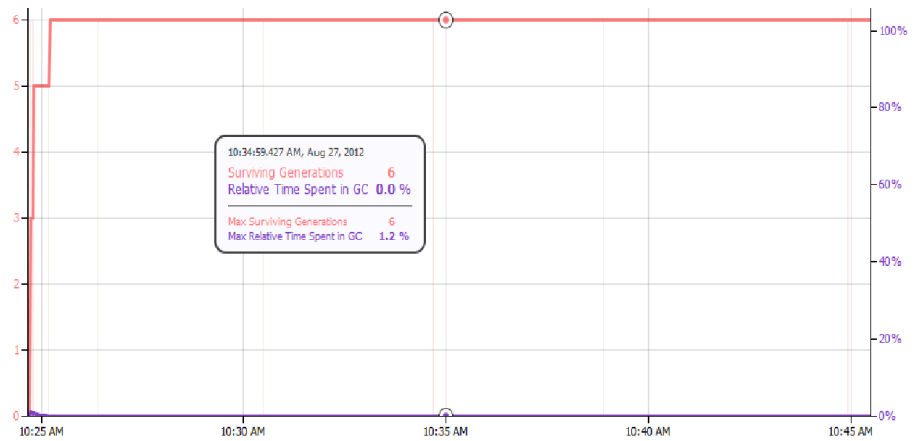
5.3.2 Memory Leakage

The problem of memory leakage for the proposed framework have analyzed by finding various surviving generations and relative time spent in the garbage collections. From Fig. 5.4(a) and Fig. 5.4(b), one can find that once the framework gets initialized total number of surviving generations becomes constant and remains at 6 for both the modes,

till the login-duration ends, which represent the framework behaves similarly for both of our proposed modes. So, there is no problem of memory leakage in proposed framework. Here, maximum relative time spent in garbage collection is 0.6% and 1.2% respectively for both of the modes as shown in Fig. 5.4(a) and Fig. 5.4(b).



(a)

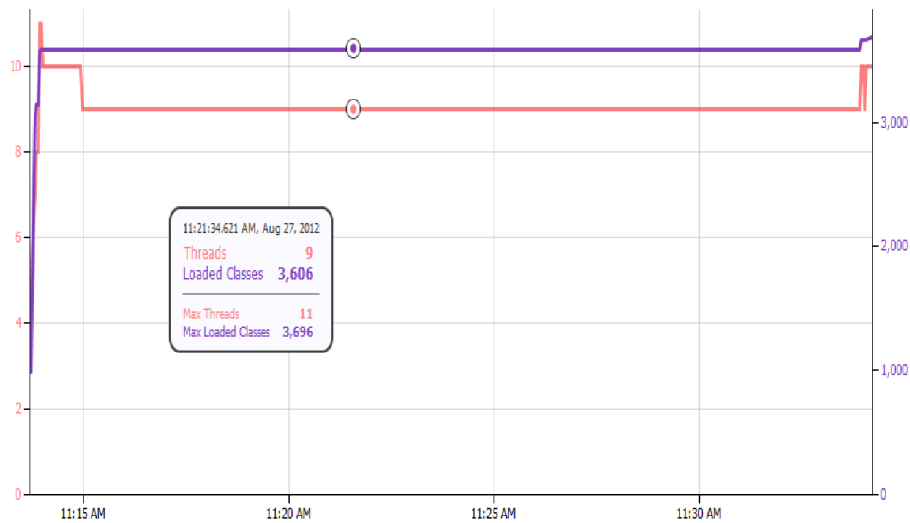


(b)

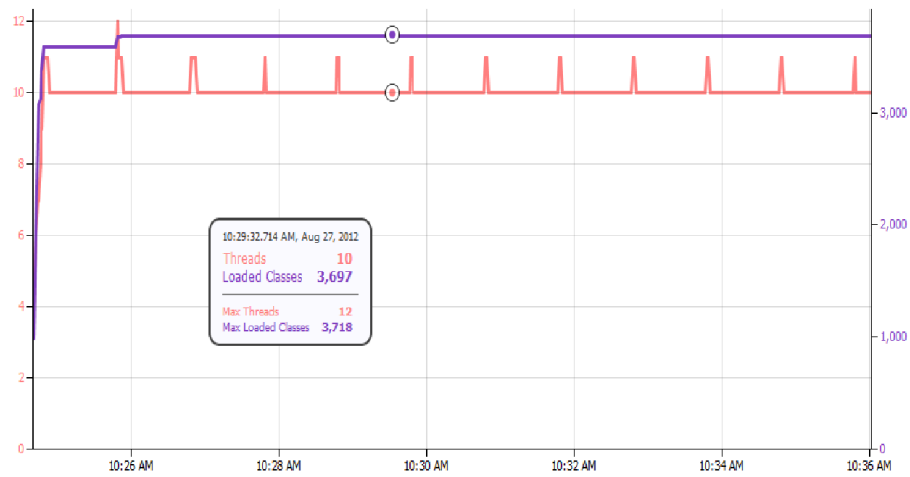
Figure 5.4: Analysis of memory performance for Surviving generations vs. Relative time spent in GC. For each graph, x-axis denotes the time in (HH:MM) and y1-axis shows the surviving generations and y2-axis shows the relative time spent in GC (%) (a) Swift Mode and, (b) Exhaustive Mode.

5.3.3 Thread Analysis

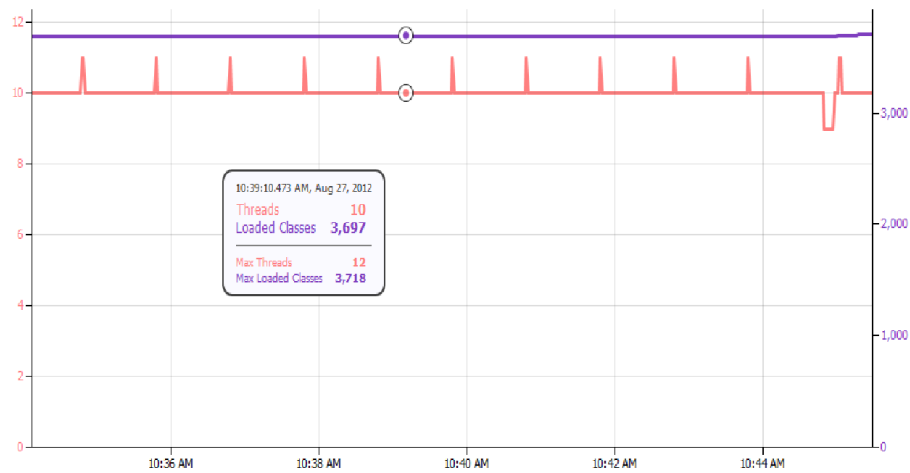
In this subsection, we have analyzed the detailed view of the memory performance for various running threads versus loaded classes. It is very much similar to the scenario discussed in thread monitoring section. From Fig. 5.5(a), Fig. 5.5(b1) and Fig. 5.5(b2), one can find that the maximum number of threads remains at 10 and 11, respectively for both the modes, with a little variation in maximum loaded classes in exhaustive mode. In Fig. 5.5(a), this can be observed that during the supplied login duration of 20 minutes no new thread is created and total number of threads remains constant, which reveals that proposed framework is executed under swift mode and no popup window is activated during the supplied log-duration. The main reason behind this is in the swift mode, we are very much concern about the user and not about their activity for each minute on the computer system. Due to this, we are not concern about the percentage of total CPU usage in each snapshot, though we have recorded this CPU usage percentage in this mode too for future purpose. At the end of login-duration, some threads get activated from sleep mode and some are created newly, details of which can be found in the thread monitoring section. It is also because at this time popup window gets activated as the supplied login-duration time gets over to know the user status on the machine and threads are created.



(a)



(b1)



(b2)

Figure 5.5: Analysis of memory performance using ESSA algorithm for threads versus loaded classes. 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. (a) Swift mode, (b1) and (b2) Exhaustive mode. (b2 is an extension of b1).

In the exhaustive mode, as shown in Fig. 5.5(b1) and Fig. 5.5(b2) various peaks are shown after each minute of time interval, which represents that the ESSA 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.

5.4ANALYZE CPU PERFORMANCE

By using CPU performance measurement, the proposed framework have been 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. We have analyzed the CPU performance separately in both the modes: Swift mode and Exhaustive mode. For both the modes, we have profiled the CPU only for project related classes that includes the core java classes only.

5.4.1 CPU performance in Swift Mode

In this mode, the CPU performance analysis is performed for entire supplied login duration time up to 20 minutes. This analysis is shown in the following Fig. 5.6 (a1) and Fig. 5.6 (a2) that shows the call tree method for login-duration, CPUInfo and various threads created to monitor the proposed framework. Fig. 5.6 (a1) shows the call tree methods AWT-EventQueue-0, Thread-8 and main. The significant description about these methods is given in Table 5.1. We can observe that methods are invoked whenever they are required. Here, Thread-8 works till the end of login-duration.

