Vectorization Quality

Georg Zitzlsberger

georg.zitzlsberger@vsb.cz

IT4Innovations
national supercomputing
center

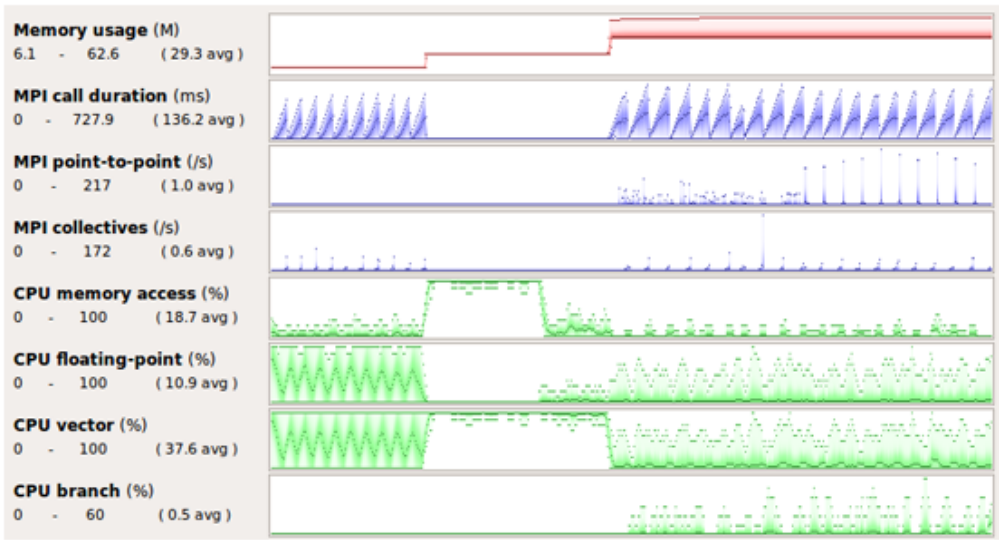
5th of July 2017

Agenda

- How to Measure Vectorization Quality?
- Why Intel Advisor
- Vectorization Efficiency
- Memory Access Patterns
- Roofline Model
- Roofline Model - In Practice
- Analyzing Non-Executed Code
- Which Intel Advisor Data is Important?

How to Measure Vectorization Quality?

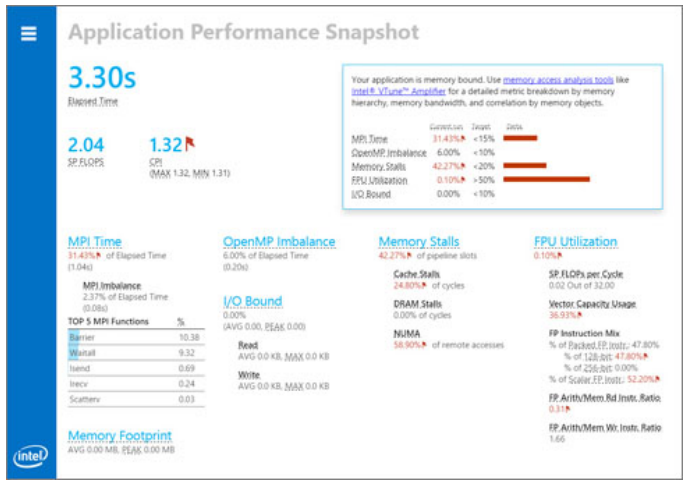
- Look at the assembly
  ⇒ Requires in-depth knowledge of underlying architecture
- Using profiling tools like *GNU perf* (see here) or *Intel VTune Amplifier XE*
  ⇒ "FLOP" related event counters might not work correctly
- *Performance API* (PAPI) (see here)
  ⇒ Requires PAPI to be implemented in the application
- *Allinea MAP* can collect FLOPS
  ⇒ Uses PAPI underneath and easy to use



(Image: Allinea)

How to Measure Vectorization Quality?

