Saving Energy without Performance Degradation

Jan Richling
Jan H. Schönherr

Kommunikations- und Betriebssysteme
Technische Universität Berlin

Gero Mühl

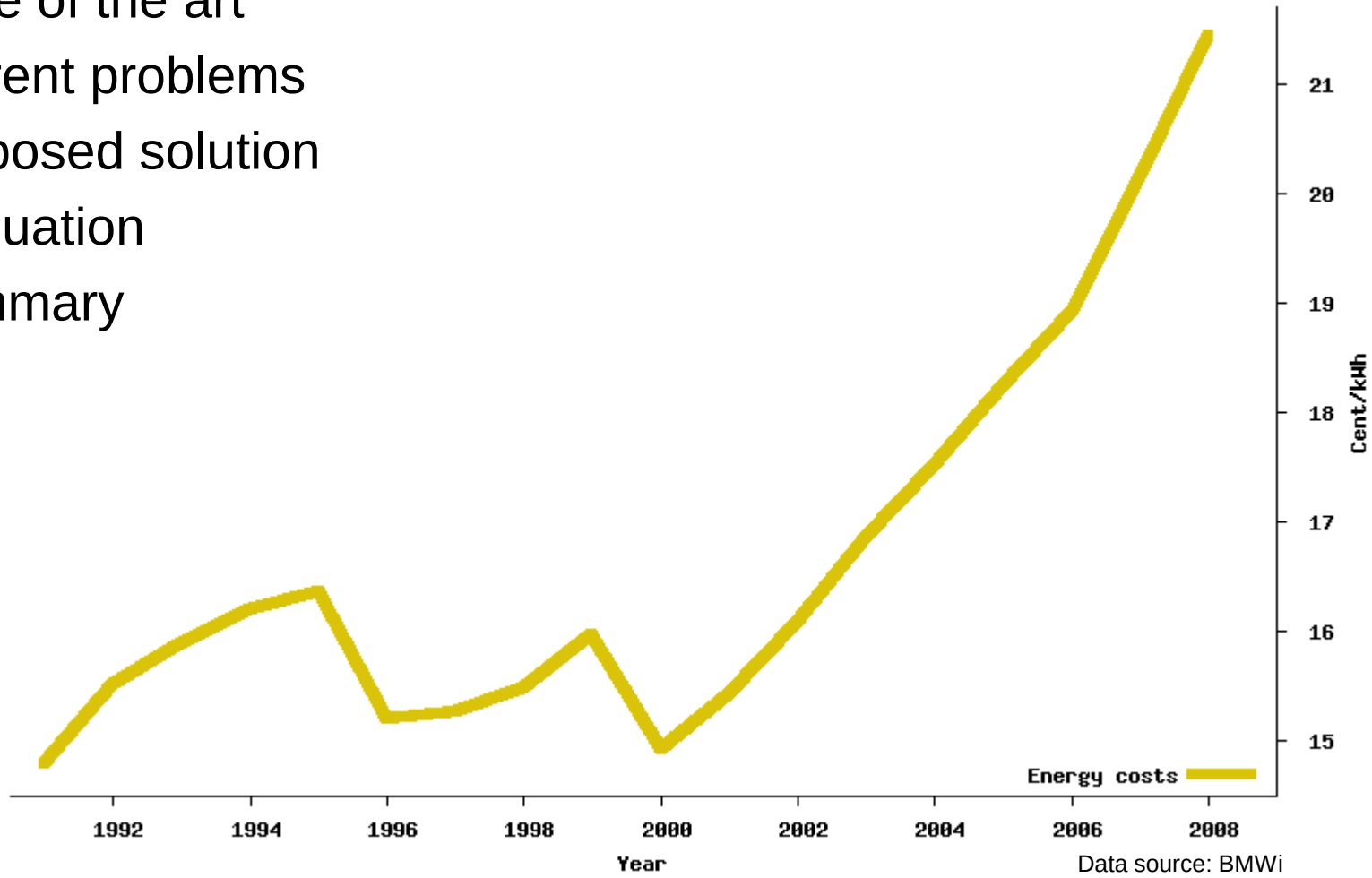
Architektur von Anwendungssystemen
Universität Rostock

