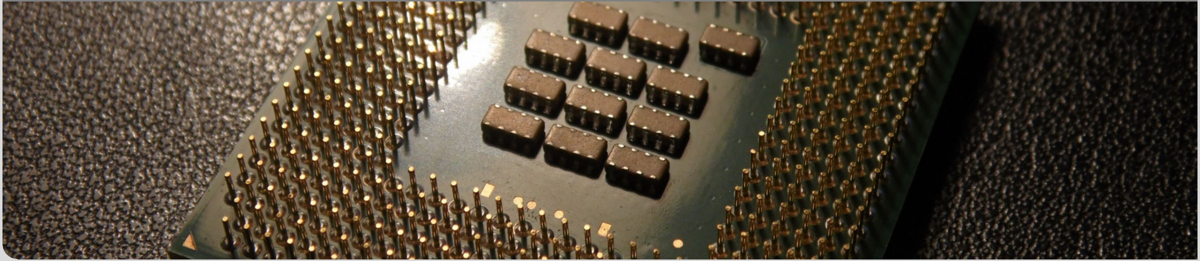
Karlsruhe Institute of Technology

Remote AVX Overhead: Detection and Mitigation

Mathias Gottschlag | March 12, 2021

KARLSRUHE INSTITUTE OF TECHNOLOGY (KIT) - OPERATING SYSTEMS GROUP



Impact of AVX2/AVX-512

Karlsruhe Institute of Technology

- AVX2/AVX-512: SIMD instructions for data parallelism
- 256-bit (AVX2), 512-bit (AVX-512)

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- AVX-512: 2.2x speedup for machine learning
  - Complex, high power dissipation
  - CPU cores reduce their frequency
  - 10%-30% slowdown for applications executed in parallel
- Similar effects in other workloads

- *Remote AVX overhead:* AVX2/AVX-512 slows other code down
- **Today: OS should manage hardware-controlled frequency scaling!**

Aubrey Li: *Core scheduling: Fixing when fast instructions go slow.* LPC'19, Sep. 2019

Impact of AVX2/AVX-512

- AVX2/AVX-512: SIMD instructions for data parallelism
- 256-bit (AVX2), 512-bit (AVX-512)

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- AVX-512: 2.2x speedup for machine learning
  - Complex, high power dissipation
  - CPU cores reduce their frequency
  - 10%-30% slowdown for applications executed in parallel
- Similar effects in other workloads

- *Remote AVX overhead:* AVX2/AVX-512 slows other code down
- **Today: OS should manage hardware-controlled frequency scaling!**

Aubrey Li: *Core scheduling: Fixing when fast instructions go slow.* LPC'19, Sep. 2019

Impact of AVX2/AVX-512

- AVX2/AVX-512: SIMD instructions for data parallelism
- 256-bit (AVX2), 512-bit (AVX-512)

- AVX-512: 2.2x speedup for machine learning
  - Complex, high power dissipation
  - CPU cores reduce their frequency
  - 10%-30% slowdown for applications executed in parallel
- Similar effects in other workloads

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

- *Remote AVX overhead:* AVX2/AVX-512 slows other code down
- **Today: OS should manage hardware-controlled frequency scaling!**

Aubrey Li: *Core scheduling: Fixing when fast instructions go slow.* LPC'19, Sep. 2019

Power-Limited Computing

- Modern CPUs: Limited by power dissipation
- Thermal headroom = wasted performance
- Usually: Select frequency close to power limits

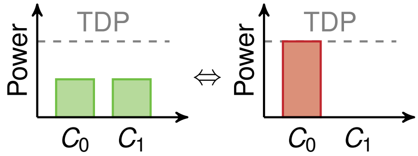
- Traditional techniques:
  - Turbo Boost: Higher frequency when some cores idle
  - Computational sprinting: Higher frequency when heatsink is cold

- Per-core power limits
- Intructions differ in their power dissipation
- “Simple” code has more thermal headroom

$\Rightarrow$ **Increase frequency!**

Power-Limited Computing

- Modern CPUs: Limited by power dissipation
- Thermal headroom $=$ wasted performance
- Usually: Select frequency close to power limits



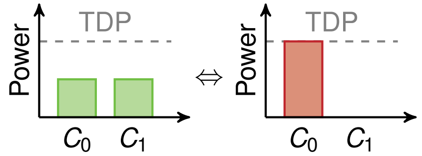
- Traditional techniques:
  - Turbo Boost: Higher frequency when some cores idle
  - Computational sprinting: Higher frequency when heatsink is cold