Call Tree - Method	Time [%]	Time	Invocations
AWT-EventQueue-0		11746 ms (100%)	1
org.jutbw.timer.Duration\$1.actionPerformed (java.awt.event.ActionEvent)		11553 ms (98.6%)	1
org.jutbw.visual.LoginDuration\$1.actionPerformed (java.awt.event.ActionEvent)		157 ms (1.3%)	1
org.jutbw.visual.PowerSaverMainWindow\$1.stateChanged (javax.swing.event.ChangeEvent)		6.38 ms (0.1%)	1
Thread-9		4923 ms (100%)	1
org.jutbw.visual.LoginDuration\$2.run ()		4923 ms (100%)	1
Self time		3699 ms (75.1%)	1
org.jutbw.process.collector.CPUInfo.<init> ()		880 ms (17.9%)	1192
org.jutbw.process.collector.CPUInfo.output (String[])		341 ms (6.9%)	1192
main		192 ms (100%)	1
powersaver.PowerSaver.main (String[])		192 ms (100%)	1
org.jutbw.visual.LoginDuration.<init> ()		152 ms (79.2%)	1
org.jutbw.visual.LoginDuration.initComponents ()		118 ms (61.8%)	1
Self time		111 ms (58.3%)	1
org.jutbw.visual.LoginDuration\$1.<init> (org.jutbw.visual.LoginDuration)		0.010 ms (0%)	1
Self time		33.2 ms (17.3%)	1
Self time		39.8 ms (20.8%)	1
Thread-7		121 ms (100%)	1
Thread-10		15.0 ms (100%)	1
Thread-9		5.8 ms (100%)	1

(a1)

Call Tree - Method	Time [%] ▼	Time	Invocations
AWT-EventQueue-0		11716 ms (100%)	1
Thread-8		4923 ms (100%)	1
main		192 ms (100%)	1
Thread-7		121 ms (100%)	1
org.juitw.process.collector.CalculateCPUUsages\$.run ()		121 ms (100%)	1
org.juitw.process.collector.CalculateCPUUsages.getCurrentCPULoad ()		121 ms (99.8%)	60
Self time		0.211 ms (0.2%)	1
org.juitw.process.collector.CalculateCPUUsages.access\$900 (org.juitw.process.collector.CalculateCPUUsages)		0.018 ms (0%)	1
Thread-10		15.0 ms (100%)	1
org.juitw.process.collector.CalculateCPUUsages\$.run ()		15.0 ms (100%)	1
org.juitw.process.collector.CalculateCPUUsages.getCurrentCPULoad ()		15.0 ms (99.8%)	15
org.juitw.process.collector.CPUInfo.<init> ()		10.8 ms (72%)	15
org.juitw.process.collector.CPUInfo.output (String[])		4.0 ms (26.8%)	15
Self time		0.175 ms (1.2%)	15
Self time		0.026 ms (0.2%)	1
Thread-9		5.8 ms (100%)	1
org.juitw.timer.Duration\$1\$.run ()		5.8 ms (100%)	1
Self time		5.5 ms (99.4%)	1
org.juitw.timer.Duration\$1\$1\$.<init> (org.juitw.timer.Duration, 1, 1)		0.023 ms (0.5%)	1
org.juitw.timer.Duration.access\$002 (org.juitw.timer.Duration, javax.swing.Timer)		0.004 ms (0.1%)	1

(a2)

Figure 5.6: Analysis of CPU performance in Swift mode (a1) call tree methods for AWT-EventQueue-0, Thread-8 and main (a2) call tree methods for Thread-7, Thread-10 and Thread-9.

In Fig. 5.6 (a2) call tree methods for user Thread-7, Thread-10 and Thread-9 are shown. Here, user Thread-7 gets stopped as soon as the framework start its functioning, whereas Thread-9 and Thread-10 created when the login-duration time gets over and the new login-duration time is asked to input for continue its functioning.

5.4.2 CPU performance in Exhaustive Mode

In this mode, we have analyzed the CPU performance by using ESSA algorithm that checks the user activity for each minute and record the snapshots of total CPU uses for each second in a file. The results obtained from this analysis are shown in Fig. 5.7 (a1), Fig. 5.7 (a2), Fig. 5.7 (a3) etc.

Cal Tree - Method	Time [%] ▾	Time	Invocations
AWT-EventQueue-0		9974 ms (100%)	1
org.jutbw.timer.Durations\$1.actionPerformed (java.awt.event.ActionEvent)		5391 ms (54.1%)	1
org.jutbw.process.collector.CalculateCPUUsages\$1.actionPerformed (java.awt.event.ActionEvent)		4351 ms (43.6%)	19
org.jutbw.visual.LoginDuration\$1.actionPerformed (java.awt.event.ActionEvent)		227 ms (2.3%)	1
org.jutbw.visual.PowerSaverMainWindow\$1.stateChanged (javax.swing.event.ChangeEvent)		2.96 ms (0%)	1
Thread-8		4682 ms (100%)	1
org.jutbw.visual.LoginDuration\$2.run ()		4682 ms (100%)	1
Self time		3702 ms (75.8%)	1
org.jutbw.process.collector.CPUInfo.<init> ()		861 ms (17.6%)	1180
org.jutbw.process.collector.CPUInfo.output (String[])		316 ms (6.5%)	1180
main		198 ms (100%)	1
powersaver.PowerSaver.main (String[])		198 ms (100%)	1
org.jutbw.visual.LoginDuration.<init> ()		154 ms (78%)	1
org.jutbw.visual.LoginDuration.initComponents ()		121 ms (61%)	1
Self time		114 ms (57.6%)	1
org.jutbw.visual.LoginDuration\$1.<init> (org.jutbw.visual.LoginDuration)		0.008 ms (0%)	1
Self time		33.6 ms (16.9%)	1
Self time		43.6 ms (22%)	1
Thread-7		143 ms (100%)	1
org.jutbw.process.collector.CalculateCPUUsages\$3.run ()		143 ms (100%)	1
org.jutbw.process.collector.CalculateCPUUsages.getCurrentCPULoad ()		143 ms (99.8%)	60
Self time		0.206 ms (0.1%)	1
org.jutbw.process.collector.CalculateCPUUsages.access\$900 (org.jutbw.process.collector.CalculateCPUUsages)		0.018 ms (0%)	1

(a1)

Cal Tree - Method	Time [%] ▾	Time	Invocations
Thread-7		143 ms (100%)	1
Thread-10		91.1 ms (100%)	1
Thread-12		67.1 ms (100%)	1
Thread-18		66.8 ms (100%)	1
Thread-14		66.5 ms (100%)	1
Thread-20		65.9 ms (100%)	1
Thread-16		65.9 ms (100%)	1
Thread-24		64.7 ms (100%)	1
Thread-30		64.3 ms (100%)	1
Thread-22		63.5 ms (100%)	1
Thread-26		62.3 ms (100%)	1
Thread-28		61.0 ms (100%)	1
Thread-32		60.9 ms (100%)	1
Thread-9		30.3 ms (100%)	1
Thread-11		27.3 ms (100%)	1
Thread-31		9.88 ms (100%)	1
Thread-17		8.96 ms (100%)	1
Thread-19		8.29 ms (100%)	1
Thread-27		8.20 ms (100%)	1
Thread-15		8.15 ms (100%)	1
Thread-13		7.96 ms (100%)	1
Thread-25		7.89 ms (100%)	1
Thread-29		7.87 ms (100%)	1
Thread-23		7.27 ms (100%)	1

(a2)

Call Tree - Method	Time [%] ▾	Time	Invocations
Thread-10		91.1 ms (100%)	1
org.juitw.process.collector.CalculateCPUUsages\$.run ()		91.1 ms (100%)	1
org.juitw.process.collector.CalculateCPUUsages.getCurrentCPULoad ()		90.9 ms (99.8%)	60
Self time		0.175 ms (0.2%)	1
org.juitw.process.collector.CalculateCPUUsages.access\$900 (org.juitw.process.collector.CalculateCPUUsages)		0.006 ms (0%)	1
Thread-12		67.1 ms (100%)	1
org.juitw.process.collector.CalculateCPUUsages\$.run ()		67.1 ms (100%)	1
org.juitw.process.collector.CalculateCPUUsages.getCurrentCPULoad ()		66.8 ms (99.7%)	60
Self time		0.211 ms (0.3%)	1
org.juitw.process.collector.CalculateCPUUsages.access\$900 (org.juitw.process.collector.CalculateCPUUsages)		0.017 ms (0%)	1
Thread-18		66.8 ms (100%)	1
org.juitw.process.collector.CalculateCPUUsages\$.run ()		66.8 ms (100%)	1
org.juitw.process.collector.CalculateCPUUsages.getCurrentCPULoad ()		66.7 ms (99.8%)	60
Self time		0.102 ms (0.2%)	1
org.juitw.process.collector.CalculateCPUUsages.access\$900 (org.juitw.process.collector.CalculateCPUUsages)		0.004 ms (0%)	1
Thread-14		66.5 ms (100%)	1
org.juitw.process.collector.CalculateCPUUsages\$.run ()		66.5 ms (100%)	1
org.juitw.process.collector.CalculateCPUUsages.getCurrentCPULoad ()		66.4 ms (99.9%)	60
Self time		0.089 ms (0.1%)	1
org.juitw.process.collector.CalculateCPUUsages.access\$900 (org.juitw.process.collector.CalculateCPUUsages)		0.003 ms (0%)	1
Thread-20		65.9 ms (100%)	1
Thread-16		65.9 ms (100%)	1
Thread-24		64.7 ms (100%)	1

(a3)

Figure 5.7: Analysis of CPU performance in exhaustive mode (a1) call tree methods for AWT-EventQueue-0, Thread-8, main, and Thread-7 (a2) various user threads to monitor user activity (a3) few expanded user threads with methods.

In the obtained results, we have analyzed the CPU performance up to 20 minutes using only all project related classes. In these results, we can find the number of variations from the previously discussed scenario. In Fig. 5.7 (a1) shown various call tree methods are similar to methods shown in Fig. 5.6 (a1). 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. 5.7 (a2) various user threads are shown and this scenario is very much similar to discussed scenario for Fig. 5.2 in the thread monitoring section. In Fig. 5.7 (a2), 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. 5.7 (a3) 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 ESSA algorithm invoking the popup window whenever the percentage of total CPU usage found below the threshold for that interval because we have considered the snapshots timing for a minute.

