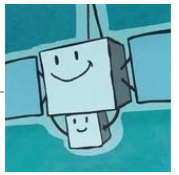
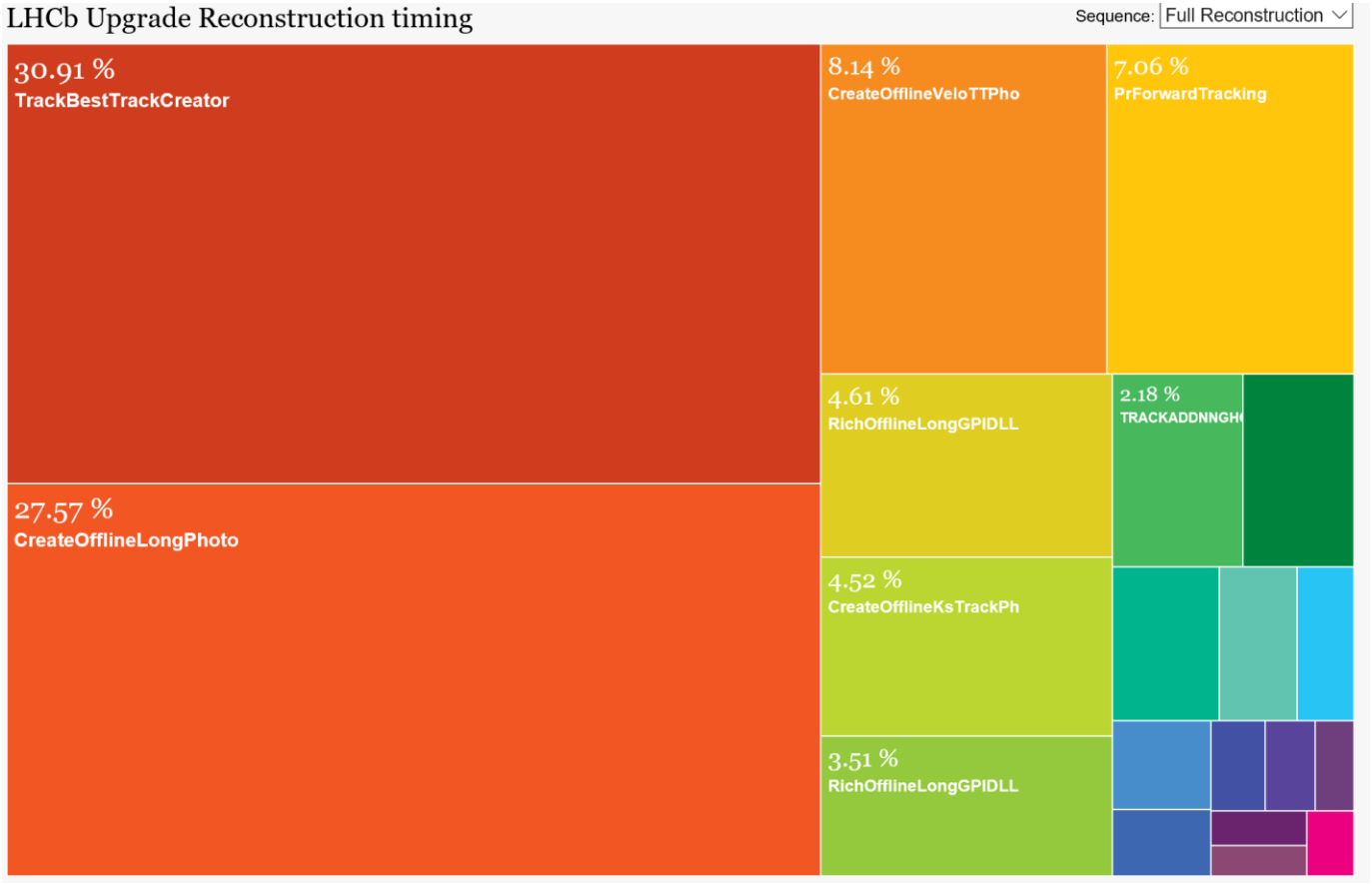
cross kalman

