# Developing Green Software

DR. BOB STEIGERWALD AND ABHISHEK AGRAWAL
SOFTWARE & SERVICES GROUP, INTEL CORPORATION, FOLSOM, CA, USA

## Introduction

For years mobile platform vendors have sought means to extend the battery life for mobile platforms. Battery technologies have gradually improved, processors have new low-power states, and displays have dramatically improved their power consumption. There is still room for improvement. Software can play an important role in reducing the power used on mobile platforms and extend the battery life. This paper describes the characteristics of Green Software and the software design considerations and methodologies to improve software energy efficiency.

*Processor P and C States*

As software developers, the component of primary interest is the CPU. Understanding the CPU's defined energy states can help developers make design decisions in favor of energy efficiency. It makes sense that if the CPU is not actively processing information or performing computations, that it should be consuming minimal energy. The CPU has what are called C-states and P-states. C-states are core power states that define the degree to which the processor is "sleeping". In state C0 the processor is active and executing instructions. While in C0, the processor can operate at various frequency levels, designated as P-States.

State C0 corresponds to a state when the CPU is active, i.e. it is busy carrying out some task, and it is performing that task at whatever frequency (P-state) is currently appropriate. Between periods of activity the CPU can take the opportunity to rest or "sleep". In fact, C-States are often referred to as "Sleep" states. Intel® processors support several levels of core and package (resources shared by all the cores) C-states, that provide a flexible selection between power consumption and responsiveness. With each successively deeper sleep state, some new part of the CPU is turned off and more energy is saved. The deeper the sleep, the

greater the energy savings. Even if the sleep period is only 100 microseconds, a considerable amount of energy can be saved over time.

P-states, or performance states, define the frequency at which the processor is running. Different processor brands showcase their P-states as features, such as SpeedStep in Intel processors, PowerNow! or Cool'n'Quiet in AMD processors, and PowerSaver in VIA processors. Typical P-States are:

- P0 - max power and frequency
- P1 - less than P0, voltage/frequency scaled
- Pn - lowest rated voltage/frequency

## Energy Saving Software Techniques

Now that we have taken a quick look at the energy efficiency components and states of the CPU, we should address why a software developer needs to know this. While most of the energy saving features in the platform are transparent to the software developer and applications have very little direct control, the behavior of the software has significant influence on whether the energy saving features built into the platform are effective. Well behaved software allows the energy saving features to work. Poorly behaved software inhibits the energy saving features and leads to lower battery life and higher energy costs.

Before describing the software energy-saving techniques, it is important to understand the distinction between *active* and *idle* software. Active software is software that is fulfilling its intended purpose such as computing a spreadsheet, playing music or a movie, uploading photos to a web site, browsing the internet, etc. In all of these cases, there is a *workload* that the CPU or GPU is busy working on. Idle software is software that is essentially running but waiting for an event at which point it will become active. Examples of idle software are: a browser that is started but has not been pointed to a web site, an open word processor program in the background or whose window is minimized, or an instant messaging program that is running but not sending or receiving a message. In the sections below, Computational Efficiency, Data Efficiency, and Context Awareness primarily apply to active software, whereas Idle Efficiency, as the name implies, covers techniques to develop well behaved software at idle.

### Computational Efficiency

Simply put, computational efficiency means getting the work done quickly. The amount of effort computer scientists have put into software performance not only saves time, it saves energy as well. We call this the *race to idle*. The faster we can complete the workload and get the computer back to idle, the more energy we can save. To achieve computational efficiency, use software techniques that achieve better performance such as efficient algorithms, multi-threading, and vectorization.

*Efficient algorithms* - Algorithms and data structures are a long-standing area of research in computer science. The choice of algorithms and data structures can make a vast difference in the performance of an application. For a particular problem, a stack may be better than a queue and a B-tree may be better than a binary tree or a hash function. The best algorithm or data structure to use depends on many factors and a study of the problem and a careful consideration of the architecture, design, algorithms, and data structures can lead to an application that delivers better performance and lower power.

