

# Porting CMS Pixel Track Reconstruction to Julia: Final Results

Mohamad Khaled Charaf

Mohamad Ayman Charaf

Maya Ali

Dr. Andrea Bocci

Dr. Phillipe Gras

Dr. Mateusz Jakub Fila

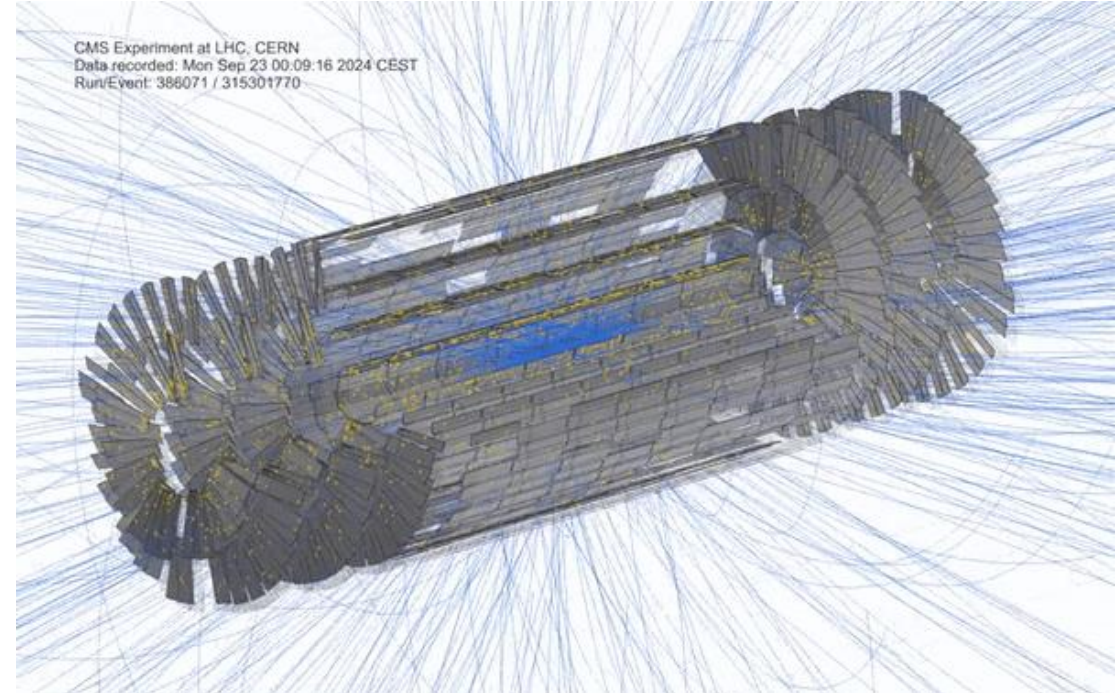


**NextGen**  
Next Generation Triggers

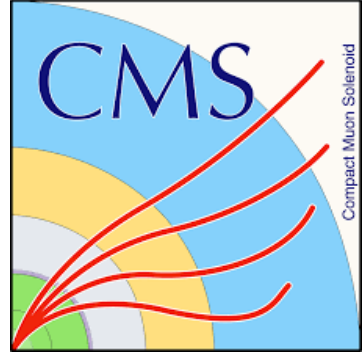
# Outline

---

1. Project Context
2. Main Goal
3. Overview of the Patatrack Application
4. Highlights from the Past Year
5. Recent Developments
6. Performance
7. Next Steps and Future Work
8. Conclusion



# Project Context



## Evaluation of Julia as a Language for High Energy Physics:

- Potential of the Julia Programming Language for High Energy Physics Computing ([doi:10.1007/s41781-023-00104-x](https://doi.org/10.1007/s41781-023-00104-x))
- Jet reconstruction (reclustering) with Julia ([github.com/JuliaHEP/JetReconstruction.jl](https://github.com/JuliaHEP/JetReconstruction.jl))

## Patatrack Pixel Reconstruction:

- Standalone application extracted from CMS software
- Pixel Reconstruction: the process of identifying and reconstructing particle trajectories by analyzing data from pixel detectors
- Tested over the years across multiple parallel programming models and accelerator platforms (OpenMP, CUDA, HIP, SYCL, Kokkos, etc.)

# **Main Goal**

---

**To evaluate the feasibility of using Julia for large-scale HEP applications by porting the CMS pixel-only track reconstruction pipeline from C++ to Julia**

**Assess Julia's capabilities in:**

- Single-threaded performance on CPU
- Multi-threading and parallel CPU performance
- GPU acceleration and heterogeneous execution

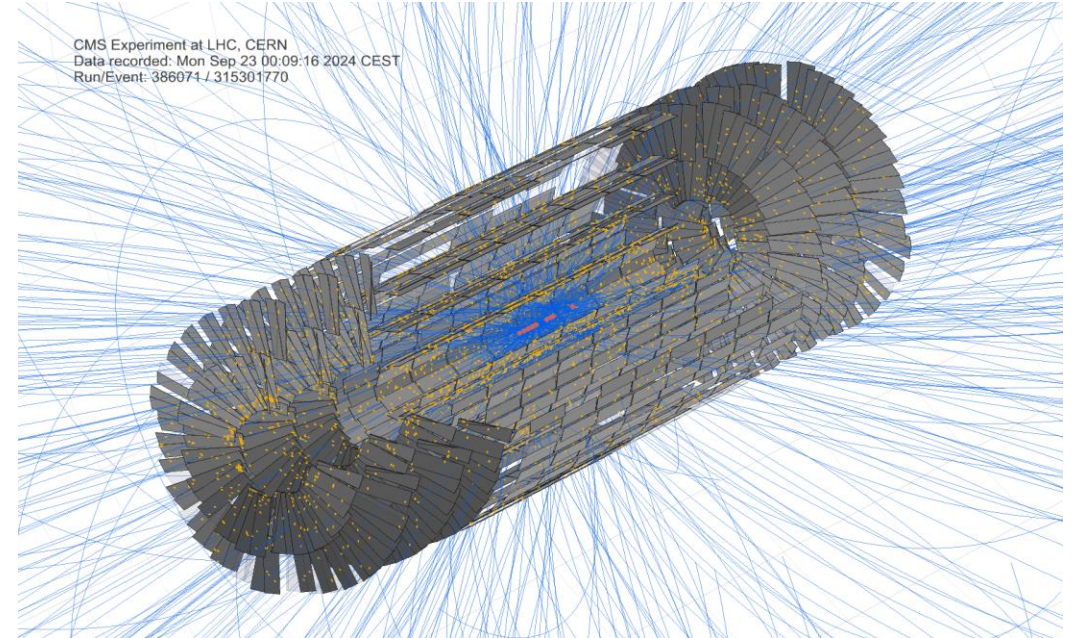
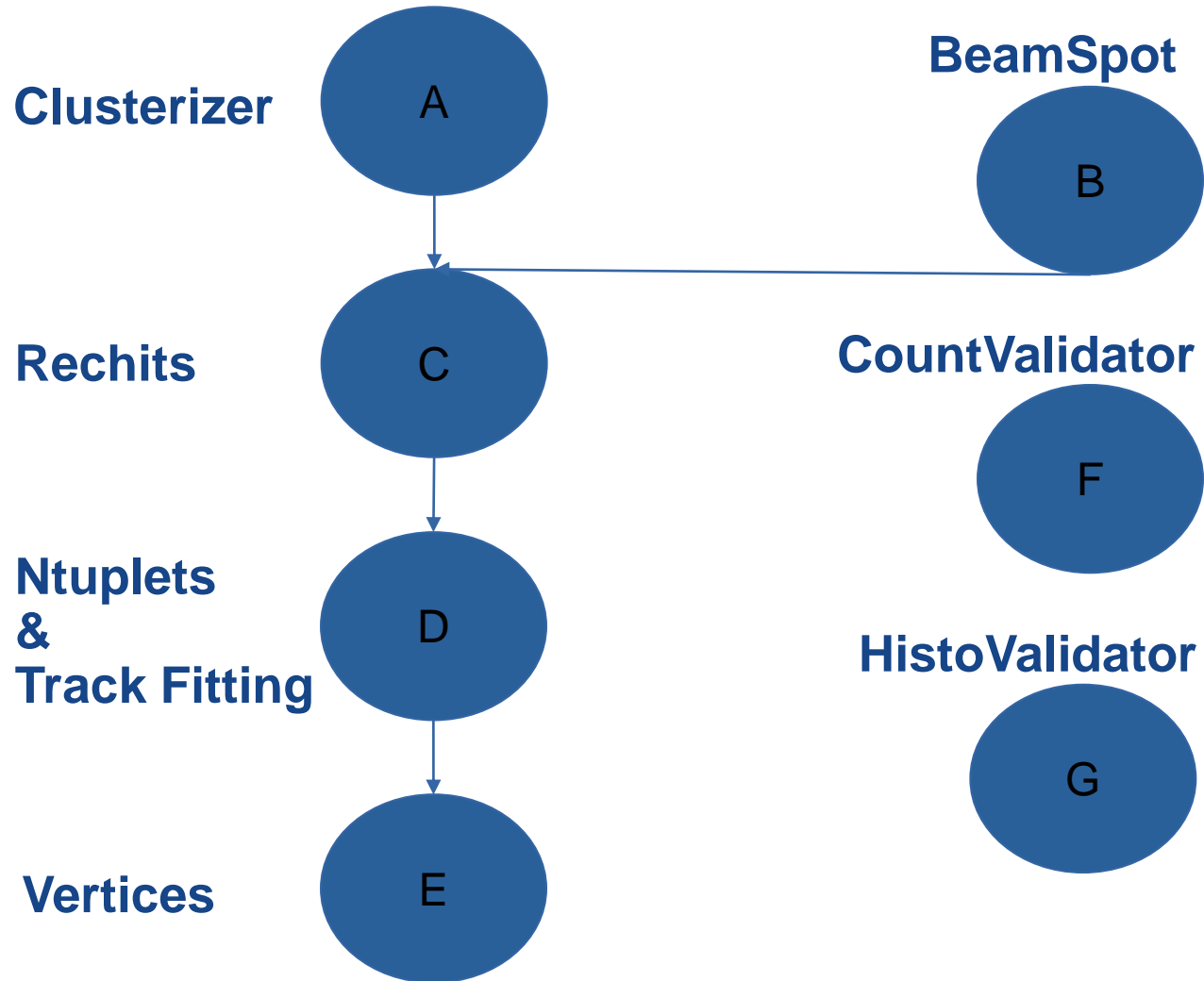
# Overview of the Application Framework

