Core Energy Efficiency

Seminar "Energy-Efficient Programming"

Dr. Manuel Dolz, Michael Kuhn, Dr. Julian Kunkel,
Konstantinos Chasapis, Prof. Dr. Thomas Ludwig

Marcus Soll

Universität Hamburg
Fakultät für Mathematik,
Informatik und Naturwissenschaften
Department Informatik

2014-11-19

Motivation

- Goal: Computers with one ExaFLOPs
  - 10^{18} float operations per second
- Important for more accurate simulations and massive data analysis
  - Biotechnology
  - Nanotechnology
  - Materials science
- Biggest problem: Energy consumption
  - Power consumption needs to be around 20 MW maximum

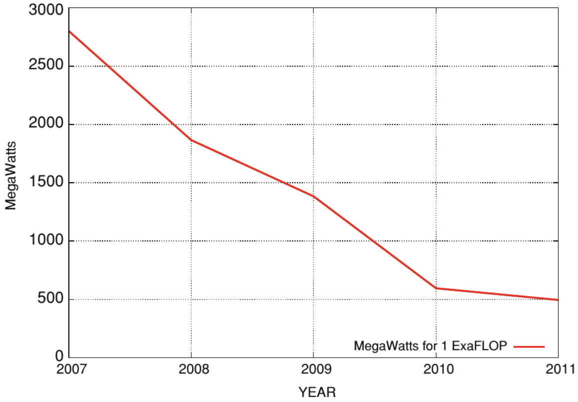
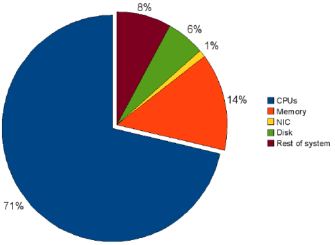
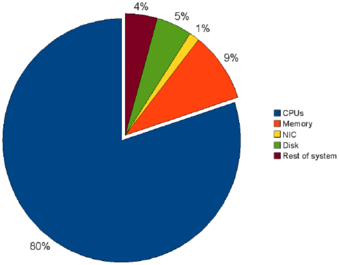


Figure: Energy needed for one ExaFLOP based on Green 500. Source: [LPK+13]

- Formula for power consumption: $P = C \cdot f \cdot V^2$
  - But each frequency need a specific minimal voltage
  - Reducing voltage also reduces frequency
  - Requirement of advanced power management
- This talk will discuss basic principles concerning energy efficiency
- Basic principles of other methods
- Focus: CPU, Memory



(a) Idle power consumption, all components are utilized 0 %.



(b) Load power consumption, all components are utilized 100 %.

Figure: Distribution of energy consumption. Source: [Min09]

Introduction

CPU
- General
- ACPI
- Implementations

Memory
- General
- Movement of data
- Energy reduction

Examples
- ACPI
- Memory

Conclusion

CPU

General information

- The CPU (processor) is the main component of a computer
- It fetches instructions and executes them
- Contains a limited amount of “registers” and gets all other data from the memory

History

- 1965: Moores Law: Computer performance double every 18 month
- Around 2000: Slower growth on single chip - shift to multi core
- Today: Physical limits of multi core systems - shift to many core

ACPI

- Specification defines an interface for power management
- First released December 1996
- Each device can be controlled through power states
- OS is in control of power management
- Bytecode language (AML)

ACPI

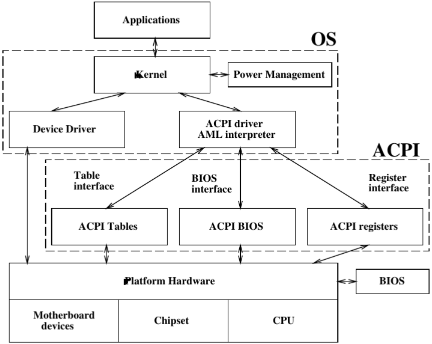


Figure: Basic ACPI structure. Source: [LSM99]