Matthias Werner

Professur für Betriebssysteme
Technische Universität Chemnitz

Overview

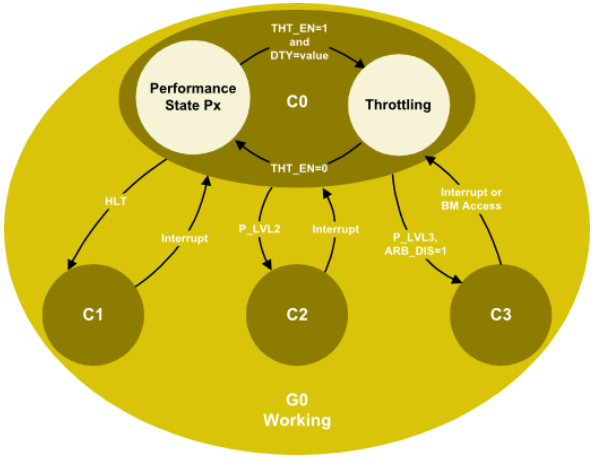
- State of the art
- Current problems
- Proposed solution
- Evaluation
- Summary



Data source: BMWi

Saving Energy: State of the Art

- When nothing has to be done
  - Switch off components
  - ACPI defines C-states for processors
    - C0: working
    - C1..C*n*: sleep-states
    - Managed by OS idle routine



ACPI processor power states
Image source: ACPI Specification, rev. 4.0

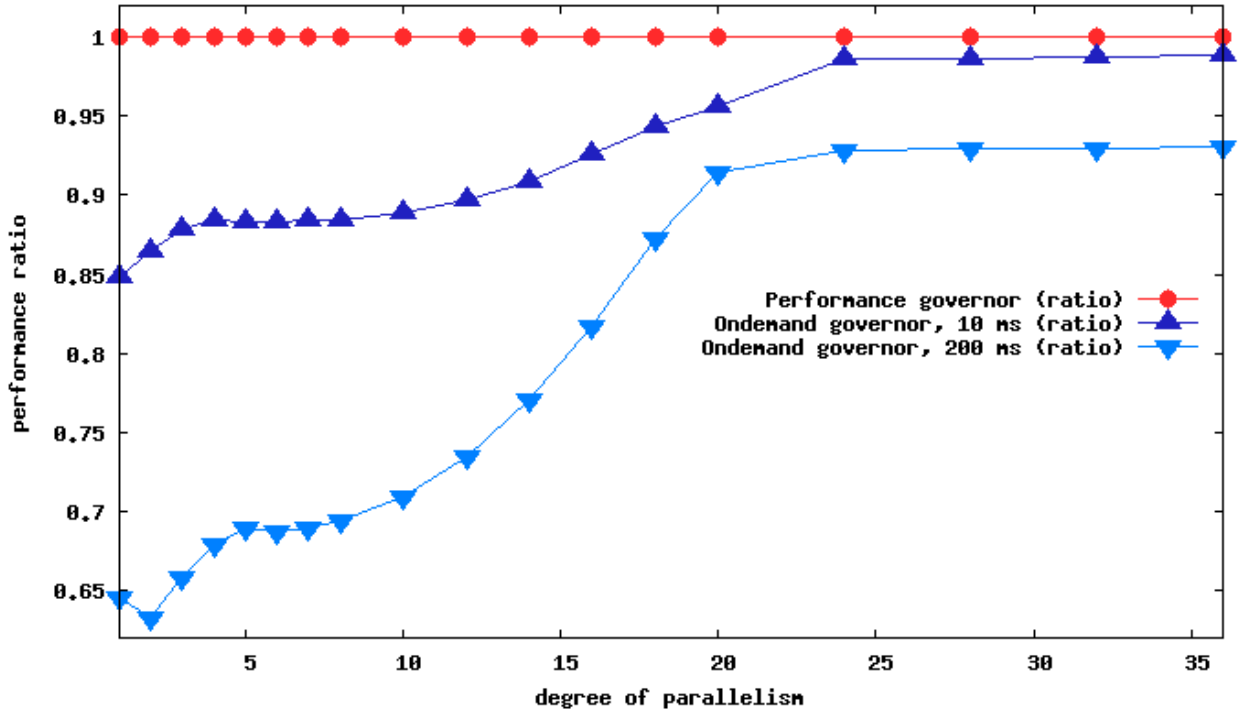
- When only a little bit has to be done
  - Reduce performance of components
  - ACPI defines P-states for processors
    - Frequency/voltage combinations
    - P0: high performance
    - P1..P*n*: lower performance
    - Managed by OS frequency governor

