



Portability for the Patatrack Pixel Track Reconstruction

Abhinav Ramesh

Mentors: Felice Pantaleo and Wahid Redjeb

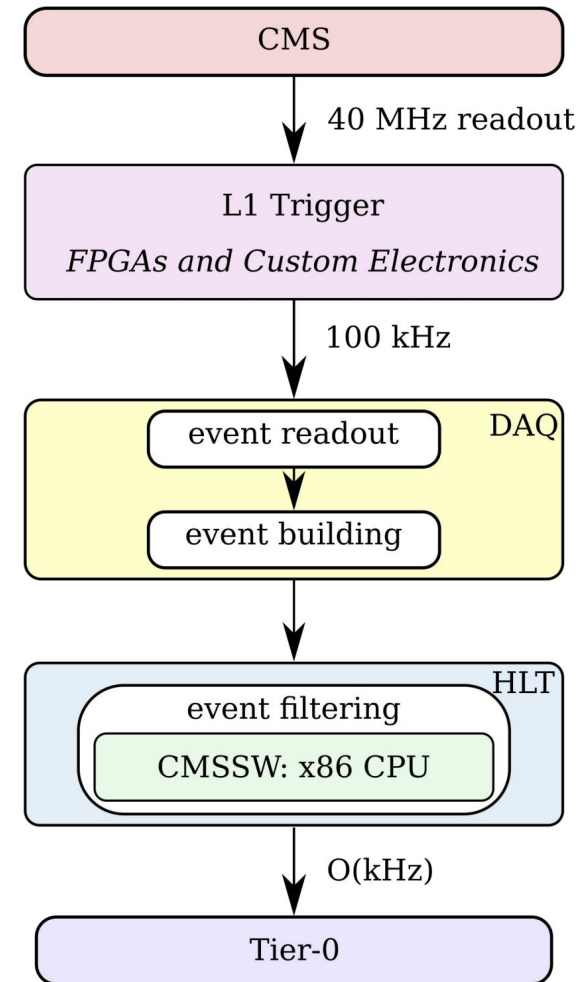
Heterogeneous Computing

- **Heterogeneous systems** consist of different types of computational units such as multicore CPUs, GPUs and FPGAs.
- **Hardware accelerators** can perform specific functions more efficiently compared to software running on a general-purpose CPU.
- They've always been used for 3D graphics acceleration, compression, encryption and pattern recognition.
- Each of CPUs and GPUs have their strengths and weaknesses and the key lies in offloading the right kind of computationally intensive tasks to GPUs.