5.5 CONCLUSION

This chapter presents different alternatives to the use of Java Technology for real time implementation and evaluation of the proposed energy sustainable framework. To evaluate the performance of Java code is really a difficult task. Performance-wise Java virtual machine (JVM) is a black box for any system analyst, which hides a lot of detail in comparison to the native systems. JVM hides the various performance measures like memory allocation, performance monitoring counters on the CPU, and thread monitoring by abstracting the underlying hardware and employing custom byte code execution mechanism. These JVMs also employ different just-in-time compilers and different garbage collection algorithms. Therefore, it is difficult to understand the flow of program execution. Moreover, the behaviour of the Java execution environment is not predictable and a number of events can lead to non-deterministic behaviour and consequently to various difficulties in performance evaluation. A variety of tools for JVM monitoring and application profiling are available in the market like Netbeans profiler, JProfiler and Websphere console which can provide a good overview of application behaviour. We have used the Profiler available in NetBeans for the evaluation of proposed framework and monitored various threads, memory allocation, memory leakage, garbage collection and CPU performance. In the various proposed performance evaluation results, we have observed the constant behaviour of the application under swift mode and exhaustive mode. We have obtained no problem regarding memory leakage and garbage collection is performed regularly after a certain period of time. In the thread monitoring analysis, threads are created accordingly as the events taken place in both the modes. During the execution of the proposed framework no unnecessary user threads are created and the CPU performance analysis methods are executed equal to supplied login duration time and no method or process is noticed for CPU over utilization, which in turn is responsible for overall system slowdown. Therefore, these performance evaluation results revealed that the proposed algorithm

achieved the proposed goals both theoretically and practically for designing a complete energy-sustainable framework and algorithms and suggested that changes can be incorporated into the power schemes of the operating systems.

Chapter 6

CONCLUSIONS AND FUTURE SCOPE

6.1 CONCLUSIONS

The rate at which information and communication technology devices are being produced is proportional to the increase in the energy consumed and heat dissipated by these devices that poses the problem of an energy crisis and exacerbation of the greenhouse gas problem and global warming. We cannot escape the fact that the world is becoming more and more dependent upon the use of ICT, and that personal computers are one of the means. It is predicted that the sales of computers are increasing explosively, therefore reducing the energy consumption of electronic systems as well as personal computers are the biggest challenge for researchers and scientists. This emerging issue of the power dissipation has imposed a very significant question on the system and software design and it is believed that in the future there will be a great demand for energy-sustainable software.

Energy consumption is now becoming an emerging area as a dominant performance measure of the computer systems in place of considering the speed of the system. In the computer systems, the energy management can achieve it variety of ways and these methods varies from hardware to software. We have focused on the various techniques required to minimize the energy consumption by the computer systems. These techniques are getting from hardware centric to operating system centric as the software based techniques are easy to implement as well as they are flexible too. We have also discussed the case scenario of Microsoft Windows power schemes and showed the various discrepancies of these power schemes. We found that most of the time these power schemes are not configured properly and all the power saving modes defined by these schemes consume the 3W – 15W of energy in these modes, which is critical from energy saving point of view. We have also presented the general view of DVFS scheduling algorithms, which are more common in modern computer systems for the energy-aware computing and focus mainly on the abstraction of CPU utilization, the

prediction of the trend in CPU utilization, and the association of the voltage and frequency values with CPU utilization. From the various discussed approaches the hardware and software based approaches are getting more attention in this direction. In future, the operating system should supply a module to do power profiling as well as supply configurable accuracy.

For designing an energy efficient computer system, ultimately require the development of fundamental frameworks, algorithmic techniques, and principles that can be used to guide practical solutions. For the proposed energy sustainable framework, we have considered the principle of user centric management as personal computers, televisions, personal media gadgets etc. All these systems are user centric as they receive inputs from the user and deliver services to them so their energy management must be user centric. The main objective of the proposed techniques is to structure concepts, strategies, and activities to design an energy-sustainable power scheme. We have implemented the proposed energy sustainable user centric framework for personal computers, which monitors system workload as well as the user behavior and represents a good alternative for the existing power schemes of Windows operating system. This framework is very much user centric as during login to the system it tries to know the user consensus by knowing the approximate user's work time on the machine and implements the two different modes of working for this framework. The main objective of this framework 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 framework is that it required minimal input and calculations for saving energy. We have also compared the functioning of existing power scheme in Windows operating systems for the proposed user centric energy sustainable framework and it is find that the proposed modes, Swift mode and Exhaustive mode detects the human activity on the computer system in an effective manner and based on the time value supplied by the user during login to the system. In our comparison results, we have found that Swift mode provides more than 66% of energy savings and exhaustive mode provides more than 93% of energy savings over existing power scheme in Windows operating system. For the designed energy sustainable framework, we have proposed two algorithms known as Swift mode algorithm and ESSA algorithm. These algorithms are based on user interactions with the computer system. Swift mode algorithm tries to know the user consensus before

starting work on the computer system by prompting the user to enter time for login duration and continue its working till the login duration time comes at end. Once the given login duration time expires, the algorithm again tries to know the user consensus. In case, user is available, a new login time value may be supplied otherwise algorithm will switch the computer system to hibernate or shutdown mode to minimize energy consumption. However, the proposed ESSA algorithm is very much different in its functioning with the swift mode and claims for more energy saving. 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 algorithm switches the state of the system from idle or inactive to hibernate or shutdown. The working principle 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 various results for cluster machines, it should be below the threshold limit defined by the user to enable the system make the decision to hibernate or shutdown.

In the several performance evaluation results for the proposed framework, we have observed the constant behaviour of the application under swift mode and exhaustive mode. We have observed that there is no any problem regarding memory leakage and garbage collection is performed regularly after a certain period of time. In thread monitoring analysis, threads are created accordingly as the events occurred in both the modes and any unnecessary user threads are not created during the execution of proposed framework. During CPU performance, the analysis methods are executed equal to supplied login duration time and no method or process is noticed for CPU over utilization which in turn is responsible for overall system slowdown. Therefore, these performance evaluation results revealed that the proposed algorithm achieved the proposed goals both theoretically and practically for designing a complete energy-sustainable framework and algorithms. However, the suggested changes can be incorporated into the power schemes of operating systems. We also hope that our framework and algorithms will be useful to the researchers and scientists for the development of a comprehensive solution for the energy-sustainable computing.

6.2 FUTURE SCOPE

Beyond the discussed framework and algorithms in this work, the proposed energy sustainable framework has potential challenges such as the implementation of this kind of framework in to Windows operating systems is a tricky process. In the presented work, we apparently restrict ourselves to the percentage of total CPU usage for performing user centric energy management. However, during the analysis and finding of our results using CPU utilization, we have not considered the usage of primary memory and other peripherals which is not covered in this work but it can play decisive role in making decision by the proposed framework and can be considered into account in the future works. In another decisive factor for the proposed energy sustainable framework that is we have obtained the various results on a Virus and Trojans free computer systems whereas the presence of Virus and Trojans can increase the total CPU utilization so this can be a situation where user is not present on the machine and proposed algorithm fails to make decision. So, this can also be taken into account in the future works.

In another approach about the scalability of the proposed framework, as at present it is very much restricted to personal computers only and in future works, we can use it for servers, mobile devices and other more user centric devices like TV etc. by proposing or enhancing the current algorithms and framework. We can extend this work to make it more users centric by introducing the concept of image processing. As we know that modern mobile devices and computers are equipped with webcam facility in them, which can use it to know the presence of the user on the machine. We hope that, this concept will not only strengthen our ESSA algorithm but also will be able to make more appropriate decision to minimize energy consumption.

REFERENCES

- [1] <http://www.thehindu.com/news/national/article8498.ece>
- [2] Ishfaq Ahmad and Sanjay Ranka, "Handbook of Energy-Aware and Green Computing," Chapman & Hall/CRC Computer and Information Science series, CRC Press, USA, 2012.
- [3] T. Mudge, "Power: a first-class architectural design constraint," *Computer*, vol. 34, no. 4, pp. 52-58, 2001.
- [4] Ruediger Kuehr and Eric Williams, "Computers and the Environment: Understanding and Managing their Impacts," Kluwer Academic Publishers: The Netherlands, 2010.
- [5] http://www.etforecasts.com/products/ES_pcww1203.htm
- [6] J. Romm, A. Rosenfeld, and S. Herrmann, "The internet economy and global warming," Center for Energy and Climate Solutions, December 1999.
- [7] K. Kawamoto, J. G. Koomey, B. Nordman, R. E. Brown, M. A. Piette, M. Ting, and A. K. Meier, "Electricity used by office equipment and network equipment in the US," *Energy*, vol. 27, no. 3, pp. 255-269, 2002.
- [8] M. Bilec, R. Ries, and H. Scott Matthews, "Sustainable development and green design — who is leading the green initiative?" *Journal of Professional Issues in Engineering Education and Practice*, vol. 133, no. 4, pp. 265-269, 2007.
- [9] A. Horvath and H. Scott Matthews, "Advancing sustainable development of infrastructure systems," *Journal of Infrastructure Systems*, vol. 10, no. 3, pp. 77-78, 2004.
- [10] Sukhdeep Singh Sandhu, Arushi Rawal, Prabhjot Kaur, Niyati Gupta, "Major components associated with green networking in information communication technology systems," *Proceedings of International Conference on Computing, Communication and Applications (ICCCA)*, India, 2012, pp. 1-6.
- [11] S. K. S. Gupta, T. Mukherjee, G. Varsamopoulos and A. Banerjee, "Research directions in energy-sustainable cyber-physical systems," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 1, pp. 57 – 74, 2011.
- [12] Marty Poniatowski, "Foundations of Green IT: consolidation, virtualization, efficiency, and ROI in the data center," Prentice Hall, pp. 1 – 321, 2010.

