Automatic Core Specialization for AVX-512 Applications

Mathias Gottschlag, Peter Brantsch, Frank Bellosa | October 13, 2020
Effects of AVX-512

- AVX-512: SIMD instructions for data parallelism

- AVX-512 speeds up Poly1305 MAC
  ⇒ web server slowed down by 10% if AVX-512 is used

- AVX-512 can speed up machine learning by up to 2.2x
  ⇒ Applications running in parallel run 10% slower

- This talk: How to prevent this slowdown?

Vlad Krasnov: On the dangers of Intel's frequency scaling. Cloudflare, Nov. 2017
Aubrey Li: Core scheduling: Fixing when fast instructions go slow. LPC’19, Sep. 2019
Effects of AVX-512

- AVX-512: SIMD instructions for data parallelism

- AVX-512 speeds up Poly1305 MAC
  ⇒ web server slowed down by 10% if AVX-512 is used

- AVX-512 can speed up machine learning by up to 2.2x
  ⇒ Applications running in parallel run 10% slower

This talk: How to prevent this slowdown?

---

Vlad Krasnov: On the dangers of Intel's frequency scaling. Cloudflare, Nov. 2017

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

M. Gottschlag, P. Brantsch, F. Bellosa – Automatic Core Specialization for AVX-512 Applications
Effects of AVX-512

- AVX-512: SIMD instructions for data parallelism
- AVX-512 speeds up Poly1305 MAC
  ⇒ web server slowed down by 10% if AVX-512 is used
- AVX-512 can speed up machine learning by up to 2.2x
  ⇒ Applications running in parallel run 10% slower

This talk: How to prevent this slowdown?

---

Vlad Krasnov: On the dangers of Intel’s frequency scaling. Cloudflare, Nov. 2017

Aubrey Li: Core scheduling: Fixing when fast instructions go slow. LPC’19, Sep. 2019
AVX Frequency Reduction

- Complex SIMD instructions cause momentary high power consumption
  ⇒ Result: High power *variability*

- Power is limited (heat, voltage drops)
  ⇒ Different max. frequencies possible depending on instructions

- **Intel**: Particularly low frequency for AVX-512 code!
AVX-512 Overhead

- Frequency reduction affects non-AVX-512 code

- Local speedup, global slowdown?
- Hard to predict, sometimes changing at runtime

⇒ Solution to prevent slowdown at runtime
Core Specialization

- Idea: Spatial separation

<table>
<thead>
<tr>
<th>“non-AVX core” (fast)</th>
<th>“AVX-512 core” (slow)</th>
</tr>
</thead>
<tbody>
<tr>
<td>non-AVX task</td>
<td>AVX-512 task</td>
</tr>
<tr>
<td>non-AVX task</td>
<td>AVX-512 task</td>
</tr>
</tbody>
</table>

- Result: Only “AVX-512 cores” slowed down

⇒ Reduced frequency impact
Implementation

- Categorization of tasks

- AVX-512 instruction (trapped)

  - Non-AVX task
  - AVX-512 task

  system call

- Non-AVX tasks allowed on AVX-512 cores
  - But: Prioritize AVX-512 tasks

- More details in the paper.

M. Gottschlag, P. Brantsch, F. Bellosa – Automatic Core Specialization for AVX-512 Applications
Implementation

- Categorization of tasks

AVX-512 instruction (trapped)

Non-AVX task \[\rightarrow\] AVX-512 task \[\rightarrow\] system call

- Non-AVX tasks allowed on AVX-512 cores
  - But: Prioritize AVX-512 tasks

- More details in the paper.
Evaluation

- CPU time for heterogeneous workloads
- Usage of AVX-512 configurable

**Two-program workloads**

(Parsec + x265)

<table>
<thead>
<tr>
<th>Workload</th>
<th>No AVX2/AVX-512</th>
<th>AVX-512 – Baseline</th>
<th>AVX-512 – Core Specialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>nginx+openssl</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>blackscholes</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>fluidanimate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>swaptions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vips</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>x264</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Original: 11.3% overhead due to AVX-512
- **Our approach:** 3.4%
Discussion

- Traps detect *all* 512-bit register accesses
  - Impossible to detect “energy-intensive” instructions
  - Better hardware/software interface?

- Missing: NUMA support
  - Prevent migration between NUMA domains

- Missing: Automatic allocation of AVX-512 cores
  - Number of cores based on load

⇒ Future work
Summary

- AVX-512 slows other code down
  - 10% overhead reported for several scenarios
  - Impact hard to predict

- Contribution: Scheduler modification to reduce slowdown
  - Core specialization
  - Intercept AVX-512 instructions
  - Restrict AVX-512 code to AVX-512 cores

- Evaluation: Slowdown reduced to 3.4% (was: 11.3%)