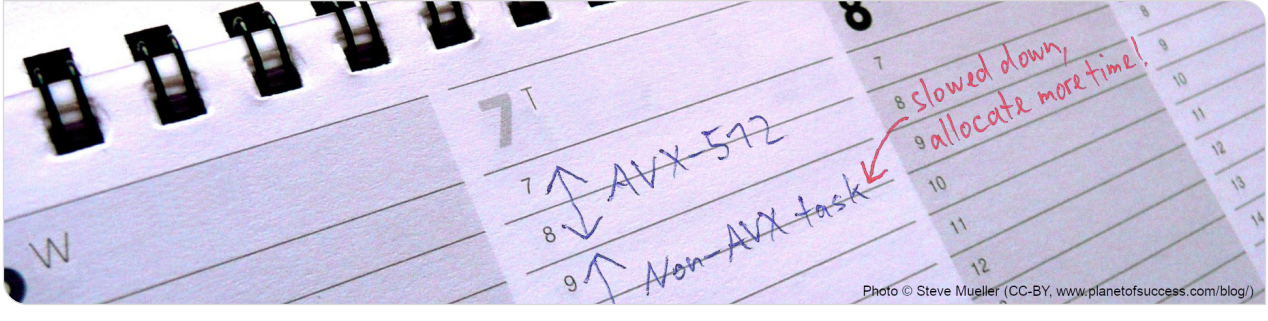
Karlsruhe Institute of Technology

Fair Scheduling for AVX2 and AVX-512 Workloads

Mathias Gottschlag, Philipp Machauer, Yussuf Khalil, Frank Bellosa | July 16, 2021



AVX2/AVX-512 and Fairness

- Scheduling should be “fair”
  - Performance isolation
- Existing fair schedulers: Equal CPU time

- Power-intensive instructions reduce frequency
- Example: Intel AVX2/AVX-512
- *Other* tasks affected, fairness impacted
  - Equal CPU time ⇒ throughput reduced
  - Often more than 15% unfairness

- Need to rethink fair scheduling
- **This work: Frequency reduction compensation**
- **Unfairness reduced by 4x**

AVX2/AVX-512 and Fairness

- Scheduling should be “fair”
  - Performance isolation
- Existing fair schedulers: Equal CPU time

- Power-intensive instructions reduce frequency
- Example: Intel AVX2/AVX-512
- *Other* tasks affected, fairness impacted
  - Equal CPU time ⇒ throughput reduced
  - Often more than 15% unfairness

- Need to rethink fair scheduling
- **This work: Frequency reduction compensation**
- **Unfairness reduced by 4x**

AVX2/AVX-512 and Fairness

- Scheduling should be “fair”
  - Performance isolation
- Existing fair schedulers: Equal CPU time

- Power-intensive instructions reduce frequency
- Example: Intel AVX2/AVX-512
- *Other* tasks affected, fairness impacted
  - Equal CPU time $\Rightarrow$ throughput reduced
  - Often more than 15% unfairness

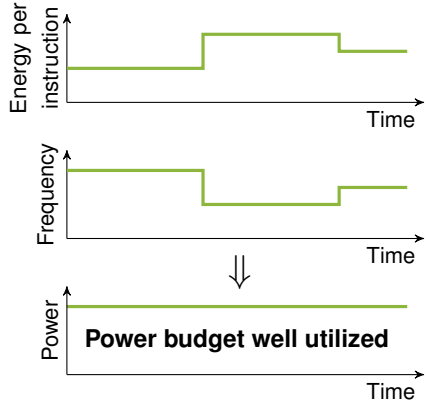
- Need to rethink fair scheduling
- **This work: Frequency reduction compensation**
- **Unfairness reduced by 4x**

Power-Limited Computing

- Restricted transistor scaling
  - $\Rightarrow$ higher power density, “dark silicon”
- Performance commonly limited by power
- Maximize frequency within power limits
  - Example: Turbo Boost
- Power depends on instructions
- Higher frequencies for “simple” instructions