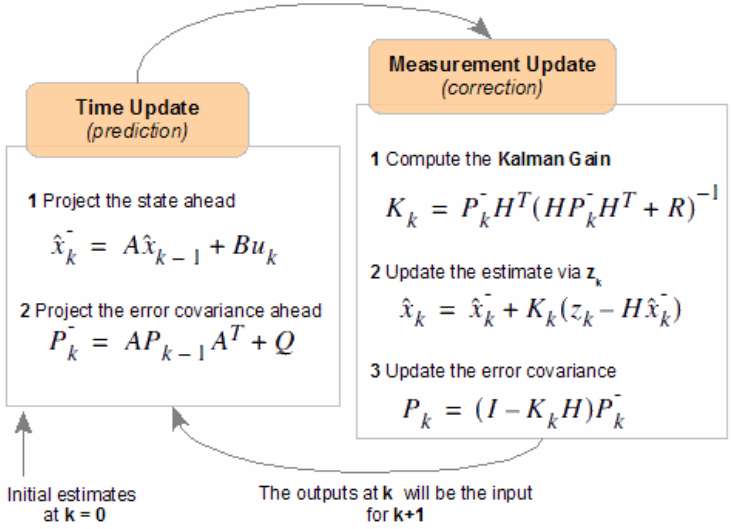
a fit and smooth kalman filter

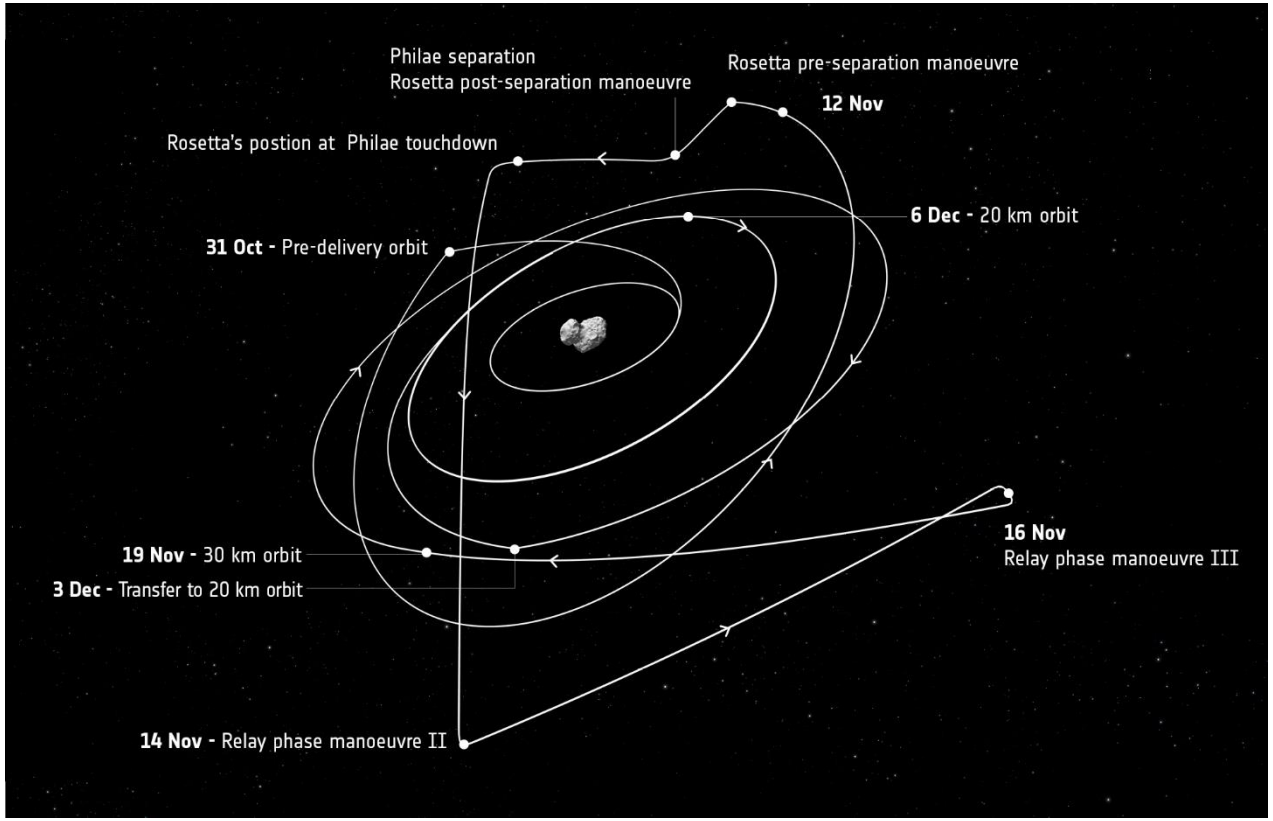


Where is it at?

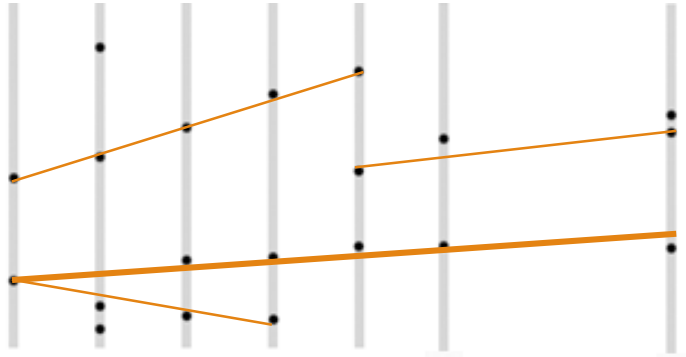


What is a Kalman Filter?





The LHCb case



p_0u_0 p_1u_1 p_2u_2 $p_{n-1}u_{n-1}$

$bp_{n-1}bu_{n-1}$ bp_0bu_0

s_0 s_1 s_2 s_{n-1}

m at a time (at best)

m at a time (at best)

$\sum_{i=1}^{m} n_i$ at a time

until converges + # for outlier removal

Upgrade is coming

- The detector will run at an increased rate of collisions and luminosity
  - LHCb software *needs* a performance boost
- Kalman Filter is the one shop for buying time in LHCb
  - Currently, somewhere around 70 % of the HLT 1 reconstruction time
- This process *is* embarrasingly parallel, across events *and* within events

```
Upgrade MC event average (1000 events):

3.127 average stages for each KF
171.324 average tracks KF in any stage
 463.886 average tracks KF in first stage
 80.52   average tracks KF in second stage
```