*Multi-threading* - Multi-threading your software delivers better performance and as well as better energy-efficiency. To illustrate this, we have taken one of our favorite performance benchmarking applications – Cinebench 11.5 from Maxon Computer GmbH [1] – and measured the energy consumed on a 4-core, 8-thread Intel processor. Figure 1 shows the CPU energy use over time using 1 thread, 2 threads, 4 threads, and 8 threads. As you can see, completing the workload with a single thread takes considerably longer and uses

more energy than any of the multi-threaded runs. In fact the 8-thread case (normalized) uses about 25% less energy than the single threaded case. The added advantage is that the 8-thread run was completed in about one-fourth the time and the processor is available for other computation.

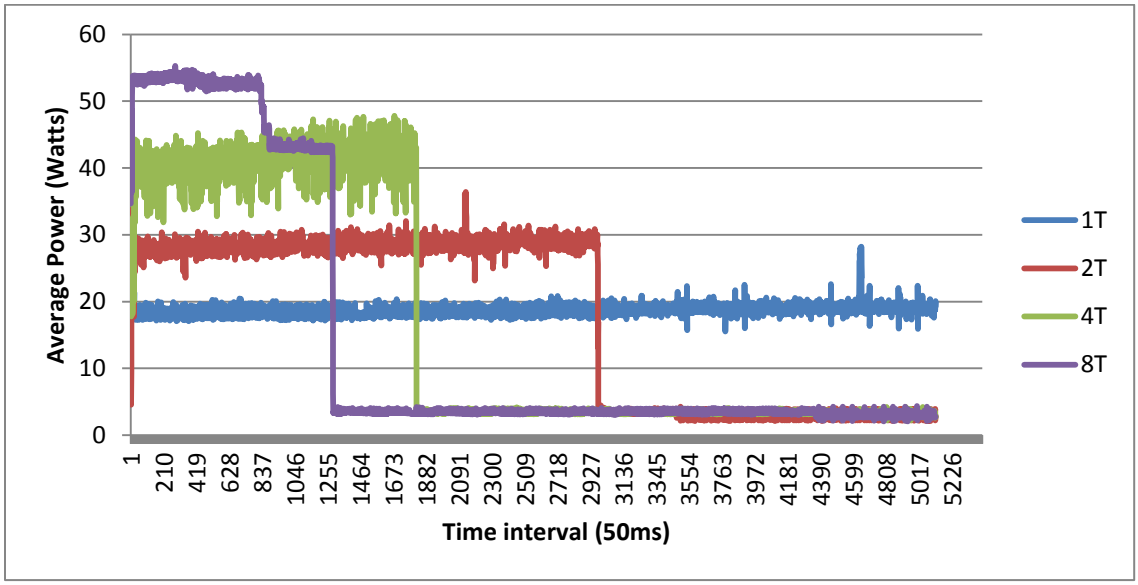



Figure 1: CPU Energy Measurements from various runs of The CINEBENCH 11.5 on an Intel Core i7 System

*Vectorization* - Another method used to achieve better computational efficiency is vectorizing the code instead of using scalar C- code by using advanced instructions such as SIMD (Single-Instruction Multiple Data) for instruction-level data parallelism. If you can *vectorize* your solution, you will get better performance and a corresponding power benefit. To test this, we took 2 different audio decode algorithms and optimized them using Intel® Advance Vector Extensions (Intel® AVX) instruction set. Using a method we have to turn the AVX feature On and Off, we measured both the performance and the average power of an Audio decode workload. When AVX is enabled, the workloads run considerably faster – by 1.65X and 1.34X with a huge savings in power. Table 1 summarizes the performance and power impact of using AVX.

| Audio Decode | Time (s) AVX Off | Time (s) AVX On | Speedup | Avg. Power (W) AVX Off | Avg. Power (W) AVX On | Savings on Avg Power (W) |
|---|---|---|---|---|---|---|
| #1 | 144 | 87 | 1.65X | 12.7 | 8.4 | 4.3 |
| #2 | 203 | 151 | 1.34X | 13.7 | 11.1 | 2.6 |

Table 1: Performance and Power Impact of AVX instructions

### 2.2 Data Efficiency

Data efficiency reduces energy costs by minimizing data movement. As summarized in [2], data efficiency can be achieved by designing:

- software algorithms that minimize data movement
- memory hierarchies that keep data close to processing elements
- application software that efficiently uses cache memories

As we demonstrated with computational efficiency, data efficiency delivers performance benefits and saves energy. The following sections provide examples of where data efficiency methods can be applied to save energy.

