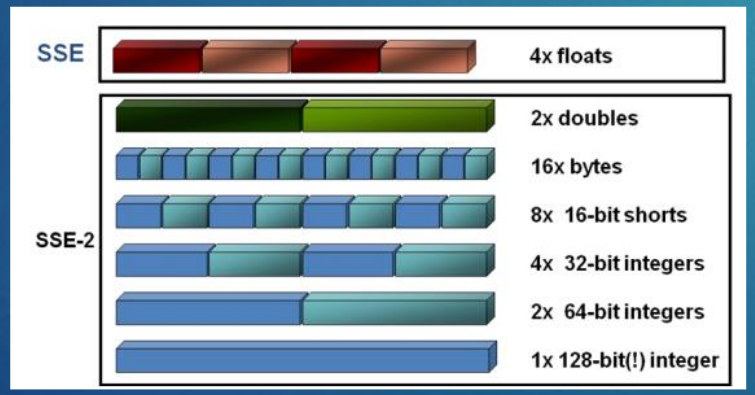
Manual vectorization @ LHCb trigger

A CASE STUDY

Vectors 101

- Vectorization is an old concept
  - Streaming SIMD Extensions (SSE) – Pentium III, 1999
  - SSE2 – 2001, SSE3, SSE4.1, SSE4.2, AVX, AVX2

SSE – 128-bit-wide registers

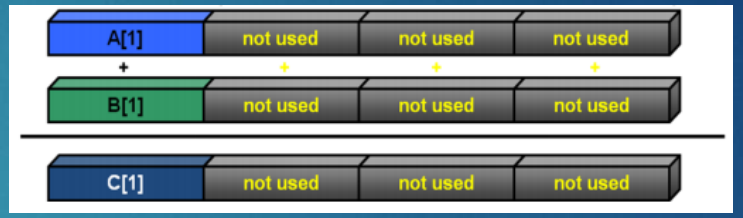


AVX – 256-bit wide registers
AVX2 – 512-bit

LHCb ONLINE

CPU does SIMD – Use it!

- Ye oldie CPU is capable of doing it



Q1'2010

```
model name      : Intel(R) Xeon(R) CPU X5650  @ 2.67GHz
...
flags           : fpu vme de pse tsc msr pae mce cx8 apic
sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr
sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm
constant_tsc arch_perfmon pebs bts rep_good xtopology
nonstop_tsc aperfmperf pni dtes64 monitor ds_cpl vmx smx
est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 popcnt
lahf_lm ida arat epb dts tpr_shadow vnmi flexpriority ept
vpid
```

Q3'2013

```
model name      : Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz
...
flags           : fpu vme de pse tsc msr pae mce cx8 apic
sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr
sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm
constant_tsc arch_perfmon pebs bts rep_good xtopology
nonstop_tsc aperfmperf pni pclmulqdq dtes64 monitor ds_cpl
vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2
x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand
lahf_lm ida arat epb xsaveopt pln pts dts tpr_shadow vnmi
flexpriority ept vpid fsgsbase smep erms
```

- 128-bit registers are there, expect increasing registers for future generations!

LHCb ONLINE

Why isn't our software already vectorized?

- Auto-vectorization – Perhaps Cilk? Only for specific cases.
- Portability
- Data
  - Data format (the recurrent AoS vs SoA)
  - Data alignment
- Control
  - Branches
- Manpower

LHCb ONLINE

Case study – LHCb trigger

- We may need a case-by-case study, identify hotspots.
- I'm familiar with the VELO Pixel code, there's where I looked.

VELO Pixel - revisited

- In the VELO, tracks are reconstructed by using a square root fit each time a hit is added.
  - Other than that, the fit variables are also utilized for calculating the covariance.

PrPixel algorithm parts	average time
searchByPair setHits, addHits	43 %
searchByPair covariance	0.6 %

It looks very nice for Amdahl's law!

LHCb ONLINE

Analyzing the code (brief)

- SoA, but looks nice. In fact, **better** for tracking vectorization than normal SIMD.
- Parameters from fit can be taken away.
- load / store overhead is not so much.
- Need to convert to float. Clear gain in vectors.

```
void addHits(A* a, B* b){
    const float x = a->x;
    const float y = a->y;
    const float z = a->z;
    const float wx = a->wx;
    const float wy = a->wy;

    ...

    b->m_tx = (b->m_sxz * b->m_s0  - b->m_sx  * b->m_sz) / den;
    b->m_x0 = (b->m_sx  * b->m_sz2 - b->m_sxz * b->m_sz) / den;
    b->m_ty = (b->m_uyz * b->m_u0  - b->m_uy  * b->m_uz) / den2;
    b->m_y0 = (b->m_uy  * b->m_uz2 - b->m_uyz * b->m_uz) / den2;
}
```

Can you spot whether it's vectorizable?

