Adaptive Approach in Handling Human Inactivity in Computer Power Management

Ria Candrawati, Nor Laily Hashim
School of Computing, Universiti Utara Malaysia, 06010 UUM Sintok, Kedah Darul Aman, Malaysia.
riacandrawati@yahoo.com

***Abstract*—Human inactivity is handled by adapting the behavioral changes of the users. Human inactivity refers to as unpredictable workload of a complex system that is caused by increments of amount in power consumption and it can be handled automatically without the need to set a fixed time for changing the computer state. This is happens due to lack of knowledge in a software system and the software self-adaptation is one approach in dealing with this source of uncertainty. This paper observes human inactivity and Power management policy through the application of reinforcement learning approach in the computer usage and finds that computer power usage can be reduced if the idle period can be intelligently sensed from the user activities. This study introduces Control, Learn and Knowledge model that adapts the Monitor, Analyze, Planning, Execute control loop integrates with Q Learning algorithm to learn human inactivity period to minimize the computer power consumption. An experiment to evaluate this model was conducted using three case studies with same activities. The result show that the proposed model obtained those 5 out of 12 activities shows the power decreasing compared to others.**

***Index Terms*—Reinforcement Learning; Dynamic Power Management; Human Inactivity.**

I. INTRODUCTION

Studies to reduce computer power usage have been actively conducted. Efficiency of computer power management has been the goal in different approaches on various studies. This is because users tend to leave computer system running for a long time, and these activities lead to waste of power consumption. Several methods for optimizing power consumption for computers have been proposed. These techniques can be categorized into different methods; hardware, software, minimizing the CPU usage, and also by the implementation of policy based approach [1]. In addition, computer manufacturers also have equipped computers with some power saving plan, but unfortunately some users often turn off this function because they feel that this function disrupts their activities [1]. This fact is also supported by [2] that stated, the power saving plan has also resulted in additional power usage when power state changes occur too frequent.

However, there is problem that previous approaches has yet to be resolved in handling uncertainty changes in power states caused by human inactivity and due to unpredictable workload of a system that affects an additional power usage [3], [2]. Human inactivity refers to uncontrollable or unpredictable human behaviors that turn into creation of uncertainty in the software system [4]. Therefore, an improved approach that has an ability to observe, learn and adapt computer power management environment to solve the uncertainty of changes in power state is needed [5]. In addition to that, this approach must also capable of reducing additional power usage when the power management shutdowns a computer component as soon as an idle state is detected to an active state during a short period of time.

Here, [3] raises a direction for future research by providing some intelligent methods to detect human inactivity and unnecessary power consumption. This future method will reduce the power consumption automatically without the need to set a fixed time for changing the computer state [6]. Based on the direction proposed by this paper proposed a new model of self-adaptation into computer power management system which incorporates reinforcement learning policy that learns a new power control policy dynamically at runtime from the information it collects. This model consists of three components; which are Control (Control Loop), Learn (Reinforcement Learning) and Knowledge (Knowledge approach in Database) known as Control Learn Knowledge (CLK) Model. This paper extend the work of [6], by presenting the implementation of CLK components, and presented an evaluated of CLK model against idle time that are manually and factory set.

The rest of this paper is structured as follows. Section 2 contains the background of this study. Section 3 describes the implementation of CLK Model. Section 4 provides the findings on this CLK Model evaluation phase and Section 5 concludes this paper.

II. MODEL DESCRIPTION

Human inactivity is one of uncertainty that is handled in the self-adaptability area. It happens due to lack of knowledge, whether it is possible because of the complexity of the models or loose coupling, ambiguity or distribution [4].

There are different areas of study that contribute in handling uncertainty of human inactivity. In verification and validation area, in order to overcome human-based model verification problem, [7] integrates the representation of human choices within the process models that is designed by proposing modelling approach state chart-embedded assertions within an executable model. This model facilitates runtime verification of models that integrates the representation of human choices. In an automated analysis area, [8] proposed Amplified

Reasoning Technique (ART)-based approach that puts human and machine in an interactive loop. For each of iteration in this approach, it involves human intelligence in guiding a tool to generate refined evidences that bring closer to the final conclusion.

In computer energy saving, [9] studies human behavior regarding to the distractions in computer usage and develop a human-in-the-loop control that can put workstations into sleep by early detection of distraction. However, this approach is different from CLK model which focus is in handling the human inactivity using self-adaptive approach that applies control feedback loops [10] and reinforcement learning [5]. This model adapts behavioral changes of the human.

