Cross-architecture Kalman filter benchmarks on modern hardware platforms

Daniel Hugo Cámpora Pérez
Omar Awile
Othman Bouizi
Niko Neufeld
ACAT, 28th August, 2017

1CERN
2Intel
Outline

Introduction

Cross-Kalman

Results

Conclusions
Introduction
The LHCb data acquisition (DAQ) system

- Data comes at a rate of 30 MHz
- A throughput of 40 Tbit/s needs to be processed in real-time
- All data will be processed in software
High Throughput Computing Collaboration (HTCC)

Apply upcoming Intel® technologies in an *Online* computing context at the Large Hadron Collider

- Data acquisition (DAQ) and event building
- Accelerator assisted decision taking on collected data

Use LHCb upgrade as an example, but applicable and useful for other experiments too!
Status of LHCb codebase

More than 5 MLOCs of C++ (and some Python)

- Under redesign for SIMD and shared-mem parallelism
- Baseline remains Xeon\textsuperscript{®} CPUs
- New framework uses TBB to dispatch algorithms to process events in a multi-threaded fashion
- Possible avenues to accelerate algorithms:
  - Offload critical functions to FPGA
  - Rewrite most time-consuming algorithms in a parallel fashion and use Xeon Phi\textsuperscript{TM}
Cross-Kalman
The Kalman filter

The most time consuming algorithm, taking 60% of the time in the Online reconstruction, is the Kalman filter. The Kalman filter is a well-known linear quadratic estimator. For every particle node,

- **Predict** - The state of the system is projected according to a given model
- **Update** - The state is adjusted taking into account a measurement
We have developed a cross-architecture Kalman filter, targeting SIMD architectures. Our design considers three key components:

- Control flow
- Data structures
- Arithmetic backend
Every particle can be considered a succession of nodes with an implicit computing order. Given that we receive hundreds of particle trajectories as an input, we are given the scheduling problem of assigning particle nodes to processors, where we attempt to minimize the number of computing iterations.

This problem is a variant of the number partitioning problem $\mathbb{NP}$, which is NP-complete.

Nevertheless, a Decreasing Time Algorithm (DTA) behaves well as a scheduling algorithm.

---

We use a *static scheduler*

- Data locality is maximized
- Data is guaranteed to be aligned

```
  it   in   out   act   vector (#particle-#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 }
```
We use a static scheduler

- Data locality is maximized
- Data is guaranteed to be aligned
- Data is AOSOA

\[
\begin{array}{cccc}
  x_0 & x_1 & x_2 & x_3 \\
  y_0 & y_1 & y_2 & y_3 \\
  tx_0 & tx_1 & tx_2 & tx_3 \\
  ty_0 & ty_1 & ty_2 & ty_3 \\
  q & q & q & q \\
  p_0 & p_1 & p_2 & p_3 \\
  \sigma_{0,0} & \sigma_{1,0} & \sigma_{2,0} & \sigma_{3,0} \\
  \vdots & \vdots & \vdots & \vdots \\
  \sigma_{0,14} & \sigma_{1,14} & \sigma_{2,14} & \sigma_{3,14} \\
  \chi_0^2 & \chi_1^2 & \chi_2^2 & \chi_3^2 \\
\end{array}
\]
Arithmetic backend

The bulk of the math of the application is written in an architecture-aware programming extension / library,

- VCL
- UMESIMD
- OpenCL
- CUDA
Methodology

- All programs were compiled with gcc 6.2.0, and flags -O2 -march=native
- Montecarlo events, generated with the official LHCb generator, were used
- All of the results are validated against existing Kalman filter implementation
- For each experiment, 500 000 events were run
- NUMA pinning was used where pertinent, spawning processes
- TBB distributes the workload across threads
## Hardware platforms

<table>
<thead>
<tr>
<th></th>
<th>Intel® Xeon® E5-2630 v3</th>
<th>Intel® Xeon® E5-2683 v4</th>
<th>Intel® Xeon Phi™ 7210</th>
<th>Intel® Xeon® Platinum 8170</th>
</tr>
</thead>
<tbody>
<tr>
<td># cores</td>
<td>8</td>
<td>16</td>
<td>64</td>
<td>26</td>
</tr>
<tr>
<td>base frequency</td>
<td>2.40 GHz</td>
<td>2.10 GHz</td>
<td>1.30 GHz</td>
<td>2.10 GHz</td>
</tr>
<tr>
<td>Cache</td>
<td>20 MB L3</td>
<td>40 MB L3</td>
<td>32 MB L2</td>
<td>35.75 MB L3</td>
</tr>
<tr>
<td>TDP</td>
<td>85 W</td>
<td>120 W</td>
<td>215 W</td>
<td>165 W</td>
</tr>
<tr>
<td>DRAM configuration</td>
<td>64 GB</td>
<td>64 GB</td>
<td>96 GB (+16 GB MCDRAM)</td>
<td>192 GB</td>
</tr>
<tr>
<td>STREAM bandwidth</td>
<td>41 GB/s</td>
<td>76 GB/s</td>
<td>77 GB/s (DRAM)</td>
<td>101 GB/s</td>
</tr>
<tr>
<td>Recommended price</td>
<td>$667.00 - $671.00</td>
<td>$1846.00</td>
<td>$1881.00</td>
<td>$7405.00 - $7411.00</td>
</tr>
</tbody>
</table>

- The E5-2630 v3 platform represents the current production configuration
- The Broadwell and Skylake platforms show both medium-range and high-end alternatives for the future TDAQ cluster
- The Knights Landing platform shows a manycore alternative to more traditional Xeon based solutions
Cross-architecture speedup

Note: Scalar performance is the Kalman filter performance when vectorization is disabled.
Cost of purchase for CPUs to achieve a throughput of 10’000 events/s for the 4 variants.
Scalability and parallel efficiency

- The transition to HyperThreads is clearly visible
- We suspect the superlinear behaviour of Skylake is due to caching issues
• gcc 7.1.1 gets a little more performance regardless of processor count
• We observe a similar performance for icc 18.0 and clang 4.0.0
Results in upcoming multithreaded framework

- In the transition to the framework, Xeon Phi\textsuperscript{TM} loses performance wrt. Xeon\textsuperscript{⃝}
- This is probably due to memory contention in the framework
• The Roofline model shows our fitter surpasses the DDR roof.
• This Skylake model is tough to program. Only arithmetic-heavy codes will achieve close to peak performance of such a processor.
Conclusions
Conclusions

We have developed a cross-architecture Kalman filter for LHCb

- It works well with modern SIMD architectures, and scales with upcoming ones
- It is integrated in our software framework

We are evaluating different architectures for the upcoming upgrade

- Many-core architectures are an option only if the framework and algorithms are prepared to take advantage of it
- Memory model will have a big impact - 3D memory?
- There is no *one-fits-all* solution, if one requires best performance. As of now, each architecture requires a slightly different approach
- We need reconstruction software and farm capable of processing 40 Tbit/s of data, performance matters!