



WHITE PAPER

Intel Software Solutions Group

Petter Larsson

April 2011

Energy-Efficient Software Guidelines

The purpose of this document is to provide energy efficient software guidelines extending the items described in the “Energy-Efficient Software Checklist” document. The guidelines are OS and architecture agnostic except where otherwise noted.

The following guidelines focus on how to optimize applications for energy efficiency. Note that the goal is not to provide system-level optimization suggestions. That said, it is beneficial to the application developer to look at system-wide power efficiency, as running background applications might affect or interact with the target application. For instance, a background virus checker might slow down file access or impact generic performance when active. To eliminate or reduce background impact, try to disable or minimize potential system culprits when measuring application energy efficiency.

Before performing any power optimizations, it is also suggested that you first create a baseline measurement using existing code base so that you have a good reference of comparison modifying your application for energy efficiency.

For a more detailed overview of the topics discussed here, please refer to the Intel® Software Network Energy-Efficient Software portal, <http://softwarecommunity.intel.com/articles/eng/1621.htm>, and the paper “Creating Energy-Efficient Software,” at <http://softwarecommunity.intel.com/articles/eng/1458.htm>.

Contents

1 Computational Efficiency	3
1.1 Loops	3
1.2 Performance Libraries/Extensions	3
1.3 Algorithms	4
1.4 Compiler Optimization	4
1.5 Drivers	4
1.6 Programming language	4
2 Maximize Idle	4
2.1 Multithreading	4
2.2 Reduce Use of High-Resolution Periodic Timers.....	5
3 Data Efficiency.....	5
4 Context/Power-Aware Behavior	6
4.1 Handling Sleep Transitions Seamlessly	6
4.2 Respond/Adapt to System Power Events	7
4.3 Scale Behavior Based on Machine Power State.....	7
4.4 Context Awareness Toolkits.....	7
4.5 Unused Peripherals	8
5 Tools and Testing for Energy-Efficiency	8
5.1 Tools.....	8
5.1.1 Intel Power Checker (Microsoft Windows XP/Windows Vista*/Windows 7).....	8
5.1.2 Perfmon (Microsoft Windows XP/Windows Vista/Windows 7)	8
5.1.3 PwrTest/Windows Driver Kit (Microsoft Windows XP/Windows Vista/Windows 7)	9
5.1.4 Windows Event Viewer/Event Log (Microsoft Windows XP/Windows Vista/Windows 7)	9
5.1.5 Windows ETW (Microsoft Windows XP/Windows Vista/Windows 7).....	9
5.1.6 PowerInformer (Microsoft Windows XP/Windows Vista/Windows 7)	9
5.1.7 Intel® Application Energy Toolkit (Microsoft Windows XP/Windows Vista/Windows 7, Linux, Apple Mac OS*).....	9
5.1.8 PowerTOP (Linux).....	9

5.1.9 Battery Life Toolkit (BLTK) (Linux).....	10
5.1.10 Linux command line tools	10

1 Computational Efficiency

1.1 Loops

Minimize the use of tight loops. To reduce the overhead associated with small loops, the performance/power relationship can be improved by loop unrolling. To achieve this goal, the instructions that are called in multiple iterations of the loop are combined into a single iteration. This approach will speed up the program if the overhead instructions of the loop impair performance significantly. Side effects may include increased register usage and expanded code size.

Greater power savings can often be achieved with the 2nd generation Intel® Core™ processor family by exploiting Intel’s most recent Loop Stream Detection (LSD) technology than by performing loop unrolling. Refer to sections 2.1.2.3 and 3.4.2.4 in the Optimization Reference Manual, <http://www.intel.com/design/processor/manuals/248966.pdf> for details. For best results, investigate both approaches and compare the energy efficiency achieved with each.

It is advisable to convert polling loops to being event driven. If polling loops are necessary, use efficient polling loops (i.e., the largest possible polling interval).

Try to eliminate busy wait (spinning) loops, although in some cases such as when locking or synchronizing using shared memory, the best approach might still be to use a spin-wait loop. For efficient spin-wait loops, it is recommended to use the “pause” instruction.