---

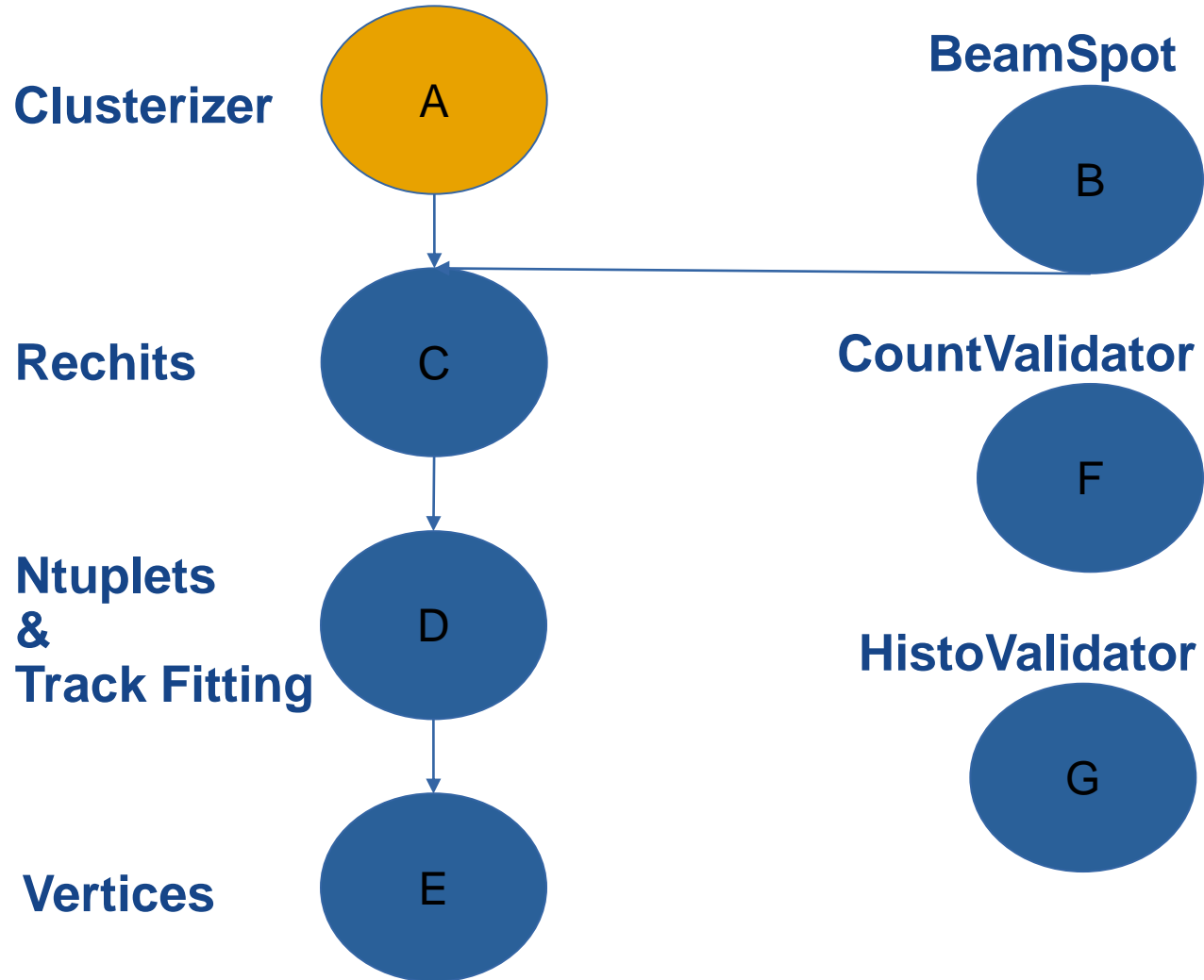
- Event and Event Setup
  - Container of data and metadata
- Product Registry
  - Interface between modules and the event
- ED Tokens
  - Used to access data from the event
- Task Scheduling & Multithreading
  - Currently implemented using Julia's native `@threads` macro
- Plugin Factory
  - Interface to instantiate modules



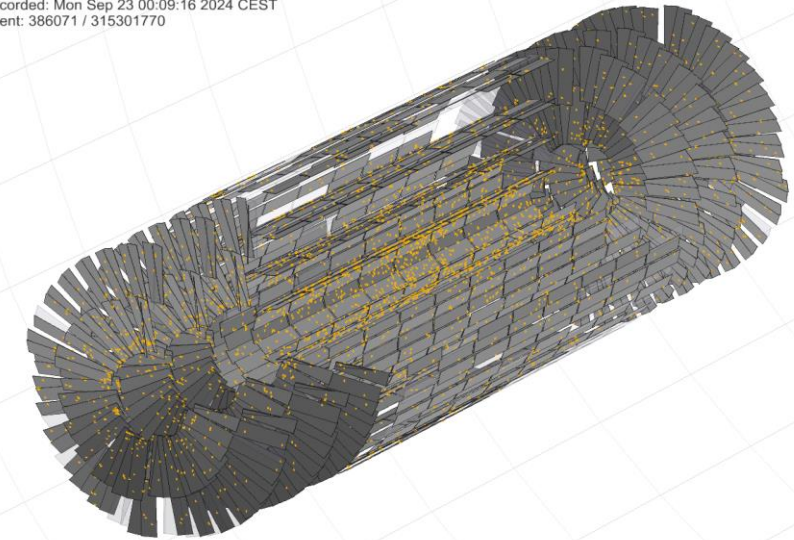
# Overview of the Application Modules



# Overview of the Application Modules

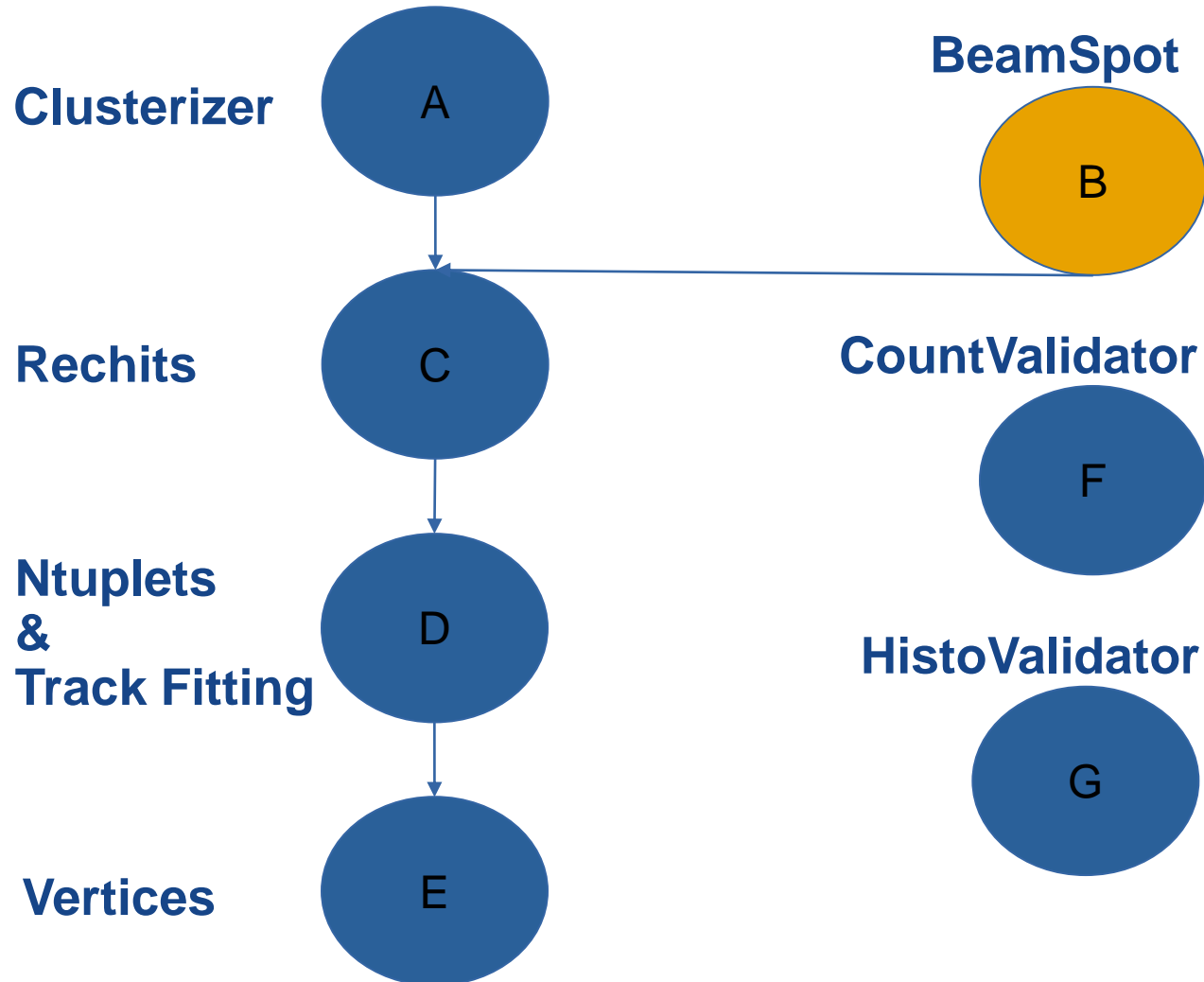


CMS Experiment at LHC, CERN  
Data recorded: Mon Sep 23 00:09:16 2024 CEST  
Run/Event: 386071 / 315301770