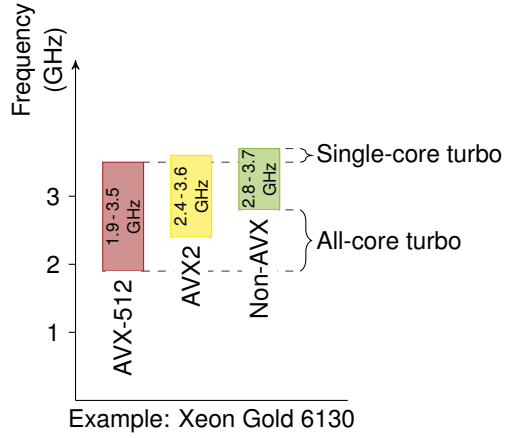
Power-Limited Computing

- Restricted transistor scaling
  - $\Rightarrow$ higher power density, “dark silicon”
- Performance commonly limited by power
- Maximize frequency within power limits
  - Example: Turbo Boost
- Power depends on instructions
- Higher frequencies for “simple” instructions



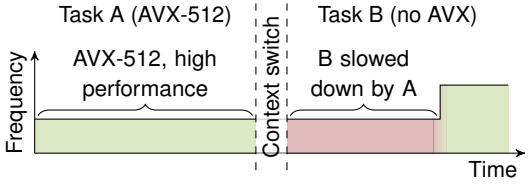
Example: AVX2/AVX-512

- Intel: AVX2/AVX-512
- SIMD instructions for 256/512-bit vectors
- Large vectors = high power
- Three frequency levels

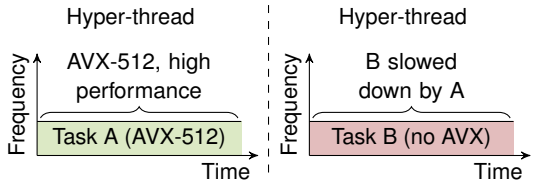


Unfairness in Existing Schedulers

- Frequency reduction affects other less power-intensive code...
  - ... after context switches:



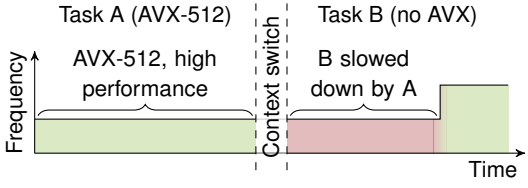
  - ... due to hyper-threading:



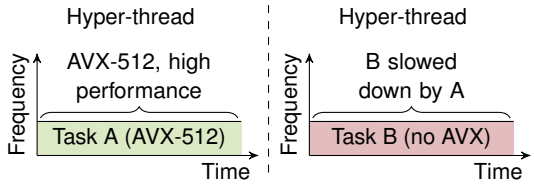
- Previously: Equal CPU time = equal relative performance
  - Also: Good performance isolation
- **Here: Task B slowed down (unfair)**

Unfairness in Existing Schedulers

- Frequency reduction affects other less power-intensive code...
  - ... after context switches:



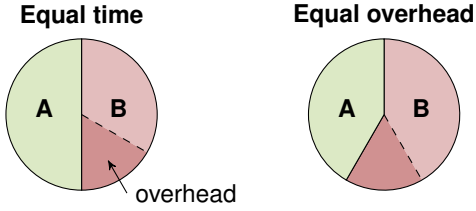
  - ... due to hyper-threading:



- Previously: Equal CPU time = equal relative performance
  - Also: Good performance isolation
- **Here: Task B slowed down (unfair)**

Rethinking Fairness

- Example: A causes frequency reduction, B is slowed down
- Proposal: Equal overhead ⇒ “relative performance”

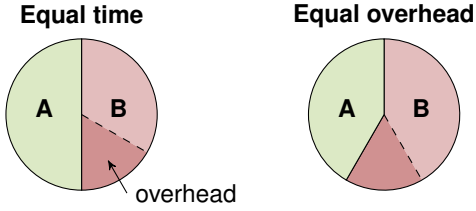


- Energy as primary metric?
  - Energy modelling difficult

  ⇒ Proposal: Modify time-based fair schedulers

Rethinking Fairness

- Example: A causes frequency reduction, B is slowed down
- Proposal: Equal overhead ⇒ “relative performance”