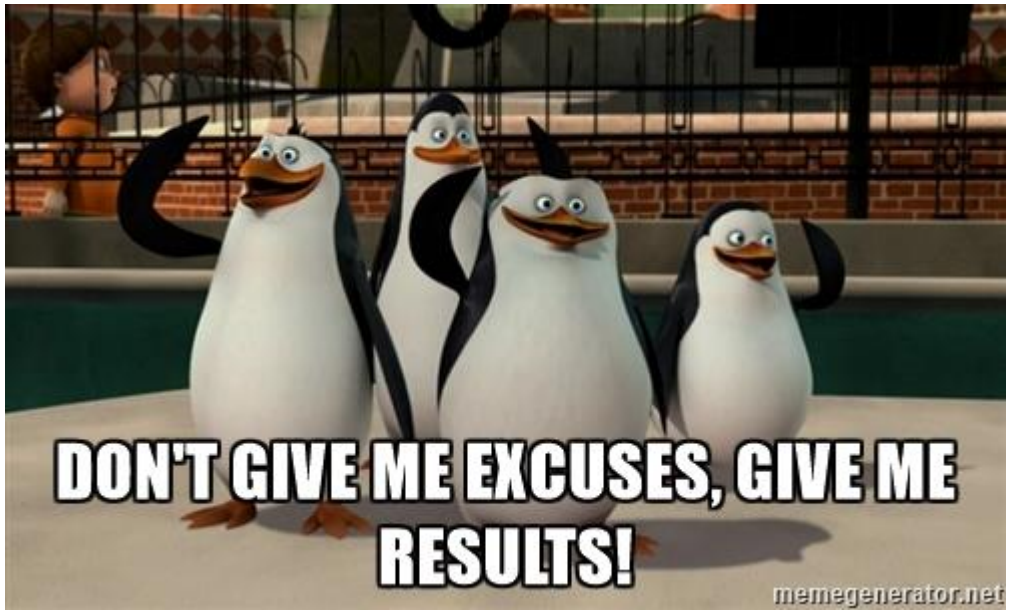
Our weapons of choice

- A fully vectorised algorithm
  - Configurable *precision* and *vector width* at compile time
- Data structures favoring locality
  - AOSOA across processing tracks
  - Data alignment set according to precision and vector width
- In the least amount of iterations possible
  - Lightweight static scheduler fills up exposed vector units efficiently
- Across architectures
  - Find the best price / performance / watt balance

```
it    in   out  act  vector (#track-#node)
[...]
#540: 0000 0001 1111 { 112-9 80-11 81-11 113-10 }
#541: 0001 1110 1111 { 112-10 80-12 81-12 79-3 }
#542: 1110 0000 1111 { 107-2 109-1 108-2 79-4 }
#543: 0000 0000 1111 { 107-3 109-2 108-3 79-5 }
#544: 0000 0000 1111 { 107-4 109-3 108-4 79-6 }
#545: 0000 0000 1111 { 107-5 109-4 108-5 79-7 }
#546: 0000 0000 1111 { 107-6 109-5 108-6 79-8 }
#547: 0000 0000 1111 { 107-7 109-6 108-7 79-9 }
#548: 0000 0000 1111 { 107-8 109-7 108-8 79-10 }
#549: 0000 0000 1111 { 107-9 109-8 108-9 79-11 }
#550: 0000 0001 1111 { 107-10 109-9 108-10 79-12 }
#551: 0001 1110 1111 { 107-11 109-10 108-11 111-1 }
#552: 1110 0000 1111 { 114-1 78-3 77-3 111-2 }
#553: 0000 0000 1111 { 114-2 78-4 77-4 111-3 }
#554: 0000 0000 1111 { 114-3 78-5 77-5 111-4 }
#555: 0000 0000 1111 { 114-4 78-6 77-6 111-5 }
#556: 0000 0000 1111 { 114-5 78-7 77-7 111-6 }
#557: 0000 0000 1111 { 114-6 78-8 77-8 111-7 }
#558: 0000 0000 1111 { 114-7 78-9 77-9 111-8 }
#559: 0000 0000 1111 { 114-8 78-10 77-10 111-9 }
[...]
```

