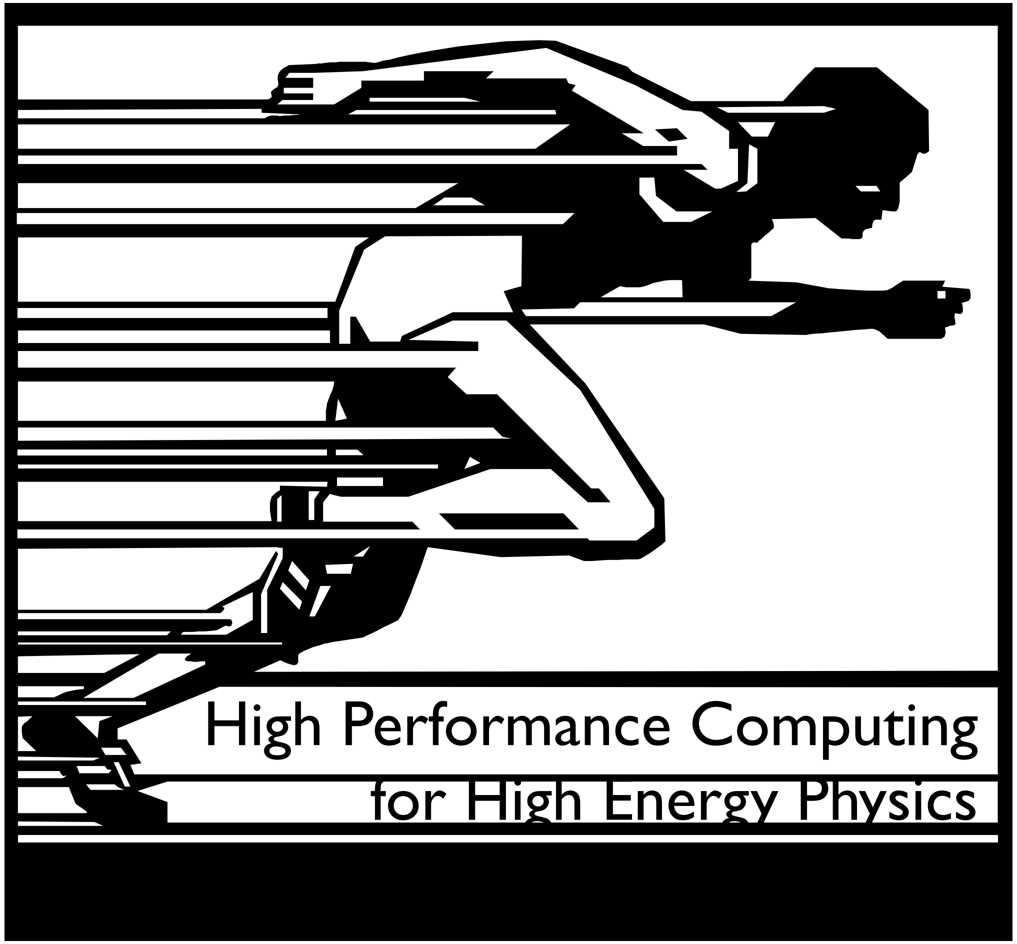
# Haswell Conundrum: AVX or not AVX?




Concurrency Forum
CERN, June `14

Vincenzo Innocente
CERN/SFT
CMS Experiment

# Summary

- Haswell is a great new Architecture:
  - » Not because of AVX
- Long SIMD vectors are worth only for intensive vectorized code
  - » Are not GPUs then a better option?
- Power Management cannot be ignored while assessing computational efficiency
- On modern architecture, extrapolation based on synthetic benchmarks is mission impossible

# Motivations

- Haswell is the latest INTEL architecture for laptop, workstations and Servers
  - Workstations are available since Jan 2013. Servers will be available in Sep 2014
  - It sports, among other features, AVX2 and advanced power management
  - It promised, w/r/t SB and IVB, up to a factor ~2 in performance in particular for a full box or w/r/t power consumption.
- For LHC Run2, CMS plans to deliver half of its HLT computing power in form of Haswell Xeon servers

- Here we will report expectations for Haswell-server performance based on tests made on high-end HSW- SB- workstations and SB- IVB-Xeon servers.