Self-adaptation is a system that allows modifying behaviors and/or structures in response to perceptions of the environment and the system itself, and their goals [11]. The generic mechanism for this self-adaptation system is provided by feedback loops [12]. In 2001, [10] made a breakthrough by introducing an autonomic computing initiative that building an autonomic system using Monitor Analysis Planning Execute Knowledge (MAPE-K) feedback loops inspired by 'self-*' capabilities and self-adaptability as part of it. CLK model adapts the MAPE-K control loops and uses Q-Learning algorithm [6] to produce a solution to minimize the computer power consumption intelligently.

III. Computer Power Management

This section provides a review on the trends of power management approaches for computers. In general, power management techniques are divided into two types of techniques: static and dynamic. Static techniques is also called as Static Power Management (SPM) techniques are used at design time (off-line) and are targeting different levels of hardware and software. In contrast, dynamic techniques also called as Dynamic Power Management (DPM) techniques are applied at runtime and these techniques are used to reduce power when systems are serving light workloads or idle [13].

Moreover, SPM techniques are divided into two target areas; first approach is targeting the CPU and investigating power consumption of the cycles and instruction levels; second approach is a high-level approach targeting different or all system components.

However, DPM techniques are applied at three different areas; first area is applied at the CPU level, using DVS (Dynamic Voltage Scaling) which allows a processor to dynamically change speed and voltage at run time, and it saves the power by spreading run cycles into idle time. Second area is targeting the system level which considers all of the system components such as memory, hard disk, input and output devices, display, etc. The last area refers to techniques that are used on multiple systems like a server cluster, where more than one system collaborates to save overall power [13]. In addition, DPM techniques provide effective techniques for power reduction at system level because they selectively shut down or slow down system components that are idle or underutilized [5].

DPM techniques can be classified into four approaches: timeout, predictive, stochastic, machine learning. Firstly, the timeout approach is a policy that switches into low power state after a device has been idle for a certain time. This policy can be used as a static timeout or adaptive timeout. To differentiate between these two approaches, the static timeout uses a fixed timeout, and the adaptive timeout will adjust the timeout period according to the last idle period [14]. However, this policy tends to waste a substantial amount of power when waiting for a timeout to expire[15].

Secondly, the predictive approach is a policy that predicts the length of idle periods and turns the device into low power mode when the predicted idle period is longer than a certain threshold [15]. However, this policy works at idle state and will wake up the device as soon as only a request arrives, by other means this policy cannot deal when the device is already entering the off or sleep state with multiple incoming request queue [16].

Thirdly, the stochastic approach is a policy that addresses the predictive problem. This policy uses probabilistic assumption about the device usage patterns and formulates optimization problem which is finding policies that optimally tradeoff performance for power that derives an optimal DPM strategy [17]. However, by solving the stochastic optimization problem to find the optimal DPM strategy either in a time-driven or event-driven manner increases the complexity of nonstationary request patterns [18].

Fourth, the machine learning (ML) policy is a current policy that applies ML to learn a request arrival pattern for DPM such as by applying genetic policy, Bayesian classifiers and recently Reinforcement Learning (RL) approaches [17]. ML is a study of methods for constructing and improving software systems by analyzing examples of their behaviors rather than by directly programming them [17]. RL is a model-free approaches, it works by interacting with the system, implementing certain actions, evaluating the effects and adjusting the action while running [19].

[20] proposed a DPM approach with RL. This approach sets several DPM policies to control power management under certain workloads. This study achieves the optimal DPM policy but it is limited to a chosen policy. This study has been extended [19] where they proposed a model-free approach that does not require selected experts or policies but yet the size of proposed approach used is quite large and return the complexity of the result. Recently approach by [21] has proposed a model-free RL based DPM policy by estimating workload information to be applied by the learning policy. The result shows this approach adapts the given power and performance constrains, and has been applied in traffic surveillance system.

To conclude, from this section highlight that the RL gives more flexibility and advantages related to reducing power usage of a device. The RF policy deals with the overhead state space result and also convergence (speed) of the policy [21]. Besides the limitation, this policy also has several advantages such as the power-performance trade-off can be flexibly controlled; non-stationary workloads can be efficiently dealt with; online learning and policy implementation can take place in parallel; easy to implement, and computationally efficient [22].