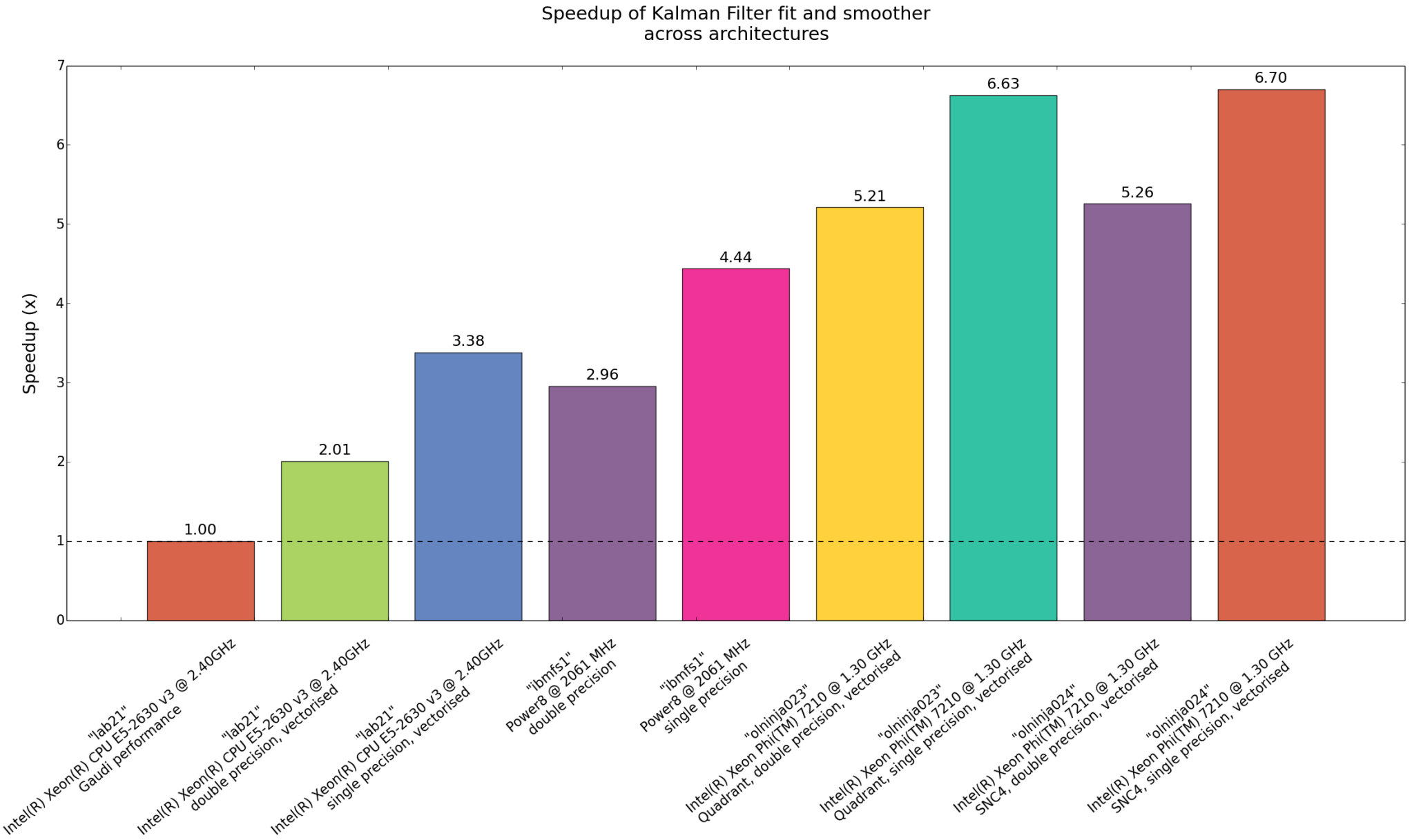
cross kalman scheduler

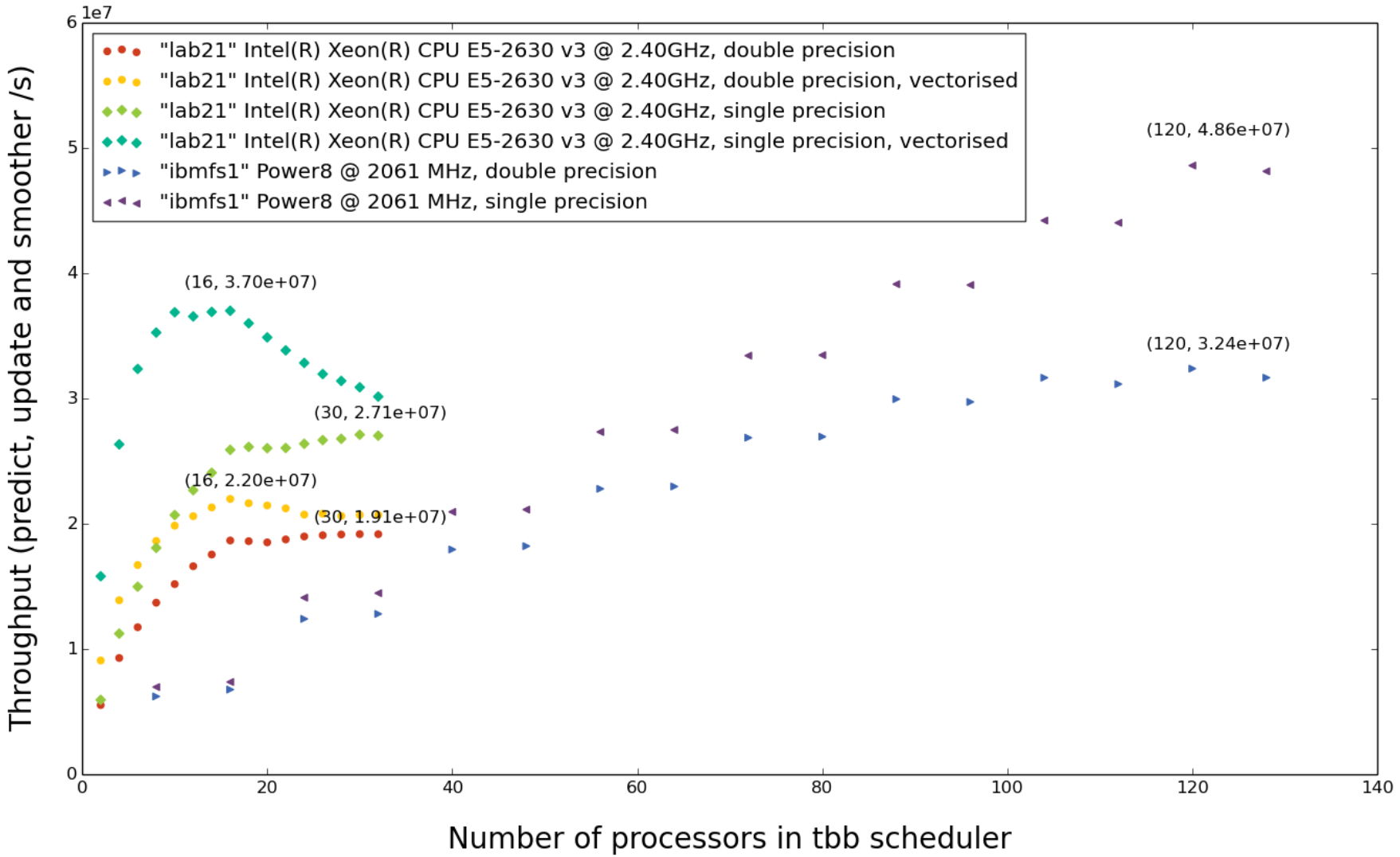
Results

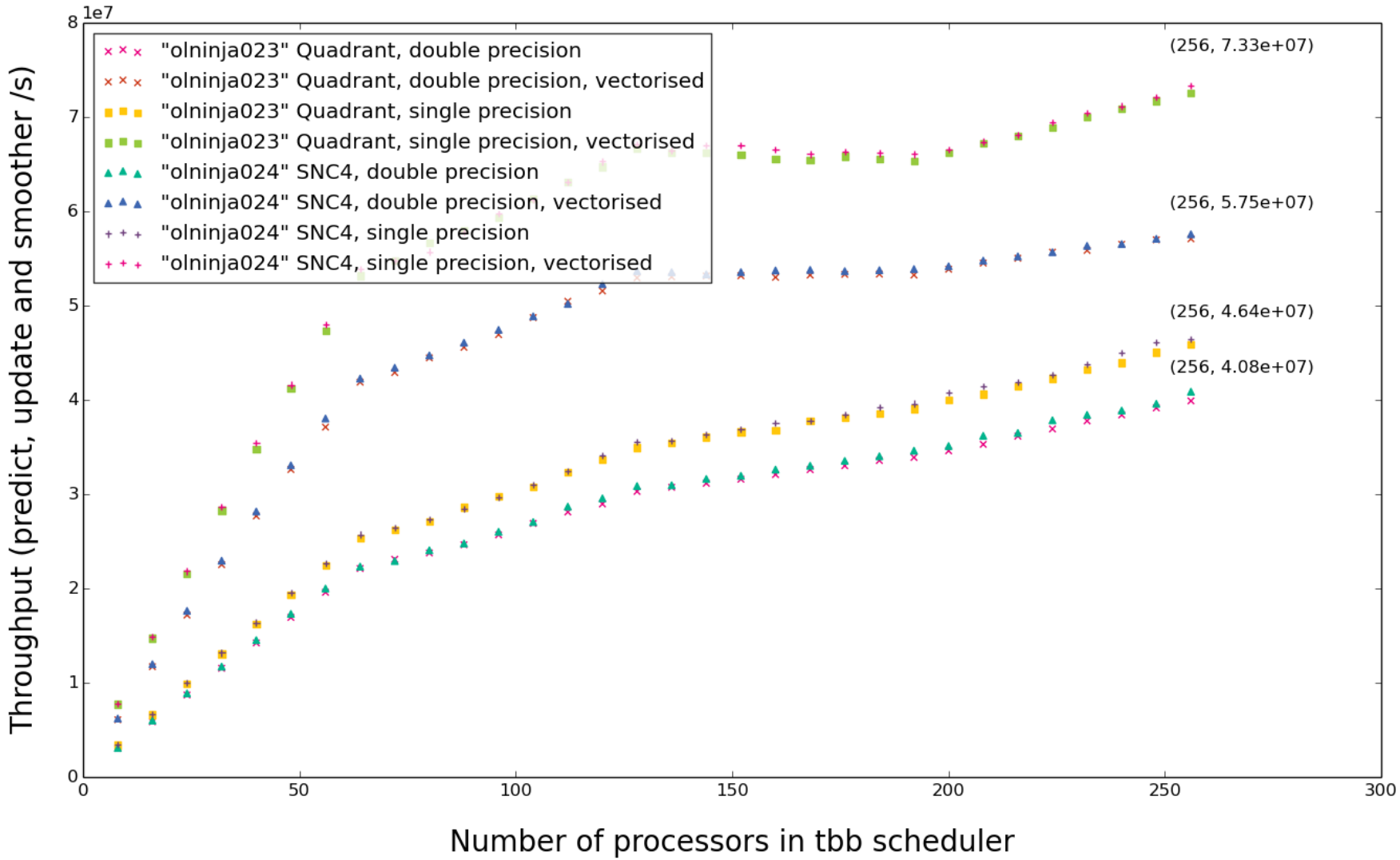