ACPI

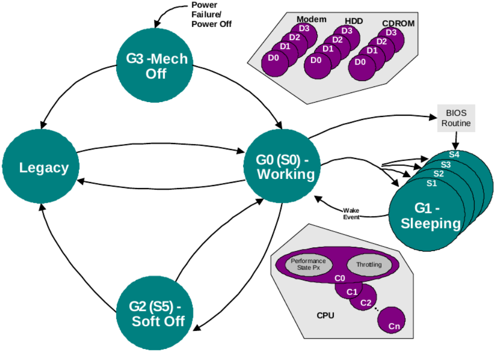


Figure: ACPI power states. Source: [CCC+13]

G-States / S-States

- The “global states” (“sleeping states”) define the overall system state
  - G0 (Working)
  - G1/S1-S4 (Sleeping)
  - G2/S5 (Soft off)
  - G3 (Mechanical off)
- Only in G0 user application are executed
- G0 offers further customisation
- G2 and G3 require restart of OS

C-States

- The “processor power states” (c-states) can be used to control the CPU while the system is in G0-state
- The states differ in latency and power consumption
  - C0
  - C1
  - C2 $\cdots$ Cn
- In C0 the processor executes instructions
- In C1 the processor does not execute instructions. Switching to C0 has almost no latency
- All other states are optional and can be defined by the manufacturer

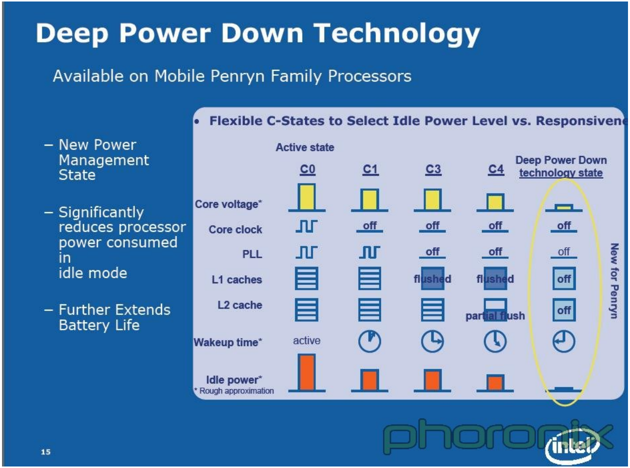


Figure: C-states of the "Intel Penryn Family" architecture. Source: [Lin07]

P-States

- "Performance states" (p-states) enable further control over CPU (and devices) when in active state (C0/D0)
- Up to 16 states (P0 $\cdots$ P15)
- Controls the power and frequency of the processor
- Implementation is optional

ACPI

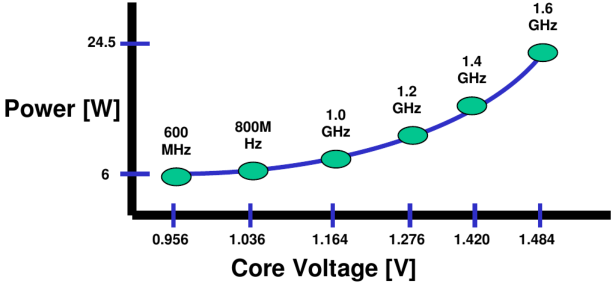


Figure: P-states of an "Intel Pentium M". Source: [Cor04]

Throttling

- Throttling provides an alternative interface to performance control
- A throttling-value may be specified
- This value determines how much performance (in percent) the CPU should run on
- Throttling is ineffective compared to p-states

D-States

- Used to control devices like CD-reader, printer, modems, drives...
- Four states
  - D0 (full-on)
  - D1
  - D2
  - D3 (off)
- Latency and power saving highly dependent on device