	MHz	TDP
P0	2600	115.0 W
P1	2100	93.2 W
P2	1700	80.8 W
P3	1400	71.5 W
P4	800	53.2 W

P-states of an
AMD Opteron 8435
Data source: AMD

Current Approach: Time-Driven Governor

> Example: Linux kernel compilation



Governor	Energy
Performance	329 Wh
Ondemand (10 ms)	368 Wh
Ondemand (200 ms)	475 Wh

Energy consumption for a parallelism degree of 1

> Severe performance degradation

> Increased energy consumption

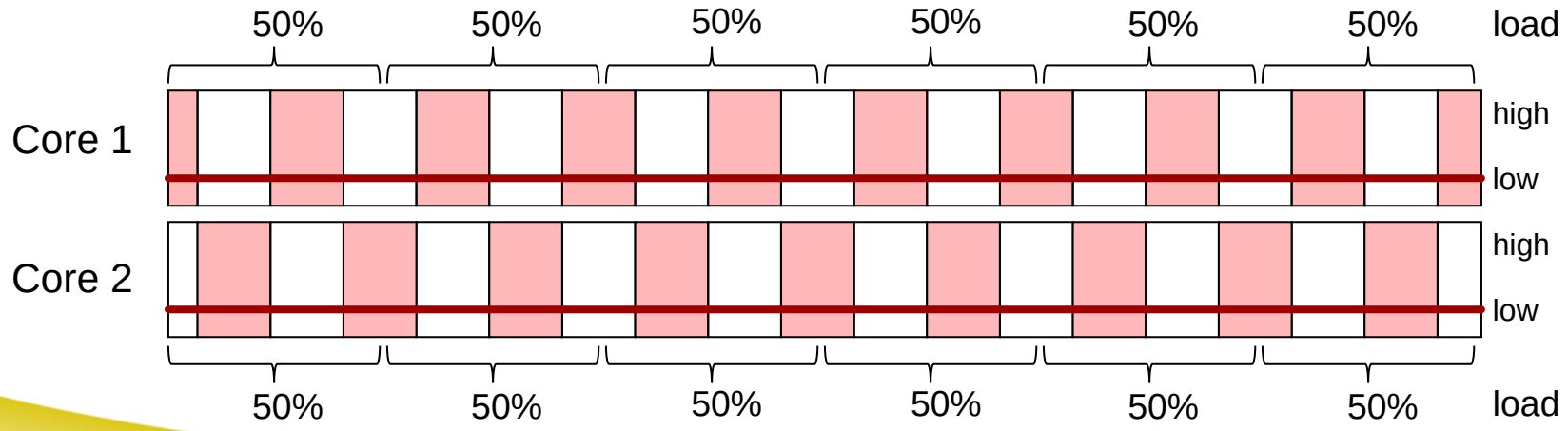
Analyzing Problems: Problem 1

> Governor determines load by polling

> Incorrect results in case of
  > small activity phases/short running processes, and
  > a low overall degree of parallelism