# Power Management

- Till the advent of SandyBridge "TurboBoost" is not just an option to boost single threaded applications
- On a IVB E5-2695 v2 @ 2.40GHz (12+12 core)
  - Single core cpu-bound goes @ 3.15 GHz
  - 48 cpu-bound processes go **@ 2.7GHz**
- IVB core on same node still coupled
  - Memory-bound process: any between 2.1 and 2.7 GHz (with clear effect on memory access latency)
  - Stabilize at ~3.0GHz when a cpu-bound job is run in parallel
- HSW core are decoupled (tests on i7-4770K CPU @ 3.50GHz)
  - Single core cpu-bound goes @ 3.9 GHz
  - 8 cpu-bound processes go @ 3.7 GHz
  - Memory-bound goes @3.85 GHz (not affected by a cpu-bound job aside)
    - (takes ~ the same number of reference cycles as on IVB. But 3.5GHz vs 2.4 !)

# AVX: myth and reality

- Myth
  - » AVX will speed up scientific application by a factor 2 (w/r/t SSE)
- Truth
  - » A factor ~2 possible only for linear algebra in L1 cache
- Facts
  - » Memory Bandwidth is what it is
  - » AVX is de facto a twin SSE
  - » div/sqrt is implemented only in SSE (latency is twice for 256-wide vectors)
  - » Instructions are longer: affects front-end latency
  - » ILP and various pipeline effects often dominate over vector width
- Any loop accessing memory beyond L1/L2 and/or performing double precision div/sqrt or permutations will not fully profit of AVX

# Wide vectors and branch-predication

– In strict SIMD, ALL branches are evaluated sequentially for all elements of the vector and results eventually "blended"
  - » All branches should be valid for any element
  - » Slowest branch will dominate even if "rare"
– Global conditions on the whole vector (voting) can mitigate the effect of rare slow conditionals
  - » less and less effective wider is the vector
– Cuda, since long time, supports a variety of instructions to deal with this kind of issues at high-level
– AVX512 is introducing new sets of instructions to manipulate wide vectors
– Nothing exists in C/C++ even not in OpenCL, OpenMP etc.
  - » Hard to see seamless solution in the medium term

# Wide vectors and branch-predication

– Correctly rounded vector elementary function

– CPE = Cycles per element

| Description | arch. | CPE |
|---|---|---|
| VCR log | SSE3 | 35.34 |
| VCR log | AVX | 21.81 |
| VCR log | AVX2 | 17.98 |
| VCR log | Xeon-Phi | 45.03 |
| VCR exp | SSE3 | 29.98 |
| VCR exp | AVX | 20.99 |
| VCR exp | Xeon-Phi | 63.1 |

# AVX and power management

- AVX consumes power (difficult to find official numbers…)
- Higher-half is usually off (leaving standard SSE on)
  - Requires few 100 cycles to start. Seems to stay on "long"
  - Its hysteresis seems longer than CPU frequency (t.b.c.)
- On Haswell AVX competes for power with other components
  - AVX On ➔ CPU slow-down
- Consequences:
  - Compiling with AVX, or even just using a handful of AVX-256 instructions at runtime, will most probably make your program globally slower