IV. CLK MODEL IMPLEMENTATION

This study defines the CLK model that has smart control to sense the human inactivity. CLK model adapts the self-adaptive feedback control loop to reduce the human inactivity period, in order to minimize the computer power consumption. The model consists of Hardware, Application and the Control Loop, as shown in Figure 1.

Hardware consists of an existing hardware in a desktop computer. A comparative study has been conducted in order to compare and acknowledge the monitored PC components, where 6 components have been selected; Processor, Graphic Card, Memory, HDD, Monitor and Power Supply. These components are selected according to [23], which revealed that the energy consumption of an average PC generated by these component. [24] Also agree that these PC components are energy consuming. Therefore, in this study, these components are monitored in the CLK model of autonomic power management software for computer.

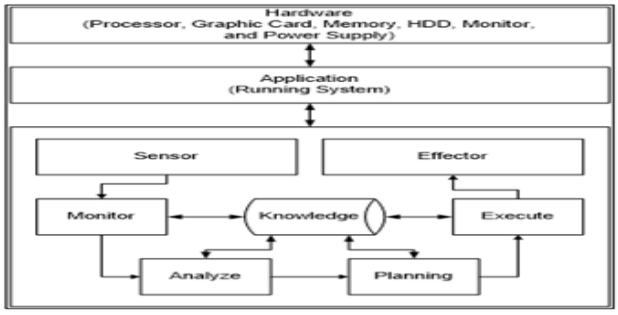

Figure 1: Model for Control Learning, Knowledge Model of Desktop Computer [6].

Applications are running through user activities. This application consists of software to support user such as browser, word application and etc. This application becomes part of the model because CLK model will capture every change in computer power mode. The power modes captured are active and idle.

The control loop consists of 7 (seven) components: Sensor, Effector, Monitor, Analyze, Planning, Execute and Knowledge.

The Sensor captures current profiles and states of hardware computer. WMI Class script is used to collect the information regarding hardware profile. The hardware component covers desktop monitor, power supply, processor, memory and graphic card. An example of the WMI script used is presented in Figure 2.

```
Public Class MyWMIQuery
  Public Overloads Shared Function Main() As Integer
    Try
      Dim searcher As New ManagementObjectSearcher("root\CIMV2", "SELECT * FROM Win32_DesktopMonitor")
      Dim searcher1 As New ManagementObjectSearcher("root\CIMV2", "SELECT * FROM Win32_Processor")
      Dim searcher2 As New ManagementObjectSearcher("root\CIMV2", "SELECT * FROM Win32_ComputerSystem")
      Dim searcher3 As New ManagementObjectSearcher("root\CIMV2", "SELECT * FROM Win32_LogicalDisk")
      Dim searcher4 As New ManagementObjectSearcher("root\CIMV2", "SELECT * FROM Win32_VideoController")
    Catch err As ManagementException
      MessageBox.Show("An error occurred while querying for WMI data: " & err.Message)
    End Try
  End Function
End Class
```

Figure 2: Sensor Script

The Effector component responds to the application and hardware, and then counters the effects of the changes by changing the system and maintaining the environment. The action that needs to be taken is whether to turn the state into active or idle. The script is shown in Figure 3.

```
Private Sub userTimer_UserActiveEvent(ByVal sender As Object, ByVal e As EventArgs) Handles userTimer.UserActiveEvent
        Return UserCurrentState()
End Sub
Private Sub userTimer_UserIdleEvent(ByVal sender As Object, ByVal e As EventArgs) Handles userTimer.UserIdleEvent
        Return UserCurrentState()
End Sub
```

Figure 3: Effector Script

Monitor component invokes the GetLastInput() method to detect current user state once the state change and lastInput.dwTime() provided by sensor phase.

```
Private Sub GetLastInput(ByVal  userState As Object)
    Me.idleThrshld = idleThreshold
    Me.lstInput = New LASTINPUTINFO()
    Me.lstInput.cbSize = Clnt(Marshal.SizeOf(Me.lastInput))
    GetLastInputInfo(Me.lastInput)
    Me.lstActivity = Me.lastInput.dwTime
    Me.activityCheckTime = New Timer(AddressOf Me.GetLastInput, Nothing, Timeout.Infinite, 1000)
    Me.lastResetTime = DateTime.Now
    Me.Enabled = active
End Sub
```

Figure 4: Monitor Script

Analyse component is a phase where the model will examine the current state of the active user and also the user state during idle, then data will be sent to the planning phase of the next action. Selection is also based MaxQ value data for each state involved.