1.2 Performance Libraries/Extensions

Using instruction set extensions such as SSE instructions, Intel® Advanced Vector Extensions (Intel® AVX) or more recent extensions, performance and energy efficiency can often be improved for computation-intensive applications. The instruction set extensions are often based on the concept of processing multiple data using one instruction (SIMD). For additional details about Intel AVX, refer to: [http://software.intel.com/en-us/avx/?wapkw=\(avx\)](http://software.intel.com/en-us/avx/?wapkw=(avx)).

Application energy efficiency can also be improved by utilizing libraries that are optimized for performance. Intel provides library packages addressing this specific topic, including Intel® Integrated Performance Primitives and the Intel® Math Kernel Library. This collection of libraries contains optimized implementations of common algorithms in areas such as audio, video, imaging, cryptography, speech recognition, and signal processing. For additional details refer to: <http://www3.intel.com/cd/software/products/asm-na/eng/perflib/219780.htm>.

Media applications can benefit from greater power efficiency by taking advantage of video encode and decode hardware acceleration that is available on the 2nd generation Intel Core processor family with Intel® HD Graphics, for example. The Intel® Media Software Development Kit (Intel® Media SDK) provides developers with a standard application programming interface (API) to create high-performance video solutions that run on a variety of current and future Intel® processors. Learn more about the Intel Media SDK at <http://www.intel.com/software/mediasdk>.

Tools such as Intel® VTune™ Performance Analyzer can provide more in-depth performance analysis of your application. Using the VTune environment, developers can drill down on specific performance bottlenecks in an application. See <http://www3.intel.com/cd/software/products/asmo-na/eng/vtune/239144.htm> for more information.

The tools and packages listed above are available for both Windows* and Linux.*

1.3 Algorithms

Generally, it is advised to improve energy efficiency by using high performance algorithms and data structures that complete tasks faster, allowing the processor to idle. If requirements allow, an alternate approach is to investigate the suitability of a less complex (and more energy efficient) algorithm. The solution can also be augmented with the ability to “hot switch” the algorithm depending on the machine power context (refer to the Context/Power-Aware section of this document). For instance, an application might select a lower-quality video encoder/decoder when running on batteries.

Be aware that heavily recursive algorithms can be energy inefficient, as they often add overhead by using or exercising more stack than non-recursive algorithms.

1.4 Compiler Optimization

Energy efficiency can often be improved by optimizing applications for speed using available compiler options such as “O2” or “O3” on Intel® compilers and GNU compilers.

Extended optimizations can be achieved by using application profiling to provide insights such as the most common execution paths. This undertaking generally entails instrumenting the application, executing a profile run, and feeding back profiling information to the compiler. The Intel compilers provide options such as **prof-gen** and **prof-use** for profiled compilation.

For more information on the Intel compilers, visit <http://www3.intel.com/cd/software/products/asmo-na/eng/compilers/284132.htm>.

1.5 Drivers

Identify the kernel, drivers, and libraries used by the application and determine whether there are alternative implementations of components that are more power friendly. For instance, a more recent Linux kernel may feature scheduling optimizations that can make the application run more efficiently. Another possibility could be to update to a more recent and energy efficient Bluetooth* device driver.

1.6 Programming language

If possible, consider using a programming language implementation and libraries that are idle-power-friendly. Some high-level run-time languages may cause more frequent wakeups compared to lower-level system programming languages such as C.

2 Maximize Idle

2.1 Multithreading

Execution can be accelerated by taking advantage of multiple threads and cores, leading to increased idle time that in turn leads to energy savings.

Try to balance your threads, as imbalanced threads may lead to increased energy consumption. The threaded workload can be decomposed using functional decomposition or data decomposition. By threading the workload using data decomposition, multithreaded performance is less likely to be affected by future functional changes. It is preferred to let the OS handle scheduling of threads as opposed to affinizing threads to a certain core. For additional details on multithreading and thread balancing, please refer to the following article:

<http://softwarecommunity.intel.com/articles/eng/2702.htm>.

To analyze how your application performs with regard to threading, we suggest using the Intel® Threading Analysis Tools: <http://www3.intel.com/cd/software/products/asm-na/eng/threading/219785.htm>.