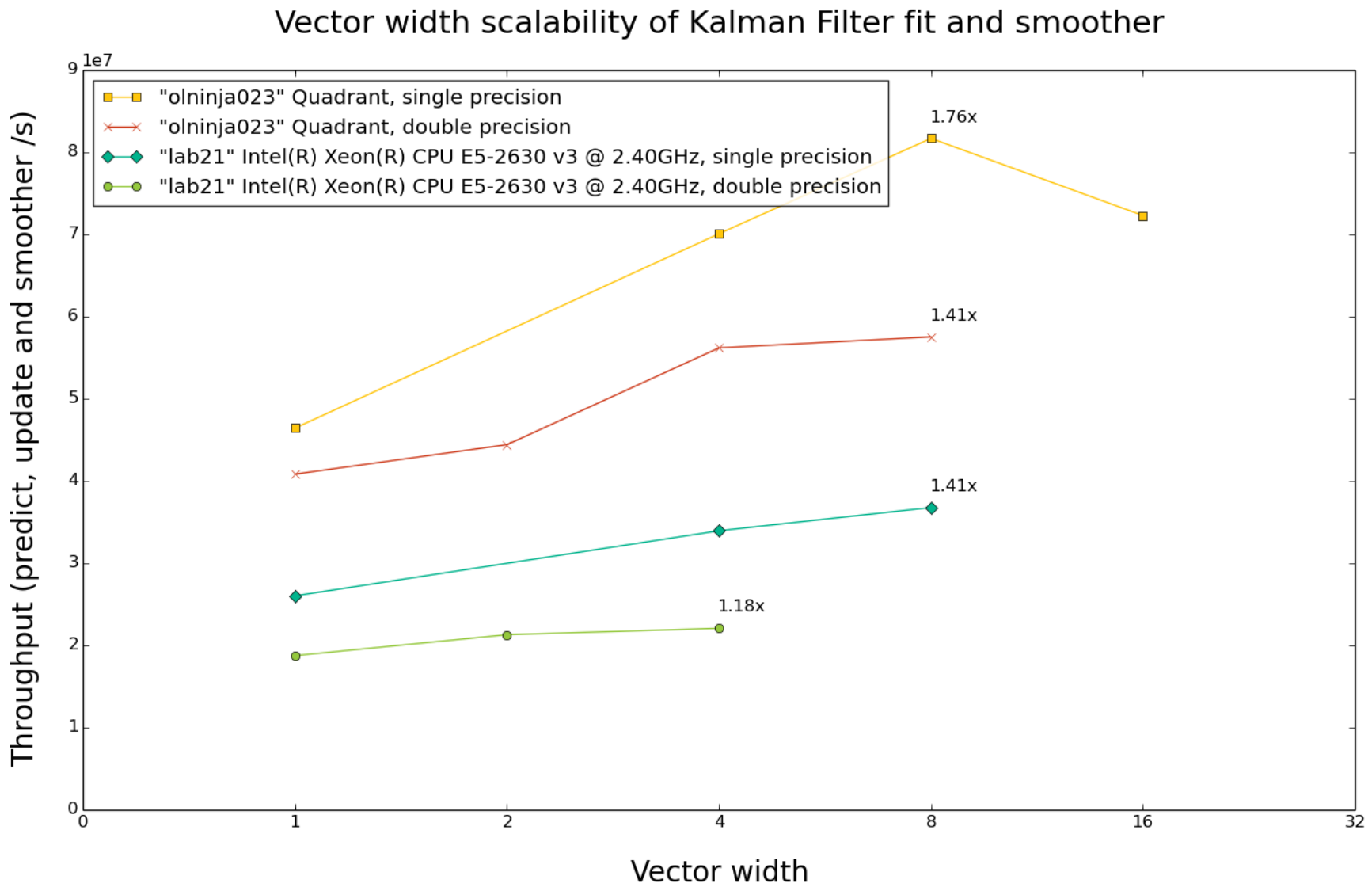
Running conditions

- Compiled with gcc 6.2.0
- Ran 500.000 experiments (events), each event is a *task*
- Used Montecarlo events from the LHCb Upgrade, checked against Gaudi implementation results
- Spawned one process per numa domain, with as many TBB threads as cores in domain and pinned to its memory