Blog https://twiki.cern.ch/twiki/bin/view/LCG/VIHaswell

# BENCHMARKS

# Scimark2

- Scimark2 contains 5 benchmarks http://math.nist.gov/scimark2/about.html
  - » Fast Fourier Transform (FFT)
    - Does not vectorize
  - » Jacobi Successive Over-relaxation (SOR)
    - Does not vectorize
  - » Monte Carlo integration
    - Sensitive to inlining
  - » Sparse matrix multiply
    - Vectorize with "gathering" instruction
  - » dense LU matrix factorization
    - Partially vectorize
- Comes in two versions
  - » Small (in cache), large (out of cache)
  - » On current hardware only FFT goes out of L2 cache for large version
- Very small kernels: sensitive to compiler heuristics
  - » Performance can be further boosted by fine-tuning (20% easy), not necessarily representing feasible options for large applications

# Scimark2 (MFlops, larger is better)

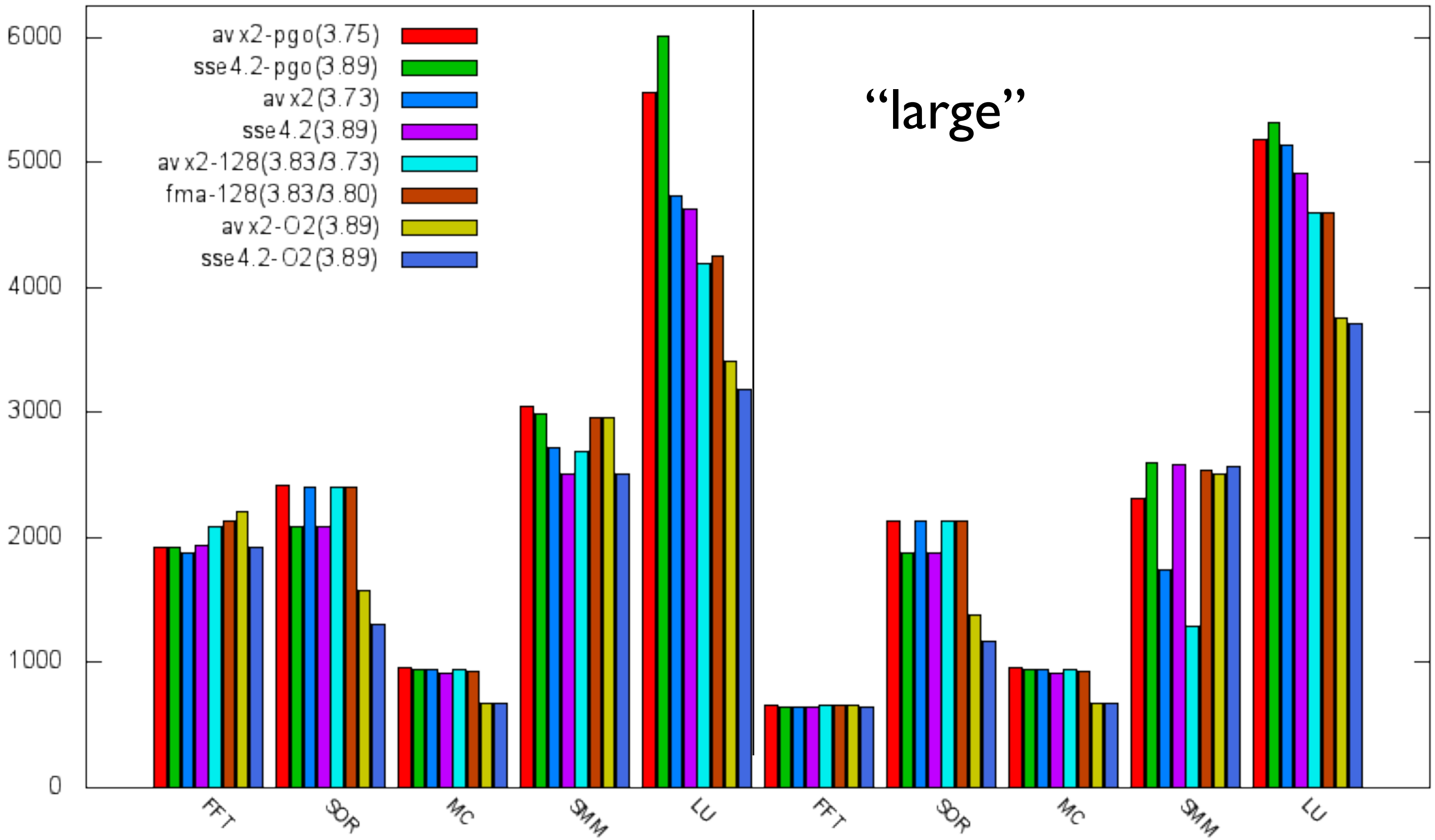
All run on HSW 4770K, compiled with gcc 4.9 (pgo=profile-guided-opt)