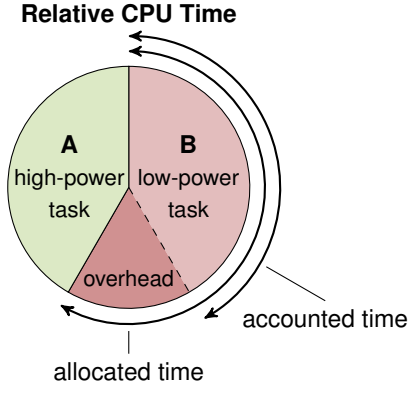


- Energy as primary metric?
  - Energy modelling difficult

  ⇒ Proposal: Modify time-based fair schedulers

Frequency Reduction Compensation

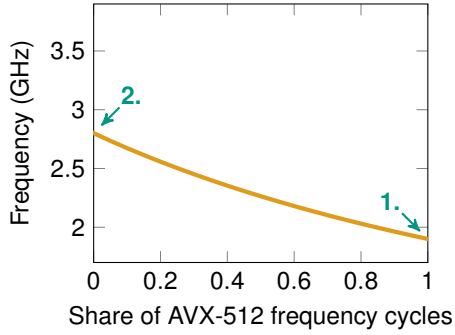
- Modify CPU time accounting, subtract overhead
- Existing scheduler ensures fair performance



Estimating the Frequency Drop

- Scale accounted CPU time – but how much?
- Ratio between ideal and actual frequency
- Non-AVX task...
  1. ... at AVX-512 frequency: $f_{AVX-512}/f_{non-AVX}$
  2. ... at non-AVX frequency: 1
- Count cycles with performance counters

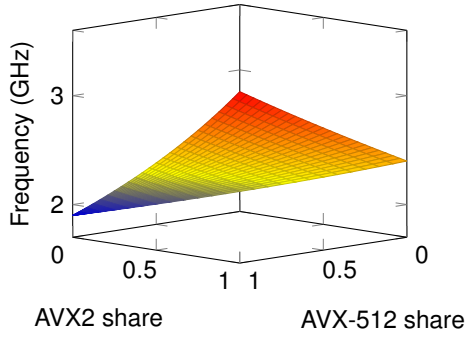
- Complications:
  - Three frequency levels
  - Turbo Boost levels
- See the paper for details



Estimating the Frequency Drop

- Scale accounted CPU time – but how much?
- Ratio between ideal and actual frequency
- Non-AVX task...
  1. ... at AVX-512 frequency: $f_{AVX-512} / f_{non-AVX}$
  2. ... at non-AVX frequency: 1
- Count cycles with performance counters

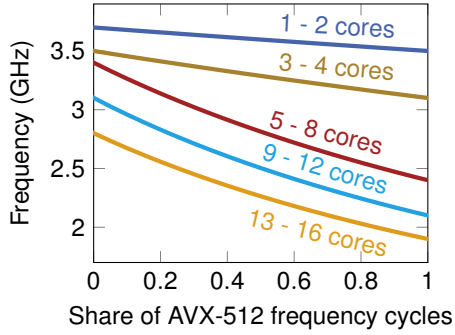
- Complications:
  - Three frequency levels
  - Turbo Boost levels
- See the paper for details



Estimating the Frequency Drop

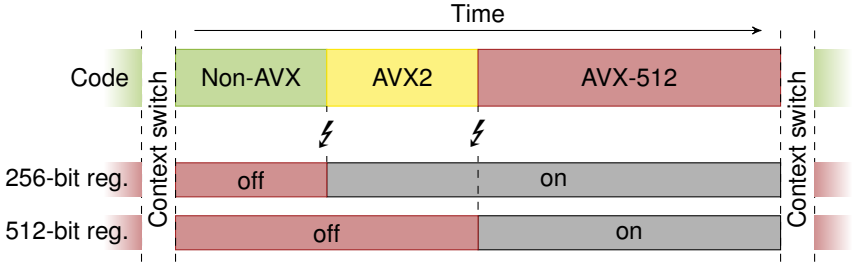
- Scale accounted CPU time – but how much?
- Ratio between ideal and actual frequency
- Non-AVX task...
  1. ... at AVX-512 frequency: $f_{AVX-512}/f_{non-AVX}$
  2. ... at non-AVX frequency: 1
- Count cycles with performance counters