- Quotient method: Back-to-back comparison of SIMD and non-SIMD version
  ⇒ Only very rough information but good for beginning
- Performance summary with ▸ Intel® Performance Snapshot:
  Application Performance Snapshot (now contains MPI Performance Snapshot)



(Image: Intel)

  ⇒ Only very high level information, but scales
- So-called "Vector Tool" from *Intel Advisor*
  ⇒ Easy to use and no changes needed

Why Intel Advisor

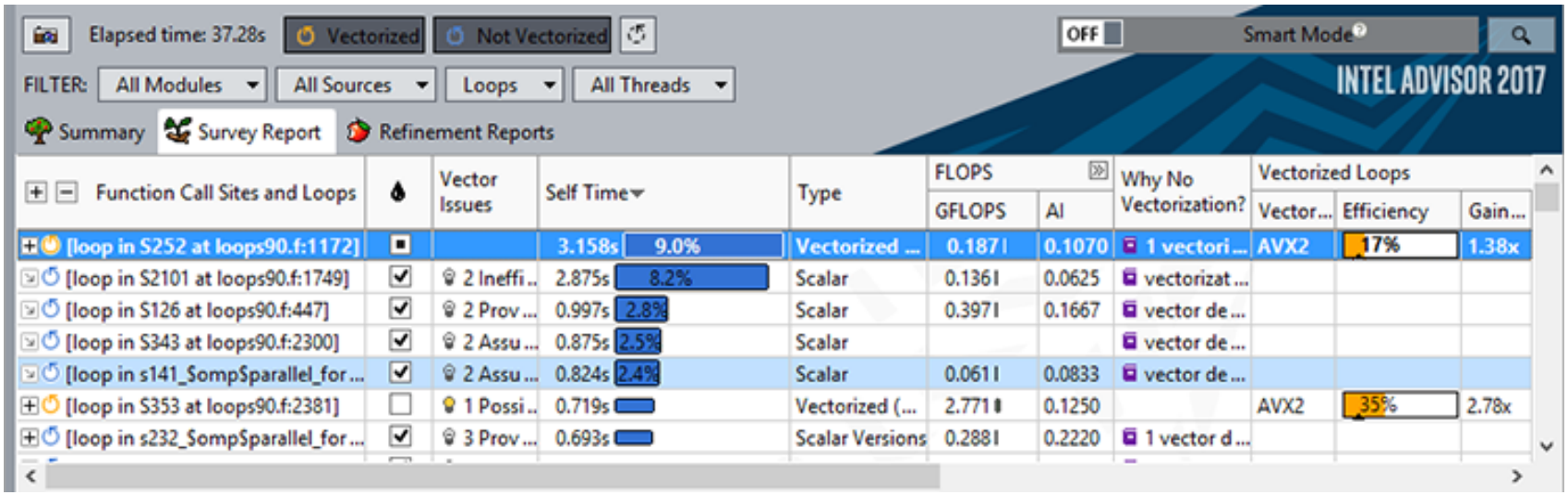
- Today we only focus on the "Vector Tool" part of Advisor
- It helps to measure the quality (efficiency) of vectorization
- It combines Intel compiler optimization reports with dynamic analysis
- It identifies the data access patterns of loops
- It can also statically analyze non-executed code (e.g. for other SIMD extensions)

Key Metrics:

- Identify the vectorization efficiency
- Memory access patterns
- Get arithmetic intensity vs. Performance (FLOPS)

Vectorization Efficiency

- Computes FLOPS (very precisely) and arithmetic intensity
- Informs about SIMD extension and FP types used for loops
- Estimation of speedup due to vectorization
- Estimation of efficiency (potential room for improvement?)
- Shows loop trip counts
- Provides information about vectorization carried out by the compiler (using also Intel compiler optimization reports)
- Gives hints how to vectorize (common problems)