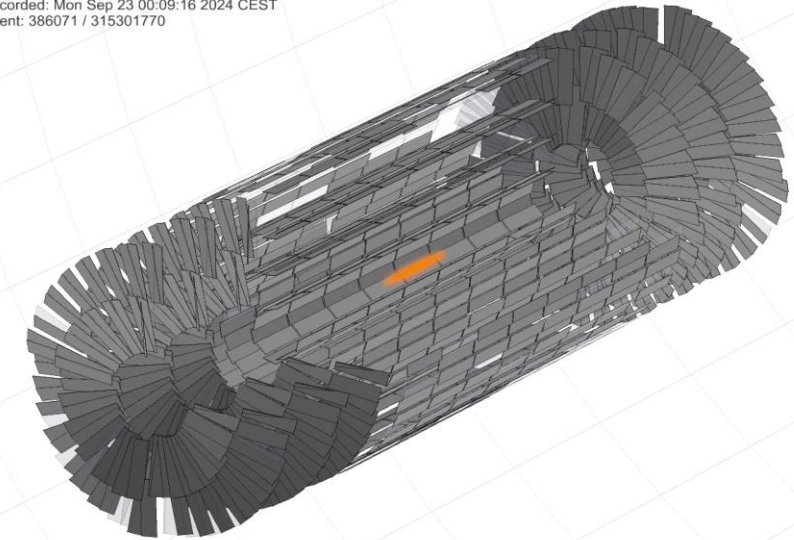


## Pixel Clusters

# Overview of the Application Modules



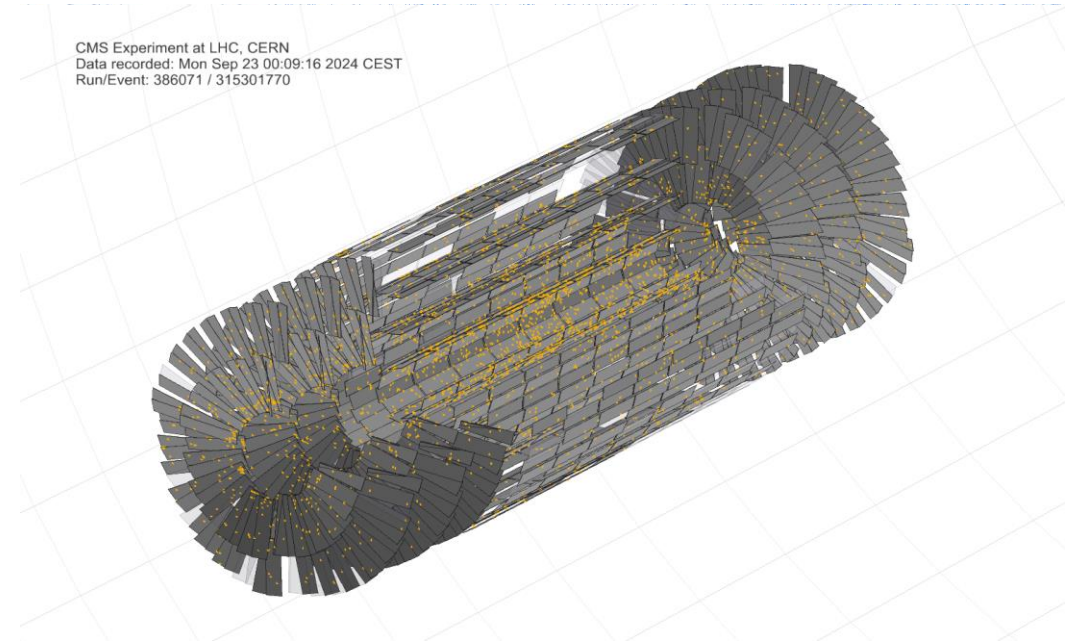
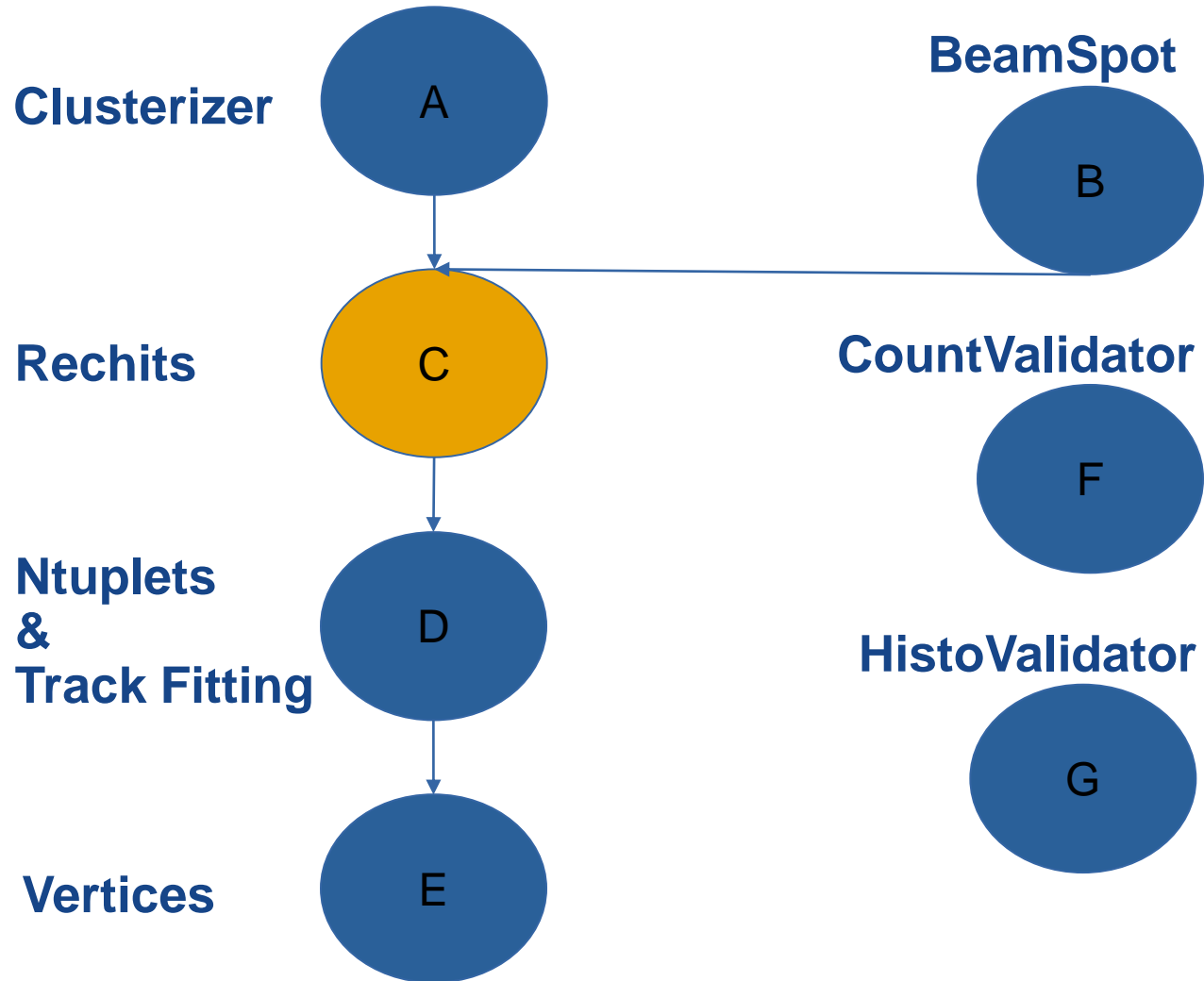
CMS Experiment at LHC, CERN  
Data recorded: Mon Sep 23 00:09:16 2024 CEST  
Run/Event: 386071 / 315301770