If you are designing threaded components, also consider Intel® Threading Building Blocks (available via the above link).

Finally, the Intel Software Network Concurrency Improvement Center is designed to provide parallel programming tools and resources, starting with the basics, written by our Community experts, to help you improve the concurrency level of your software: <http://software.intel.com/en-us/articles/concurrency-improvement-center/>.

2.2 Reduce Use of High-Resolution Periodic Timers

A good way of reducing application energy footprint is to let it idle as often as possible. Make sure the application is optimized to use the longest timer rate possible while still fulfilling the requirements. Using timer intervals shorter than 15ms has small benefit for most applications. Always make sure to disable periodic timers in case they are not in use, letting the OS adjust the minimum timer resolution accordingly.

Intel® Power Checker provides a quick method of assessing platform timer tick behavior during both application active and idle modes (Microsoft Windows) on mobile platforms using the Intel Core processor family or Intel® Atom™ processor: <http://www.intel.com/partner/sat/pe>.

For driver or kernel applications running under Linux, there are additional timer optimization techniques available. Using “round jiffies”, http://www.lesswatts.org/projects/tickless/round_jiffies.php, non critical timers can be grouped, decreasing the number of wakeups. Using “deferrable timers,” <http://www.lesswatts.org/projects/tickless/deferrable.php>, non critical timers can be queued until processor wakes up from idle by non-deferrable timer.

3 Data Efficiency

Efficient handling of application data can often reduce the energy required to perform a given task.

One approach used to reduce data movement is to buffer data transferred to and from typical storage devices such as hard disks and optical disks. By pre-fetching and/or buffering data, thereby avoiding frequent reads and writes, the device is left more time to idle. Examine your application to determine whether data requests can be buffered and batched into one operation. For additional details, refer to <http://softwarecommunity.intel.com/articles/eng/1462.htm#dataefficiency>.

Another method to minimize data movement is to optimize how data is stored in memory. It is preferable to store data as close as possible to the processing entity. For instance, data efficiency will

improve if an algorithm is optimized so that it uses data in cache as often as possible instead of accessing data from RAM.

It is also beneficial to study how resources (such as memory) are shared between processor cores. One core using a shared resource may prevent other cores from descending into a lower sleep state (C-state). To resolve this issue, try to synchronize threads on different cores to work simultaneously and idle simultaneously.

To analyze how memory is exercised in your application, it is advisable to use memory-profiling tools such as the VTune analyzer and Intel® Performance Tuning Utility (<http://softwarecommunity.intel.com/articles/eng/1437.htm>).

4 Context/Power-Aware Behavior

4.1 Handling Sleep Transitions Seamlessly

Applications can improve power awareness and user experience by reacting/adapting to platform sleep/hibernate/wake-up power transitions. Reacting to platform power transitions usually means that applications should handle the transitions without requiring a restart, loss of data, and change in state. In addition, for a good user experience, applications should handle the power transitions transparently with no user interaction.

Following are some tasks that applications should consider with regard to sleep transitions:

- Saving state/data prior to the sleep transition and restoring state/data after the wake-up transition
- Closing all open system resource handles such as files and I/O devices prior to the sleep transition
- Disconnecting all communication links prior to the sleep transition and re-establishing all communication links after the wake-up transition
- Synchronizing all remote activity (such as writing back to remote files or to remote databases) after the wake-up transition
- Stopping any ongoing user activity (such as streaming video)

On Windows operating systems, the **SetThreadExecutionState** API is used to prevent the system from transitioning to sleep mode. The application should use the API with care and only prevent system idle timeouts when necessary. Remember to reset execution state when the task (such as presentation mode) is complete.

For additional details on this topic, refer to the article, “Application Power Management for Mobility,” <http://softwarecommunity.intel.com/articles/eng/2402.htm>, including some examples on how to handle system transitions on the Windows OS (using **WM_POWERBROADCAST**, **setThreadExecutionState**, etc.) and to the article, “Graceful Application Suspension,” <http://softwarecommunity.intel.com/articles/eng/3437.htm>.