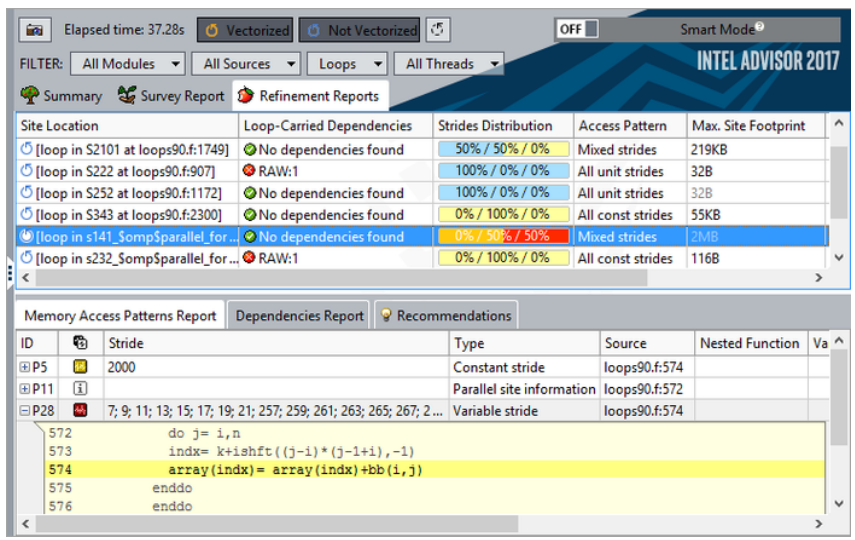


(Image: Intel)

Memory Access Patterns

- Three different patterns are distinguished:
  - **Unit strides:**
    Contiguous memory accesses
    ⇒ Best performance and recommended
  - **Constant strides:**
    Sequential but non-contiguous memory accesses
    ⇒ Not optimal but can be mitigated by prefetching
  - **Mixed strides/random:**
    Worst case because not always predictable
    ⇒ Maybe mitigated by gather/scatter instructions

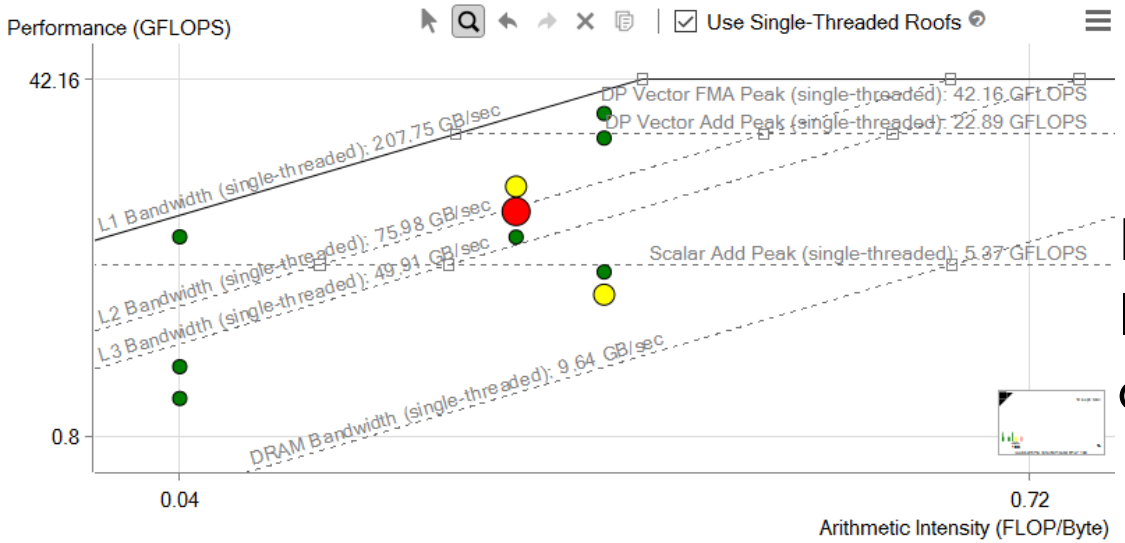
- Analysis is done per loop
- Stride information including memory footprint



(Image: Intel)

Roofline Model

- The Roofline model plots loops characterized by their arithmetic intensity and performance in FLOPS:
  - **Arithmetic intensity:** Ratio of FLOP to bytes needed from memory - lower means more memory for a FLOP. Limited by algorithm and optimizations.
  - **Performance:** Measured in FLOPS and limited (so-called roof) by architecture. There are multiple roofs depending on the levels of memory hierarchy.
- It helps to identify and track the performance of loops



More information about Intel Advisor's Roofline can be found here

(Image: Intel)