The planning component performs the Q-Learning algorithm using the prior knowledge stored to decide what the optimal stage should maintain. Below is the flow chart for adapting algorithm.

```
Start
Initialize α, γ
Repeat
        Identify current state
        Choose action from state using policy e-greedy (select max Q(s,a))
        Take action, set next idle time (difference time between current state and previous state
        using e-greedy)
        Update Q value using:
        Q(s,a)_i = (1 − α)Q(s,a)_{i−1} + α[R(s,a)_i + γ_{a'} max Q(s',a')]
Until next state determine
End
```

Figure 5: Flow Chart

The Executing component executes when the planning function is given the task to optimize the power usage when the usage achieve the warning point and also to activate or idle the computer based on the threshold time that will determine based on learning history in the knowledge part.

The Knowledge component adapt knowledge management process, stores the resources details and also the result of all process from loop progress. These also record the parameters from hardware, mode state from application and execution history of loop.

V. FINDINGS

An evaluation process was conducted through an experiment using three case studies: (1). Case 1: uses the factory default idle time, (2). Case 2: uses an idle time of 5 minutes, and (3). Case 3: uses CLK model that is based on adaptive idle time. The activities used in each case study are adapted from [25], where they have suggested 11 usages scenario. However, in this study only 8 usage scenarios were implemented. In addition to that, 2 additional idle activities were added, where for each activities, 5 minutes duration were set. The activities are web navigation, e-mail, productivity suite, data transfer (disk), data transfer (USB), presentation, multimedia (audio) and multimedia (video). These scenarios were executed and to collect power consumption using HWMonitor Software [26].

Our experimental result shows that 5 out of the 10 activities reveal in power saving using CLK model, when compared to Case 1 and Case 2. The activities are: web navigation, productivity suite, 2nd idle time, presentation, and multimedia (video). This is because the value of the time idle in CLK is set based on the learning process from the users' activities. Q-learning as a controller in this model learns the last state changes in knowledge and run the formula to choose the data with same state, then find differences time to set as next idle time for the user. The model handles the power consumption caused by the workload of uncertainty that leads to the use of additional power when switching from idle state to an active state within a short period. As a conclusion, Case 1 occurred 4 times minimum time in 10 activities, meanwhile in Case 2, only 1 activity, in Case 3, 5 times minimum time occurred, this prove highest percentages in Case 3. The results shows a graph which is derived from the data collected from HWMonitor Software based on the user activities as shown in Figure 6.

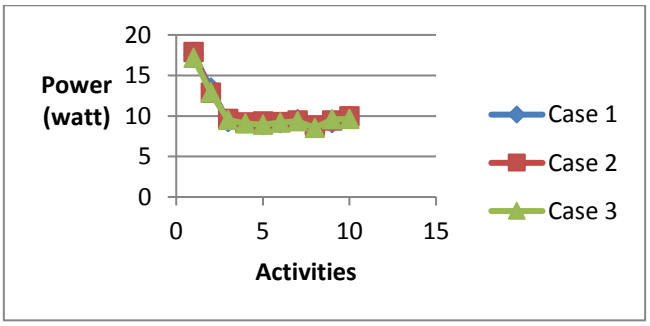

Figure 6: Case studies result

VI. CONCLUSION

As a conclusion, this study observed the eligibility of human inactivity period issues from computer power consumption point of view and the policy of dynamic power management used in the computer power management. This study found the need to improve it by applying the learning element which is flexible to sense the human inactivity while using the computer. This study applies self-adaptive concept by conducting the feedback control loop. It also adapts the Q learning algorithm that is flexible to sense the human inactivity period while users are using the computer. This paper has presented the results of power saving obtained from an experiment conducted to compare CLK model against factory default idle time and manually set idle time. The result shows that 5 out of 12 activities, shows the power decreasing compared to others.

ACKNOWLEDGMENT

The authors would like to thank University Utara Malaysia for providing the research funds that allows this study possible.

REFERENCES

[1] Gupta, P. K. and Singh, G., "Minimizing Power Consumption by Personal Computers: A Technical Survey," *Int. J. Inf. Technol. Comput. Sci.* 4(10):57–66, 2012.

[2] Irani, S. Shukla, S. and Gupta, R., "Online strategies for dynamic power management in systems with multiple power-saving states," *ACM Trans. Embed. Comput. Syst.* 2(3):325–346, 2003.