*Managing disk I/O* - This section contains brief summaries of our analysis of power characteristics of the hard disk power during various activities as well as guidelines on optimizing the power during disk I/O. For additional details as well as sample code sample code, see [3]. The analyses are based on the typical performance characteristics of hard disk drives (HDD) which is affected by RPM, seek time, rotational latency, and the sustainable transfer rate. Furthermore, the actual throughput of the system will also depend on the physical location of the data on the drive. Results of four separate experiments to better understand the energy usage of hard disk drives using various I/O methodologies are summarized below:

HYPOTHESIS: When reading a large volume of sequential data, reading the data in larger chunks requires lower processor utilization and less energy
RECOMMENDATION: Use block sizes of 8KB or greater for improved performance.

HYPOTHESIS: Effective use of asynchronous I/O with native command queuing (NCQ) improves performance and saves energy.
RECOMMENDATION: Applications that deal with random I/O or I/O operations with multiple files should use asynchronous I/O to take advantage of NCQ. Queue up all the read requests and use events or callbacks to determine if the read requests are complete.

HYPOTHESIS: The performance and energy costs to read a fragmented file are greater than that of a contiguous file.
RECOMMENDATION: Avoid by pre-allocating large sequential files when they are created, e.g. SetLength in .Net* framework. Use NtFsControlFile() to aid in defragmenting files. End users can defragment their volumes periodically

HYPOTHESIS: The performance and energy costs to read a fragmented file are greater than that of a contiguous file.
RECOMMENDATION: For multiple threads competing simultaneously for disk I/O, queue the I/O calls and utilize NCQ. Reordering may help optimize the requests, improve performance, and save energy. When multiple threads competing for the disk causes significant disk thrashing, consolidate all the read/write operations in a single thread to reduce read/write head thrashing and reduce

*Pre-fetching and caching* - We also conducted a study to determine if pre-fetching and caching can save energy during DVD playback. More details of the study can be obtained from [4]. In this study we analyzed the power consumption of three different DVD playback software applications (DVD App #1, #2, and #3) with the multiple out-of-box configurations available while taking power measurements - primarily maximum power-saving mode vs. no power-saving mode. The workload used for the analysis was a standard definition DVD movie that was included with the MobileMark 2005 [5] benchmarking tool. Form our experiments it was clear that DVD drive spin-up is the most power hungry (requiring over 4W for a brief period) and that reducing spin-up can lead to energy savings. Continuous DVD read consumes about 2.5W of power and is another area for potential savings.

Table 2 shows the actual measured energy usage for the three DVD Playback applications. It's valuable to note from this data that difference between the worst case energy consumption (App 2, 10143 mWHrs) and the best case (App 3, 6023 mWHrs) energy consumption is over 4000 mWHrs and appears to be entirely due to the design choices made by the application developers. This is about a 40% energy savings. Even a 10% energy savings on a 4-hour battery would provide 24 minutes more battery life on a standard notebook PC.

| Application | Mode | DVD Energy (mWHrs) | CPU Energy (mWHrs) | Platform Energy (mWHrs) |
|---|---|---|---|---|
| DVD App 1 | No Save | 869.84 | 663.92 | 6618.76 |
| | Max Save | 263.41 | 762.99 | 6039.41 |
| DVD App 2 | No Save | 897.82 | 3329.53 | 10143.56 |
| | Max Save | 895.57 | 1064.02 | 7509.18 |
| DVD App 3 | No Save | 780.93 | 703.25 | 6202.04 |
| | Max Save | 781.04 | 554.87 | 6023.57 |

**Table 2**: Energy Consumed during DVD Playback

From the results of the studies performed, three guidelines emerge that can help save energy during DVD playback:

- **Buffering**: The studies shown above indicate that the technique of buffering implemented by DVD App #1 reduces DVD power consumption by 70% and overall platform power consumption by about 10%, as compared to other techniques.
- **Minimize DVD drive use**: Reduce DVD spin-up, spin-downs, and read accesses in order to save energy.
- **Let the OS manage the CPU frequency**: We do not recommend changing the CPU power scheme to run the processor at the highest available frequency. Allow the OS to set the appropriate P-state, adjusting the CPU frequency as needed.

