

Multithreading Muon Collider Software

Sam Ferraro
(samuel_ferraro@g.harvard.edu)



December 9, 2025

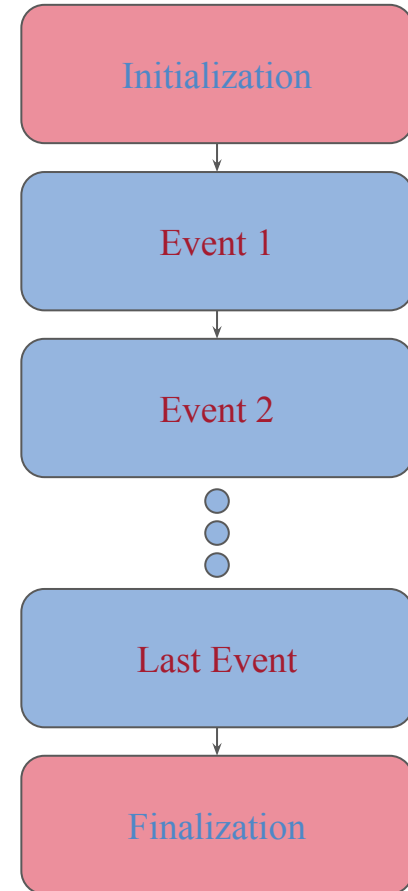
Fully Gaudi-based Image

- See [Federico's slides](#).
 - And [steering files](#)!
- Note that the setup script for the steering files finds all of the geometry files for you!

```
source mucoll-benchmarks/k4MuCPlayground/setup_digireco.sh \
mucoll-benchmarks/ \ # So the script knows where it is
MuSIC_v2             # Name of geometry
```

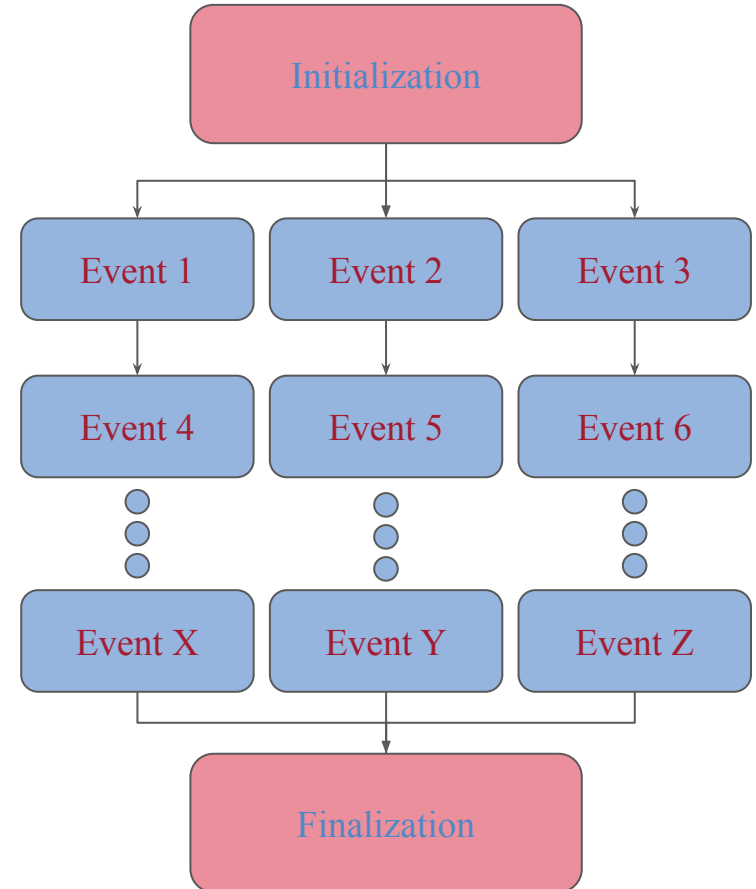
Multithreading Over Events

- Key4hep (Gaudi) is a thread-safe framework
- Some transitioned software did not have internal states:
 - [k4Reco](#)
 - Tracker Digitization ✓
 - Calorimeter Digitization ✓
 - Overlay? ✗
 - [k4ActsTracking](#) ✓
 - [k4GaudiPandora](#) ✗