→ No increase of frequency

→ Performance degradation

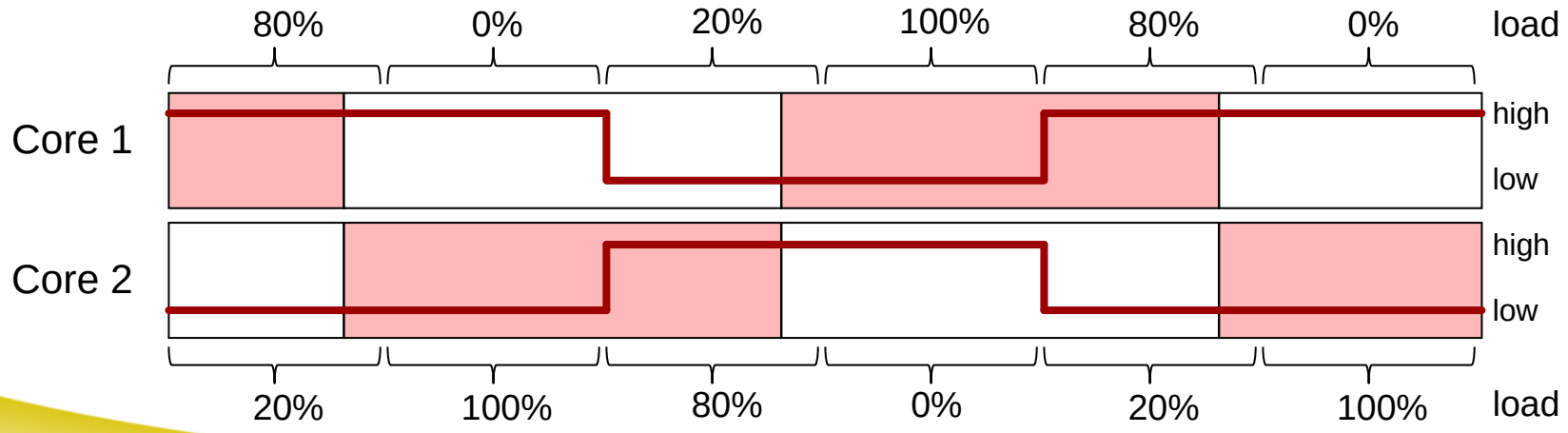


Analyzing Problems: Problem 2

- Governor reduces frequency of idling cores
- Scheduler favors idle cores

- New load is scheduled on cores with low frequency
  - in case of a low overall degree of parallelism

→ Performance degradation



Problem: Separation of Concerns

- Clear separation between
  - Scheduling (which task when and where) → Scheduler
  - Energy control (clock frequencies of cores) → Governor

- But
  - The functions mentioned above depend on each other in reality!
  - Overhead: Governor reconstructs information already available to scheduler

- Energy consumption is a non-functional property
  - Separation of concerns may raise problems

Approach: Integration

> Non-functional interdependency between scheduling and frequency control
  → Integration instead of separation
  → Combination of scheduler and governor

> Two-step approach
  1. Event-driven frequency control by scheduler

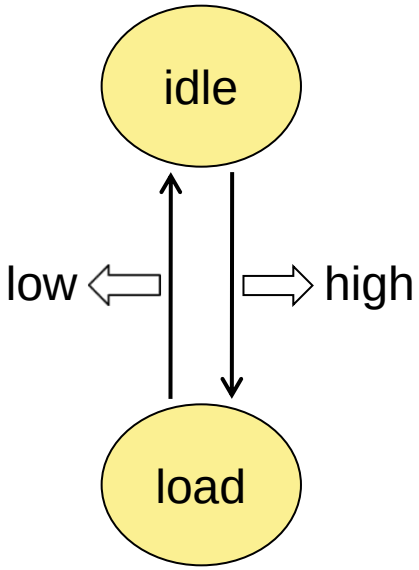
  2. Adaption of scheduler (future work)
     > Scheduling in space according to power states of cores
     > Concentration of load