LHCb ONLINE

Unit test results

- SSE performance varies across CPU
- Implemented with intrinsics

CPU	addHits u	addHits a	cov u	cov a
Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz (Q1'10)	2.22 x	2.25 x	2.18 x	2.18 x
Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz (Q3'13)	1.92 x	1.95 x	1.36 x	1.37 x
HLT node - Intel(R) Xeon(R) CPU X5650 @ 2.67GHz (Q1'10)	**1.75 x**	1.77 x	2.06 x	2.10 x

LHCb ONLINE

The **real** scenario – PrPixel Physics

- Results from an unaligned vectorized version
- v45r0 (latest two weeks ago)

Vanilla PrPixel v45r0

```
PrChecker.Velo      INFO **** Velo   42422 tracks including        9502 ghosts
PrChecker.Velo      INFO      velo :  22582 from    30102 [ 75.0 %]   7460 clones [24.8 %]
```

Vector PrPixel v45r0

```
PrChecker.Velo      INFO **** Velo   42427 tracks including        9506 ghosts
PrChecker.Velo      INFO      velo :  22583 from    30102 [ 75.0 %]  7459 clones [24.8 %]
```

LHCb ONLINE

The **real** scenario – PrPixel Performance

Current HLT node

PrPixel	vanilla	vectorized	speedup
prepare	0.026	0.026	1 x
findByPairs	0.858	0.724	1.19 x
store	0.036	0.032	1.13 x
total	0.919	0.783	**1.17 x**

The vectorized version is about **14.6 % faster!**

LHCb ONLINE

More on the results

- Results vary between **1.13 – 1.17x** depending on CPU.
- **No additional includes** were necessary. It's coded in intrinsics.
- X-compiler tests didn't change much.
  - After all, code is not auto-vectorized, and intrinsics map very much the same way to ASM.
- Aligning code to match vectorized implementation (sort of reverse-engineering) didn't help.
- A conversion to float was done, with no visible impact on Physics efficiency (although we should test on bigger datasets).
- AVX would allow a double implementation (but why?)
  - Or perhaps a slightly better vectorized one.

LHCb ONLINE

Aligned?

- Good luck.

- It's tough. Specially in Gaudi.
- I stepped into at least three cases to align:
  - Stack – Override new
  - Heap – Libraries (ie. std::vector can be aligned by changing the allocator)
  - Heap – Attributes (is this solved? Architecture-dependent instructions)
- Unaligned instructions **will benefit** of the performance boost when data is aligned (no performance penalty). They are simply **more flexible**.

Is the gain worth it?

LHCb ONLINE

Other considerations
Agner Fog's library

- Agner Fog provides a higher level library, preferred over intrinsics where possible.
- However my tests were not very successful.
  - 29 lines in C, intrinsics compiled into 50 ASM, Fog's library into 62.

```
Times: seq: 0.0266874, fog_vec: 0.0191835, intrinsics: 0.013965
Speedup (fog_vec vs seq): 1.39117x
Speedup (fog_vec vs intrinsics): 0.727969x
Speedup (intrinsics vs seq): 1.91103x

Results are not identical!

vanilla results:
m_x0 7.50546, m_tx 0.0618172, m_y0 2.81588e+15, m_ty -3.51984e+14

fog_vec:
m_x0 7.50546, m_tx 0.061817, m_y0 2.81591e+15, m_ty -3.51989e+14

intrinsics:
m_x0 7.50546, m_tx 0.0618172, m_y0 2.81588e+15, m_ty -3.51984e+14
```

Some conclusions

- Current PrPixel received a boost of 12-15%. On **current** HW.
- Vectors have been there since long. Use them.
- Sure, we need to use them now.
  - But **manycore** is next stop, and there you need vectors even more.
- We need to rethink our trigger.
  - Identify **hotspots**, discuss.
  - Do small test units.
  - Why not? Learn intrinsics.

LHCb ONLINE

Do it yourself!

- Intel Intrinsics guide
- Microsoft guide to Intrinsics
- http://d3f8ykwhia686p.cloudfront.net/1live/intel/An_Introduction_to_Vectorization_with_Intel_Compiler_021712.pdf
- http://d3f8ykwhia686p.cloudfront.net/1live/intel/CompilerAutovectorizationGuide.pdf
- http://www.agner.org/optimize/

LHCb ONLINE

Backup. We need backup!

More figures on performance

Other nodes

CPU	PrPixel speedup
Xeon L5520 @ 2.27GHz Q1'09	1.15 x
Xeon E5-2620 v2 @ 2.10GHz Q3'13	1.12 x
Xeon E5-2670 v2 @ 2.50GHz Q3'13	1.13 x

LHCb ONLINE