## Beam Spot

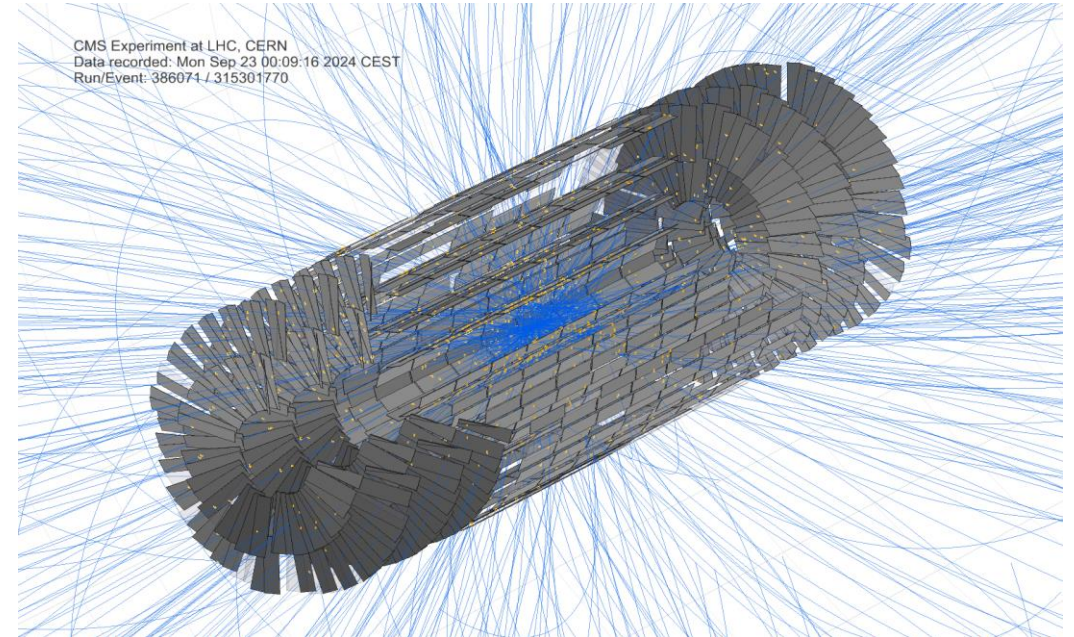
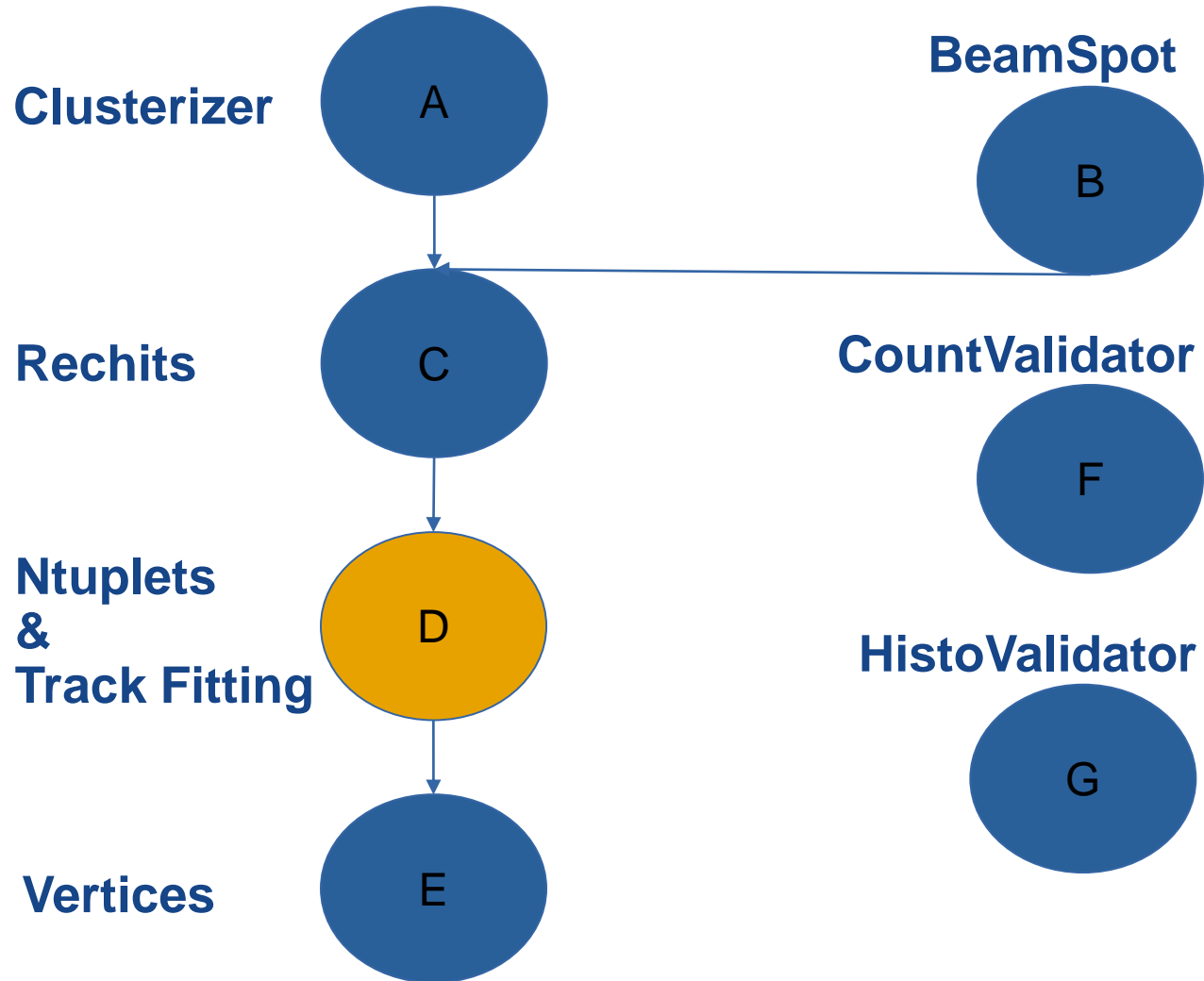


# Overview of the Application Modules



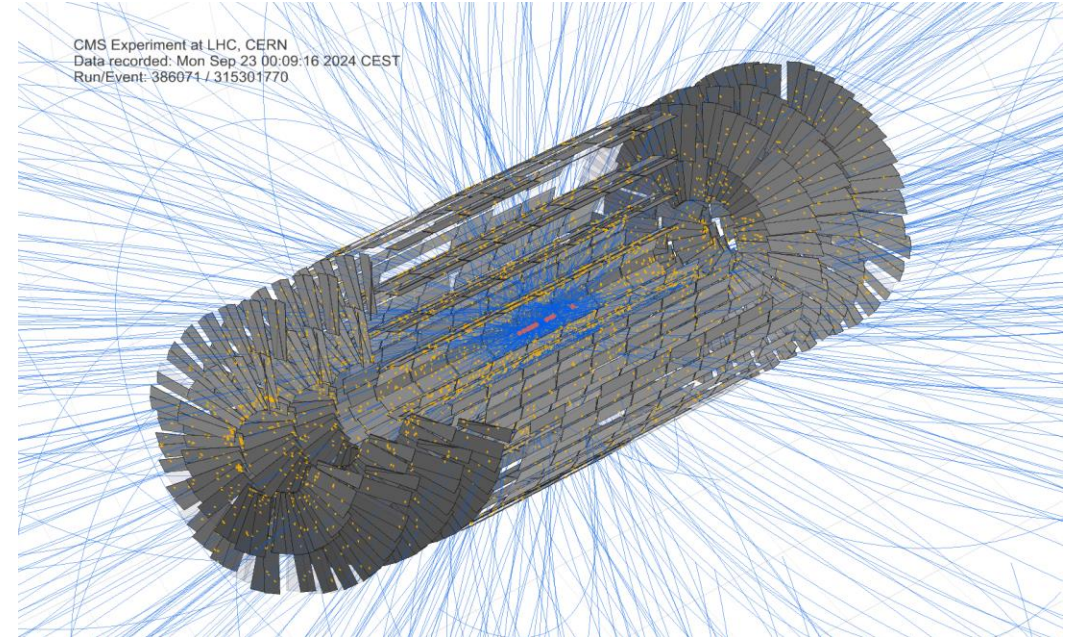
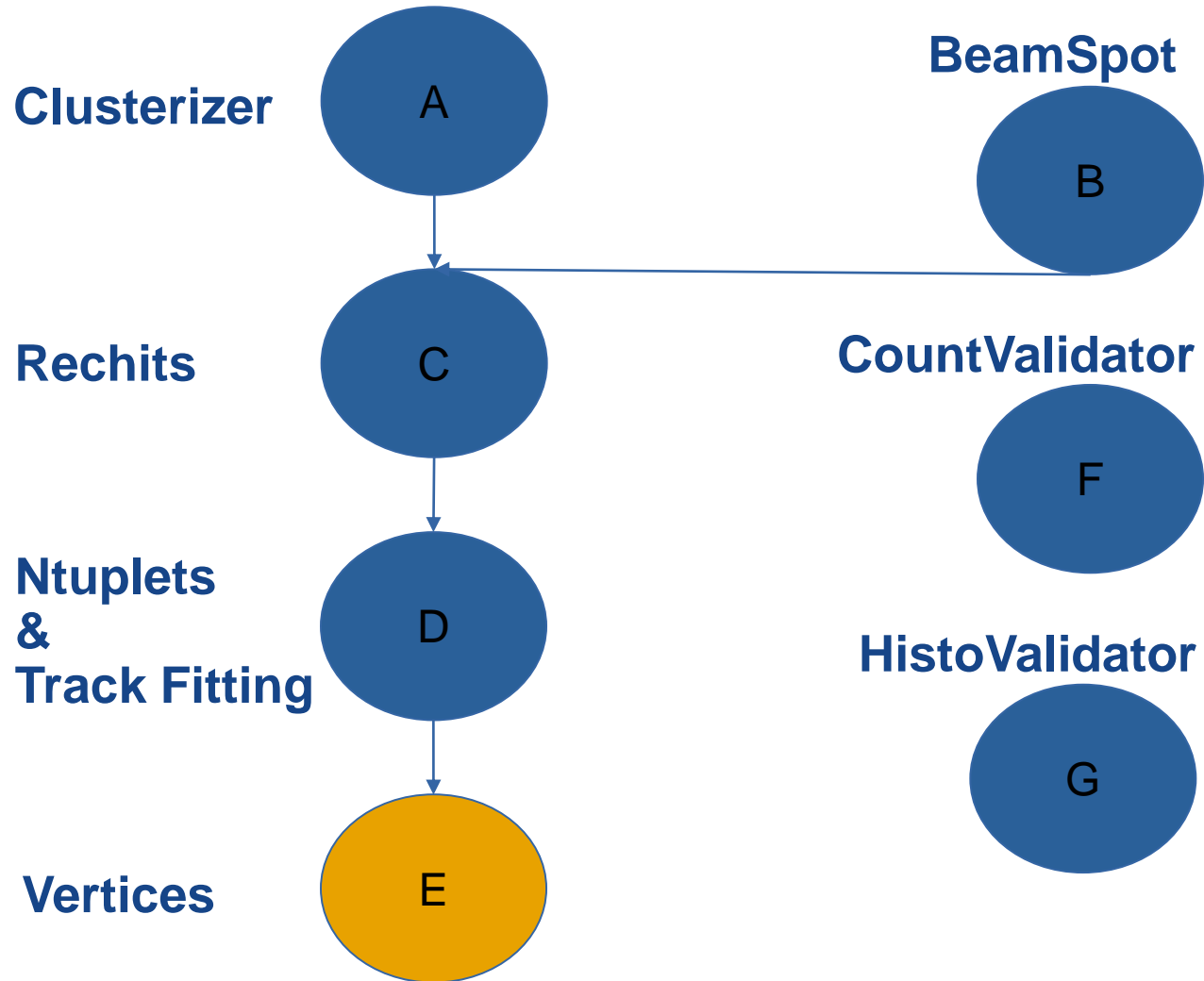
## Reconstructed Hits