- [13] J. G. Koomey, T. Oey, and Eric Bergman, "The economics of cycling personal computers," *Energy Policy*, vol. 21, no. 9, pp. 937-943, 1993.
- [14] J. G. Koomey, M. D. Levine, J. E. McMahon, A. H. Sanstad, and Eric Hirst, "Energy efficiency policy and market failures," *Annual Review of Energy and the Environment*, vol. 20, no. 1, pp. 535-555, 1995.
- [15] J. G. Koomey, C. A. Webber, and R. E. Brown, "Savings estimates for the Energy Star[®] voluntary labeling program," *Energy Policy*, vol. 28, no. 15, pp. 1137-1149, 2000.
- [16] M. C. Sanchez, R. E. Brown, C. Webber, and G. K. Homan, "Savings estimates for the United States Environmental Protection Agency's ENERGY STAR voluntary product labeling program," *Energy Policy* vol. 36, no. 6, pp. 2098-2108, 2008.
- [17] L. K. Norford, and C. B. Dandridge, "Near-term technology review of electronic office equipment," *Industry Applications Society Annual Meeting, IEEE Conference Record of the 1993, Toronto, Ont., 1993*, pp. 1355-1362.
- [18] C. B. Dandridge, J. Roturier, and L. K. Norford, "Energy policies for energy efficiency in office equipment case studies from Europe, Japan and the USA," *Energy Policy*, vol. 22, no. 9, pp. 735-747, 1994.
- [19] J. A. Roberson, G. K. Homan, A. Mahajan, B. Nordman, C. A. Webber, R. E. Brown, M. McWhinney, and J. G. Koomey, "Energy use and power levels in new monitors and personal computers," *Energy Analysis Department, Environmental Energy Technologies Division, Ernest Orlando Lawrence Berkeley National Laboratory: University of California, Berkeley, CA, 2002*, pp. 1-32.
- [20] C. A. Webber, J. A. Roberson, M. C. McWhinney, R. E. Brown, M. J. Pinckard, and J. F. Busch, "After-hours power status of office equipment in the USA," *Energy*, vol. 31, no. 14, pp. 2823-2838, 2006.
- [21] B. Howarth, B. M. Haddad, and B. Paton, "The economics of energy efficiency: insights from voluntary participation programs," *Energy Policy*, vol. 28, no. 6, pp. 477-486, 2000.
- [22] B. Howarth and Bo Andersson, "Market barriers to energy efficiency," *Energy Economics*, vol. 15, no. 4, pp. 262-272, 1993.
- [23] B. Nordman, M. A. Piette, and K. Kinney, "Measured energy savings and performance of power-managed personal computers and monitors," *Lawrence*

- Berkeley National Laboratory: University of California, Berkeley, CA, 1996, pp. 267-278.
- [24] Chamara Gunaratne, Ken Christensen, and Bruce Nordman, "Managing energy consumption costs in desktop PCs and LAN switches with proxying, split TCP connections, and scaling of link speed," *International Journal of Network Management*, vol. 15, no. 5, pp. 297-310, 2005.
 - [25] R. Brown, C. Webber, and J. G. Koomey, "Status and future directions of the ENERGY STAR program," *Energy*, vol. 27, no. 5, pp. 505-520, 2002.
 - [26] M. Sanchez, C. A. Webber, R. E. Brown, and G. K. Homan, "2007 Status Report: Savings Estimates for the ENERGY STAR(R) Voluntary Labeling Program," 2007.
 - [27] http://publications.gc.ca/collections/collection_2007/nrcan-rncan/M144-145-1-2007E.pdf.
 - [28] A. Walldius, Y. Sundblad, L. Bengtsson, B. Sandblad, and J. Gulliksen. "User certification of workplace software: assessing both artefact and usage." *Behaviour & Information Technology* vol. 28, no. 2, pp.101-120, 2009.
 - [29] <http://center.sustainability.duke.edu/sites/default/files/documents/ecolabelsreport.pdf>
 - [30] <http://www.jeita.or.jp/english/>
 - [31] <http://www.ecolabelindex.com/ecolabel/nordic-ecolabel-or-swan>
 - [32] http://www.cpcb.nic.in/ecomark_logo.php
 - [33] http://beeindia.in/miscellaneous/documents/rti_act/organization_function_duties.pdf
 - [34] http://www.blauer-engel.de/en/blauer_engel/index.php
 - [35] <http://ec.europa.eu/environment/ecolabel>
 - [36] <http://www.ecomark.jp/english/>
 - [37] Z. Bako-Biro, P. Wargocki, C. J. Weschler, and P. O. Fanger, "Effects of pollution from personal computers on perceived air quality, SBS symptoms and productivity in offices," *Indoor Air*, vol. 14, no. 3, pp. 178–187, 2004..
 - [38] P. Wargocki, D. P. Wyon, J. Sundell, G. Clausen, and P. Fanger, "The effects of outdoor supply rate in an office on perceived air quality, sick building syndrome (SBS) symptoms, and productivity," *Indoor Air*, vol. 10, no.4, pp. 222–236, 2000.
 - [39] M. McWhinney, A. Fanara, R. Clark, C. Hershberg, R. Schmeltz, and J. Roberson, "ENERGY STAR product specification development framework:

- using data and analysis to make program decisions,” *Energy Policy*, vol. 33, no. 12, pp.1613-1625, 2005.
- [40] S. Murugesan, “Harnessing green it: principles and practices,” *IEEE IT Professional*, vol. 10, no. 1, pp. 24–33, 2008.
 - [41] Kivel, Seppo, and Kari Vigelius, “Method for display power management and monitor equipped with a power management function,” U.S. Patent 6,404,423, June 11, 2002.
 - [42] P. M. Greenawalt, “Modeling power management for hard disks,” *Proceedings of the 2nd IEEE International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'94)*, Durham, NC, 1994, pp. 62-66.
 - [43] J. Steele, “ACPI thermal sensing and control in the PC,” *Conference Proceedings of IEEE Wescon*, Anaheim, CA, Sept. 15-17, 1998, pp. 169-182.
 - [44] T. Lewis, “Method to reflect BIOS set up changes into ACPI machine language,” U.S. Patent 6,167,511, December 26, 2000.
 - [45] J. H. Ewertz, “Method of dynamically changing the lowest sleeping state in ACPI,” U.S. Patent 6,499,102, issued December 24, 2002.
 - [46] B. Nordman, M. A. Piette, K. Kinney, and C. Webber, “User guide to power management for PCs and monitors,” *Environmental Energy Technologies Division, Lawrence Berkeley National Laboratory: University of California, Berkeley, CA*, 1997, pp. 1-64.
 - [47] B. Nordman, A. Meier, and M. A. Piette, “PC and monitor night status: Power management enabling and manual turn-off,” *Lawrence Berkeley National Laboratory: University of California, Berkeley, CA*, 1998, pp. 1-12.
 - [48] J. G. Koomey, M. A. Piette, M. Cramer, and J. H. Eto, “Efficiency improvements in US office equipment: expected policy impacts and uncertainties,” *Energy Policy*, vol. 24, no. 12, pp. 1101-1110, 1996.
 - [49] C. A. Webber, J. A. Roberson, R. Brown, C. Payne, B. Nordman, and J. Koomey, “Field surveys of office equipment operating patterns,” *Draft Report, Lawrence Berkeley National Laboratory: University of California, Berkeley, CA*, 2001, pp. 1-30.
 - [50] C. Webber, D. Korn, and M. Sanchez, “Savings potential of ENERGY STAR external power adapters and battery chargers,” *Lawrence Berkeley National Laboratory: University of California, Berkeley, CA*, 2007, pp. 1-15.

- [51] R. E. Picklum, B. Nordman, and B. Kresch, "Guide to reducing energy use in office equipment," Bureau of Energy Conservation, City and County of San Francisco and Energy Analysis Department, Lawrence Berkeley National Laboratory: University of California, Berkeley, CA, 1999, pp. 1-24.
- [52] C. A. Balaras, K. Droutsas, A. A. Argiriou, and K. Wittchen, "Assessment of energy and natural resources conservation in office buildings using TOBUS," *Energy and Buildings*, vol. 34, no. 2, pp. 135-153, 2002.
- [53] Hui Chen, Youhui Li and Weisong Shi, "Fine-grained power management using process-level profiling", *Sustainable Computing: Informatics and Systems*, vol. 2, no. 1, pp. 33 – 42, Mar 2012.
- [54] P. K. Gupta and G Singh, "Minimizing power consumption by personal computers: a technical survey," *International Journal of Information Technology and Computer Science*, vol. 4, no. 10, pp. 57- 66, 2012.
- [55] <http://www.zdnet.com/blog/microsoft/idc-windows-server-still-rules-the-server-roost/6424>
- [56] <http://blogs.msdn.com/b/b8/archive/2012/02/07/improving-power-efficiency-for-applications.aspx>.
- [57] http://www.verismic.com/pdf/Verismic_VPM_vs_Windows.pdf
- [58] D. Brooks, V. Tiwari, and M. Margaret, "Wattch: a framework for architectural-level power analysis and optimizations," *ACM SIGARCH Computer Architecture News*, vol. 28, no. 2, pp. 83-94, 2000.
- [59] D. Brooks, M. David, P. Bose, S. E. Schuster, J. Hans, N. K. Prabhakar, B. Alper, J. Wellman, Z. Victor, M. Gupta, and P. W. Cook, "Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors," *IEEE Micro*, vol. 20, no. 6, pp. 26-44, 2000.
- [60] A. Vahdat, A. Lebeck, and S. E. Carla, "Every joule is precious: The case for revisiting operating system design for energy efficiency," *Proceedings of the 9th ACM SIGOPS European Workshop "Beyond the PC: new challenges for the operating system"*, Denmark, 2000, pp. 31-36.
- [61] H. Zeng, S. E. Carla, A. Lebeck, and A. Vahdat, "ECOSystem: managing energy as a first class operating system resource," *ACM SIGPLAN Notices*, vol. 37, no. 10, pp. 123-132, 2002.