### Context Awareness

Humans naturally use context to understand our world, makes decisions, and adapt to the environment. Context awareness in computers means that they can sense the environment in which they are operating and software can be designed to react to changes in the environment. Context awareness was first introduced by Schilit in 1994 [6]. The objective is to create applications that can respond or adapt to changes in the environment. For the physical environment this requires sensors and the ability to generate events or state changes to which the applications can react. Some examples of context aware behavior are when a notebook PC responds to a change from AC to DC power by automatically dimming the display or when a notebook PC quickly parks the hard drive heads when sensors detect that the device is falling – to avoid a head crash. Embedded systems are particularly context aware because in many cases, they are designed specifically to monitor environmental conditions from sensor data and react to them. The use of sensors is growing rapidly in smartphones and tablets and includes light sensors, gyros, accelerometers, GPS receivers, near-field communications, and others.

Context-awareness makes our devices "smarter" and the behavior of applications can be passive or active. A passive response to a change in context would be to ask the user what action to take ("Switch to power-save mode?") or to acknowledge that the state change has occurred ("You have 10% battery left. OK?"). An active response would be to take action automatically either as a built-in feature (dim the display in a dark room) or as a user configurable option (skip full-system virus scan when on battery). We believe software applications can take advantage of context-awareness to save energy. We'll look at two examples here and suggest some others for further research.

*AC or DC?* - Does it benefit an application to know if a notebook PC is plugged into an AC power source or operating on battery? In many cases, the answer is yes. In Windows, you can achieve this by querying a unique GUID called GUID_ACDC_POWER_SOURCE [7]. Armed with this information, you can adapt the application's behavior and possibly deliver extended battery life for the current usage.

*Platform Power Policies* - Microsoft Windows provides built-in power policies – "High performance", "Balanced", and "Power saver". They give the system user the option to choose between better performance and better

battery life. Application software can use power polices in the following ways: Adjust application behavior based on the user's current power policy, change application behavior in response to a change in power policy, or change the power policy to suit the application behavior Once again, Windows system GUIDs are available to implement these behaviors. For more information on how to use this API, please see [8].

*Additional Context-Aware Behaviors* - Developers may want to consider the status of other components on the platform and use that information for intelligent application behavior and energy savings. It might be useful to know information about the status of components such as network cards, Bluetooth, Wi-Fi, USB devices, monitors, etc. For example, should an application continue to render video if the monitor has turned off due to a timeout or if the application window is minimized? The Microsoft System Event Notification Service (SENS) can help alleviate t mobile application issues. The SENS API, distributed with the Intel Mobile Platform SDK, provides a simple function call for checking if the network connection is alive and another that will ping a specified address for you. In addition to these simple functions, you can also register with the service to receive events when a connection is made or lost, to ping a destination, or even as an alternative method to detect when the system changes power states (battery on, AC ON, or battery low).

### Idle Efficiency

*Idle Power* for mobile platforms is defined as the power consumed when the system is running in ACPI S0 state (S3-Sleep or S4-Hibernate) with software applications & services running but not actively executing workloads. In this state, there should be minimal background activity. Figure 2 below (see also [9]) shows energy measurements taken while a computer was running Mobile Mark 2007 [5]. The *idle floor* sits at around 8 Watts, about as low as you can go in S0 on this particular platform. If you remember from section 1, C0 is the state where the processor is considered fully active. By examining the C-state data from this run, we determined that even with an active benchmark like MM07, the platform was in C0 only 5-10% of the time. In other words, the processor was in an *idle* state, something lower than C0, 90-95% of the time. The challenge is to lower the idle floor by improving application *idle efficiency* which will lead to a significant increase in battery life. This also benefits average power scenarios and helps all but the most demanding (TDP-like) workloads.