[3] Sandhu, S. Rawal, A. Kaur, P. and Gupta, N., "Major components associated with green networking in information communication technology systems," *Commun. Appl. (ICCCA), 2012 Int. Conf. Comput.*1–6, 2012.

[4] Esfahani, N. and Malek, S. Uncertainty in self-adaptive software systems, *Softw. Eng. Self-Adaptive Syst. II*, 2013.

[5] Shen, H. Tan, Y. Lu, J. Wu, Q.and Qiu, Q., "Achieving autonomous power management using reinforcement learning," *ACM Trans. Des. Autom. Electron. Syst.*18(2):1–32, 2013.

[6] Candrawati, R. Hashim, N. L. Mahmuddin, M. and Irwan, H., "A Model of Framework of Control , Learn , and Knowledge for Computer Power Management" In Proceeding of *Knowledge Management International Conference*, 2014.

[7] Schumann, M. a. Drusinsky, D. Michael, J. B.and Wijesekera, D., "Modeling human-in-the-loop security analysis and decision-making processes," *IEEE Trans. Softw. Eng.* 40(2):154–166, 2014.

[8] Kothari, S. Deepak, A. Tamrawi, A. Holland, B. and Krishnan, S., "A ' Human-in-the-loop ' Approach for Resolving Complex Software Anomalies," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2014,1971–1978, 2014.

[9] Munir, S. Stankovic, J. Liang C., and Lin, S. Reducing Energy Waste for Computers by Human-in-the-Loop Control, 2013.

[10] IBM. An architectural blueprint for autonomic computing, 2005.

[11] Lemos, R. De Giese, H. and Müller, H., "Software engineering for self-adaptive systems: A second research roadmap," *Softw. Eng.*1–32, 2013.

[12] Brun, Y. Di, G. Serugendo, M. Gacek, C. Giese, H.and Shaw, M. Engineering Self-Adaptive Systems through Feedback Loops.. 48–70, 2009.

[13] Chedid W. and Yu, C. Survey on power management techniques for energy efficient computer systems, 2002.

[14] Lu Y.-H. and Micheli, G. De, "Comparing system level power management policies," *IEEE Des. Test Comput.* 18(2):10–19, 2001.[15] Srivastava, M. B. Chandrakasan, A. P. and Brodersen, R. W., "Predictive System Shutdown and Other Architectural Techniques for Energy Efficient Programmable Computation," *IEEE Trans. VERY LARGE SCALE Integr. Syst.* 4(I):42–55, 1996.

[16] Benini, L. Bogliolo, A. and De Micheli, G., "A survey of design techniques for system-level dynamic power management," *IEEE Trans. Very Large Scale Integr. Syst.*, 8(3):299–316, 2000.

[17] Barto, A. and Dietterich, T. Reinforcement learning and its relationship to supervised learning, *Handb. Learn. Approx. Dyn. Program, 2004*.

[18] Gurumurthi, S. Sivasubramaniam, A. Irwin, M. J. Vijaykrishnan, N. and Kandemir, M. Using complete machine simulation for software power estimation: The softwatt approach, *High-Performance Comput. Arch*, 2002.

[19] Tan, Y. and Qiu, Q., "A Framework of Stochastic Power Management Using Hidden Markov Model," 92–97, 2008.

[20] Dhiman, G. and Rosing, T., "Dynamic Power Management Using Machine Learning," in *2006 IEEE/ACM International Conference on Computer Aided Design*, 2006. 747–754, 2006.

[21] Khan U. A.and Rinner, B., "A Reinforcement Learning Framework for Dynamic Power Management of a Portable, Multi-camera Traffic Monitoring System," *IEEE Int. Conf. Green Comput. Commun.* Nov. 2012. 557–564, 2012.
[22] Khan, U.and Rinner, B., "Online learning of timeout policies for dynamic power management," *ACM Trans. Embed. Comput.* 13(4), 2014.
[23] Higgs, T. Energy Efficient Computing, *IEEE*,210–215, 2007.
[24] Moshnyaga, V. G. How to Really Save Computer Energy?, *Computer (Long. Beach. Calif).*3, 2008.
[25] Moshnyaga, V. G. he use of eye tracking for PC energy management, *Proc. 2010 Symp. Eye-Tracking Res. Appl. - ETRA '10*, 113, 2010.
[26] CPUID, "HWMONITOR-PRO," 2016. [Online]. Available: http://www.cpuid.com/softwares/hwmonitor-pro.html. [Accessed: 31-Dec-2015].