- [62] A. Merkel and F. Bellosa, "Balancing power consumption in multiprocessor systems," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 4, pp. 403-414, 2006.
- [63] D. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 6, pp. 663-670, 1994.
- [64] H. S. Wang, Z. Xinping, Li-Shiuan Peh and Sharad Malik, "Orion: a power-performance simulator for interconnection networks," *Proceedings of 35th Annual IEEE/ACM International Symposium on Microarchitecture*, 2002, pp. 294-305.
- [65] W. Ye, V. Narayanan, M. Kandemir and J. I. Mary, "The design and use of simple power: a cycle-accurate energy estimation tool," *ACM Proceedings of the 37th Annual Design Automation Conference*, 2000, pp. 340-345.
- [66] R. Joseph, D. Brooks and M. Margaret, "Live, runtime power measurements as a foundation for evaluating power/performance tradeoffs," *Workshop on Complexity Effective Design (WCED)*, vol. 28, 2001, pp. 1-9.
- [67] S. Kamil, S. John, and S. Erich, "Power efficiency in high performance computing," *Proc. IEEE International Symposium on Parallel and Distributed Processing*, Miami, FL, 2008, pp. 1-8.
- [68] J. R. Lorch, and A. J. Smith, "Apple Macintosh's energy consumption," *IEEE Journal of Microelectronics*, vol. 18, no. 6, pp. 54-63, 1998.
- [69] W. L. Bircher and L. K. John, "Complete system power estimation using processor performance events," *IEEE Trans. on Computers*, vol. 61, no. 4, pp. 563-577, 2012.
- [70] R. Bertran, M. Gonzalez, M. Xavier, N. Navarro, and E. Ayguade, "Decomposable and responsive power models for multicore processors using performance counters," *Proceedings of the 24th ACM International Conference on Supercomputing*, Oregon, USA, 2010, pp. 147-158.
- [71] C. Isci, and M. Martonosi, "Runtime power monitoring in high-end processors: methodology and empirical data," *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, San Diego, CA, 2003, pp. 93.
- [72] T. Do, S. Rawshdeh, and W. Shi, "ptop: A process-level power profiling tool," *Proceedings of the 2nd Workshop on Power Aware Computing and Systems (HotPower'09)*, Big Sky, MT, 2009, pp. 1-5.

- [73] A. Kansal, Z. Feng, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," Proceedings of the 1st ACM symposium on Cloud computing (SoCC '10), Indianapolis, IN, 2010, pp. 39-50.
- [74] Y. H. Lu, L. Benini, and G. D. Micheli, "Power-aware operating systems for interactive systems," IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 10, no. 2, pp. 119-134, 2002.
- [75] T. Li, and L. K. John, "Run-time modelling and estimation of operating system power consumption," Proceedings of ACM SIGMETRICS International Conference on Measurement and Modeling of Computer systems, USA, 2003, pp. 160 - 171
- [76] G. Dhiman, M. Kresimir, and R. Tajana, "A system for online power prediction in virtualized environments using Gaussian mixture models," Proceedings of the 47th ACM/IEEE Design Automation Conference (DAC), Anaheim, CA, 2010, pp. 807-812.
- [77] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," Proceedings of the 9th Workshop on ACM SIGOPS European Workshop: Beyond the PC: New Challenges for the Operating System, Denmark, 2000, pp. 37-42.
- [78] R. Joseph, and M. Martonosi, "Run-time power estimation in high performance microprocessors," Proceedings of International Symposium on Low Power Electronics and Design, 2001, California, USA, pp. 135-140.
- [79] G. Contreras, and M. Margaret, "Power prediction for intel XScale® processors using performance monitoring unit events," Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED'05), San Diego, CA, 2005, pp. 221-226.
- [80] W. L. Bircher, M. Valluri, J. Law, and L. K. John, "Runtime identification of microprocessor energy saving opportunities," Proceedings of International Symposium on Low Power Electronics and Design (ISLPED'05), San Diego, CA, 2005, pp. 275-280.
- [81] Dam Sunwoo, Hassan Al-Sukhni, Jim Holt, and Derek Chiou, "Early models for system-level power estimation," Proceedings of IEEE International Workshop on Microprocessor Test and Verification, Austin, TX, 2008, pp. 8 – 14.
- [82] Sudhanva Gurumurthi, Anand Sivasubramaniam, Mary Jane, Irwin N., Vijaykrishnan, and Mahmut Kandemir, "Using complete machine simulation for

- software power estimation: the softwatt approach,” Proceedings of International Symposium on High-Performance Computer Architecture (HPCA.02), Austin, TX, 2002, pp. 1 – 10.
- [83] Mendel Rosenblum, Stephen A. Herrod, Emmett Witchel, and Anoop Gupta, “Complete computer system simulation: the simOS approach,” IEEE Parallel and Distributed Technology: Systems and Applications, vol. 3, no. 4, pp. 34- 43, 1995.
 - [84] J. Chen, M. Dubois and, P. Stenstrom, “Simwattch: integrating complete-system and user-level performance and power simulators,” IEEE Micro, vol. 27, no. 4, pp. 34 – 48, 2007.
 - [85] P. Bohrer, E. N. Elnozahy, K. Tom, K. Michael, L. Charles, M. Chandler, and R. Rajamony, “The case for power management in web servers,” Power Aware Computing, Kluwer Academic Publisher, Norwell, MA, USA, 2002.
 - [86] Song Shuaiwen, Rong Ge, X. Feng, and Kirk W. Cameron, "Energy profiling and analysis of the HPC challenge benchmarks," International Journal of High Performance Computing Applications , vol. 23, no. 3, pp. 265-276, 2009.
 - [87] Rong Ge, X. Feng, Song Shuaiwen, Hung-Ching Chang, Dong Li, and Kirk W. Cameron, "Powerpack: energy profiling and analysis of high-performance systems and applications," IEEE Trans. on Parallel and Distributed Systems, vol. 21, no. 5, pp. 658-67, 2010.
 - [88] X. Feng, Rong Ge, and Kirk W. Cameron, "Power and energy profiling of scientific applications on distributed systems," Proceedings of IEEE International Parallel and Distributed Processing Symposium, Denver, Colorado, 2005, pp. 34-34.
 - [89] D. Foster, L. Shaun, B. Mark, and C. Paul, “Wattsup?: motivating reductions in domestic energy consumption using social networks,” Proceedings of ACM Nordic Conference on Human-Computer Interaction: Extending Boundaries, Reykjavik, Iceland, 2010, pp. 178-187.
 - [90] A. M. Viredaz and A. W. Deborah, “Power evaluation of Itsy version 2.3,” Technical Note TN-57, WRL, Compaq, Palo Alto, USA, 2000.
 - [91] D. C. Snowdon, R. Sergio, and G. Heiser, “Power management and dynamic voltage scaling: Myths and facts,” Proceedings of the Workshop on Power Aware Real-time Computing, New Jersey, USA, 2005, pp. 1-7.

- [92] C. Xian, L. Cai, and Y. H. Lu, "Power measurement of software programs on computers with multiple I/O components," *IEEE Trans. on Instrumentation and Measurement*, vol. 56, no. 5, pp. 2079 – 2086, October 2007.
- [93] Y. H. Lu and G. D. Micheli, "Comparing system-level power management policies," *Proceedings of IEEE Design and Test, Munich*, 2001, pp. 10 – 19.
- [94] Gary Cameron, "Baseline measurement of software driven power consumption," *Proceedings of Instrumentation and Measurement Technology Conference (IMTC)*, Ottawa, Canada, 2005, pp. 2088 – 2090.
- [95] V. Tiwari, S. Malik, A. Wolfe, and M. T.C. Lee, "Power analysis of embedded software: a first step towards software power minimization," *IEEE Trans. Very Large Scale Integr. (VLSI) System*, vol. 2, no. 4, pp. 437– 445, Dec. 1994.
- [96] T. Laopoulos, P. Neofotistos, C. A. Kosmatopoulos, and S. Nikolaidis, "Measurement of current variations for the estimation of software-related power consumption," *IEEE Trans. on Instrumentation and Measurement*, vol. 52, no. 4, pp. 1206 – 1212, August 2003.
- [97] M. Arunadevi, and R. S. D. Wahidabanubb, "Design of Power Efficient Schema for Energy Optimization in Data Center With Massive Task Execution Using DVFS," *International Journal of Computer Science Issues*, vol. 9, no. 1, pp. 407-414, 2012.
- [98] J. M. Baron, "Electronic tour guide and photo location finder," U.S. Patent 6,459,388, October 1, 2002.
- [99] P. M. Cohen, A. M. Christopher, and J. C. Ronald, "Power supply with interface to determine power requirements of devices coupled thereto," U.S. Patent 6,512,682, January 28, 2003.
- [100] Loper, Joe, and Sara Parr, "Energy efficiency in data centers: a new policy frontier," *Environmental Quality Management*, vol. 16, no. 4, pp. 83-97, 2007.
- [101] T. Daim, J. Justice, M. Krampits, M. Letts, G. Subramanian, and M. Thirumalai, "Data center metrics: an energy efficiency model for information technology managers," *Management of Environmental Quality: An International Journal*, vol. 20, no. 6, pp. 712-731, 2009.
- [102] M. Poess, and R. O. Nambiar, "Tuning servers, storage and database for energy efficient data warehouses," *Proc. IEEE International Conference on Data Engineering (ICDE)*, California, USA, 2010, pp. 1006-1017.

- [103] W. Feng and K. Cameron, "The green500 list: encouraging sustainable supercomputing," *Computer*, vol. 40, no.12, pp. 50–55, 2007
- [104] <http://www.spec.org/specpower/>
- [105] M. Poess, R. O. Nambiar, K. Vaid, J. M. Stephens, K. Huppler, and E. Haines, "Energy benchmarks: a detailed analysis," *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, Passau, Germany, 2010, pp. 131-140.
- [106] R. G. Matthew, J. S. Ringenberg, Dan Ernst, T. M. Austin, Trevor Mudge, and R. B. Brown, "MiBench: a free, commercially representative embedded benchmark suite," *Proc. IEEE International Workshop on Workload Characterization*, 2001. pp. 3-14.
- [107] AMD-ACP, http://www.amd.com/us/documents/43761c_acp_wp_ee.pdf
- [108] I. Ahmad and J. Luo, "On using game theory to optimize the rate control in video coding," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 16, no. 2, pp. 209-219, 2006.
- [109] S. U. Khan and I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids," *IEEE Trans. on Parallel and Distributed Systems*, vol. 20, no. 3, pp. 346-360, 2009.
- [110] L. Benini, A. Bogliolo, and G. D. Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 299 – 316, June 2000.
- [111] L. Benini, A. Bogliolo, G. A. Paleologo, and G. D. Micheli, "Policy optimization for dynamic power management," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 6, pp. 813 – 834, June 1999.
- [112] Keqin Li, "Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed," *IEEE Trans. on Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1484 – 1497, November 2008.
- [113] S. Wang, J. Liu, J. J. Chen, and X. Liu, "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*, vol. 1, no. 3, pp. 289 – 298, September 2011.