# Overview of the Application Modules



## Pixel Tracks

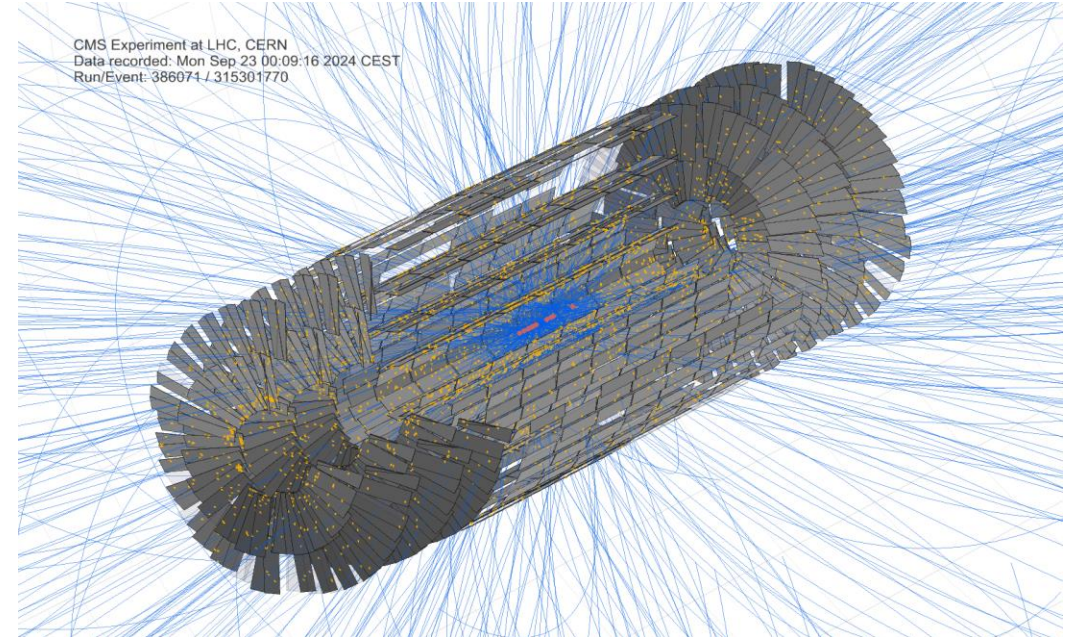
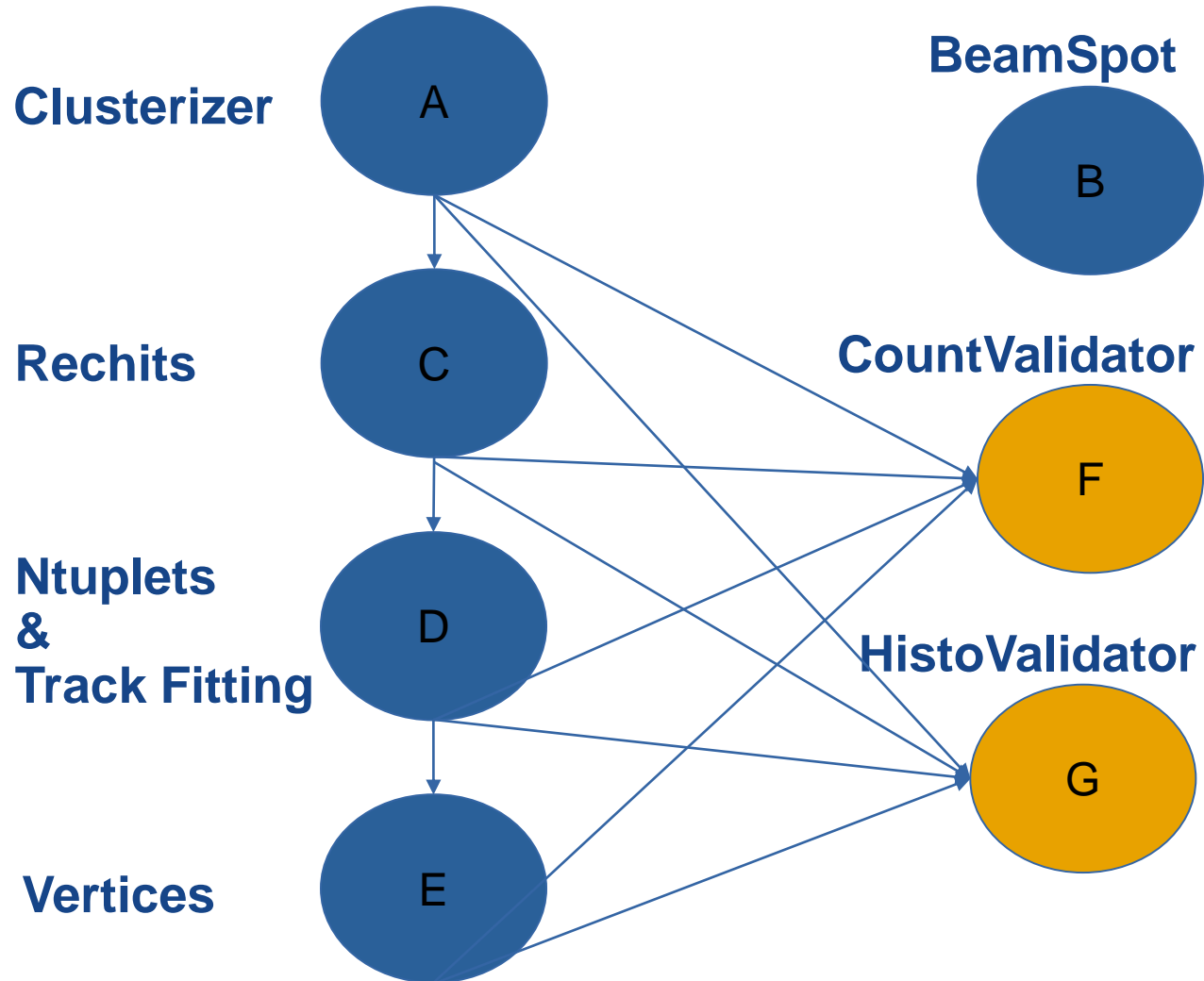
# Overview of the Application Modules



## Vertices



# Overview of the Application Modules

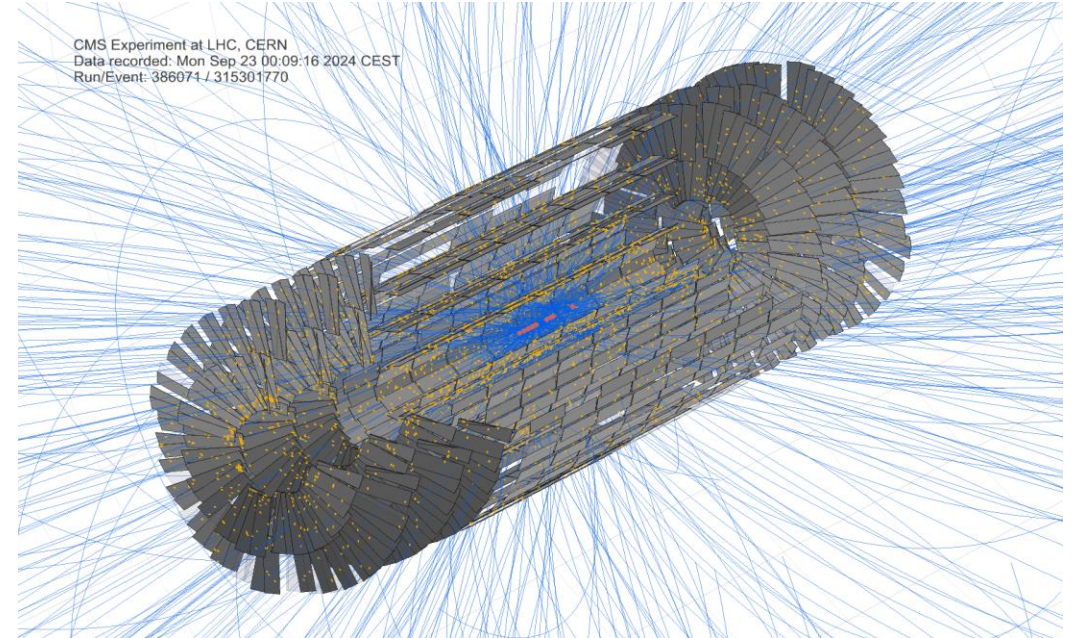
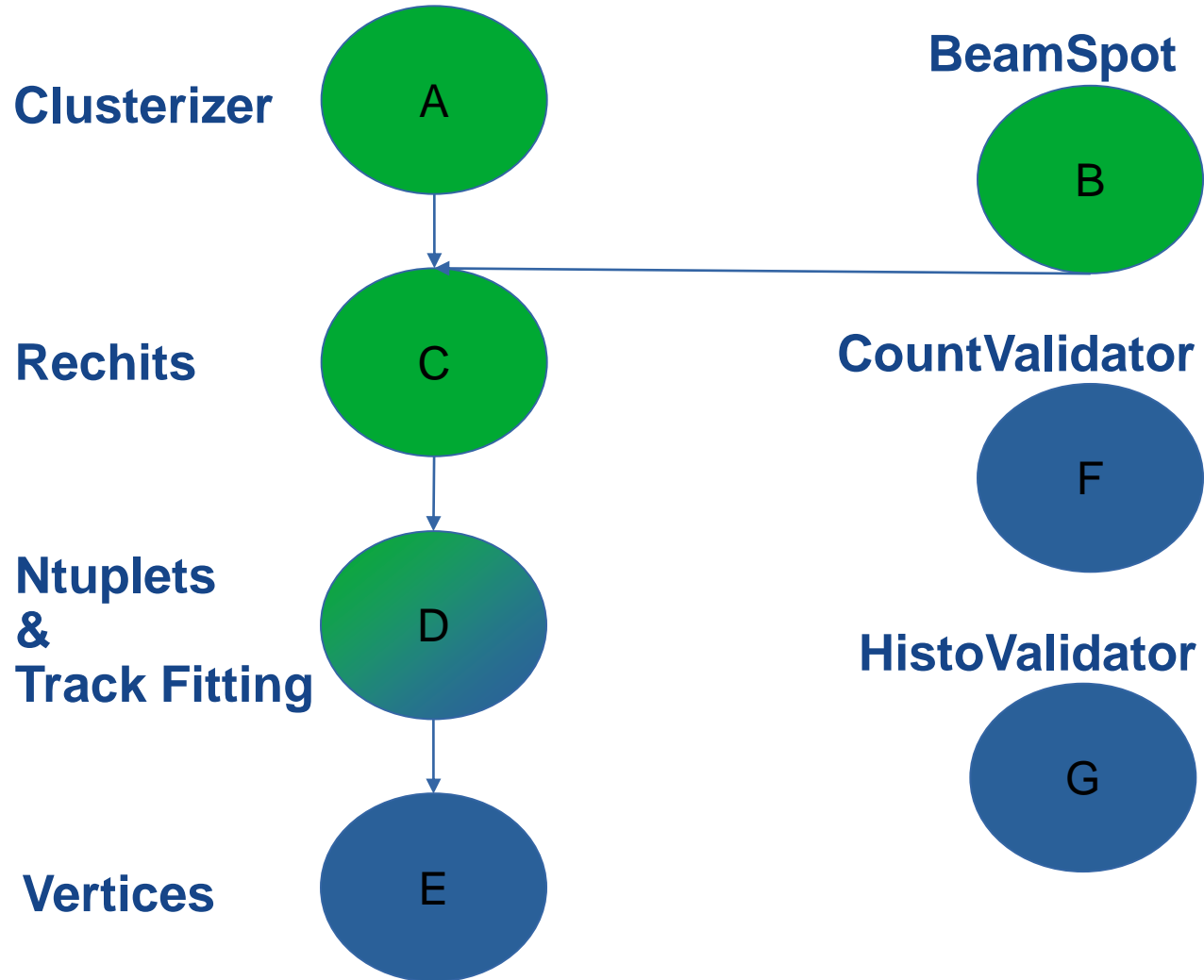