- Per-core power limits
- Intructions differ in their power dissipation
- “Simple” code has more thermal headroom

$\Rightarrow$ **Increase frequency!**

Power-Limited Computing

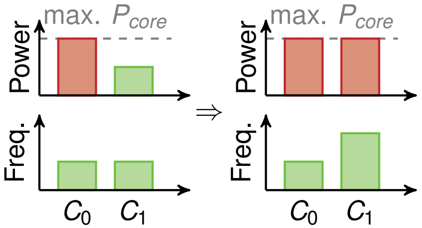
- Modern CPUs: Limited by power dissipation
- Thermal headroom $=$ wasted performance
- Usually: Select frequency close to power limits



- Traditional techniques:
  - Turbo Boost: Higher frequency when some cores idle
  - Computational sprinting: Higher frequency when heatsink is cold

- Per-core power limits
- Intructions differ in their power dissipation
- “Simple” code has more thermal headroom

$\Rightarrow$ **Increase frequency!**

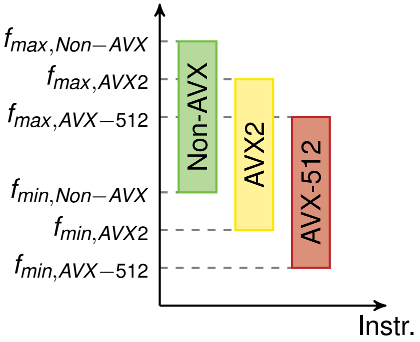


Power-Limited Computing

- Intel CPUs:
  - Low frequency for AVX-512 code
  - Intermediate frequency for AVX2
  - High frequency for non-AVX code

⇒ All code fully utilizes available power

- Optimization to speed up “simple” code (+30%)



- Future CPUs will remain power-limited

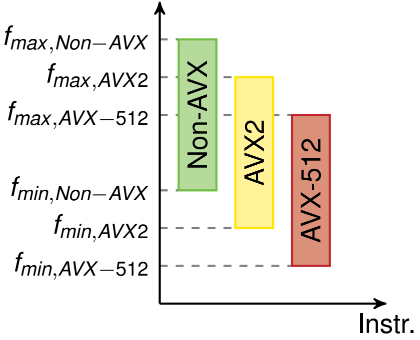
⇒ **Following effects increasingly visible on other CPUs**

Power-Limited Computing

- Intel CPUs:
  - Low frequency for AVX-512 code
  - Intermediate frequency for AVX2
  - High frequency for non-AVX code

⇒ All code fully utilizes available power

- Optimization to speed up “simple” code (+30%)

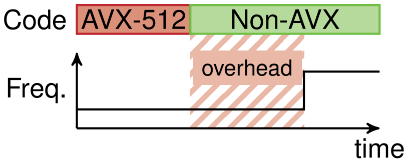


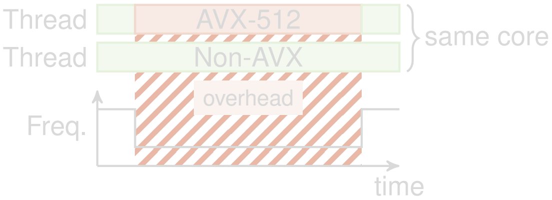
- Future CPUs will remain power-limited

⇒ **Following effects increasingly visible on other CPUs**

Remote AVX Overhead

- Frequency reduction affects other code





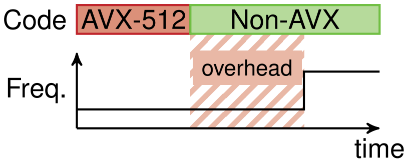
- Example:
  - AVX-512-enabled OpenSSL + nginx
  - 10% slowdown

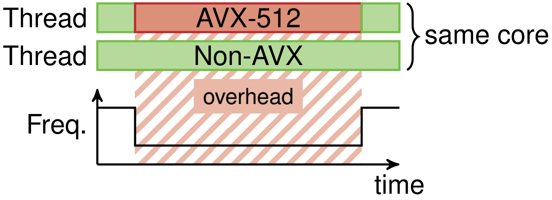
- Example:
  - Tasks executed in parallel to AVX-512 ML task
  - Tasks executed in parallel to AVX-512 video encoder
  - 10%-30% slowdown