- [114] K. Huang, L. Santinelli, J. J. Chen, L. Thiele, and G. C. Buttazzo, "Adaptive power management for real-time event streams," *Proceedings of IEEE Asia and South Pacific Design Automation Conference*, Taipei, Taiwan, 2010, pp. 7 – 12.
- [115] A. Abbasian, S. Hatami, A. A. Kusha, M. Nourani, and C. Lucas, "Event-driven dynamic power management based on wavelet forecasting theory" *Proceedings of International Symposium on Circuits and Systems (ISCAS)*, 2004, pp. 325 – 328.
- [116] A. H. Hwang, and Allen C.H. Wu, "A predictive system shutdown method for energy saving of event-driven computation," *ACM Transactions on Design Automation of Electronic Systems*, vol. 5, no. 2, pp. 226–241, April 2000.
- [117] M. B. Srivastava, A. P. Chandrakasan, and R. W. Brodersen, "Predictive system shutdown and other architecture techniques for energy efficient programmable computation," *IEEE Trans. on VLSI Systems*, vol. 4, no. 1, pp. 42-55, March 1996.
- [118] Q. Jiang, H.S. Xi, and B.Q. Yin, "Adaptive optimisation of timeout policy for dynamic power management based on semi-Markov control processes," *IET Control Theory and Applications*, vol. 4, no. 10, pp. 1945 – 1958, 2010.
- [119] R. Golding, P. Bosch, and J. Wilkes, "Idleness is not sloth," *Hewlett- Packard Laboratories*, Palo Alto, CA, 1996, pp. 96-140.
- [120] F. Douglass, P. Krishnan, and B. Bershad, "Adaptive disk spin-down policies for mobile computers," *Proceedings of 2nd USENIX Symposium Mobile Location-Independent Computing*, MI, USA, Apr. 1995, pp. 121–137.
- [121] D. Helmbold, D. Long, and E. Sherrod, "Dynamic disk spin-down technique for mobile computing," *Proceedings of IEEE Conference Mobile Computing*, NewYork, USA, 1996, pp. 130–142.
- [122] Y. Lu and G. D. Micheli, "Adaptive hard disk power management on personal computers," *Proceedings of IEEE Great Lakes Symposium VLSI*, MI, USA, Mar. 1999, pp. 50–53.
- [123] E. Chung, L. Benini, and G. D. Micheli, "Dynamic power management using adaptive learning trees," in *Proceedings of International Conference on Computer-Aided Design*, San Jose, California, Apr. 1999, pp. 221–227.

- [124] L. Benini, R. Hodgson, and P. Siegel, "System-level power estimation and optimization," Proceedings of International Symposium Low Power Electronics Design, California, USA, Aug. 1998, pp. 173–178.
- [125] P. Krishnan, P. Long, and J. Vitter, "Adaptive disk spin-down via optimal rent-to-buy in probabilistic environments," International Conference on Machine Learning, California, USA, July 1995, pp. 322–330.
- [126] T. Simunic, L. Benini, and G. D. Micheli, "Event-driven power management," Proceedings of International Symposium System Synthesis, Boca Raton, Florida, USA, Apr. 1999, pp. 18–23.
- [127] T. Simunic, L. Benini, Peter Glynn, and Giovanni de Micheli, "Dynamic Power management of laptop hard disk," Proceedings of Design, Automation, Test Eur., Paris, France, Mar. 2000, pp. 736.
- [128] T. Simunic, H. Vikalo, P. Glynn, and G. de Micheli, "Energy efficient design of portable wireless devices," Proceedings of International Symposium Low Power Electronics Design, Rapallo, Italy, Aug. 2000, pp. 49–54.
- [129] T. Simunic, L. Benini, P. Glynn, and G. D. Micheli, "Dynamic power management for portable systems," Proceedings of International Conference on Mobile Computing and Networking, Boston, USA, Oct. 2000, pp. 22–32.
- [130] E. Chung, L. Benini, and G. D. Micheli, "Dynamic power management for non-stationary service requests," Proceedings of Design, Automation Test Eur., Munich, Germany, Mar. 1999, pp. 77–81.
- [131] Q. Qiu and M. Pedram, "Dynamic power management based on continuous-time Markov decision processes," Proceedings of Design Automation Conference, New Orleans, LA, USA, June 1999, pp. 555–561.
- [132] Q. Qiu, Q. Wu and M. Pedram, "Dynamic power management of complex systems using generalized stochastic petri-nets," Proceedings of Design Automation Conference, Los Angeles, California, USA, June 2000, pp. 352–356.
- [133] Q. Qiu, Q. Wu, and M. Pedram, "Stochastic modelling of a power-managed system—construction and optimisation," IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems, vol. 20, no. 10, pp. 1200–1217, 2001.

- [134] B. Kveton, P. Gandhi and G. Theocherous, "Adaptive timeout policies for fast fine-grained power management", Proceedings of National Conference on Artificial Intelligence, Vancouver, July 2007, pp. 1795–1800
- [135] Y. H. Lu, T. Hnwzic, and G. D. Micheli, "Software Controlled Power Management," Proceedings of the seventh international workshop on Hardware/software codesign, 1999, Rome, Italy, pp. 157 – 161.
- [136] M. Sarkar, and R. L. Cruz, "An adaptive "Sleep," algorithm for efficient power management in WLANs," Proc. IEEE 61st Vehicular Technology Conference, Stockholm, Sweden, vol. 3, June 2005, pp. 2101 – 2104.
- [137] R. Zheng, J. C. Hou, and L. Sha, "On timeout driven power management policies in wireless networks," Proceedings of IEEE Global Telecommunications Conference, Dallas, 2004, vol. 6, pp. 4097–4103
- [138] P. Rong, and M. Pedram, "Determining the optimal timeout values for a power-managed system based on the theory of Markovian processes: offline and online algorithms," Proceedings of Design Automation and Testing Europe (DATE), Munich, March 2006, pp. 1128–1133.
- [139] Hewlett-Packard, Intel, Microsoft, Phoenix, and Toshiba, "Advanced configuration and power interface specification," ACPI Specification Document, Revision 3, 2004.
- [140] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd. and Toshiba Corporation, "Advanced Configuration and Power Interface Specification," Rev – 5, December 2011, pp. 1 – 958.
- [141] C. H. Hsu and W. C. Feng, "A feasibility analysis of power awareness in commodity-based high-performance clusters," IEEE International Cluster Computing, Cardiff, UK, 2005, pp. 1-10.
- [142] V. W. Freeh, D. K. Lowenthal, P. Feng, N. Kappiah, R. Springer, L. B. Rountree, and E. M. Femal, "Analyzing the energy-time trade-off in high-performance computing applications," IEEE Transactions on Parallel and Distributed Systems, vol. 18, no. 6, pp. 835-848, 2007.
- [143] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation, Monterey, CA, 1994, pp. 449-471.

- [144] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," *Proceedings of Annual Symposium on Foundations of Computer Science*, Milwaukee, WI, October 1995, pp. 374-382.
- [145] F. Krisztián, S. Reinhardt, and T. Mudge, "Automatic performance setting for dynamic voltage scaling," *Wireless Networks*, vol. 8, no. 5, pp. 507-520, 2002.
- [146] F. Jason, and M. Satyanarayanan, "Managing battery lifetime with energy-aware adaptation," *ACM Trans. on Computer Systems*, vol. 22, no. 2 pp. 137-179, 2004
- [147] K. Govil, E. Chan, and H. Wasserman, "Comparing algorithm for dynamic speed setting of a low-power CPU," *Proceedings of the first International Conference on Mobile Computing and Networking*, Berkeley, CA, November 1995, pp. 13-25.
- [148] D. Grunwald, P. Levis, K. Farkas, C. Morrey III, and M. Neufeld, "Policies for dynamic clock scheduling," *Proceedings of the 4th Conference on Symposium on Operating System Design and Implementation*, San Diego, CA, October 2000, vol. 4, pp. 6-6.
- [149] J. Lorch and A. Smith, "PACE: a new approach to dynamic voltage scaling," *IEEE Trans. on Computers*, vol. 53, no. 7, pp. 856-869, 2004.
- [150] B. C. Mochocki, S. H. Xiaobo, and G. Quan, "A unified approach to variable voltage scheduling for nonideal DVS processors," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 9, pp. 1370-1377, 2004.
- [151] T. Pering, T. Burd and R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms," *Proceedings of International Symposium on Low-Power Electronics Design (ISLPED 98)*, Monterey, CA, August 1998, pp. 76-81.
- [152] T. Pering, T. Burd, and R. Brodersen, "Voltage scheduling in the IpARM microprocessor system," *Proceedings of IEEE International Symposium on Low Power Electronics and Design (ISLPED'00)*, Monterey, CA, 2000, pp. 96-101.
- [153] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1571-1580, 2000.