## Validators



# Highlights From The Past Year:

## 100% of Local Reconstruction



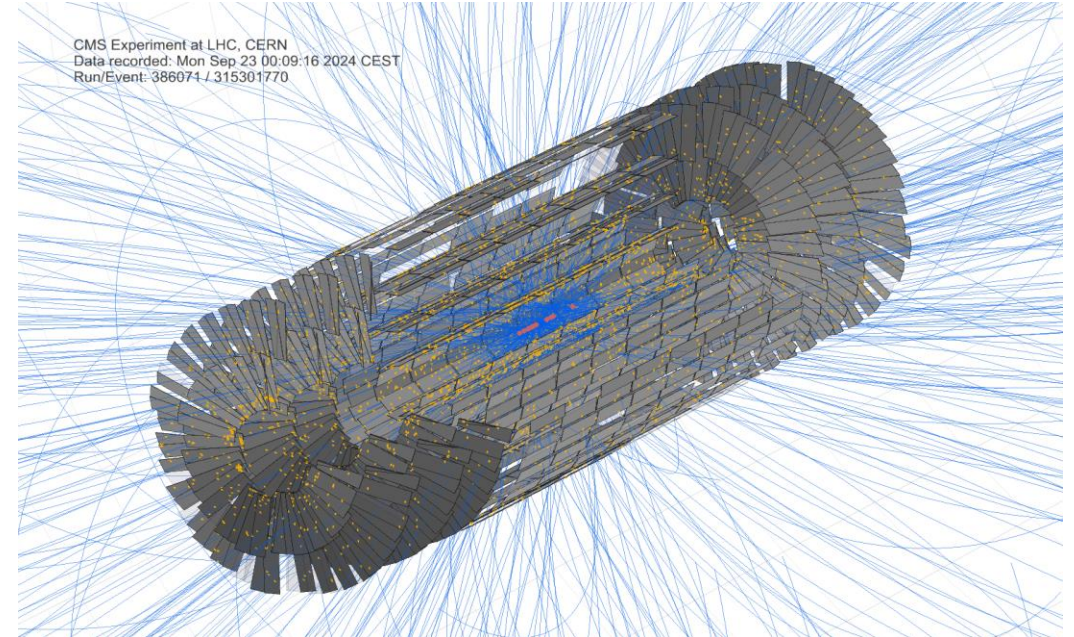
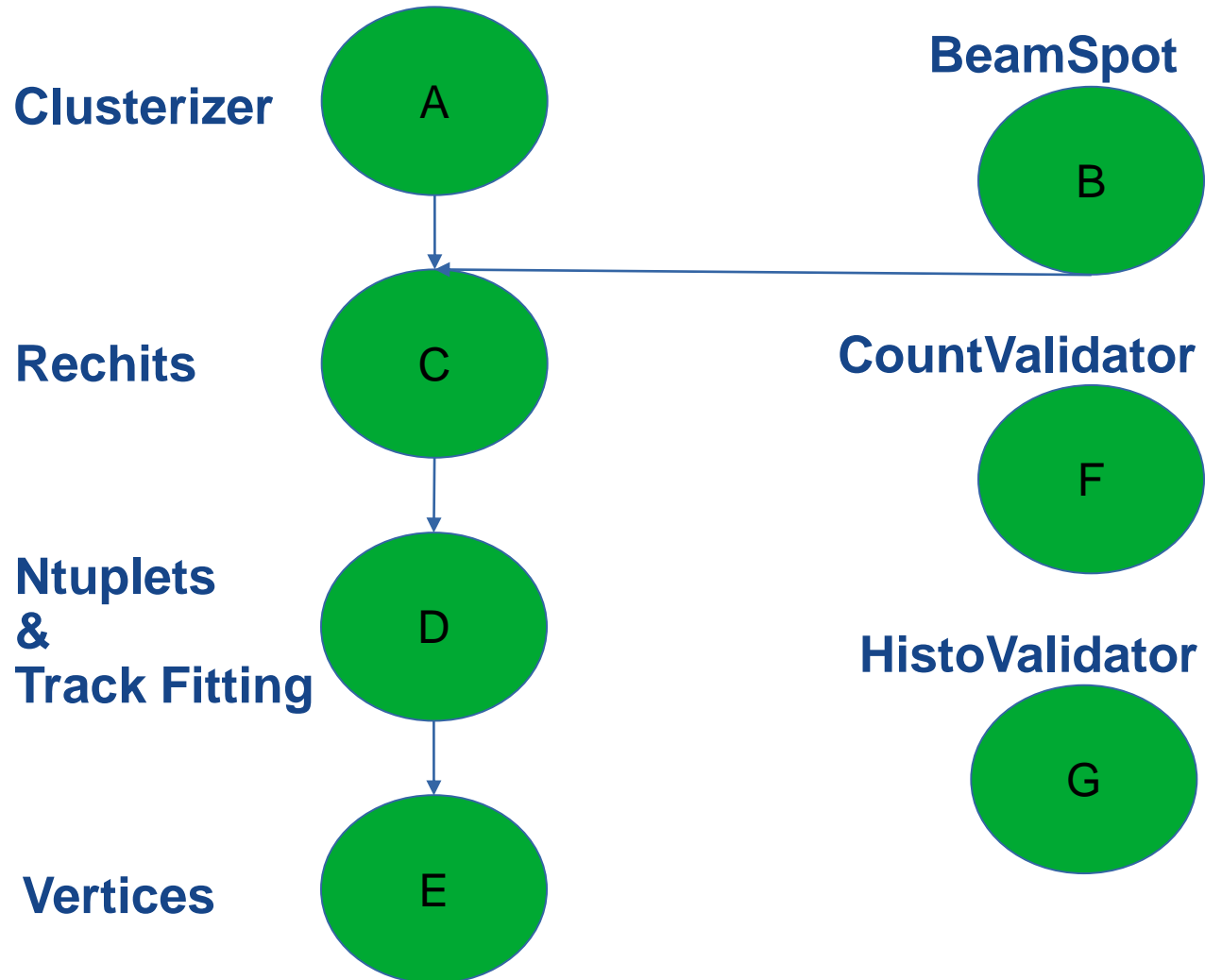
**C++**

**31.4 evts/s**

**Julia**

**28.5 evts/s**

# Recent Developments: **100%** of Global Reconstruction



## Ported & Validated:

- Track Reconstruction
- Vertex Reconstruction
- Validators (CountValidator, HistoValidator)

# Performance

---



# Benchmark Setup

---

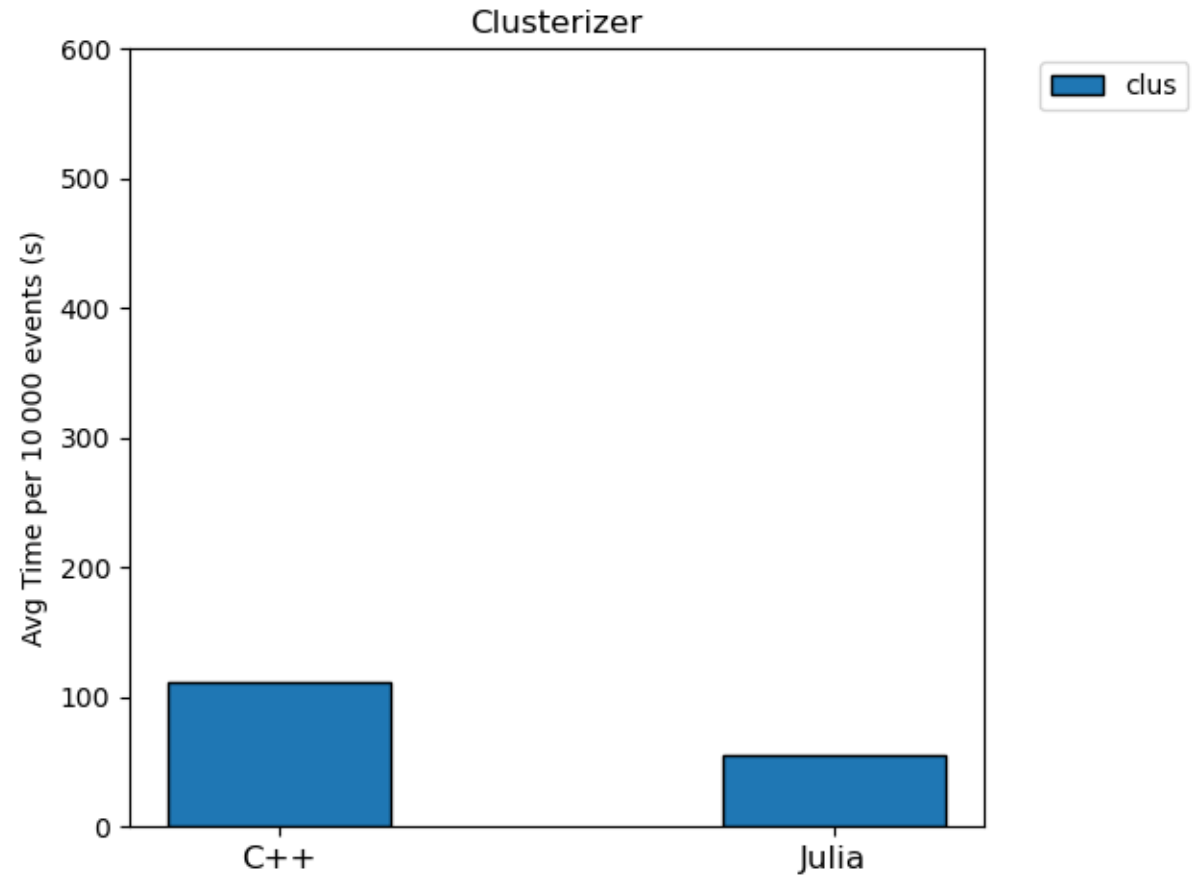
- **CPU: 2 x AMD EPYC 9534 64-Core Processor (125 cores available)**
- **Operating System: AlmaLinux 9.6 (Sage Margay)**
- **Execution Mode: Single-threaded**
- **Warm-up run: 1000 events**
- **Main performance test: 10000 events**
- **Repetitions: 10**



