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
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)
Figure: Basic ACPI structure. Source: [LSM99]
**Figure**: ACPI power states. Source: [CCC+13]
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
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 ··· 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.
Deep Power Down Technology

Available on Mobile Penryn Family Processors

- New Power Management State
- Significantly reduces processor power consumed in idle mode
- Further Extends Battery Life

**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 ··· P15)
- Controls the power and frequency of the processor
- Implementation is optional
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
ACPI

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

- Core ACPI system implementation called “ACPI-ICA”
  - 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%
Movement of data

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

**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%) \([\text{DFG}^+11]\)
  - Experimental data suggest minimal slowdown of average 0.17% (max. 1.69%) \([\text{DFG}^+11]\)
  - Problem: Data transfers take longer \(\Rightarrow\) 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, 1.47 GHz, 1.33 GHz, 1.20 GHz, 1.07 GHz, 933 MHz
    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, 1.47 GHz, 1.33 GHz, 1.20 GHz, 1.07 GHz, 933 MHz
    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:
```

Marcus Soll

Core Energy Efficiency
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
Core Energy Efficiency

```
~ $ 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

<table>
<thead>
<tr>
<th>Power State/Transition</th>
<th>Power</th>
<th>Time</th>
<th>Active Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>300mW</td>
<td>-</td>
<td>Refresh, clock, row, col decoder</td>
</tr>
<tr>
<td>Standby</td>
<td>180mW</td>
<td>-</td>
<td>Refresh, clock, row decoder</td>
</tr>
<tr>
<td>Nap</td>
<td>30mW</td>
<td>-</td>
<td>Refresh, clock</td>
</tr>
<tr>
<td>Powerdown</td>
<td>3mW</td>
<td>-</td>
<td>Refresh</td>
</tr>
<tr>
<td>Standby To Active</td>
<td>240mW</td>
<td>+6ns</td>
<td></td>
</tr>
<tr>
<td>Nap To Active</td>
<td>160mW</td>
<td>+60ns</td>
<td></td>
</tr>
<tr>
<td>Powerdown To Active</td>
<td>150mW</td>
<td>+6000ns</td>
<td></td>
</tr>
</tbody>
</table>

**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.


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

Memory power management via dynamic voltage/frequency scaling. 
CoScale: Coordinating CPU and Memory System DVFS in Server Systems.

[Gee05] David Geer.
Chip makers turn to multicore processors.

Quantifying the Energy Cost of Data Movement in Scientific Applications.

*Top Ten Exascale Research Challanges.*
Intel Penryn & Nehalem Information.
http://www.phoronix.com/scan.php?page=article&item=672
March 2007.

Software Controlled Power Management.

[MCH07] Prof. Rajat Moona, Sharad Chole, and Sanchay Harneja.

[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.  

*Modern Operating Systems*.  