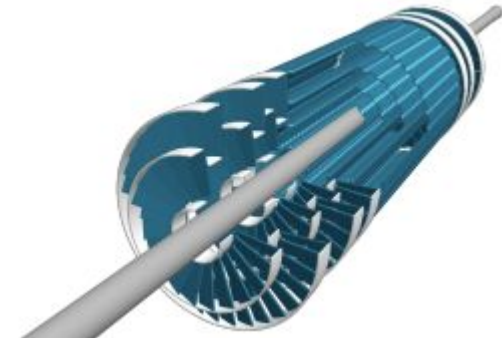
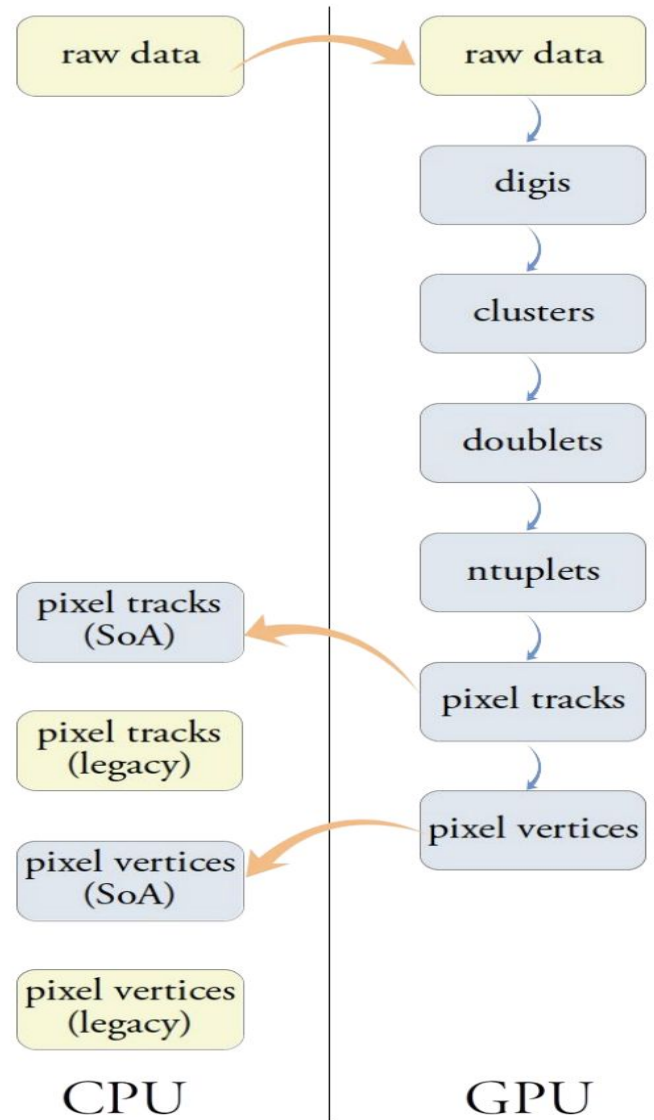
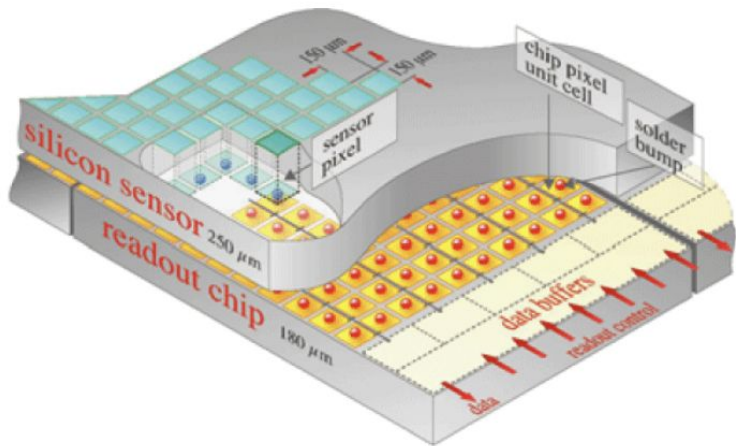


The need for accelerators in CMSSW

- CMS Phase-2 upgrade
- Increasing luminosity (5x)
- Lot of data being produced
- L1 Trigger : 100 kHz -> 750 kHz (7.5x)
- HLT (5-7x)
- More complex detectors
- Pile-up : 50 -> 200 (4x)
- Reconstruction algorithms don't scale linearly with pile-up
- Using CPUs alone won't cut it. Importantly, better Physics algorithms that are more complex can be run at the same cost