avx2 = -march=hashwell -Ofast

sse4.2 = -march=nehalem –Ofast

avx2-128 = -mavx2 -mprefer-avx128 –Ofast (No higher-half of avx)

fma-128 = -mavx -mfma -mprefer-avx128 –Ofast (no gather)



# ParallelPdf (VIFIT)

**ParallelPdf** is a *RooFit-like* multi-threaded prototype that evaluates the gradient of a Log-Likelihood for a given number of *events*.

- Presented last year in this forum
  - including comparison among SB/HSW/Atom and Numa effects
- it can run either evaluating all pdfs each time or reading the values of a pdf from a cache if its parameters did not change.
- For not too small number of events it is essentially dominated by memory access (L3-cache critical).
- Numerical computation is dominated by the evaluation of transcendental functions in double precision that implies at least one division and few conditions each (vdt is used).

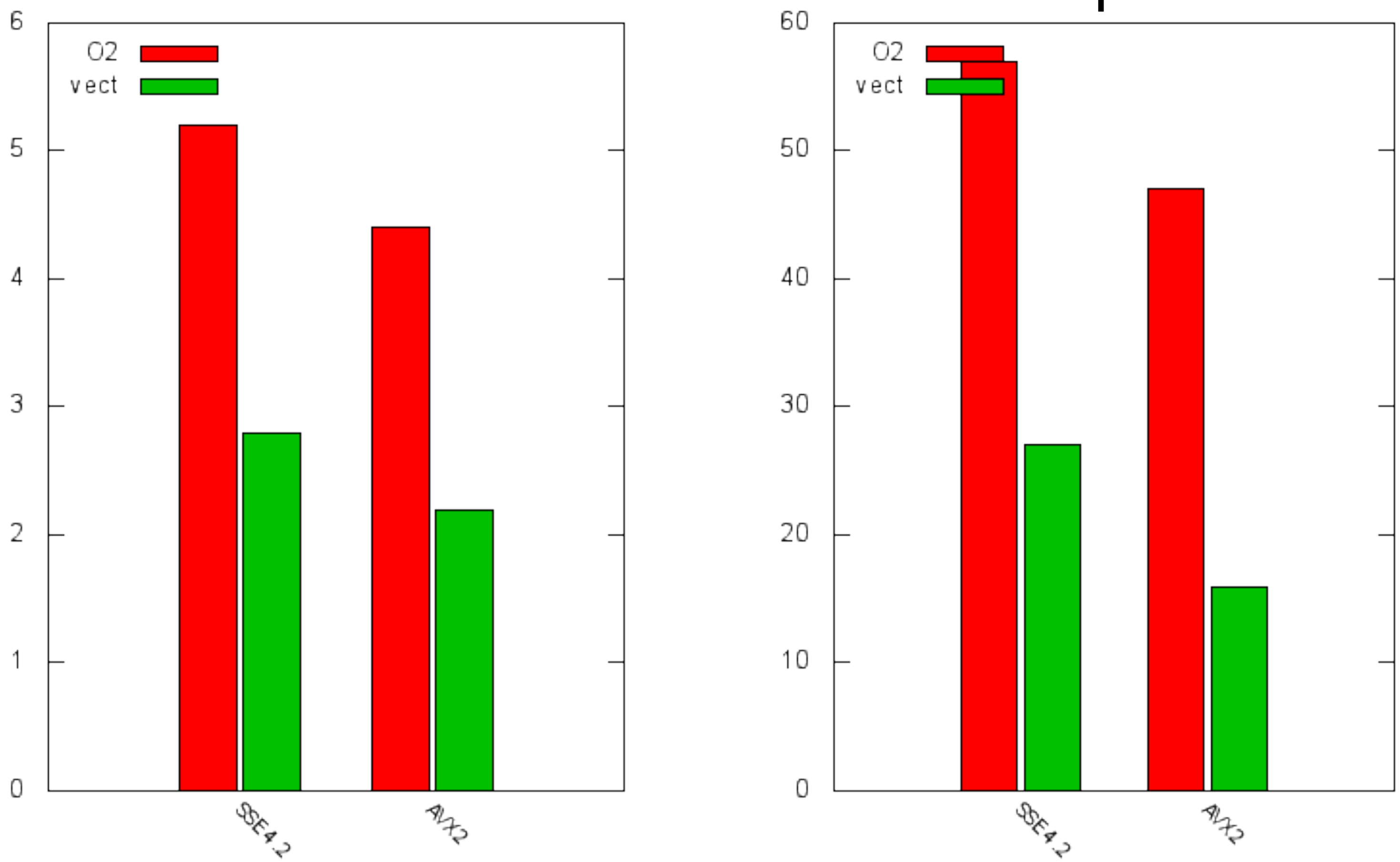
- It is heavily multi-threaded and vectorized
  - Known to scale extremely well with core number