# Clusterizer

**C++: 111.1 Seconds**

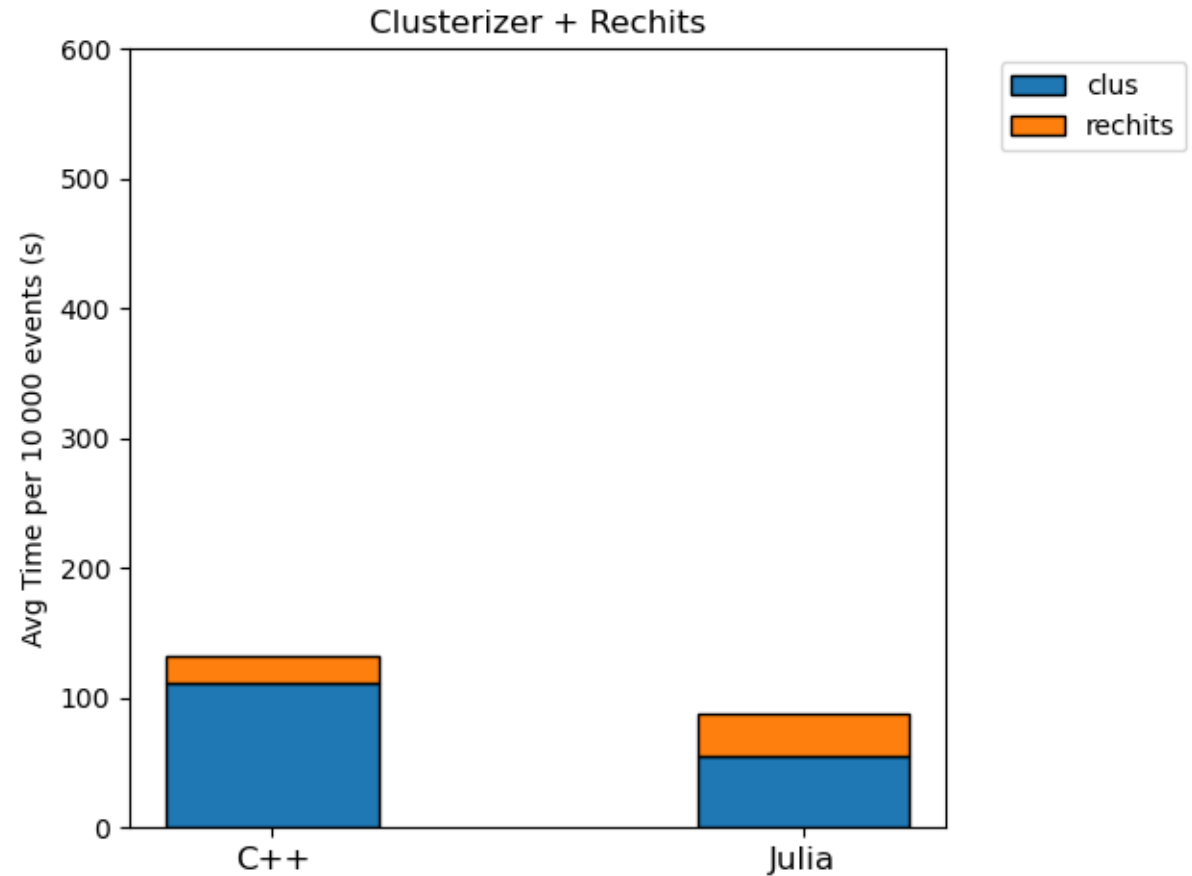
**Julia: 55.5 Seconds**



# Rechits

**C++: 20.9 Seconds**

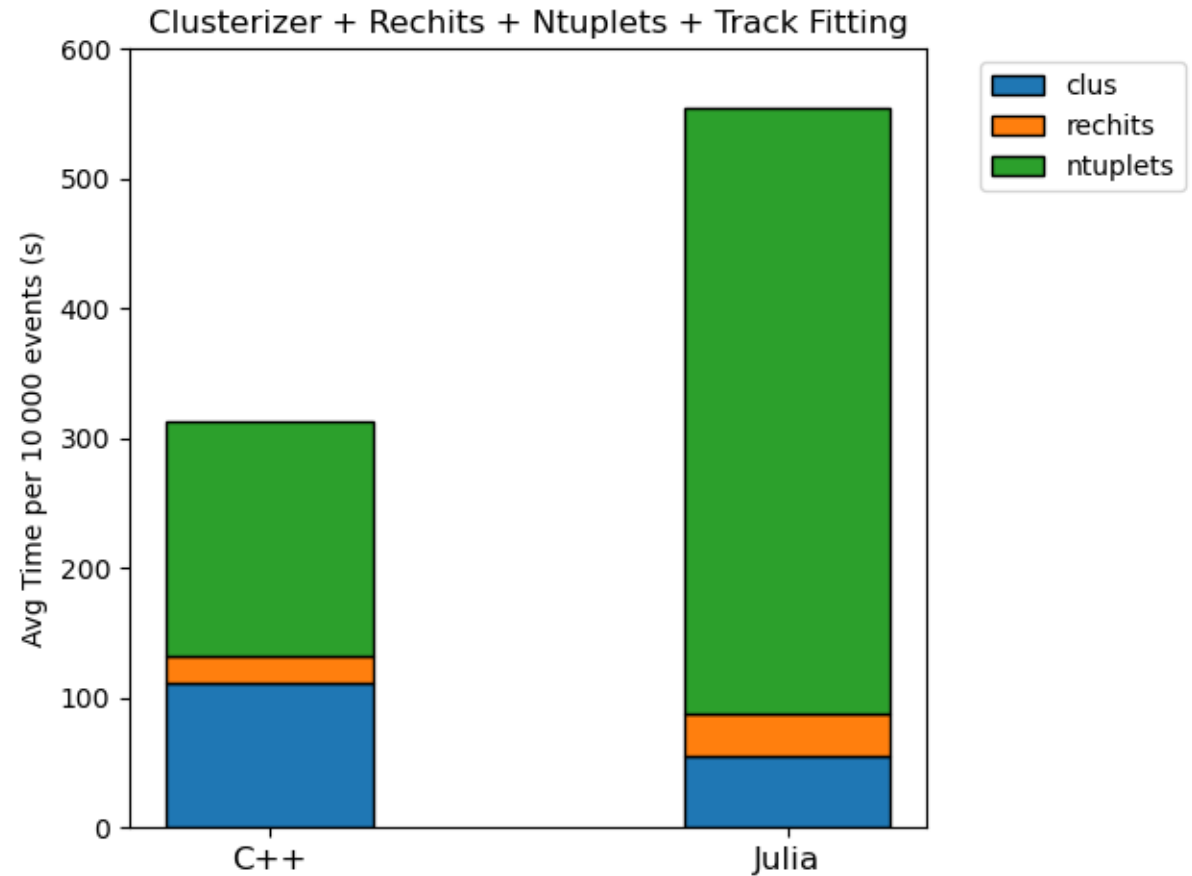
**Julia: 32.8 Seconds**



# Ntuplets + Track Fitting

**C++: 180.7 Seconds**

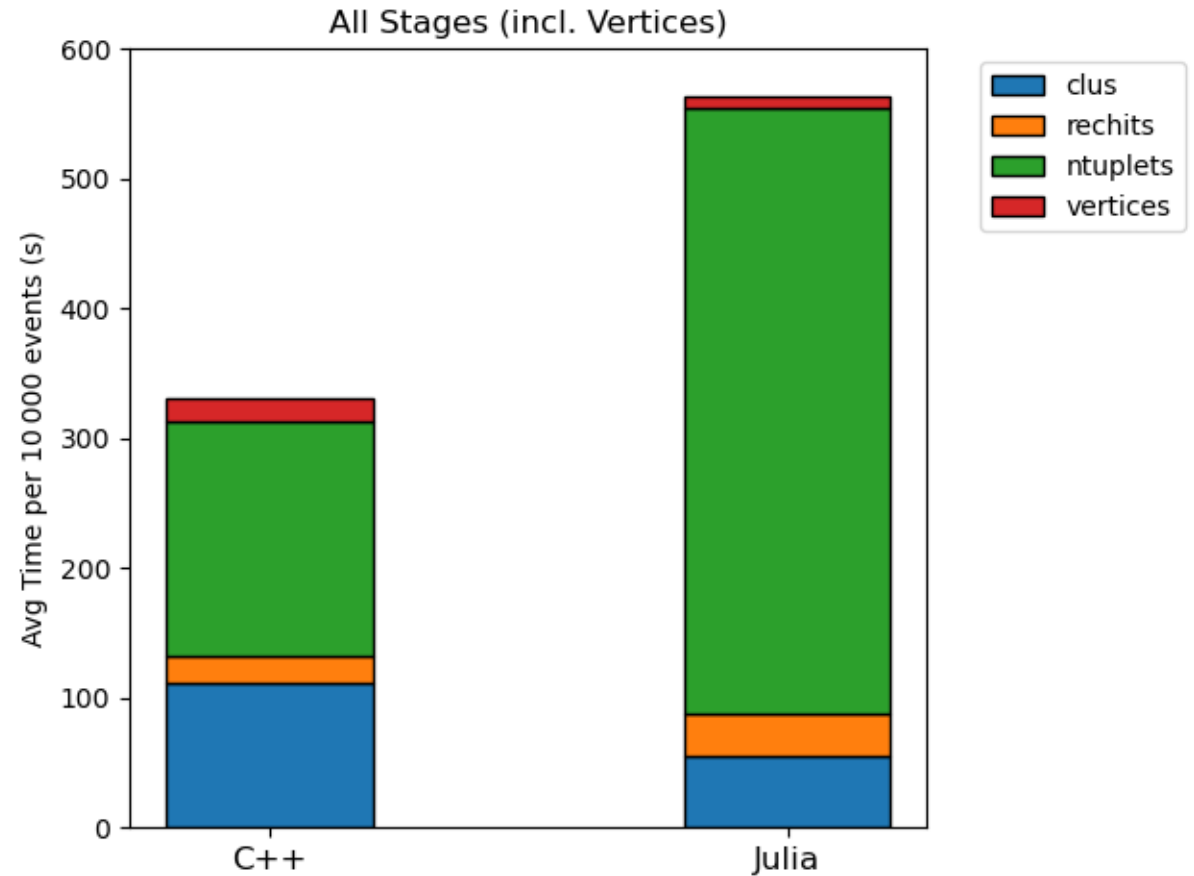
