The Effects of Wide Vector Operations on Processor Caches

Andrei Poenaru
Simon McIntosh-Smith

University of Bristol

https://uob-hpc.github.io
Background: HPC at Bristol

• At Bristol we work with a variety of HPC architectures
  • Arm: Marvell TX2, Fujitsu A64FX
  • x86: Intel Cascade Lake, AMD Rome
  • GPU: NVIDIA, AMD, Intel

• Two kinds of studies:
  • Between architectures, e.g. performance portability
  • Deep-dive in a single architecture
    • Arm Scalable Vector Extension (SVE) is one of the most interesting new platforms
Motivation: Studying SVE

• The latest generations of Arm HPC processors now use SVE
  • Fujitsu A64FX in Fugaku, Isambard 2, ...

• We began work years before hardware was available
  • Aim is to prepare to get a rolling start
  • Helps with toolchain maturity
  • Mostly emulation, some simulation

• SVE offers a choice of vector lengths
  • It is important to understand the effects of this choice
Motivation: Investigating Cache Behaviour

• Modern processors have different cache configurations
  • There are many design choices to make

• Good caching is critical for performance
  • ...and bad caching can be disastrous [1]

• Vector architectures have different requirements and cause different effects
  • SVE supports many vector lengths, adding another problem dimension [2]
Methodology: Cache Simulator

- Problem: existing tools are either ISA-/microarchitecture-specific or completely generic
  - SVE support in early stages

- Solution: lightweight in-house simulator
  - Based on memory traces from Arm Instruction Emulator (ArmIE)
    - ArmIE gives SVE support “for free”
    - Other trace formats can be converted
  - SVE is supported, but applicability is much wider

https://uob-hpc.github.io
Methodology: Benchmarks

- Benchmarks: mini-apps from most common classes of HPC codes
  - CloverLeaf – structured-grid fluid dynamics
  - MegaSweep – memory bandwidth benchmark based on SNAP
    - Originally developed to investigate caching-related effects
  - MiniFMM – C++ task-based fast multipole method

- Restrictions:
  - Single core, 2 levels of private cache, no prefetching
  - Single thread, no I/O, instrumentation restricted to kernel
Simulator Validation

• Metrics validated against hardware counter data
  • 128 bits vs TX2, 512 bits vs A64FX
  • Benchmarks: STREAM, BUDE, CloverLeaf, MegaSweep, MiniFMM

• < 10% difference between simulator data and hardware counters

• Single-core inputs chosen to maintain performance characteristics of real-world, full-node test cases
  • Bar a scaling factor, results are proportional
Results: Cache Misses

• We explored cache miss rates for hypothetical configurations
  • Line size: 64–2048 bytes
  • Associativity: 2–32 lines/set

• Increasing line size almost always helped
• Increasing set size did not always help on its own
  • But longer lines benefited from—and sometimes required—larger sets
## Cache Misses: MegaSweep and MiniFMM

### MegaSweep

<table>
<thead>
<tr>
<th>set size (lines)</th>
<th>line size (bytes)</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5.61</td>
<td>5.61</td>
<td>5.48</td>
<td>5.46</td>
<td>5.44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2.88</td>
<td>2.88</td>
<td>2.81</td>
<td>2.81</td>
<td>2.81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1.51</td>
<td>1.49</td>
<td>1.45</td>
<td>1.45</td>
<td>1.46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.88</td>
<td>0.79</td>
<td>0.77</td>
<td>0.77</td>
<td>0.78</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>1.16</td>
<td>0.48</td>
<td>0.43</td>
<td>0.43</td>
<td>0.43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>2.40</td>
<td>0.46</td>
<td>0.24</td>
<td>0.24</td>
<td>0.24</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### MiniFMM

<table>
<thead>
<tr>
<th>set size (lines)</th>
<th>line size (bytes)</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.23</td>
<td>0.16</td>
<td>0.15</td>
<td>0.15</td>
<td>0.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.16</td>
<td>0.10</td>
<td>0.09</td>
<td>0.09</td>
<td>0.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.12</td>
<td>0.06</td>
<td>0.05</td>
<td>0.05</td>
<td>0.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.14</td>
<td>0.04</td>
<td>0.03</td>
<td>0.03</td>
<td>0.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>0.24</td>
<td>0.04</td>
<td>0.03</td>
<td>0.02</td>
<td>0.02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>0.55</td>
<td>0.21</td>
<td>0.07</td>
<td>0.03</td>
<td>0.03</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results: Cache Lifetimes

• Taking the cache configurations used in the TX2 and A64FX, we compared how long data stays in cache before being evicted
  • The data was collected empirically, at run-time (cf. stack distance)
  • We scaled the vector width to observe differences

• The longer lines of A64FX and larger overall size led to more reuse
  • The difference increased with vector width
Cache Lifetimes: CloverLeaf

![Graphs showing cache lifetimes for different svewidth values.](https://uob-hpc.github.io)
Cache Lifetimes: MegaSweep

https://uob-hpc.github.io
Results: Non-Contiguous Accesses

- Performance of an operation often depends on number of cache lines touched
- At 1024-bit vectors (and above) the longer cache lines of the A64FX configuration serviced non-contiguous requests using 3X fewer cache lines
- A number of accesses were spread far enough apart that they always spanned multiple cache lines
Challenges and Improvements

• A lot of room for expansion:
  • Pre-fetching
  • More types of cache
  • More policies

• Multi-core simulations are more interesting, but implementation details are not always public
  • Some cache parameters can be guessed using micro-benchmarks
  • A64FX architecture manual is public
  • Dimensionality increases even further...
  • Memory trace size becomes a very important consideration

https://uob-hpc.github.io
Further Work: SimEng

• We are building SimEng ("Simulation Engine"), a flexible and accurate simulation toolkit designed specifically with HPC processors in mind
  • SVE micro-architecture design-space exploration is our first goal

• We already have accurate (Δ<10%) models of TX2 and A64FX cores
  • Work is underway to enable multi-core and memory hierarchy support
  • Easily extendable to future generations of these processors
Conclusion

• The cache hierarchy presents a multi-dimensional challenge
  • The optimal configuration is closely related to other design choices
  • It is not always a case of increasing capacity

• Vector length directly impacts cache usage
  • With SVE, different lengths prefer different configurations

• Advanced vector instructions benefit from particular configurations
References


Thank You

- Bristol HPC Homepage: https://uob-hpc.github.io/
  - Publications: https://uob-hpc.github.io/publications/
- Reproducibility: https://github.com/UoB-HPC/cache-effects-reproducibility
- Benchmarks: https://github.com/UoB-HPC/benchmarks
- Performance Portability: https://github.com/UoB-HPC/performance-portability