- Complications:
  - Three frequency levels
  - Turbo Boost levels
- See the paper for details



Detecting AVX2/AVX-512

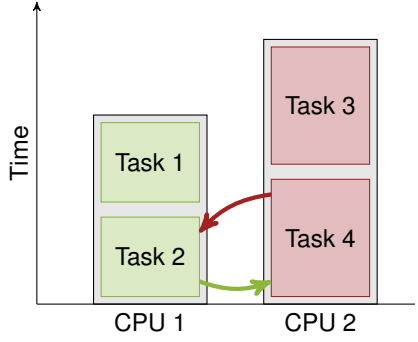
- Only scale CPU time if task did not cause frequency reduction
- In our case: Detect AVX2/AVX-512 instructions
- Disable registers to cause traps



- **Low frequency, registers *not* used ⇒ Apply compensation**

Implementation

- Time progresses at a non-constant rate
- Fairness between tasks on “fast” vs. “slow” CPUs?
  - ⇒ Sometimes, task migration needed
- CFS load balancing too slow
- Our prototype: MuQSS modified to use CFS algorithm
- Performance similar

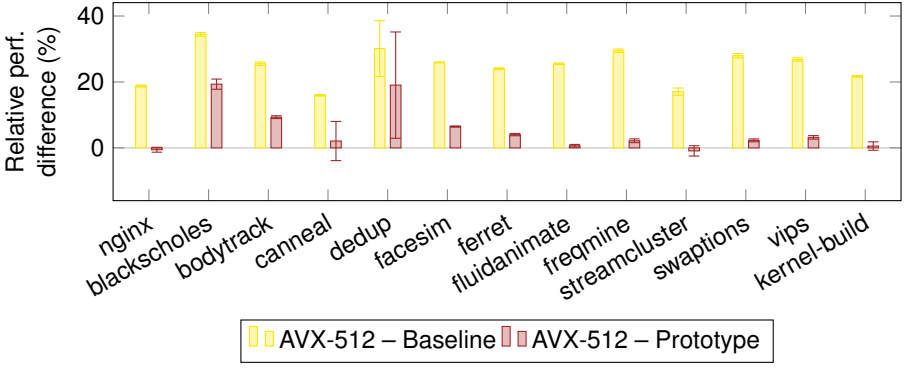


Evaluation Setup

- Questions:
  - Sensible approach to improve fairness?
  - Additional overhead introduced?

- Intel Xeon Gold 6130
  - All-core-turbos between 2.8 GHz (non-AVX) and 1.9 GHz (AVX-512)
- Diverse set of benchmarks: Parsec, nginx web server, PTS linux-kernel-build
- Executed alongside x265 which uses AVX2/AVX-512
- Calculate unfairness and overhead from completion times

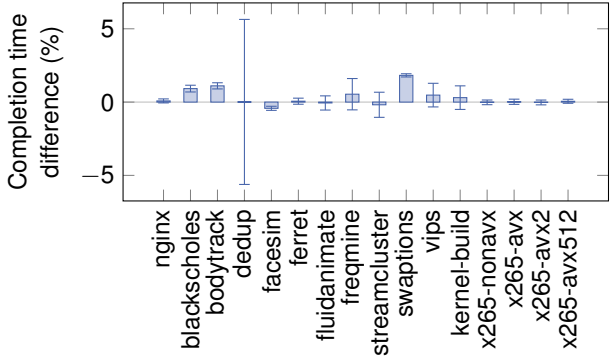
Remaining Unfairness (AVX-512)



- **Reduction from 24.9% unfairness to 5.4% unfairness**

Overhead

- Baseline: Remove frequency reduction compensation code with `#ifdef`



- **For most benchmarks, statistically insignificant overhead**

Summary

- Complex instructions reduce CPU frequencies
- Less power-intensive code also slowed down
- Problem for current fair schedulers

⇒ Need to rethink fair scheduling

- This work: Simple approach to modify existing schedulers
  - Frequency reduction compensation
  - Trap-based detection of affected tasks

- **Result: Unfairness for AVX-512 workload reduced by 4x**