4.2 Respond/Adapt to System Power Events

For some applications, responding to transitions between battery and AC operation, including displaying battery status, can improve the user experience and depending on the measures taken, it may also improve energy efficiency.

To avoid duplicate work in the case of transition to standby, it is advisable to handle low-battery events by saving work, state, or checkpoint. Applications also benefit by adapting to the user-selected OS power policy (scheme).

4.3 Scale Behavior Based on Machine Power State

Improved application energy efficiency can be achieved by scaling the application behavior in response to a change in machine power state. Following are some examples of scaled behavior:

- Reduced resource usage when on battery power (such as disabling automatic background updates/downloads)
- If possible, switch to a “low-power” algorithm when on battery power or running low on battery
- Reducing the quality of video and audio playback in a DVD application to extend playtime while travelling
- Turning off automatic spell check and grammar when on battery power

For additional details on this topic, refer to the following article:

<http://softwarecommunity.intel.com/articles/eng/2738.htm>.

Depending on machine state, also explore the option of letting the application inform the user to select a lower power profile (when necessary) for more energy efficient execution.

Intel Power Checker provides a quick method of assessing application power efficiency during both application active and idle modes (Microsoft Windows) on mobile platforms using the Intel Core processor family or Intel Atom processor: <http://www.intel.com/partner/sat/pe>.

4.4 Context Awareness Toolkits

To simplify some of the above context awareness issues, consider the use of existing context awareness toolkits such as the Intel® Mobile Platform Software Development Kit (SDK), Intel® Laptop Gaming TDK, or Intel® Web 2.0 TDK.

The Intel Mobile Platform SDK is open source and available for both Windows and Linux. The goals of the SDK include managing connectivity transparently, effectively balancing power and performance, using available memory and disk space for local data store and synchronization, adapting to different display types, managing network bandwidth, and managing platform devices and plug-in mechanisms. For SDK details and how to download, refer to <http://ossmpsdk.intel.com/>.

For examples on how to access platform power information using the Intel Mobile Platform Software Development Kit, please refer to the following articles:

<http://softwarecommunity.intel.com/articles/eng/1070.htm> and

<http://softwarecommunity.intel.com/articles/eng/1083.htm>.

The Intel Laptop Gaming TDK is available for Windows XP and Vista and provides an easy interface to help extend games by adding mobile-aware features to create a better laptop gaming experience. For TDK details and how to download, refer to <http://softwarecommunity.intel.com/articles/eng/1017.htm>.

The Intel Web 2.0 TDK allows developers to extract information about the platform's configuration (e.g., display, storage, and processor), and the platform's context (e.g., bandwidth, connectivity, power, and location) within a browser using JavaScript. For TDK details and how to download, refer to <http://softwarecommunity.intel.com/articles/eng/1026.htm>.

4.5 Unused Peripherals

To further improve energy efficiency, explore the option of disabling or even turning off unused peripherals.

For instance, in case an application exclusively uses Bluetooth on the system, the device could be disabled temporarily if there is no activity, leading to improved energy efficiency.

5 Tools and Testing for Energy-Efficiency

To analyze your application's energy efficiency, it is recommended to profile platform power usage during application runtime. Please refer to the list of tools described in the next chapter to assist analysis.

During profiling, try to explore the following power-related aspects of application execution:

- Understand the power impact of the application at Idle and Running state
- Examine C-state behavior
- Examine P-state behavior
- Examine timer interrupts
- Examine interrupt statistics
- Examine disk and file access statistics

5.1 Tools

A range of tools are available that address power-related frameworks, optimizations, and measurements.

5.1.1 Intel Power Checker (Microsoft Windows XP/Windows Vista*/Windows 7)

Intel Power Checker can be used to quickly assess idle power efficiency (C3 state residency), platform timer tick, and power-aware behavior for applications that run on mobile platforms using the Intel Core processor family or Intel Atom processor: <http://www.intel.com/partner/sat/pe>.

5.1.2 Perfmon (Microsoft Windows XP/Windows Vista/Windows 7)

Perfmon can be used to assist optimizations, monitoring results of tuning and configuration scenarios, and the understanding of a workload and its effect on resource usage to identify bottlenecks. <http://softwarecommunity.intel.com/articles/eng/3449.htm>.