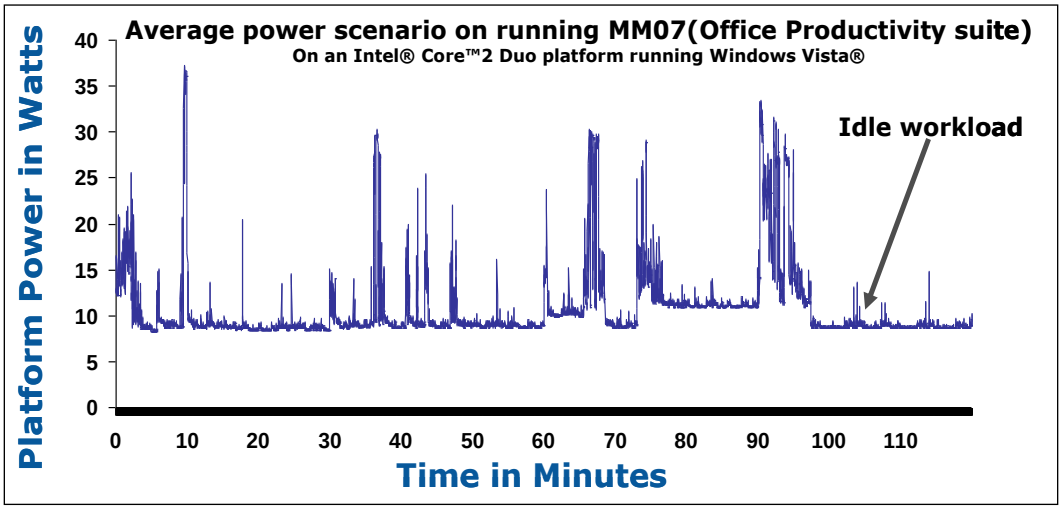



Figure 2: Platform Power Profile While Running MobileMark 2007

### Deep C-State Residency

One of the key requirements for idle efficiency is to keep the platform in deeper C-states for as long a duration as possible. For a platform in idle state, ideally the residency in the deepest C-state (C6/C7) should be more than 90%. Software should aim to keep the number of C-state transitions as low as possible. Frequent C-state transitions from idle to active are not energy efficient. Activity should be coalesced whenever possible to allow for higher C-state residencies. This sort of frequent C-state transition impacts power consumption in two ways:

- The energy requirements to enter/exit C-state are non-trivial. When the C0 (active) duration is very small, the latency to transition in and out of the C-states in comparison is appreciable and may result in net energy loss.

- The hardware policy may demote the C-state to a lower state based on heuristics. Even if the frequent C-state transition behavior occurs only for 2-3msec in a 15.6msec window, hardware polices may either demote core C-state or re-open package level cache and this will impact power for the remaining ~12-13msec of the idle period.

To reduce C-state transitions in applications and services, we recommend that you not split a task between processes/threads unless parallel execution can occur. If it is necessary that a task be split between processes, then schedule the work so that the number of C-state transitions can be reduced. Also, applications & services should coalesce activity whenever possible to increase idle period residency.

### *OS Timer Resolution*

The default system-wide timer resolution in Windows* is 15.6 ms, which means that every 15.6 ms the operating system receives a clock interrupt from the system timer hardware. When the clock interrupt fires, Windows performs two main actions: it updates the timer tick count if a full tick has elapsed, and it checks whether a scheduled timer object has expired. A timer tick is an abstract notion of elapsed time that Windows uses to consistently track the time of day and thread quantum times. By default, the clock interrupt and timer tick are the same, but Windows or an application can change the clock interrupt period.