Patatrack Pixel Reconstruction

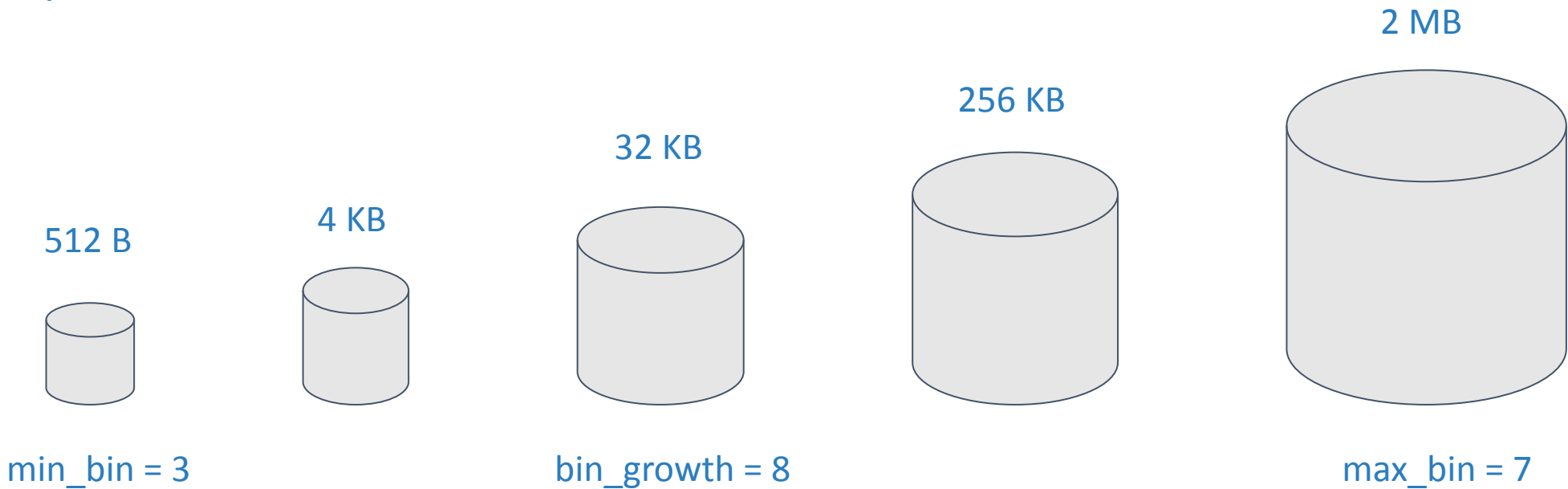


Necessity for ALPAKA

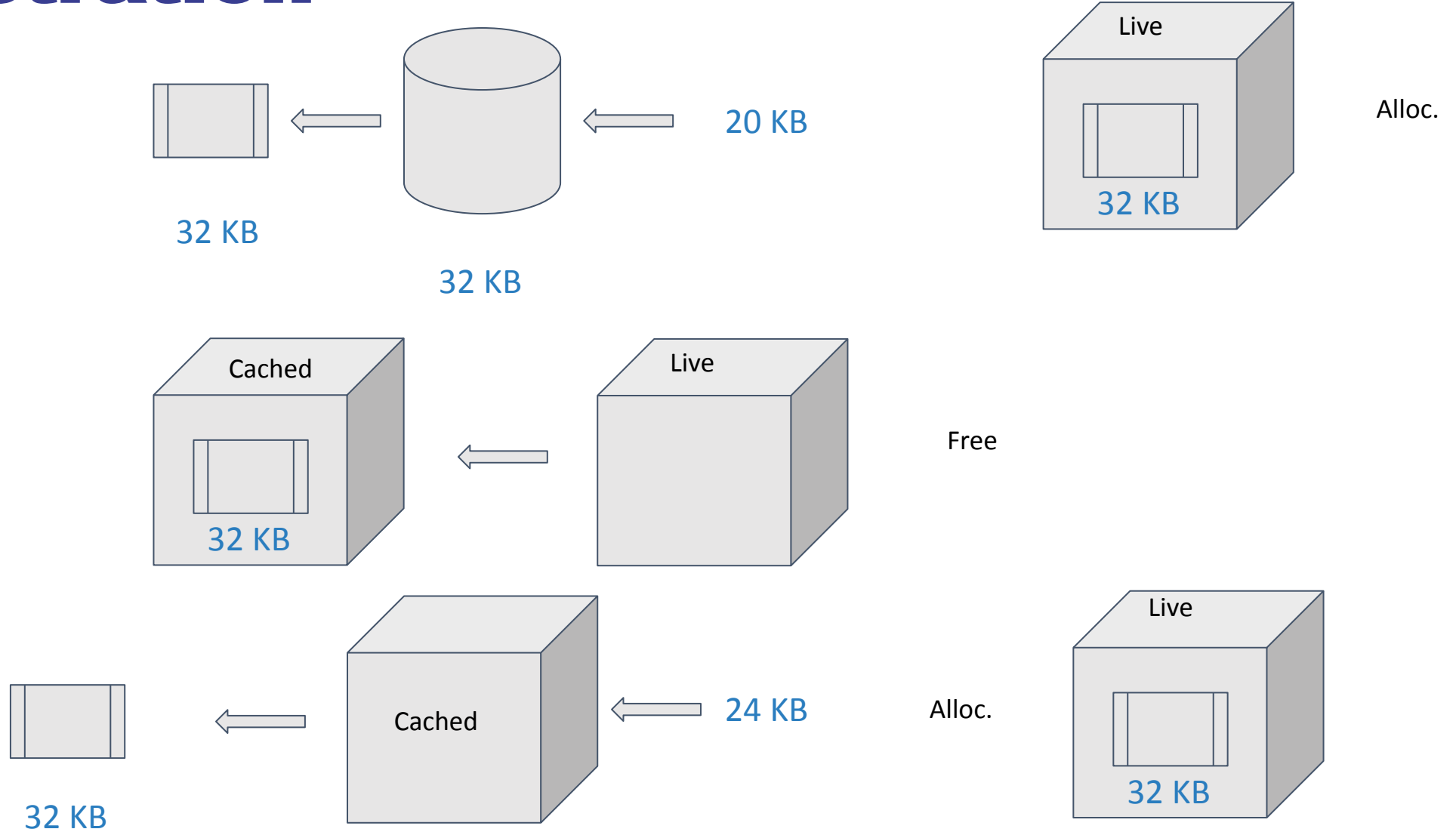
- Lots of architectures and vendors - NVIDIA GPUs (CUDA), ARM GPUs (HIP), OpenMP, TBB
- Writing code mundanely for each back-end would lead to code duplication
- Changes have to be made in many places and mistakes could crop up
- ALPAKA : header-only C++ library that provides performance portability across accelerators by abstracting the underlying parallelism
- Platform independent, and can support concurrent and cooperative use of multiple devices
- Good number of accelerator back-ends are supported and there is no need to write any specialized code for any back-end
- Abstraction used is similar to the CUDA grid-blocks-threads division strategy

Caching Allocators : Introduction

- CMS is porting pixeltrack and vertex reconstruction to Alpaka, but performance improvement is desired. It's not desirable to keep allocating memory repeatedly as we process the events. So, we wish to reuse memory. Moreover, there are 2 sub-versions for the CUDA version - 1 that uses and 1 that doesn't use the allocators.
- Idea : CUDA version uses caching allocators that can be enabled/disabled at compile-time



Illustration



Caching Allocators : Process

1. Assign a bin to the allocation request by finding the nearest power
2. If the request is greater than the max_bin, allocate exactly and don't cache for reuse when the block is returned, go to step 5
3. Search for a suitable block within cached blocks. If found, remove from cached blocks and add to live blocks, go to step 5
4. Allocate the block and add to live blocks
5. Return the block

Standalone pixel tracking repo :

<https://github.com/cms-patatrack/pixeltrack-standalone> , my branch :

https://github.com/abhinavramesh8/pixeltrack-standalone/tree/alpaka_caching_allocator

Conclusion

- The caching allocator logic has been implemented. I am still in the process of incorporating it in the framework code.
- Currently, 3 out of 6 plugins and all the data formats except for 1 are using the allocators.
- After this work is done, we have to profile workflows in order to better understand performance.



QUESTIONS?

abhinavramesh@utexas.edu

[linkedin.com/in/abhinavramesh8/](https://www.linkedin.com/in/abhinavramesh8/)