Flexible vectors use cases

Why go beyond 128-bit SIMD

Petr Penzin

Intel

August 1, 2023

Agenda

- Performance expectations
- Targeted hardware
- Software use cases

Theory

- Increase in vector size leads to increase in performance
  - Performing 2x more operations per cycle leads to 2x speedup, 4x to 4x speedup, etc
  - Realistically that it is an upper bound (Amdahl's law, etc)
- Actual gain depends on algorithm, but for easily parallelizeable it is going to be substantial
  - Think element-wide addition of two arrays

Hardware support

Popular hardware extensions:

- AVX (x86)

  supported by virtually all PCs, 128-bit instructions already supported by Wasm runtimes
- SVE (Arm)
- AVX512 (x86)

First order focus on AVX and emerging extensions.

Some practical considerations

- There is a relationship between number of elements in the vector, "complexity" of the algorithm, and resulting performance
- Relationship between vector size and performance is not linear for some algorithms
- Faster execution often allows to do "more work"
- Relatively small gain can be enough to make an algorithm usable

Software

Accelerating existing use cases, GEMM and friends:

- Machine learning
- Graphics
- Physics

Software

Accelerating existing use cases, GEMM and friends:

- Machine learning
- Graphics
- Physics

Expanding use cases:

- Larger value types (double percision)
- Text processing / pattern matching
- Cryptography / off-browser cases

GEMM and basic algebra

Matrix multiplication and basic algebra more broadly cover important existing use cases:

- Machine Learning [1]
- Graphics [2]
- Other linear algebra applictions, (example: game physics)

Matrix operations are generally succeptible to SIMD parallelization. For 256-bit operations it is safe to assume double digit performance gains, more for wider extensions.

[1] Chellapilla et al 2006, Dukhan 2019

[2] Sobel operator

Higher precision

- 128-bit SIMD only processes two double precision or 64-bit integer values at a time
- Only straight-forward kernels would be able to have substantial speedup over scalar
- 256-bit SIMD instructions process four 64-bit lanes, a lot more to work with

For current 'hard' cases we can expect more than 2x speedup for 256-bit operations.

Expanding into new algorithms

Examples:

- Pattern matching [3]
- Cryptography, compression
- Signal processing

[3] HyperScan

[4] RLBox, shipping in Firefox since 2020

[5] WASM as a portable IR for apps (Adnroid NDK)

[6] MPIWasm, Chadha et al, PPoPP 2023

Expanding into new algorithms

Examples:

- Pattern matching [3]
- Cryptography, compression
- Signal processing

Considerations:

- In general present more steps
- 128 bit SIMD implementations tend to have low performance gains over scalar
- Some important non-Web use cases

  Examples: sandboxing[4], portability[5], virtualization[6]

Need to do a better estimate, rough estimate would be about 2x for 256-bit SIMD.

[3] HyperScan

[4] RLBox, shipping in Firefox since 2020

[5] WASM as a portable IR for apps (Adnroid NDK)

[6] MPIWasm, Chadha et al, PPoPP 2023

Future work?

See tracking issue.