5.1.3 PwrTest/Windows Driver Kit (Microsoft Windows XP/Windows Vista/Windows 7)

The Power Management Test Tool enables developers, testers, and system integrators to exercise and record power-management information from the platform. PwrTest can be used to automate sleep and resume transitions and record processor power management and battery information from the platform over a period of time: see <http://msdn.microsoft.com/en-us/library/ff550682.aspx> and http://www.microsoft.com/whdc/system/pnppwr/powermgmt/PM_apps.mspc.

5.1.4 Windows Event Viewer/Event Log (Microsoft Windows XP/Windows Vista/Windows 7)

Windows Event Viewer/Log provides a centralized log service to report events that have taken place, such as a failure to start a component or to complete an action.

For instance, the tool can be used to capture “timer tick” change events, which have an indirect effect on platform energy efficiency: http://en.wikipedia.org/wiki/Event_Viewer.

5.1.5 Windows ETW (Microsoft Windows XP/Windows Vista/Windows 7)

Event Tracing for Windows (ETW) provides application programmers the ability to start and stop event tracing sessions, instrument an application to provide trace events, and consume trace events. You can use the events to debug an application and perform capacity and performance analysis: see [http://msdn.microsoft.com/en-us/library/bb968803\(v=VS.85\).aspx](http://msdn.microsoft.com/en-us/library/bb968803(v=VS.85).aspx).

5.1.6 PowerInformer (Microsoft Windows XP/Windows Vista/Windows 7)

PowerInformer provides relevant and condensed platform power information to the developer, including for instance battery status, C and P state residency, interrupt rate and disk/file IO rates: see <http://softwarecommunity.intel.com/articles/eng/3725.htm>.

5.1.7 Intel® Application Energy Toolkit (Microsoft Windows XP/Windows Vista/Windows 7, Linux, Apple Mac OS*)

Intel Application Energy Toolkit is a set of tools designed to help software developers and quality-assurance teams create and evaluate software applications for power awareness:

- **Power Status Simulator** simulates a battery-operated environment for an application running on a platform that is AC powered.
- **Application Energy Graphing Tool** is an interactive tool that can measure the battery power consumption of an application over time, log and graph the resulting data.
- **Application Energy Command Line Tool** automates the process of determining the total power consumption and net power consumption of an application running on a battery-operated platform.

See <http://softwarecommunity.intel.com/articles/eng/1631.htm>.

5.1.8 PowerTOP (Linux)

PowerTOP helps to point out the power inefficiencies of your platform. The tool shows how well the platform is using the various hardware power-saving features and culprit software components that are preventing optimal usage. It also provides tuning suggestions on how to achieve low power consumption: see <http://www.lesswatts.org/projects/powertop/>.

5.1.9 Battery Life Toolkit (BLTK) (Linux)

Battery Life Toolkit (BLTK) provides infrastructure to measure laptop battery life, by launching typical single-user workloads for power performance measurement: see

<http://www.lesswatts.org/projects/bltk/>.

5.1.10 Linux command line tools

Disk/Device activity

- **blktrace** generates traces of IO traffic: see <http://linux.die.net/man/8/blktrace>.
- **iostat** provides device utilization reports such as r/w per second: see <http://linux.die.net/man/1/iostat>.

Application activity

- **strace** traces system calls: see <http://linux.die.net/man/1/strace>.
- **frysk** monitor running processes and threads: see <http://sourceware.org/frysk/>.

Others

- **cpufreq-info** displays CPU type, available P states, and governor: see <http://linux.die.net/man/1/cpufreq-info>.

OPTIMIZATION NOTICE

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel® Compiler User and Reference Guides” under “Compiler Options.” Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

This white paper, as well as the software described in it, is furnished under license and may only be used or copied in accordance with the terms of the license. The information in this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See www.intel.com/products/processor_number for details.

The Intel processor/chipset families may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents, which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web site.

Intel, the Intel Logo, Atom, Core, MMX, and VTune are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2011, Intel Corporation. All rights reserved.