Event-driven Frequency Control

- Load changes trigger frequency transitions
- Purely event-driven: No polling, no consideration of load

- Input events: State change from idle to load or vice versa
- Output events: State change from low to high frequency or vice versa

- Optimization
  - Delays for transitions
  - Minimum time to stay in high or low frequency



Implementation

- Based on current Linux kernel (2.6.32-rc5)
- Scheduler interface
  - Scheduler notifies CPUFreq framework whenever load changes
- New scheduler governor
  - Initiates P-state transitions based on load changes
- Modified CPUFreq driver
  - Driver is now used in atomic contexts
  - Must be (sleep-)lock-free

```
on_load_change() {
    if( (is_idle && is_low) ||
            (is_load && is_high) ) {
        delete_timer();
    } else {
        if( is_due(next()) )
            initiate_transition();
        else
            add_timer(initiate_transition, next());
    }
}
```

Pseudo code of scheduler governor

Evaluation: Hardware

- Quad AMD Opteron 8435
  - Latest generation server
  - Four sockets
  - 45 nm K10 hexa-core processors
    - code-name Istanbul
  - Frequencies: 0.8, 1.4, 1.7, 2.1, and 2.6 GHz
  - Minimum idle consumption: 280W

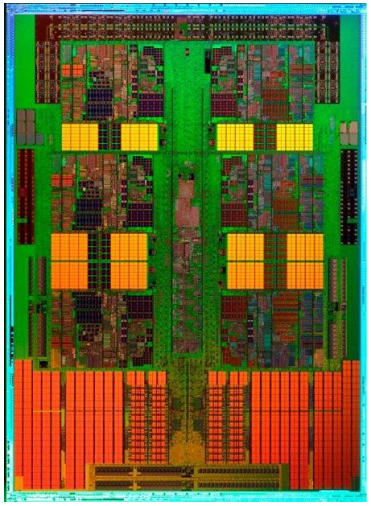


Image source: AMD

- AMD Phenom 9950
  - Previous generation desktop
  - 65 nm K10 quad-core processor
    - code-name Agena
  - Frequencies: 1.3 and 2.6 GHz
  - Minimum idle consumption: 84W