ACPI

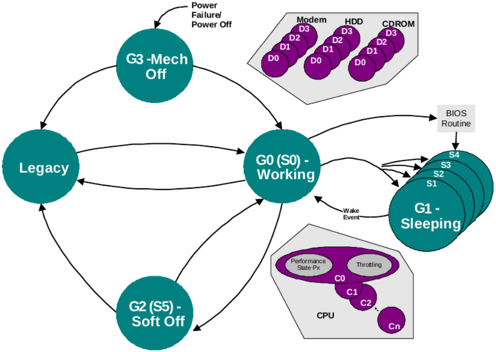


Figure: ACPI power states. Source: [CCC+13]

Implementation - Linux

- Core ACPI system implementation called “ACPICA”
  - Does not implement policies
- “ACPI drivers” implement policies
  - C-states are controlled by “idle loop”
  - P-states are controlled by different “governors”
  - Throttling is used on thermal emergencies

Implementation - Windows

- First implementation in Windows 2000 (1996)
- All driver have to register to the ACPI driver
- The ACPI driver calls registered methods on ACPI changes
- The user can influence the power management by “policies”
- Applications can disable certain parts of the power management

Memory

General

- Second major component in modern PCs
- Cache results of operations
- Goal: Fast, large and cheap
  - Can not be done with current technology
  - Combination of multiple type of memory

Memory types

- Different memory types build into a hierarchy:
  - CPU-register
  - Cache (L1-cache, L2-cache...)
  - RAM
  - Persistent cache (Hard disk drives, magnetic tape...)
- Different costs and access time

Non-uniform memory access

- Provides a single address space off all memory for all CPUs
- All memory can be accessed via unified instructions
- Access to local memory is faster than remote memory

Movement of data

- Experimental analysis of data movement costs
  - Average energy cost of moving data is 25%
  - Peak energy cost around 40%

Operation	Energy Cost (nJ)	Δ Energy (nJ)	Eq. Ops
NOP	0.48	-	-
ADD	0.64	-	-
L1->REG	1.11	1.11	1.8 ADD
L2->L1	2.21	1.10	3.5 ADD
L3->L2	9.80	7.59	15.4 ADD
MEM->L3	63.64	53.84	99.7 ADD
stall	1.43	-	-
prefetching	65.08	-	-

Figure: Energy spend accessing memory (AMD Interlagos 6227). Source: [PWnt]

Energy reduction - Reduce data movement

- Reduce amount of data movement
- Algorithmic changes
  - Keep data redundant on multiple cores
  - Calculation of data instead storing

Energy reduction- DVFS

- Dynamically scale down frequency and voltage of DRAM
  - Experimental data suggest average 2.43% power reduction (max. 5.15%) [DFG+11]
  - Experimental data suggest minimal slowdown of average 0.17% (max. 1.69%) [DFG+11]
  - Problem: Data transfers take longer ⇒ more energy consumption
  - Problem: No current implementation
- Better results when scaling CPU and DRAM together

Examples

Examples - ACPI in Linux

- You can control ACPI in Linux using `cpufrequtils`
  - `cpufreq-info` shows information about current power management settings
  - `cpufreq-set` allows changing current power management behaviour
  - `cpufreq-aperf` measures current power management stats