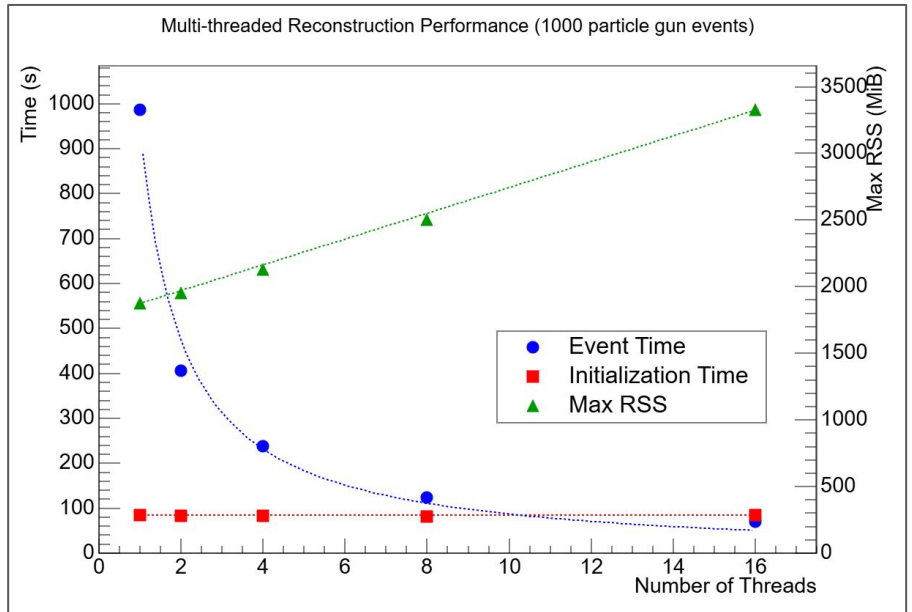
Multithreading Over Events

- Key4hep (Gaudi) is a thread-safe framework
- Redesigned k4GaudiPandora to be stateless
 - [Pull Request](#)



Multithreading Over Events

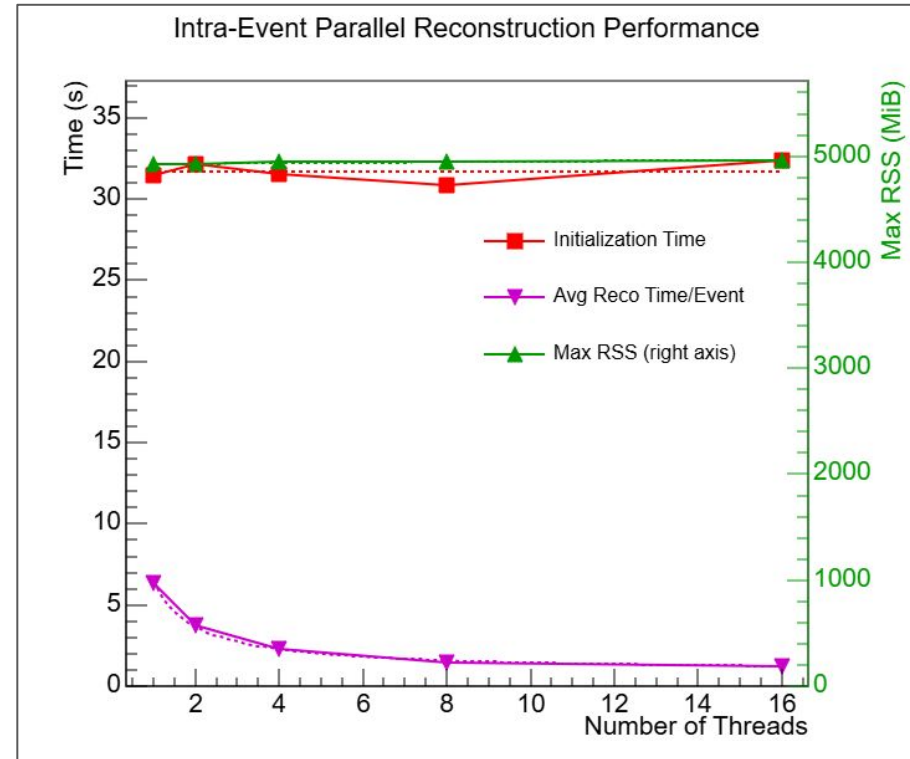
- Key4hep (Gaudi) is a thread-safe framework
- Redesigned k4GaudiPandora to be stateless
 - [Pull Request](#)
 - [New Steering Files](#)
- Performed as expected
 - Initialization Time remained Constant.
 - Event Processing went as $\frac{1}{n_{\text{threads}}}$.
 - Maximum Memory Usage Increased Linearly.
- Also works for digitization. No improvement when running without BIB



* No BIB

Multithreading Within Events

- Memory is a concern (Especially with BIB!)
- Implemented multithreading in k4ActsTracking using Gaudi's multithreading library
 - [Pull Request](#)
 - [New Steering Files](#)
- Naive Parallelization → Still gets speed up
- **Memory Constant w.r.t n_{threads} !**




* With BIB

Note on TrackTruthAlg

- Part of k4ActsTracking (and ActsTracking)
- Uses maps, but doesn't take advantage of the speed up.
- Improvements:
 - Removed map duplication
 - Took advantage of `std::map::find` ←
 - Minor parallelization

Time/Event Before (s)	Time/Event After (s)
1213.64	0.48

Mainly from map clean-up



Note on Overlay Algorithm

- I didn't check if the [Overlay algorithm](#) is thread-safe. I don't think that it is.
- Simple mutex locks on open files should fix this.
 - As I was making this I realized this would be pretty easy...
- The Overlay Algorithm continues to baffle me.
 - The version in the `full_gaudi_test` image does not work
 - Compiling the exact same version locally works fine

Note on k4GaudiPandora

- Much is outsourced to Pandora, which seems to hold internal states.
- This prevents easy intra-event parallelization without unboxing Pandora.

Next Steps

- Make Overlay Algorithm thread-safe.
 - And solve pesky bug!
- Intra-event Parallelization for k4GaudiPandora.
- Validate new image.