Many applications call ***timeBeginPeriod*** with a value of 1 to increase the timer resolution to the maximum of 1 ms to support graphical animations, audio playback, or video playback. This not only increases the timer resolution for the application to 1 ms, but also affects the global system timer resolution, because Windows uses at least the highest resolution (that is, the lowest interval) that any application requests. Therefore, if only one application requests a timer resolution of 1 ms, the system timer sets the interval (also called the "system timer tick") to at least 1 ms. Generally, setting the timer to a value less than 10 ms can negatively affect battery life. Modern processors and chipsets, particularly in portable platforms, use the idle time between system timer intervals to reduce system power consumption. Various processor and chipset components are placed into low-power idle states between timer intervals. However, these low-power idle states are often ineffective at lowering system power consumption when the system timer interval is less than 10 ms. Figure 3 below shows the impact of increasing periodic timer resolution on platform power

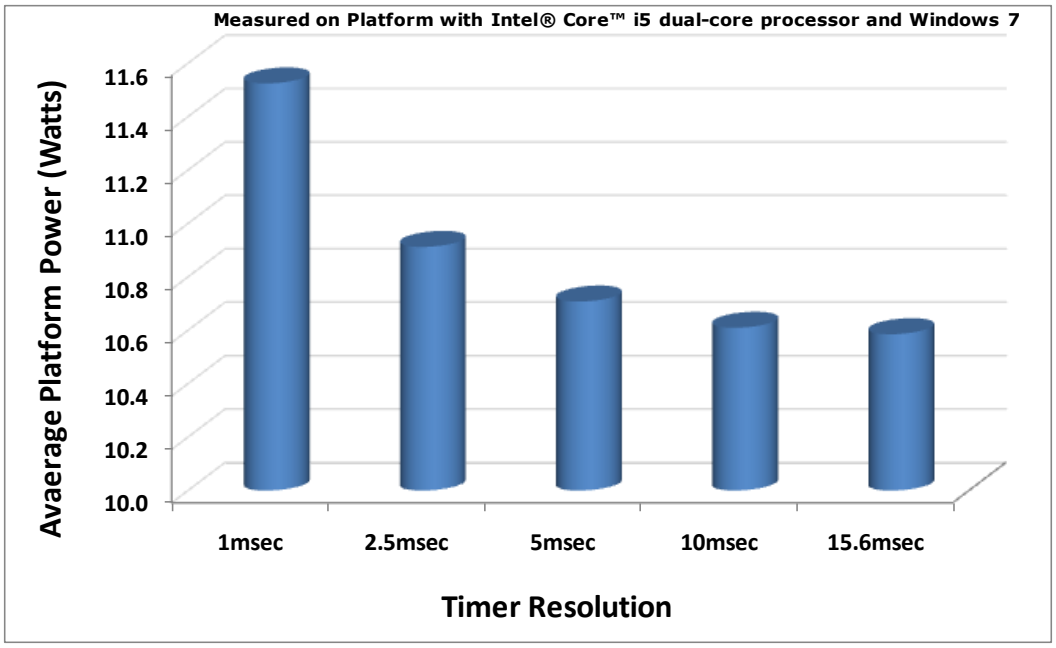



**Figure 3:** Power Impact of increasing Periodic Timer resolution

Both Windows Vista* and Windows 7* come with a command-line utility called *PowerCfg.* Using the */energy* option, powercfg can be used to determine whether an application has increased the platform timer resolution. Run the PowerCfg utility when the application is running and examine the resulting energy report to see if the application changed the platform timer resolution. Powercfg will also show the entire call stack for the request. The report lists all of the instances of increased platform timer resolution and indicates whether the process hosting the application increased the timer resolution. More details on this can be found in the tools chapter.

Other recommendations are:

- If your application must use a high-resolution periodic timer, enable the periodic timer only while the required functionality is active. For example, if the high-resolution periodic timer is required for animation, disable the periodic timer when the animation is complete.

- If your application must use a high-resolution periodic timer, consider disabling use of the periodic timer and associated functionality when a Power Saver power plan is active or when the system is running on battery power.