ACPI

```
~ $ cpufreq-info
cpufrequtils 008: cpufreq-info (C) Dominik Brodowski 2004-2009
Bitte melden Sie Fehler an cpufreq@vger.kernel.org.
analysiere CPU 0:
  Treiber: acpi-cpufreq
  Folgende CPUs laufen mit der gleichen Hardware-Taktfrequenz: 0
  Die Taktfrequenz folgender CPUs werden per Software koordiniert: 0
  Maximale Dauer eines Taktfrequenzwechsels: 10.0 us.
  Hardwarebedingte Grenzen der Taktfrequenz: 933 MHz - 2.53 GHz
  mögliche Taktfrequenzen: 2.53 GHz, 2.40 GHz, 2.27 GHz, 2.13 GHz, 2.00 GHz, 1.87 GHz, 1.73 GHz, 1.60 GHz,
  mögliche Regler: conservative, performance
  momentane Taktik: die Frequenz soll innerhalb 933 MHz und 2.53 GHz.
                    liegen. Der Regler "conservative" kann frei entscheiden,
                    welche Taktfrequenz innerhalb dieser Grenze verwendet wird.
  momentane Taktfrequenz ist 933 MHz.
analysiere CPU 1:
  Treiber: acpi-cpufreq
  Folgende CPUs laufen mit der gleichen Hardware-Taktfrequenz: 1
  Die Taktfrequenz folgender CPUs werden per Software koordiniert: 1
  Maximale Dauer eines Taktfrequenzwechsels: 10.0 us.
  Hardwarebedingte Grenzen der Taktfrequenz: 933 MHz - 2.53 GHz
  mögliche Taktfrequenzen: 2.53 GHz, 2.40 GHz, 2.27 GHz, 2.13 GHz, 2.00 GHz, 1.87 GHz, 1.73 GHz, 1.60 GHz,
  mögliche Regler: conservative, performance
  momentane Taktik: die Frequenz soll innerhalb 933 MHz und 2.53 GHz.
                    liegen. Der Regler "conservative" kann frei entscheiden,
                    welche Taktfrequenz innerhalb dieser Grenze verwendet wird.
  momentane Taktfrequenz ist 2.53 GHz.
analysiere CPU 2:
```

ACPI

```
~ $ cpufreq-info -fmc 0
933 MHz
~ $ cpufreq-info --governor
conservative performance
~ $ sudo cpufreq-set -g performance
Passwort:
~ $ cpufreq-info -fmc 0
2.53 GHz
~ $ sudo cpufreq-set -g conservative
~ $ cpufreq-info -fmc 0
933 MHz
```

ACPI

```
~ $ sudo cpufreq-aperf
CPU     Average freq(KHz)       Time in C0      Time in Cx      C0 percentage
000     1063860                 00 sec 048 ms   00 sec 951 ms   04
001     1089190                 00 sec 061 ms   00 sec 938 ms   06
002     1317160                 00 sec 021 ms   00 sec 978 ms   02
003     1266500                 00 sec 002 ms   00 sec 997 ms   00

000     1089190                 00 sec 016 ms   00 sec 983 ms   01
001     1114520                 00 sec 008 ms   00 sec 991 ms   00
002     1418480                 00 sec 023 ms   00 sec 976 ms   02
003     1393150                 00 sec 002 ms   00 sec 997 ms   00

000     0987870                 00 sec 022 ms   00 sec 977 ms   02
001     1215840                 00 sec 007 ms   00 sec 992 ms   00
002     1114520                 00 sec 011 ms   00 sec 988 ms   01
003     1215840                 00 sec 028 ms   00 sec 971 ms   02
```

Examples - Memory management in Linux

- Algorithm “Dynamic Memory Switching”
- Developed by Prof. Rajat Moona, Sharad Chole, Sanchay Harneja
- Implemented for Linux 2.6.15
- Goal: Switch off unused memory

Dynamic Memory Switching

- New kernel daemon
  - Migrates memory pages and frees parts of memory (banks)
  - Sets banks to low-power state

Power State/Transition	Power	Time	Active Components
Active	300mW	-	Refresh, clock, row, col decoder
Standby	180mW	-	Refresh, clock, row decoder
Nap	30mW	-	Refresh, clock
Powerdown	3mW	-	Refresh
Standby To Active	240mW	+6ns	
Nap To Active	160mW	+60ns	
Powerdown To Active	150mW	+6000ns	

Figure: Energy of different memory power states. Source: [MCH07]

Conclusion

- Core method of reducing energy consumption of CPU
  - ACPI
- Energy consumption of memory
  - Problems
  - Possible solutions

[BKL+05] Len Brown, Anil Keshavamurthy, David Shaohua Li, Robert Moore, Venkatesh Pallipadi, and Luming Yu.
ACPI in Linux.
In *Ottawa Linux Symposium*, 2005.

[Bor07] Shekhar Borkar.
Thousand core chips: a technology perspective.
In *Proceedings of the 44th annual Design Automation Conference*, pages 746–749, 2007.