- [154] N. Pettis, Le Cai, and Y. H. Lu, "Statistically optimal dynamic power management for streaming data," *IEEE Trans. on Computers*, vol. 55, no. 7, pp. 800-814, 2006.
- [155] V. W. Freeh, N. Kappiah, D. K. Lowenthal, and T. K. Bletsch, "Just-in-time dynamic voltage scaling: exploiting inter-node slack to save energy in MPI programs," *Journal of Parallel and Distributed Computing* vol. 68, no. 9, pp. 1175-1185, 2008.
- [156] L. M. Yeol, V. W. Freeh, and D. K. Lowenthal, "Adaptive, transparent CPU scaling algorithms leveraging inter-node MPI communication regions," *Parallel Computing*, vol. 37, no. 10 pp. 667-683, 2011.
- [157] R. Barry, D. K. Lowenthal, B. R. Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, "Adagio: making dvs practical for complex hpc applications," *Proceedings of ACM International Conference on Supercomputing*, Portland, Oregon, 2009, pp. 460-469.
- [158] Y. H. Lu, L. Benini, and G. D. Micheli, "Power aware operating systems for interactive systems," *IEEE Trans. on Very Large Scale Integration (VLSI) systems*, vol. 10, no. 2, pp. 119-134, April 2002.
- [159] A. Varma, B. Ganesh, M. Sen, S. Choudhary, L. Srinivasan, and B. Jacob, "A control theoretic approach to dynamic voltage scaling," *Proceedings of International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES '03)*, San Jose, CA, October 2003, pp. 255-266.
- [160] K.Y. Mun, D.W. Kim, D.H. Kim, and C. I. Park, "dDVS: An efficient dynamic voltage scaling algorithm based on the differential of CPU utilization," *Proceedings of 9th Asia-Pacific Conference (ACSAC 2004)*, Beijing, China, September 2004, pp. 160-169.
- [161] John Clemens, <http://www.deater.net/john/powernowd.html>, 2003-2011.
- [162] Carl Thompson, <http://www.carlthompson.net/software/cpuspeed>, 2008.
- [163] V. Pallipadi and A. Starikovskiy, "The Ondemand Governor past, present, and future," *Proceedings of the LINUX Symposium*, Ottawa, Canada, July 2006, vol. 2, pp. 223-238.
- [164] D. A. Bader, V. Agarwal, K. Madduri, and S. Kang, "High performance combinatorial algorithm design on the Cell broadband Engine processor," *Parallel Computing*, vol. 33, no. 10, pp. 720-740, 2007.

- [165] B. Francesco, and R. Giacobazzi, "A fast implementation of the octagon abstract domain on graphics hardware," *Static Analysis, LNCS*, vol. 4634, 2007, pp. 315-332.
- [166] G. Egri, Z. Fodor, C. Hoelbling, S. D. Katz, D. Nogradi, and K. K. Szabó, "Lattice QCD as a video game," *Computer Physics Communications*, vol. 177, no. 8, pp. 631-639, 2007.
- [167] I. Khaled and F. Bodin, "Implementing Wilson-Dirac operator on the cell broadband engine," *Proceedings of ACM Annual International Conference on Supercomputing*, Austin, TX, 2008, pp. 4-14.
- [168] I. Khaled and F. Bodin, "Efficient simdization and data management of the lattice qcd computation on the cell broadband engine," *Scientific Programming*, vol. 17, no. 1, pp. 153-172, 2009.
- [169] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80-113. 2007.
- [170] J. Spray, J. Hill, and A. Trew, "Performance of a lattice quantum chromodynamics kernel on the cell processor," *Computer Physics Communications*, vol. 179, no. 9, pp. 642-646, 2008.
- [171] S. Williams, J. Shalf, L. Oliker, S. Kamil, P. Husbands, and K. Yelick, "Scientific computing kernels on the cell processor," *International Journal of Parallel Programming*, vol. 35, no. 3 pp. 263-298, 2007.
- [172] R. Allen, and K. Kennedy, "Automatic translation of Fortran programs to vector form," *ACM Trans. on Programming Languages and Systems (TOPLAS)*, vol. 9, no. 4 pp. 491-542, 1987.
- [173] M. M. Baskaran, U. Bondhugula, S. Krishnamoorthy, J. Ramanujam, A. Rountev, and P. Sadayappan, "A compiler framework for optimization of affine loop nests for GPGPUs," *Proceedings of ACM Annual International Conference on Supercomputing*, Austin, TX, 2008, pp. 225-234.
- [174] S. Lee, S.J. Min, and R. Eigenmann, "OpenMP to GPGPU: a compiler framework for automatic translation and optimization," *ACM Sigplan Notices*, vol. 44, no. 4, pp. 101-110, 2008.
- [175] Kevin O'Brien, Kathryn O'Brien, Zehra Sura, Tong Chen, and Tao Zhang, "Supporting OpenMP on cell," *International Journal of Parallel Programming*, vol. 36, no. 3 pp. 289-311, 2008.

- [176] S. Ryoo, C. I. Rodrigues, S. S. Stone, S. S. Baghsorkhi, S. Z. Ueng, J. A. Stratton, and W. W. Hwu, "Program optimization space pruning for a multithreaded gpu," Proceedings of the 6th Annual IEEE/ACM International Symposium on Code Generation and Optimization, Boston, MA, USA, 2008, pp. 195-204.
- [177] S. Williams, L. Oliker, R. Vuduc, J. Shalf, K. Yelick, and J. Demmel, "Optimization of sparse matrix-vector multiplication on emerging multi-core platforms," Parallel Computing, vol. 35, no. 3, pp. 178-194, 2009.
- [178] Michael Wolfe, "Implementing the PGI accelerator model," Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units, Pittsburgh, PA, USA, 2010, pp. 43-50.
- [179] Francois Bodin and Stephane Bihan, "Heterogeneous multicore parallel programming for graphics processing units," Scientific Programming, vol. 17, no. 4 pp. 325-336, 2009.
- [180] P. Bellens, J. M. Perez, F. Cabarcas, A. Ramirez, R. M. Badia, and J. Labarta, "CellSs: Scheduling techniques to better exploit memory hierarchy," Scientific Programming, vol. 17, no. 1 pp.77-95, 2009.
- [181] Clint Whaley, Antoine Petitet, and Jack J. Dongarra, "Automated empirical optimizations of software and the ATLAS project," Parallel Computing, vol. 27, no. 1 pp. 3-35, 2001.
- [182] M. Puschel, J. Moura, J. R. Johnson, D. Padua, M. M. Veloso, B. W. Singer, and J. Xiong, "SPIRAL: code generation for DSP transforms," Proceedings of the IEEE, vol. 93, no. 2 pp. 232-275, 2005.
- [183] M. Frigo, and S. G. Johnson, "FFTW: an adaptive software architecture for the FFT," Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, Seattle, Washington, USA, 1998, vol. 3, pp. 1381-1384.
- [184] R. Vuduc, J. W. Demmel, and Katherine A. Yelick, "OSKI: a library of automatically tuned sparse matrix kernels," Journal of Physics: Conference Series, vol. 16, no. 1, pp. 521, 2005.
- [185] L. Parolini, B. Sinopoli, and B. H. Krogh, "Reducing data center energy consumption via coordinated cooling and load management," Proceedings of conference on Power aware computing and systems, (HotPower'08), San Diego, CA, USA, 2008, vol. 8, pp. 14-14.

- [186] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," *ACM SIGARCH Computer Architecture News*, vol. 38, no. 3, pp. 338-347, 2010.
- [187] R. A. Bradley, "Climate Change Mitigation and Copenhagen: An IEA Perspective," *International Energy Agency*, 2009, pp. 1-32.
- [188] R. Mooney, K. P. Schmidt, and R. S. Studham, "NWPerf: a system wide performance monitoring tool for large Linux clusters," *Proc. IEEE International Conference on Cluster Computing*, Sept. 20-23, 2004, pp. 379-389.
- [189] Shinan Wang, Hui Chen and Weisong Shi, "SPAN: A software power analyzer for multicore computer systems", *Sustainable Computing: Informatics and Systems*, vol. 1, no. 1, pp. 23 – 34, Mar 2011.
- [190] 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*, vol. 1, no. 2, pp. 113 – 133, Jun 2011.
- [191] 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*, vol. 1, no. 4, pp. 294 – 304, 2011.
- [192] Q. Tang, T. Mukherjee, S. K. S. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efficient high-performance data centers," *Proc. 4th International Conference on Intelligent Sensing and Information Processing, 2006 (ICISIP 2006)*, 2006, pp. 203-208.
- [193] T. Heath, A. P. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini, "Mercury and Freon: temperature emulation and management for server systems," *ACM SIGARCH Computer Architecture News*, vol. 34, no. 5, pp. 106-116, 2006.
- [194] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: a cyber-physical approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1458-1472, 2008.
- [195] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing power using transformations," *IEEE Trans. on Computer-Aided*

Design of Integrated Circuits and Systems, vol. 14, no. 1, pp. 12-31, January 1995.

- [196] K. K. Parhi, "Algorithm transformation techniques for concurrent processors," Proceedings of IEEE, vol. 77, no. 12, pp. 1879-1895, December 1989.
- [197] M. Potkonjak and J. Rabaey, "Fast implementation of recursive programs using transformations," Proceedings of the IEEE International Conference on Acoustics, Speech and Signal processing (ICASSP), San Francisco, CA, March 1992, vol. 5, pp. 569-572.
- [198] C. Leiserson and J. Saxe, "Optimizing synchronous systems," Proceedings of the 22nd Annual Symposium on Foundations of Computer Science (SFCS'81), Nashville, TN, USA, 1981, pp. 23-36.
- [199] H. Loomis, and B. Sinha, "High speed recursive digital filter realization," Circuits, Systems, and Signal Processing, vol. 3, no. 3, pp. 267-294, 1984.
- [200] K. K. Parhi and D.G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters," IEEE Trans. on Acoustics, Speech and Signal Processing, vol. 37, no. 7, pp. 1099-1117, July 1989.
- [201] S. Malik and S. Devadas, "A survey of optimization techniques targeting low power VLSI circuits," Proceedings of the 32nd ACM/IEEE Design Automation Conference (DAC'95), San Francisco, CA, 1995, pp. 242-247.
- [202] S. Iman and M. Pedram, "Logic Synthesis for Low power VLSI Designs," Kluwer Academic Publisher, MA, USA, pp. 236, 1998.
- [203] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Pre-computation-based sequence logic optimization for low-power," IEEE Trans. on Very Large Scale Integration (VLSI) systems, vol. 2, no. 4, pp. 426-436, December 1994.
- [204] M. C. Chang, C. S. Chang, C. P. Chao, K. I. Goto, M. Jeong, L. C. Lu, and C. H. Diaz, "Transistor and circuit design optimization for low power CMOS," IEEE Trans. on Electron Devices, vol. 55, no. 1, pp. 84-95, January 2008.
- [205] K. Chen and C. Hu, "Device and technology optimizations for low power design in deep sub-micron regime," Proceedings of ACM/IEEE International Symposium on Low-Power Electronic Design (ISLPED 97), Monterey, USA, 1997, pp. 312-316.
- [206] J. Rabaey, "Reconfigurable processing: the solution to low-power programmable DSP," Proceedings of the IEEE International Conference on