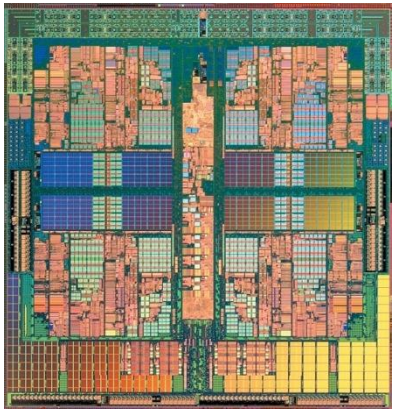
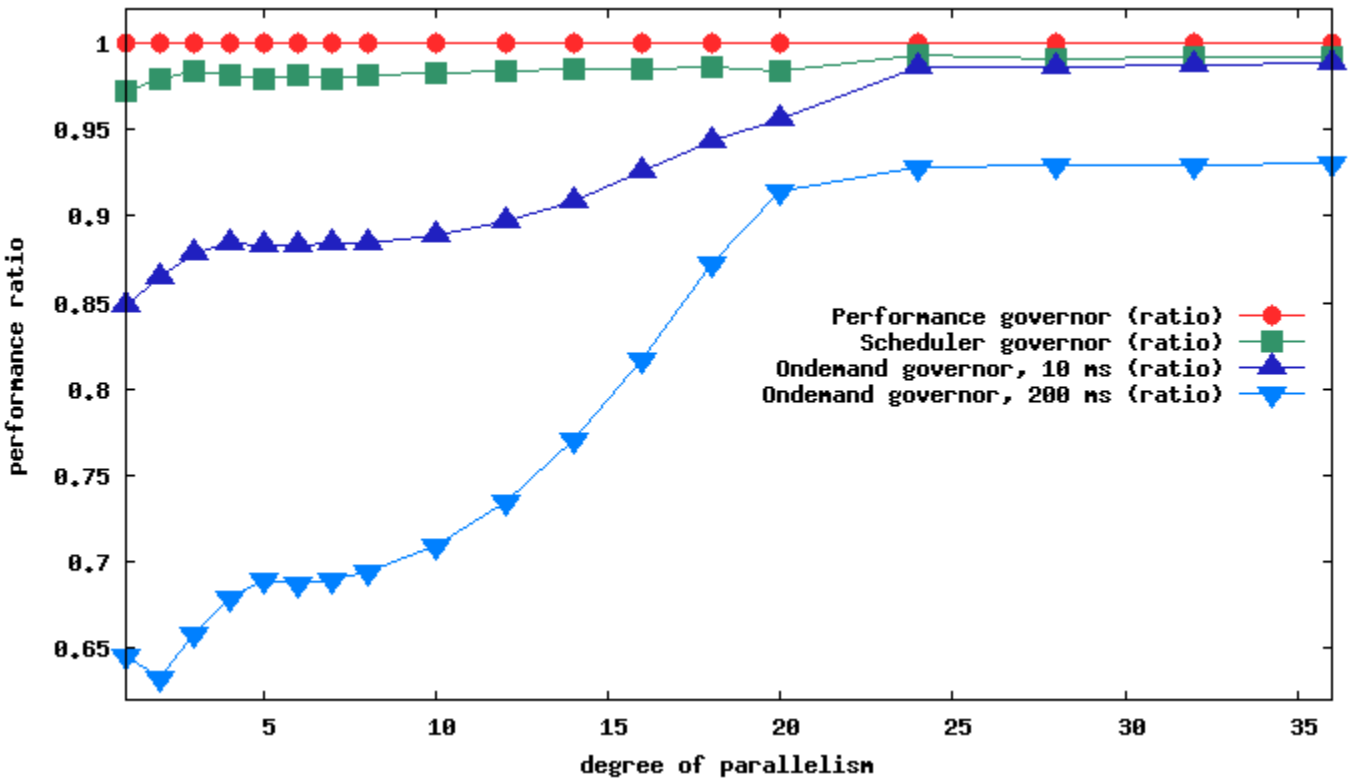