Local speedup, remote slowdown

Remote AVX Overhead

- Frequency reduction affects other code





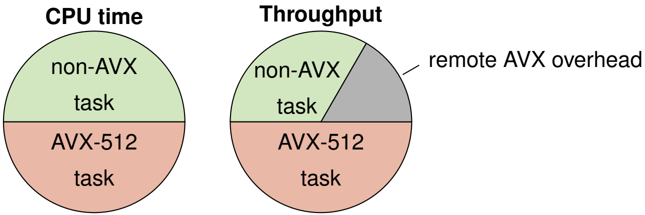
- Example:
  - AVX-512-enabled OpenSSL + nginx
  - 10% slowdown

- Example:
  - Tasks executed in parallel to AVX-512 ML task
  - Tasks executed in parallel to AVX-512 video encoder
  - 10%-30% slowdown

Local speedup, remote slowdown

Remote AVX Overhead

- Example: Typical system with fair scheduler



- Overall system performance reduced
- Unfair: Some tasks receive less performance

Why not just disable AVX2/AVX-512?

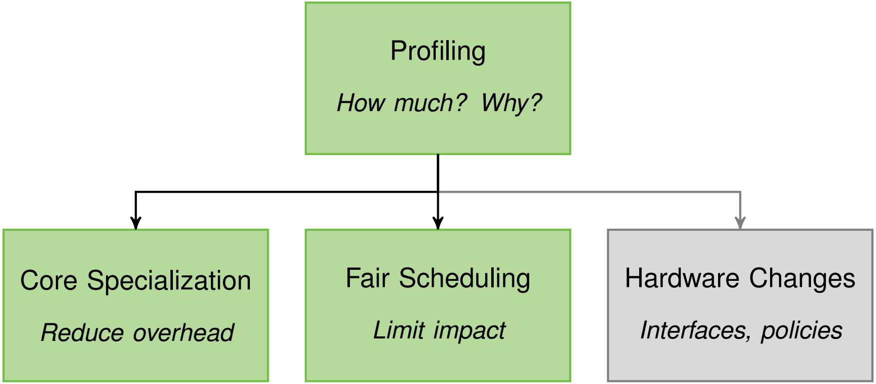
- Local decision, global impact
  - Sometimes positive, sometimes negative

- Caused by interaction at runtime
  - Hard to predict during software development
  - No information about other tasks at runtime

- **Proper location to solve these problems is in the OS**

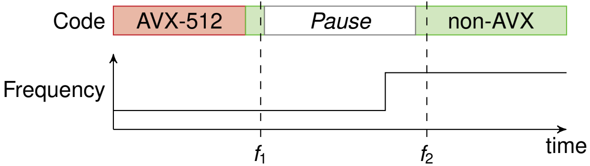
Toolbox for AVX Frequency Management

- Various techniques to mitigate remote AVX overhead



Profiling

- Question: Is there substantial remote AVX overhead?
- Problem: Differentiation from *local* AVX overhead
  - Local: AVX2/AVX-512 code is affected
  - Remote: Code can execute at higher frequency
- Approach: Periodic sampling

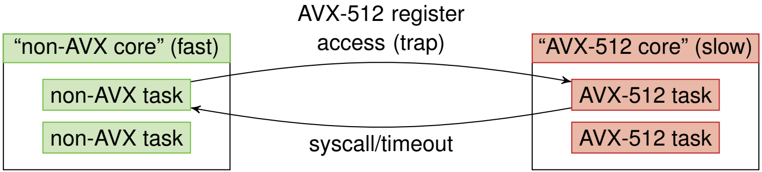


- **Calculate overhead from f_1 / f_2 (error only 1.2 percentage points)**

Gottschlag et al.: *AVX Overhead Profiling: How Much Does Your Fast Code Slow You Down?* APSys'20, Aug. 2020

Core Specialization

- *Early work presented at the fall meeting 2018*
- Observation: Problems caused by co-scheduling
- Idea: Restrict co-scheduling of AVX-512 and other code



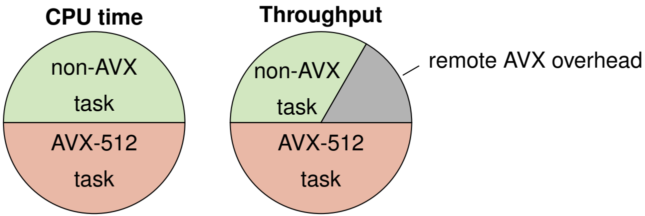
- Only AVX-512 cores execute AVX-512
- Rarely any non-AVX-512 tasks on AVX-512 cores