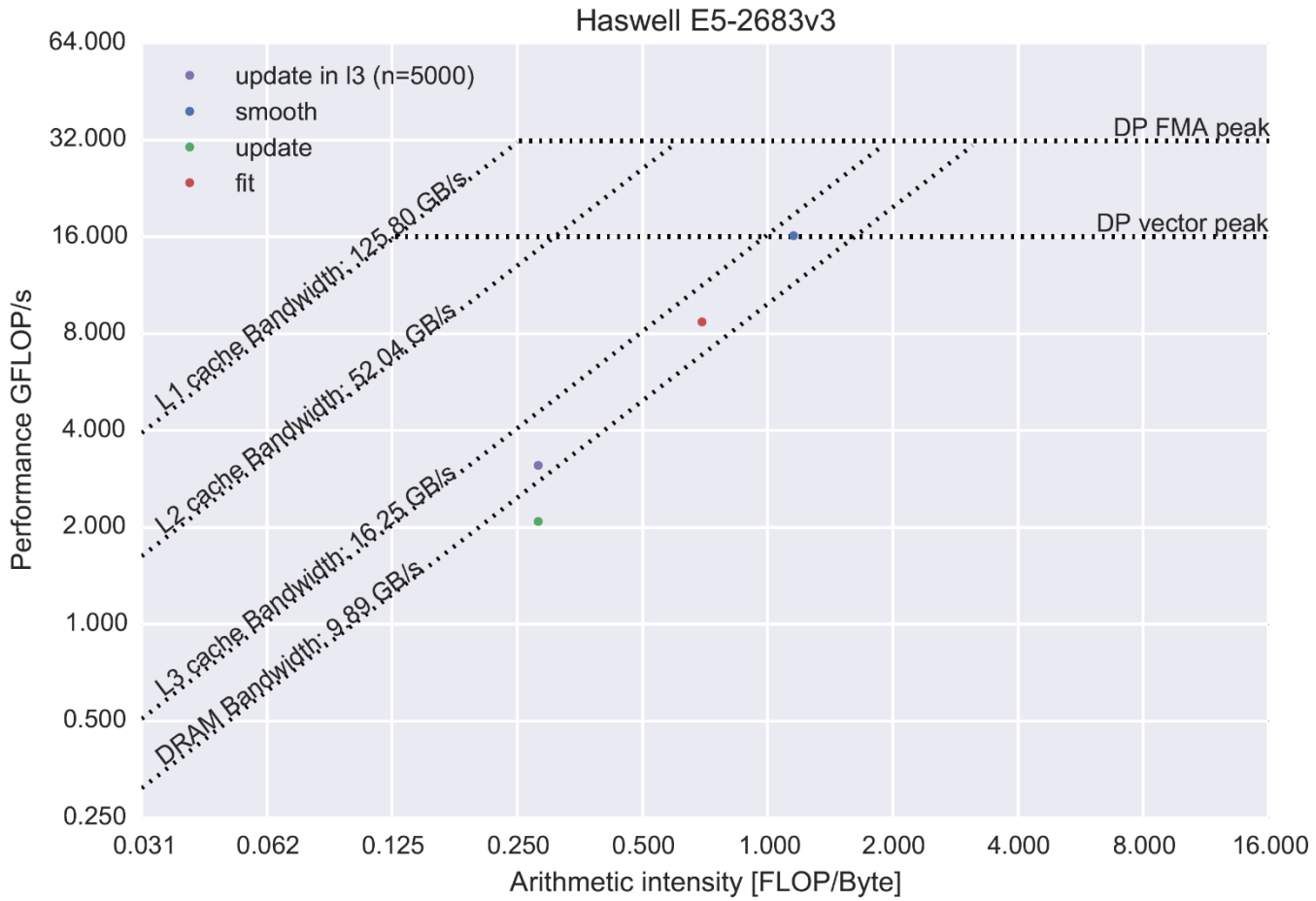

Speedup of Kalman Filter fit and smoother
across architectures
Speedup (x)
0
1
2
3
4
5
6
7
1.00
2.01
3.38
2.96
4.44
5.21
6.63
5.26
6.70
"lab21"
Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
Gaudi performance
"lab21"
Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
double precision, vectorised
"lab21"
Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz
single precision, vectorised
"ibmfs1"
Power8 @ 2061 MHz
double precision
"ibmfs1"
Power8 @ 2061 MHz
single precision
"olninja023"
Intel(R) Xeon Phi(TM) 7210 @ 1.30 GHz
Quadrant, double precision, vectorised
"olninja023"
Intel(R) Xeon Phi(TM) 7210 @ 1.30 GHz
Quadrant, single precision, vectorised
"olninja024"
Intel(R) Xeon Phi(TM) 7210 @ 1.30 GHz
SNC4, double precision, vectorised
"olninja024"
Intel(R) Xeon Phi(TM) 7210 @ 1.30 GHz
SNC4, single precision, vectorised


Scalability of Kalman Filter fit and smoother across architectures
1e7
"lab21" Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz, double precision
"lab21" Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz, double precision, vectorised
"lab21" Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz, single precision
"lab21" Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz, single precision, vectorised
"ibmfs1" Power8 @ 2061 MHz, double precision
"ibmfs1" Power8 @ 2061 MHz, single precision
(120, 4.86e+07)
(16, 3.70e+07)
(120, 3.24e+07)
(30, 2.71e+07)
(16, 2.20e+07)
(30, 1.91e+07)
Throughput (predict, update and smoother /s)
Number of processors in tbb scheduler
0
1
2
3
4
5
6
0
20
40
60
80
100
120
140


Scalability of Kalman Filter fit and smoother
on Intel(R) Xeon Phi(TM) 7210 @ 1.30 GHz
Throughput (predict, update and smoother /s)
1e7
Number of processors in tbb scheduler
"olninja023" Quadrant, double precision
"olninja023" Quadrant, double precision, vectorised
"olninja023" Quadrant, single precision
"olninja023" Quadrant, single precision, vectorised
"olninja024" SNC4, double precision
"olninja024" SNC4, double precision, vectorised
"olninja024" SNC4, single precision
"olninja024" SNC4, single precision, vectorised
(256, 7.33e+07)
(256, 5.75e+07)
(256, 4.64e+07)
(256, 4.08e+07)
0
1
2
3
4
5
6
7
8
0
50
100
150
200
250
300



- *Speedup* shown is versus the respective scalar version



- **Roofline model** of our main processes
  - Roofline: An Insightful Visual Performance Model for Multicore Architectures
- We have improved our performance by combining *predict* and *update* (into *fit*)
- The requirements of the Kalman Filter, in terms of memory operations, caps the attainable performance

A word about precision

- Single precision is worth exploring

  - It shows from 20 to 70 % performance gain, depending on architecture
  - Memory required is nominally half
  - Some architectures naturally favor single precision (ie. GPUs)

  - Mathematical error estimation hard to get
  - Cholesky decomposition can fail due to numerical imprecisions
    - It does indeed fail for about 1 % of the nodes in our tests

- Still need to check back in Gaudi what is the impact of enabling single precision
- Mixed precision?
- Configurable precision upon track properties?

Concluding

We have achieved some good things

- Cross architecture, fully vectorised Kalman Filter
- Base 2x with respect to original implementation
- Up to 6.7x on KNL
- Our roofline model shows we are getting the most out of the machine

There is more to come

- Integration with present and future Gaudi
- GPU implementation revamp
- Vectorisation for Power8
- Testing + vectorisation for ARM64

Thanks!

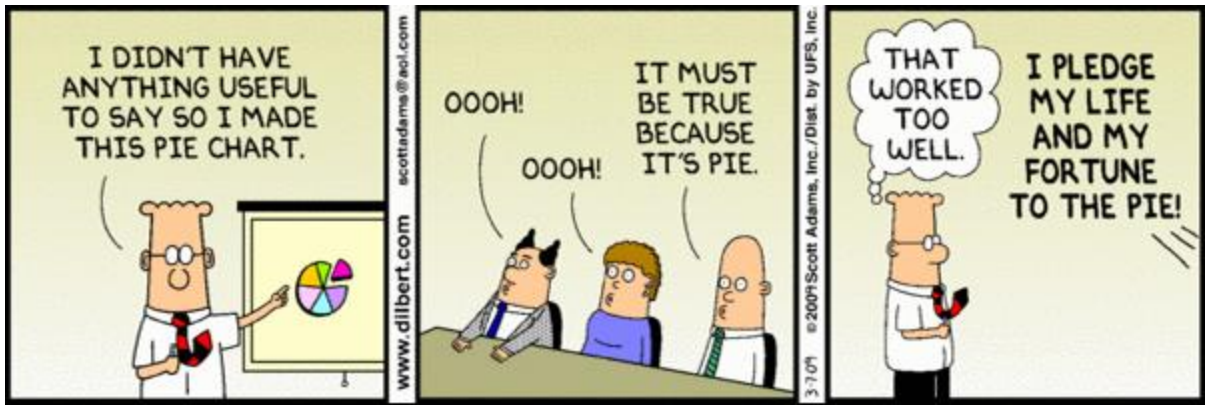
Thanks to all of these people, who greatly contributed to these results,