# ParallelPdf: timing on HSW 4770K

Absolute time in seconds (200K events, 200 times)
Smaller is better!
4 threads. SSE4.2 @3.7GHz, AVX2 @3.45GHz

use caches

Eval all pdfs



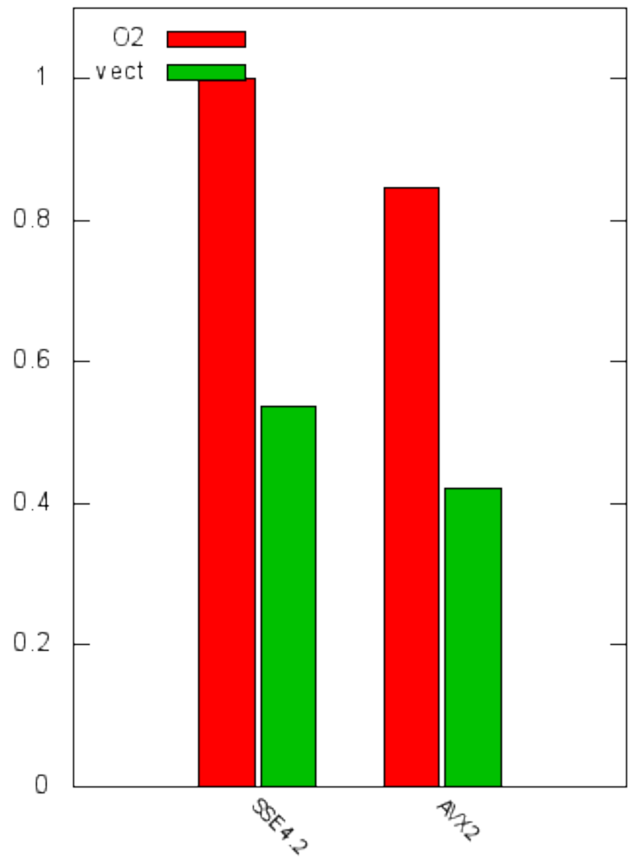
# ParallelPdf: normalized timing

Time normalized to SSE4.2 -O2

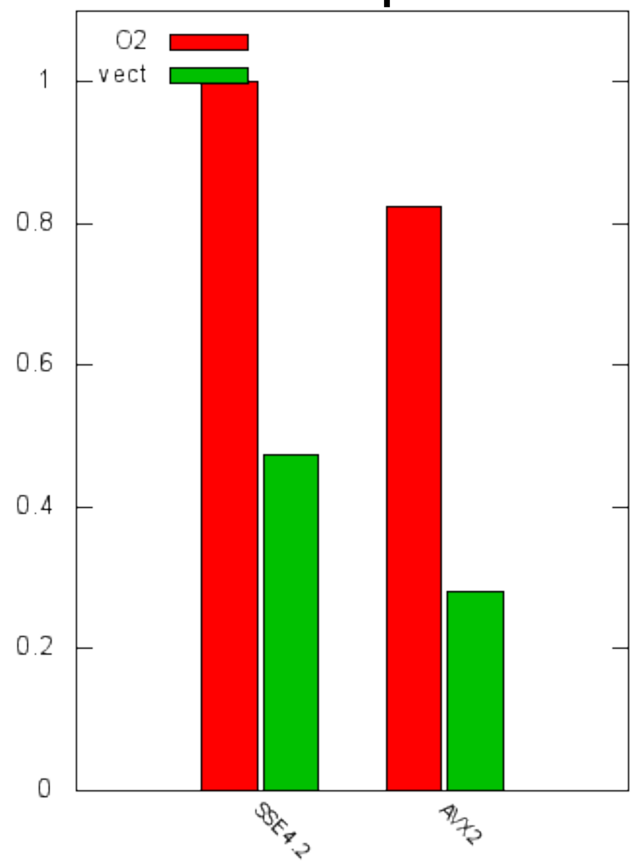
Smaller is better!

4 threads. SSE4.2 @3.7GHz, AVX2 @3.45GHz

use caches




Eval all pdfs



# CMSSW (Reco and HLT)

- Standard build (-O2, -ftree-vectorize, SSE3)
  - » HSW up to 25% faster than SB for selected modules
  - » In average 15-20% faster than SB
- Native build (-O2, -ftree-vectorize, -march=haswell)
  - » Between 2% and 4% slower than Standard build

# CMSSW (Reco 620: Igprof details)

| producer | SB | | | HW |
|---|---|---|---|---|
| | sse | sse | avx | avx2 |
| total | 275918 | 214294 | 220113 | 218826 |
| cms::CkfTrackCandidateMakerBase | 106244 | 76419 | 80907 | 78750 |
| VirtualJetProducer (fastjet: sse) | 23799 | 21075 | 20673 | 20704 |
| SeedGeneratorFromRegionHitsEDProducer | 20619 | 15348 | 16138 | 16749 |
| TrackProducer | 19607 | 14511 | 14891 | 14091 |
| ConversionTrackCandidateProducer | 15816 | 12346 | 13198 | 13082 |
| MuonIdProducer | 14797 | 11848 | 12879 | 12929 |
| GsfTrackProducer | 8876 | 7294 | 5396 | 5308 |
| PrimaryVertexProducer | 4157 | 3267 | 3233 | 2922 |
| ConversionProducer | 4200 | 3396 | 2984 | 3990 |
| PFECALSuperClusterProducer | 3291 | 2730 | 2732 | 2706 |
| RecoTauProducer | 3072 | 2417 | 2703 | 2702 |
| EcalUncalibRecHitProducer | 2690 | 2845 | 2634 | 2908 |
| PFClusterProducer | 2722 | 2656 | 2579 | 2387 |
| PFBlockProducer | 3098 | 2514 | 2561 | 2540 |

# Conclusions

- Free lunch is over
  - In 2 years the computational power of Intel workstations has increased by 30% max (including core count and freq-boost)
  - For servers even less
- Power management affects individual components:
  - Achieving maximal throughput requires to make choices among features to activate
- Memory wall is higher than ever
  - HSW improves on instruction caching though..
- Wide SIMD vectors are effective only for highly specialized code
- Little support for this new brave world in generic high level languages and libraries