⇒ **Impact on non-AVX-512 code reduced by 70%**

Gottschlag et al.: *Automatic Core Specialization for AVX-512 Applications*. SYSTOR’20, Oct. 2020

Fair Scheduling

- Sometimes, remote AVX overhead cannot be mitigated
  - Need to prevent idle cores, no precise detection of “problematic” instructions
- At least restrict impact on other threads
- Existing schedulers: Fair allocation of CPU time



- **CPU time not a suitable metric for throughput!**

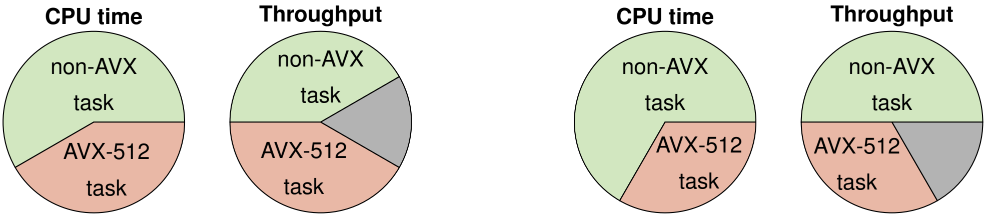
Gottschlag et al.: *Fair Scheduling for AVX2 and AVX-512 Workloads*. Submitted.

Fair Scheduling

- Scale CPU time according to remote AVX overhead
- How much?

a) Fairness

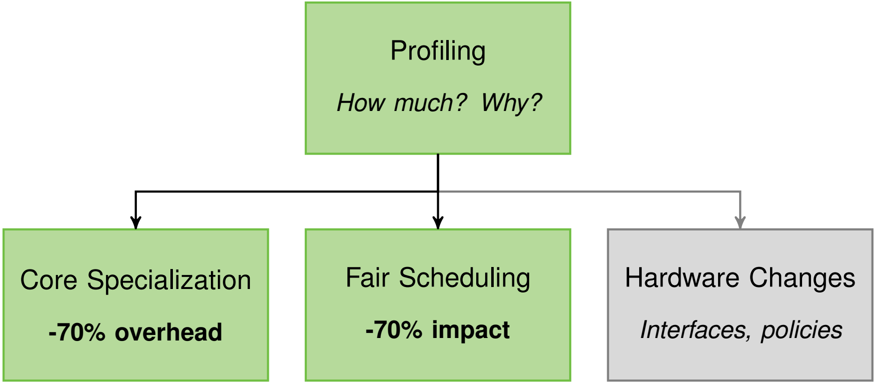
b) Performance Isolation



- Prototype based on custom scheduler

⇒ **Performance impact reduced by 70%**

Toolbox for AVX Frequency Management



- **OS should manage hardware-controlled DVFS!**

Future Hardware

- Prototypes suffer from lack of information, overhead
- Sensible improvements for future CPUs?

Current Hardware	Improvement
Information on *current* frequency	Information about *required* frequency $\Rightarrow$ easier accounting
Cannot detect/prevent energy-intensive instructions	Exception *before* reducing the frequency $\Rightarrow$ better scheduling
Delay before restoring frequency	Allow OS to increase frequency $\Rightarrow$ better DVFS policies

- **Empower the OS**

Future Hardware

- Prototypes suffer from lack of information, overhead
- Sensible improvements for future CPUs?

Current Hardware	Improvement
Information on *current* frequency	Information about *required* frequency $\Rightarrow$ easier accounting
Cannot detect/prevent energy-intensive instructions	Exception *before* reducing the frequency $\Rightarrow$ better scheduling
Delay before restoring frequency	Allow OS to increase frequency $\Rightarrow$ better DVFS policies

- **Empower the OS**

Future Hardware

- Prototypes suffer from lack of information, overhead
- Sensible improvements for future CPUs?

Current Hardware	Improvement
Information on *current* frequency	Information about *required* frequency $\Rightarrow$ easier accounting
Cannot detect/prevent energy-intensive instructions	Exception *before* reducing the frequency $\Rightarrow$ better scheduling
Delay before restoring frequency	Allow OS to increase frequency $\Rightarrow$ better DVFS policies

- **Empower the OS**

Conclusion

- AVX2/AVX-512 frequencies affect other (e.g., non-AVX) code
- Fundamental problem of power-limited computing

- This work:
  - Tools to measure and mitigate *remote AVX overhead*
  - Impact often reduced by more than 70%
- Hardware changes can improve efficacy

- **OS should manage hardware-controlled DVFS!**