- O Awile
- O Bouizi
- S Harald
- F Lemaitre
- C Potterat
- The CERN Intel Openlab HTC Collaboration
- The RGNC at the University of Sevilla

Resources

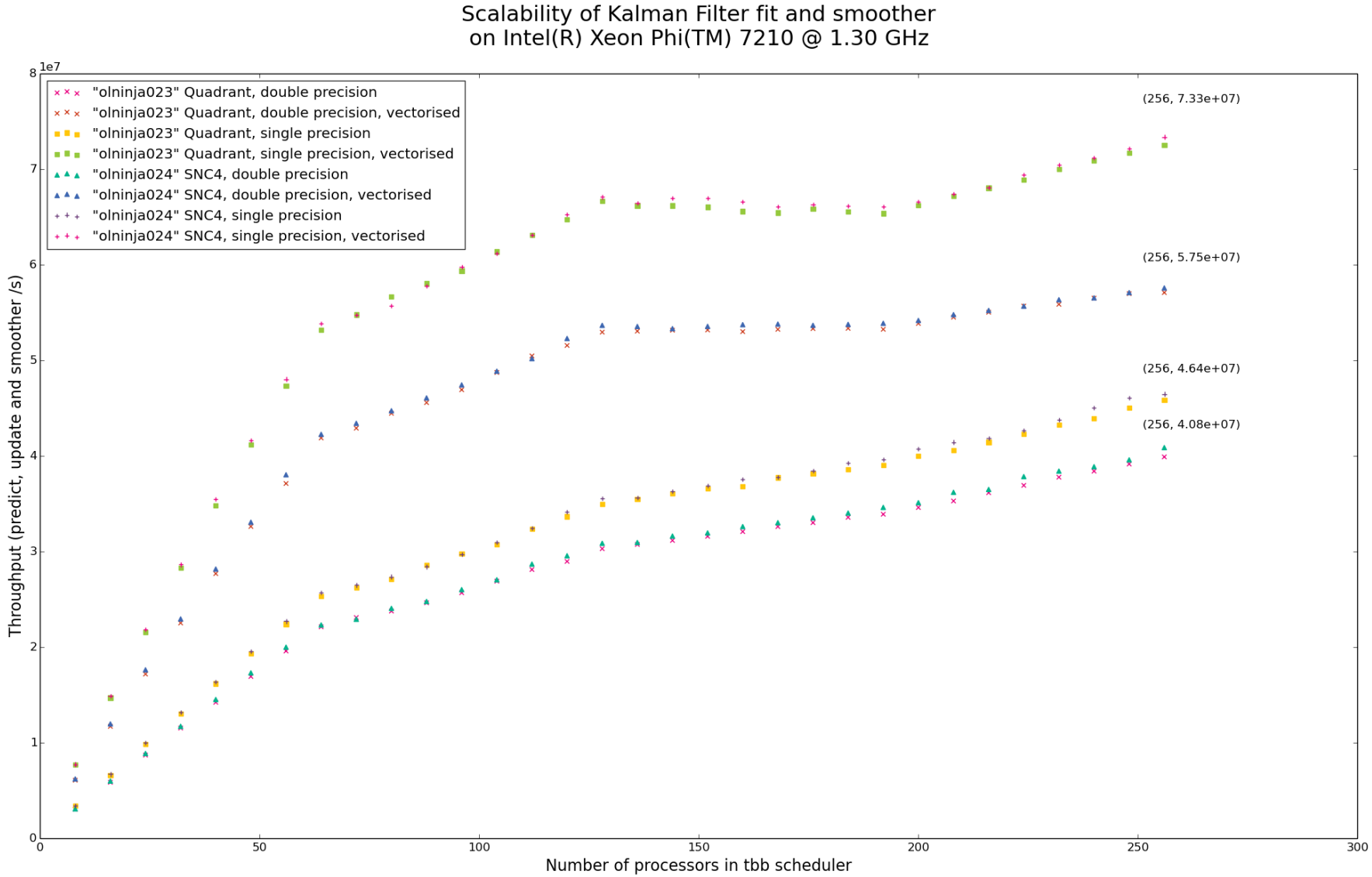
- cross kalman repository
- Roofline: An Insightful Visual Performance Model for Multicore Architectures

Questions?


I DIDN'T HAVE ANYTHING USEFUL TO SAY SO I MADE THIS PIE CHART.
www.dilbert.com scottadams@aol.com
OOOH!
OOOH!
IT MUST BE TRUE BECAUSE IT'S PIE.
©2001 Scott Adams, Inc./Dist. by UFS, Inc.
THAT WORKED TOO WELL.
I PLEDGE MY LIFE AND MY FORTUNE TO THE PIE!

They back me up when I’m feeling blue

How about you


Scalability of Kalman Filter fit and smoother
on Intel(R) Xeon Phi(TM) 7210 @ 1.30 GHz
1e7
"olninja023" Quadrant, double precision
"olninja023" Quadrant, double precision, vectorised
"olninja023" Quadrant, single precision
"olninja023" Quadrant, single precision, vectorised
"olninja024" SNC4, double precision
"olninja024" SNC4, double precision, vectorised
"olninja024" SNC4, single precision
"olninja024" SNC4, single precision, vectorised
(256, 7.33e+07)
(256, 5.75e+07)
(256, 4.64e+07)
(256, 4.08e+07)
Throughput (predict, update and smoother /s)
Number of processors in tbb scheduler
0
1
2
3
4
5
6
7
8
0
50
100
150
200
250
300