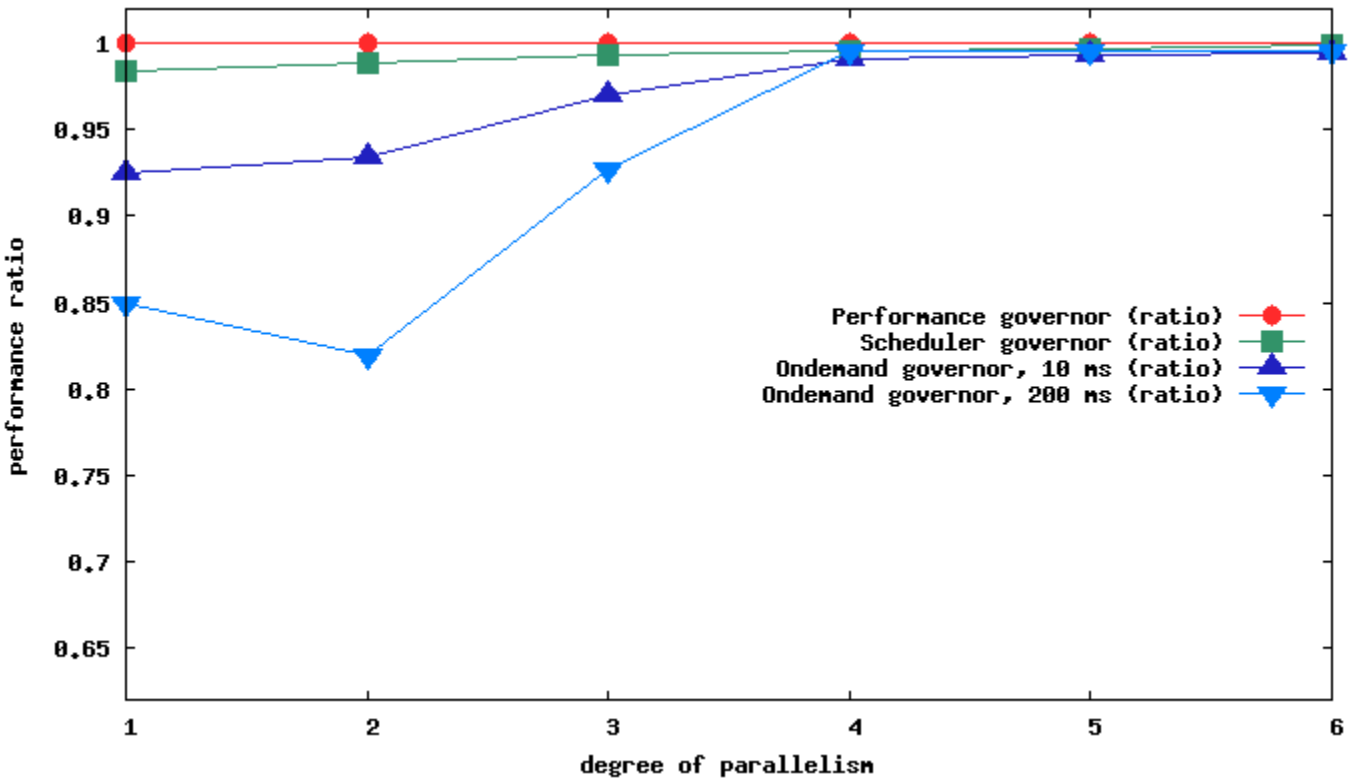
Image source: AMD

Evaluation: Quad AMD Opteron 8435



Governor	Run-Time	Power	Energy	Power over Idle	Energy over Idle
Performance	1:02:59 h	313 W	329 Wh	33 W	35 Wh
Scheduler	1:04:56 h	300 W	324 Wh	20 W	21 Wh
Ondemand (10ms)	1:14:58 h	294 W	368 Wh	14 W	18 Wh
Ondemand (200ms)	1:37:32 h	292 W	475 Wh	12 W	20 Wh
Powersave	3:08:40 h	285 W	898 Wh	5 W	17 Wh

Evaluation: AMD Phenom 9950



Governor	Run-Time	Power	Energy	Power over Idle	Energy over Idle
Performance	1:13:26 h	133 W	163 Wh	49 W	60 Wh
Scheduler	1:14:33 h	120 W	150 Wh	36 W	45 Wh
Ondemand (10ms)	1:19:43 h	117 W	156 Wh	33 W	44 Wh
Ondemand (200ms)	1:26:26 h	116 W	167 Wh	32 W	46 Wh
Powersave	2:18:19 h	107 W	246 Wh	23 W	53 Wh

Summary

- Integration of scheduler and frequency governor
- Event-driven frequency control

- Implementation based on Linux

- Reduced energy consumption
- No performance degradation
- Improved interactive behavior

- Promising foundation for future extensions

Future Work

- Event-driven governor
  - Further evaluation of parameters
  - Adaption of scheduling in space
    - Concentration of load
  - Energy efficiency by considering application behavior
  - Coordinated control of P-states and C-states

- Further approaches
  - Consideration of Intel’s turbo mode
  - Consideration of SMT