- Acoustics, Speech and Signal Processing (ICASSP 97), Munich, Germany, April 1997, vol. 1, pp. 275-278.
- [207] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen, "Dynamic voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 11, pp. 1571-1580, November 2000.
 - [208] M. Goel and N. R. Shanbhag, "Dynamic algorithm transformations (DAT) – a systematic approach to low-power reconfigurable signal processing," *IEEE Trans. on Very Large Scale Integration (VLSI) systems*, vol. 7, no.4, pp. 463-476, December 1999.
 - [209] A. Sinha, A. Wang and A. Chandrakasan, "Energy scalable system design," *IEEE Trans. on Very Large Scale Integration (VLSI) systems*, vol. 10, no. 2, pp. 135-145, April 2002.
 - [210] A. Chandrakasan, R. Amirtharajah, J. Goodman, and W. Rabiner, "Trends in low power digital signal processing," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS 98)*, Monterey, USA, June 1998, vol. 4, pp. 604-607.
 - [211] S. Lee and T. Sakurai, "Run-power control scheme using software feedback loop for low-power real-time applications," *Proceedings of the Asia-South Pacific Design Automation Conference (ASPDAC)*, Yokohama, Japan, 2000, pp. 381-386.
 - [212] 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*, vol. 1, no. 2, pp. 145 – 163, 2011.
 - [213] P K Gupta and G Singh, "User centric framework of power schemes for minimizing energy consumption by computer systems," *IEEE International Conference on Radar, Communication and Computing, (ICRCC-12)*, India, Dec. 2012.
 - [214] P. K. Gupta and G. Singh, "Energy-sustainable framework and performance analysis of power scheme for operating systems: a tool," *International Journal of Intelligent Systems and Applications*, vol. 5, no. 1, pp. 1 – 15, 2013.
 - [215] M. Webb, "SMART 2020: Enabling the low carbon economy in the information age," *The Climate Group, London*, vol. 1, no. 1 pp.1-1, 2008.

- [216] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, "Temperature-aware microarchitecture: modeling and implementation," *ACM Trans. on Architecture and Code Optimization*, vol. 1, no. 1, pp. 94-125, 2004.
- [217] N. Bansal, T. Kimbrel, and K. Pruhs, "Dynamic speed scaling to manage energy and temperature," *Proceedings of 45th Annual IEEE Symposium on Foundations of Computer Science*, Italy, Oct. 17-19, 2004, pp. 520-529.
- [218] H. L. Chan, W. T. Chan, T. W. Lam, L. K. Lee, K. S. Mak, and P. WH Wong, "Energy efficient online deadline scheduling," *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 795-804.
- [219] W. C. Kwon and T. Kim, "Optimal voltage allocation techniques for dynamically variable voltage processors," *ACM Trans. on Embedded Computing Systems (TECS)*, vol. 4, no. 1, pp. 211-230, 2005.
- [220] M. Li, A. C. Yao, and F. F. Yao, "Discrete and continuous min-energy schedules for variable voltage processors," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 11, pp. 3983-3987, 2006.
- [221] M. Li and F. F. Yao, "An efficient algorithm for computing optimal discrete voltage schedules," *SIAM Journal on Computing*, vol. 35, no. 3, pp. 658-671, 2005.
- [222] H. Aydin, R. Melhem, D. Mossé, and P. M. Alvarez, "Power-aware scheduling for periodic real-time tasks," *IEEE Trans. on Computers*, vol. 53, no. 5, pp. 584-600, 2004.
- [223] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power optimization of variable-voltage core-based systems," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 12, pp. 1702-1714, 1999.
- [224] C. Im, S. Ha and H. Kim, "Dynamic voltage scheduling with buffers in low-power multimedia applications," *ACM Trans. on Embedded Computing Systems (TECS)*, vol. 3, no. 4, pp. 686-705, 2004.
- [225] C. M. Krishna, and Y. H. Lee, "Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems," *IEEE Trans. on Computers*, pp. 1586-1593, 2003.

- [226] R. N. Mahapatra and W. Zhao, "An energy-efficient slack distribution technique for multimode distributed real-time embedded systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 16, no. 7, pp. 650-662, 2005.
- [227] G. Quan, and X. S. Hu, "Energy efficient DVS schedule for fixed-priority real-time systems," *ACM Trans. on Embedded Computing Systems (TECS)*, vol. 6, no. 4, pp. 29, 2007.
- [228] D. Shin and J. Kim, "Power-aware scheduling of conditional task graphs in real-time multiprocessor systems," *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED'03)*, Seoul, Aug. 25-27, 2003, pp. 408-413.
- [229] P. Yang, C. Wong, P. Marchal, F. Catthoor, D. Desmet, D. Verkest, and R. Lauwereins, "Energy-aware runtime scheduling for embedded-multiprocessor SOCs," *IEEE Design and Test*, vol. 18, no. 5, pp. 46-58, 2001.
- [230] X. Zhong and C. Z. Xu, "Energy-aware modeling and scheduling for dynamic voltage scaling with statistical real-time guarantee," *IEEE Trans. on Computers*, vol. 56, no. 3, pp. 358-372, 2007.
- [231] D. Zhu, D. Mosse, and R. Melhem, "Power-aware scheduling for AND/OR graphs in real-time systems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 15, no. 9 pp. 849-864, 2004.
- [232] J. Zhuo, and C. Chakrabarti, "Energy-efficient dynamic task scheduling algorithms for DVS systems," *ACM Trans. on Embedded Computing Systems (TECS)*, vol. 7, no. 2, pp. 17, 2008.
- [233] J. A. Barnett, "Dynamic task-level voltage scheduling optimizations," *IEEE Trans. on Computers*, vol. 54, no. 5, pp. 508-520, 2005.
- [234] D. P. Bunde, "Power-aware scheduling for make span and flow," *Proceedings of the 18th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, NY, USA, 2006, pp. 190-196.
- [235] S. Cho, and R. G. Melhem, "On the interplay of parallelization, program performance, and energy consumption," *IEEE Trans. on Parallel and Distributed Systems*, vol. 21, no. 3 pp. 342-353, 2010.
- [236] Y. C. Lee and A. Y. Zomaya, "Energy conscious scheduling for distributed computing systems under different operating conditions," *IEEE Trans. on Parallel and Distributed Systems*, vol. 22, no. 8 pp. 1374-1381, 2011.

- [237] Dibakar Das and Debabrata Das, “Back-off algorithm based power saving mechanism in a multi-rate UE” Proceedings of IEEE 5th Conference on Internet Multimedia Systems Architecture and Application (IMSAA), Bangalore, India, 2011, pp. 1-6.
- [238] David Meisner, Brian T. Gold and Thomas F. Wenisch, “PowerNap: eliminating server idle power”, Proceedings of the 14th ACM international Conference on Architectural Support for Programming Languages and Operating Systems, Hamilton, Washington, DC, 2009, pp. 1 – 12.
- [239] Keng-Mao Cho, Chun-Hung Liang, Jun-Ying Huang, Chu-Shing Yang, “Design and implementation of a general purpose power - saving scheduling algorithm for embedded systems”, Proc. IEEE Conference on Signal Processing, Communications and Computing (ICSPCC), Xi'an, 2011, pp. 1 – 6.
- [240] Jong-Phil Kim, Doo-Hwan Kim, Jang-Eui Hong, “Estimating power consumption of mobile embedded software based on behavioral model”, Proceedings of International Conference on Consumer Electronics (ICCE), Las Vegas, NV, 2010, pp. 105 – 106.
- [241] Shan Li, Edmund M-K. Lai, Mohammed Javed Absar, “Minimizing embedded software power consumption through reduction of data memory access”, Proceedings of the 2003 Joint Conference of the 4th International Conference on Information, Communications and Signal Processing and 4th Pacific Rim Conference on Multimedia (ICICS-PCM 2003), Singapore, Dec. 15-18, 2003, pp. 309 – 313.
- [242] Tiefei Zhang, Ying-Jheng Chen, Che-Wei Chang, Chuan-Yue Yang, Tei-Wei Kuo and Tianzhou Chen, “Power management strategies in data transmission”, Proceedings of 16th IEEE Asia and South Pacific Design Automation Conference, Yokohama, Jan. 2011, pp. 668 – 675.
- [243] Keqin Li, “Performance optimization with energy constraint in heterogeneous multiple computer system”, Proc. IEEE International Parallel & Distributed Processing Symposium, Shanghai, 2011, pp. 1930 – 1939.
- [244] Osman S. Unsal and Israel Koren, “System-level power-aware design techniques in real-time systems”, Proceedings of the IEEE, vol. 91, no. 7, pp. 1053 – 1069, 2003.

- [245] P. K. Gupta and G Singh, "Energy-sustainable snapshot algorithm for operating systems to minimize power consumption," Elsevier journal of Sustainable Computing: Informatics and Systems, 2012. (Under revision)
- [246] <http://www.softwareok.com/?seite=Microsoft/StressMyPC>
- [247] <http://netbeans.org/community/news/show/1556.html>
- [248] J. Connery, J. M. Enery, D. Hickey, and M. Boubekur, "Profiling real-time Java applications," Proc. IEEE International Conference on Computer Engineering & Systems (ICCES'07), Egypt, Nov. 27-29, 2007, pp. 319-324.
- [249] S. Tallam and R. Gupta. "Profile-guided Java program partitioning for power aware computing," Proceedings of 18th IEEE International Symposium on Parallel and Distributed Processing, Santa Fe, New Mexico, 2004, pp. 156-164.
- [250] Peng Hao-Lin, Liu Yi-Min, You Xiang-Bai, "Research on memory leakage in Java application," Proceedings of IEEE International Conference on Computer Science and Information Technology, Wuhan, China, 2010, vol. 2, pp. 146-148.
- [251] 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, vol. 1, no. 1, pp. 27-55, 2002.