## Background Activity

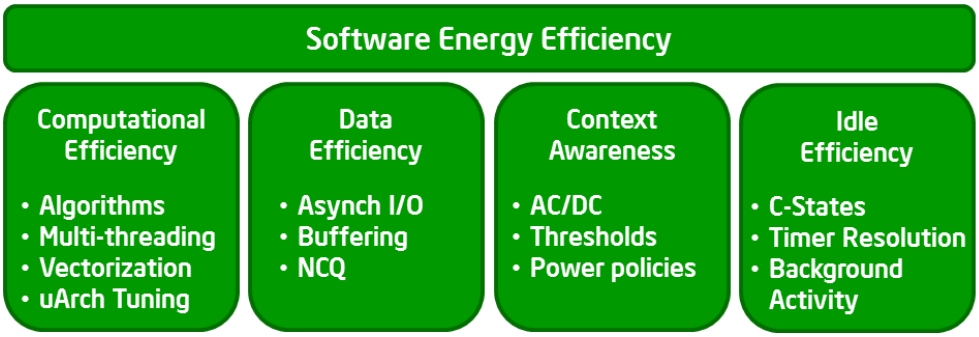
Frequent periodic background activity increases overall system power consumption. It impacts both the processor and chipset power. Long running infrequent events also prevent the system from idling to sleep. Background activity on the macro scale (minutes, hours) such as disk defragmentation, antivirus scans, etc. are also important for power. Win7 has introduced a unified background process manager (UBPM) to minimize the power impact from background activities. The UBPM drives scheduling of services and scheduled tasks and is transparent to users, IT pros, and existing APIs. It enables trigger start services. For instance many background services are configured to autostart and wait for rare events. With UBPM, it enables the trigger-

start services based on environmental changes. Some of the environmental change activities include device arrival/removal, IP address change, domain join, etc. An example of trigger start service is starting Bluetooth services only when the Bluetooth radio is currently attached. Some of the other Win7 improvements to minimize frequent idle activity are:

- Elimination of TCP DPC timer on every system timer interrupt
- Reduction in frequency of USB driver maintenance timers
- Intelligent timer tick distribution
- Timer coalescing

## Summary

The world is moving toward Green technologies and consumer demand for longer battery life in mobile devices is always increasing. The demand for higher performance and new usage models will also continue to grow. Energy-efficiency will be crucial for the computing industry in the future both to increase battery life for mobile platforms and to reduce energy expenses for desktop and server platforms. Software behavior can have a significant effect on platform power consumption and battery life.




Modern processors and platforms have many energy saving features, particularly for performance and the ability to enter low-power states when idle. Software should work in harmony with these features. To get the most benefit, developers should do the following:

- Take advantage of performance features by emphasizing Computational Efficiency.
- Be frugal with data movement to improve Data Efficiency.
- Implement intelligent application behaviors by exploiting Context Awareness.
- Seriously consider the impact of software at idle to improve Idle Efficiency.

There are many free tools from Intel and others to help you get started. Even small improvements when amplified across millions of systems can make a dramatic difference.

## References


1. Maxon GmbH, CINEBENCH 11.5, run on Windows 7 and an Intel Core i7 processor-based software development platform. CPU energy measurements obtained using a utility to capture CPU counter data. This test was performed without independent verification by Maxon GmbH and the Maxon GmbH makes no representation or warranties as to the result of the test. The CINEBENCH is Copyright ® 2010 MAXON Computer GmbH, All Rights Reserved. http://software.intel.com/sites/products/collateral/hpc/vtune/performance_analysis_guide.pdf
2. Arnout, Guido. 2005. Data-Efficient Software and Memory Architectures are Essential for Higher Performance and Lower Power. *Information Quarterly*, V.4, No.3.
3. Krishnan, Karthik, and De Vega, Jun. 2009. Power Analysis of Disk I/O Methodologies. Working paper, Intel Corp. http://softwarecommunity.intel.com/articles/eng/1091.htm
4. Chabukswar, Rajshree. 2009. DVD Playback Power Consumption Analysis. Working paper, Intel Corp. http://softwarecommunity.intel.com/articles/eng/1089.htm
5. MobileMark 2005 and 2007. For information on MobileMark, please see: http://www.bapco.com
6. B. Schilit, N. Adams, and R. Want. (1994). "Context-aware computing applications". IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94), Santa Cruz, CA, US: 89-101. (see: http://sandbox.parc.com/want/papers/parctab-wmc-dec94.pdf )2003.
7. 2007. Application Power Management Best Practices for Windows Vista. Working paper, Microsoft Corp. http://www.microsoft.com/whdc/system/pnppwr/powermgmt/pm_apps.mspx
8. 2011. PowerSettingAccessCheck Function. Working paper, Microsoft Corp. http://msdn.microsoft.com/en-us/library/aa372761(v=VS.85).aspx
9. 2009. Energy-Efficient Platforms: Designing Devices Using the New Power Management Extensions for Interconnects. Working paper, Intel Corp. http://download.intel.com/technology/pdf/322304.pdf
10. 2010. Using PowerCfg to Evaluate System Energy Efficiency. Working paper, Microsoft Corp. http://www.microsoft.com/whdc/system/pnppwr/powermgmt/PowerCfg.mspx
11. 2008. Intel PowerInformer. Working paper. Intel Corp. http://software.intel.com/en-us/articles/intel-powerinformer/
12. 2010. Intel Energy Checker SDK, http://software.intel.com/en-us/articles/intel-energy-checker-sdk/
13. 2010. Power Policy Configuration and Deployment in Windows. Working paper, Microsoft Corp. http://www.microsoft.com/whdc/system/pnppwr/powermgmt/PMpolicy_Windows.mspx