[CCC+13] Hewlett-Packard Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., and Toshiba Corporation.
AdvancedConfiguration and Power Interface Specification, November 2013.

[Cor04] Intel Corporation.
Enhanced Intel® SpeedStep® Technology for the Intel® Pentium® M Processor, March 2004.

[Cor05] Intel Corporation.
Excerpts from A Conversation with Gordon Moore: Moore's Law.
2005.

[Cor07a] Microsoft Corporation.
ACPI Driver Interface in Windows Vista, April 2007.

[Cor07b] Microsoft Corporation.
Processor Power Management in Windows Vista and Windows Server 2008, November 2007.

[Cor09] Microsoft Corporation.
Power Availability Requests, June 2009.

[DFG+11] Howard David, Chris Fallin, Eugene Gorbatov, Ulf R. Hanebutte, and Onur Mutlu.
Memory power management via dynamic voltage/frequency scaling.
In *Proceedings of the 8th ACM international conference on Autonomic computing*, pages 31–40, June 2011.

[DMB+12] Qingyuan Deng, David Meisner, Abhishek Bhattacharjee, Thomas F. Wenisch, and Ricardo Bianchini.
CoScale: Coordinating CPU and Memory System DVFS in Server Systems.
In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 143–154. IEEE, December 2012.

[Gee05] David Geer.
Chip makers turn to multicore processors.
In *Computer*, volume 38, issue: 5, pages 11–13. IEEE, May 2005.

[GGJ+13] Hormozd Gahvari, William Gropp, Kirk E. Jordan, Martin Schulz, and Ulrike Meier Yang.
Systematic Reduction of Data MovementAlgebraic Multigrid Solvers, 2013.

[Gro10] Andrew Grover.
Modern System Power Management.
Queue - Power Management, Volume 1(Issue 7):66, January 2010.

[KGKH13] Gokcen Kestor, Roberto Gioiosa, Darren J. Kerbyson, and Adolfy Hoisie.
Quantifying the Energy Cost of Data Movement in Scientific Applications.
In *2013 IEEE International Symposium on Workload Characterization (IISWC)*, pages 56–65. IEEE, September 2013.

[L+14] Robert Lucas et al.
Top Ten Exascale Research Challanges.
U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, February 2014.

[Lin07] David Lin.
Intel Penryn & Nehalem Information.
http://www.phoronix.com/scan.php?page=article&item=672
March 2007.
Accessed: 2014-11-02 12:42.

[LPK+13] James H. Laros, III, Kevin Pedretti, Suzanne M. Kelly, Wei Shu, Kurt Ferreira, John Van Dyke, and Courtenay Vaughan.
Energy-Efficient High Performance Computing.
Springer, 2013.

[LSM99] Yung-Hsiang Lu, Tajana Simunic, and Giovanni De Micheli.
Software Controlled Power Management.
Technical report, Computer System Laboratory, Stanford University, 1999.

[MCH07] Prof. Rajat Moona, Sharad Chole, and Sanchay Harneja.
Memory Management using Dynamic Memory Switching, May 2007.

[Min09] Timo Minartz.
Model and simulation of power consumption and power saving potential of energy efficient cluster hardware.
Master's thesis, Ruprecht-Karls-Universität Heidelberg, August 2009.

[Min13] Timo Minartz.
Design and Evaluation of Tool Extensions for Power Consumption Measurement in Parallel Systems.
PhD thesis, Universität Hamburg, March 2013.

[PWnt] Dhinakaran Pandiyan and Carole-Jean Wu.
Quantifying the Energy Cost of Data Movement for Emerging Smart Phone Workloads on Mobile Platforms.
In *2014 IEEE International Symposium on Workload Characterization*, pre-print.

[Sim09] Dario Simone.
Power Management in a Manycore Operating System.
Master's thesis, ETH Zurich, August 2009.

[Tan09] Andrew S. Tanenbaum.
Modern Operating Systems.
Pearson Education, Inc., third edition, 2009.