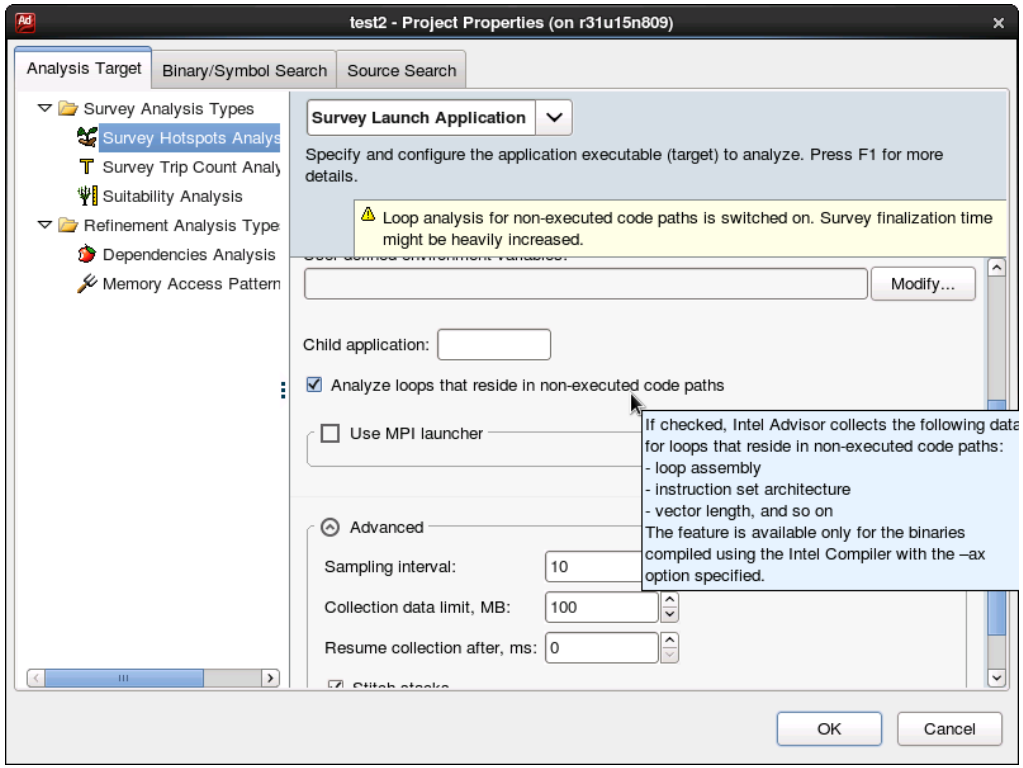
Roofline Model - In Practice

The following data from the Roofline model helps you:

- Qualitatively compare different implementations
  $\Rightarrow$ Increase arithmetic intensity first
- Don't focus on small (green) dots, they are cold loops
  $\Rightarrow$ Address big (red) dots as those are the hot loops
- When optimizing a loop, track it via the Roofline model
  $\Rightarrow$ It should move towards top right
- For well tuned loops, see whether their roof is close to which cache level
  $\Rightarrow$ Is it close to the roof of the cache level the data reuse was optimized for?

Analyzing Non-Executed Code

- Useful for multi-version code (e.g. `-axCORE-AVX512`) - works for Intel Compiler only
- Static information instead of dynamic one
- Informs what the compiler generated and which instructions were used



Which Intel Advisor Data is Important?

The following information from Intel Advisor could be important for you:

- Which SIMD extension was effectively used?
  - Speedups of both Intel Advisor (*Gain Estimate*) and Intel Compiler (*Compiler Estimated Gain*) are reported
  - Vector length (*VL*) is not always tied to SIMD type (VL can be shorter!)
  - Efficiency of vectorization ($\frac{Gain\ Estimate}{VL}$)
- *Code Analytics* tab per loop shows where time was spent (computation, memory accesses or other)
- GFLOP & GFLOPS (latter varies due profiling overhead)
- *Memory Access Patterns* (unit stride, non-unit stride and random/scatter/gather)
- Roofline model (AI & GFLOPS) - also track over time during development

Lab Time!