## Further Readings/Useful Websites


1. Steigerwald, Chabukswar, Krishnan, and De Vega. 2008. Creating Energy-Efficient Software. Working Paper. Intel Corp. http://software.intel.com/en-us/articles/creating-energy-efficient-software-part-1/
2. Chabukswar, Rajshree. 2008. Maximizing Power Savings on Mobile Platforms. Working paper, Intel Corp. http://software.intel.com/en-us/articles/maximizing-power-savings-on-mobile-platforms/
3. De Vega, Jun and Chabukswar Rajshree. Data Transfer over Wireless LAN Power Consumption Analysis. 2009. Working paper, Intel Corp. http://www.intel.com/cd/ids/developer/asmo-na/eng/333927.htm
4. Stemen, Pat and Miller Geralyn. 2005. Windows Vista: Developing Power-Aware Applications. slide presentation, SDC 2005, http://download.microsoft.com/download/c/d/5/cd5154e8-d825-4e14-89c8-8b0eb9dda203/pdc_2005_developingpower-awareapplications.ppt
5. E. Capra, G. Formenti, C. Francalanci, S. Gallazzi. 2010. The Impact of MIS software on IT Energy Consumption. European Conference of Information Systems, 2010.
6. Ellison, M. Energy Efficiency for Information Technology. 2010. Intel Press.
7. 2010. Energy Smart Software. Working Paper, Microsoft Corp. http://www.microsoft.com/whdc/system/pnppwr/powermgmt/Energy-Smart_SW.mspx

8. 2010. Mobile Battery Life Solutions for Windows 7: A Guide for Mobile Platform Professionals. Working paper, Microsoft Corp. http://www.microsoft.com/whdc/system/pnppwr/mobile_bat_Win7.mspx
9. 2010. The Science of Sleep. Working Paper, Working paper, Microsoft Corp. http://www.microsoft.com/whdc/system/pnppwr/powermgmt/Science-Sleep.mspx
10. 2010. Timers, Timer Resolution, and Development of Efficient Code. Working paper, Microsoft Corp. http://www.microsoft.com/whdc/system/pnppwr/powermgmt/Timer-Resolution.mspx
11. 2009. Windows Timer Coalescing. Working paper, Microsoft Corp. http://www.microsoft.com/whdc/system/pnppwr/powermgmt/TimerCoal.mspx
12. 2010. ACPI / Power Management - Architecture and Driver Support, Microsoft Hardware Developer Central. Working paper, Microsoft Corp. http://www.microsoft.com/whdc/system/pnppwr/powermgmt/default.mspx

## Acknowledgements

The experiments and data summarized in this chapter are based on years of experimentation and research conducted by engineers of Intel Corp. We would like to acknowledge the tireless efforts and contributions of Rajshree Chabukswar, Jun De Vega, Karthik Krishnan, Manuj Sabharwal, and Jamel Tayeb.