**Julia: 466.9 Seconds**



# Vertices

**C++: 17.7 Seconds**

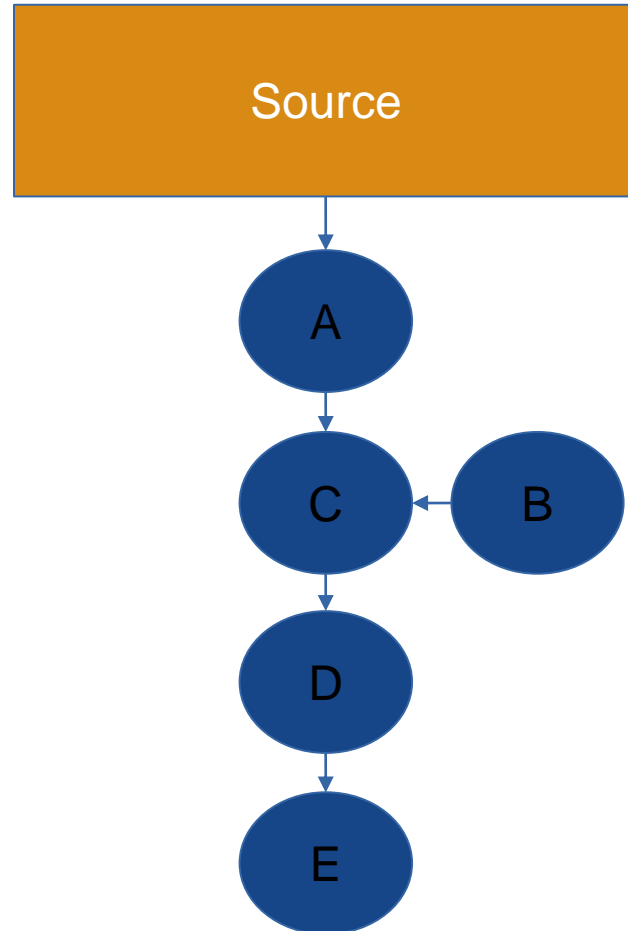
**Julia: 7.9 Seconds**



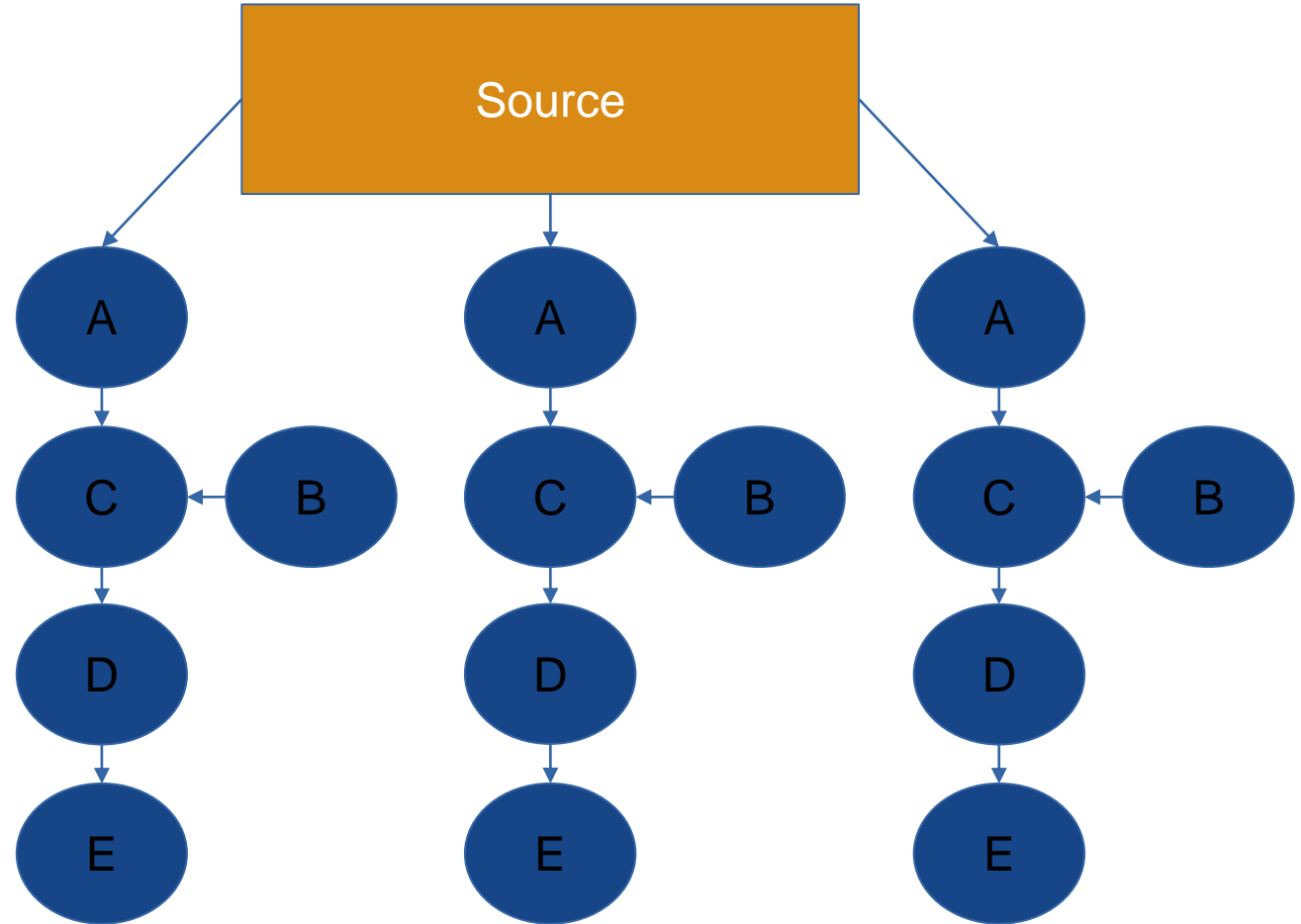


# Multithreading

## Sequential Framework



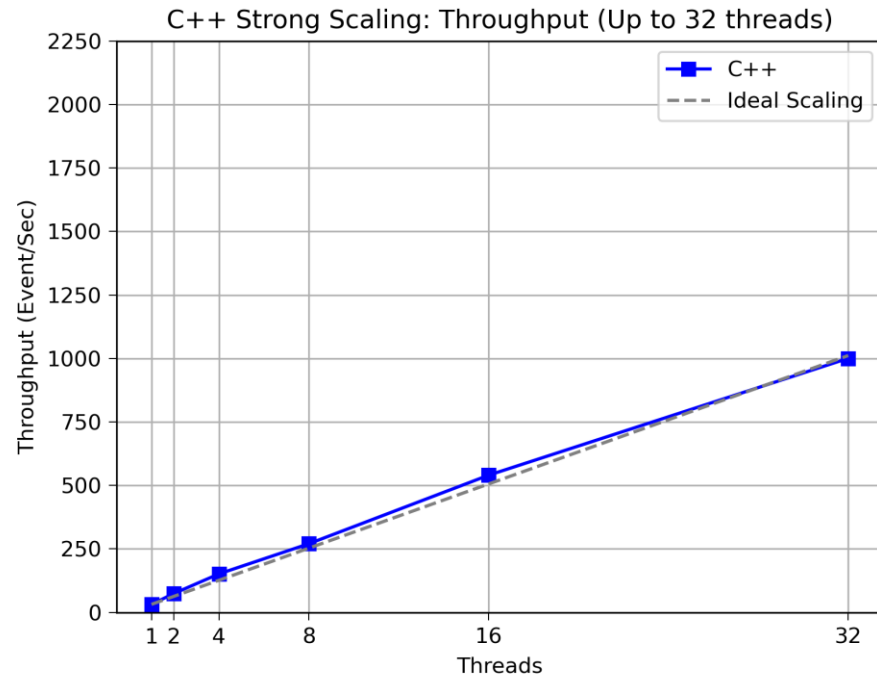
## Parallelized Framework



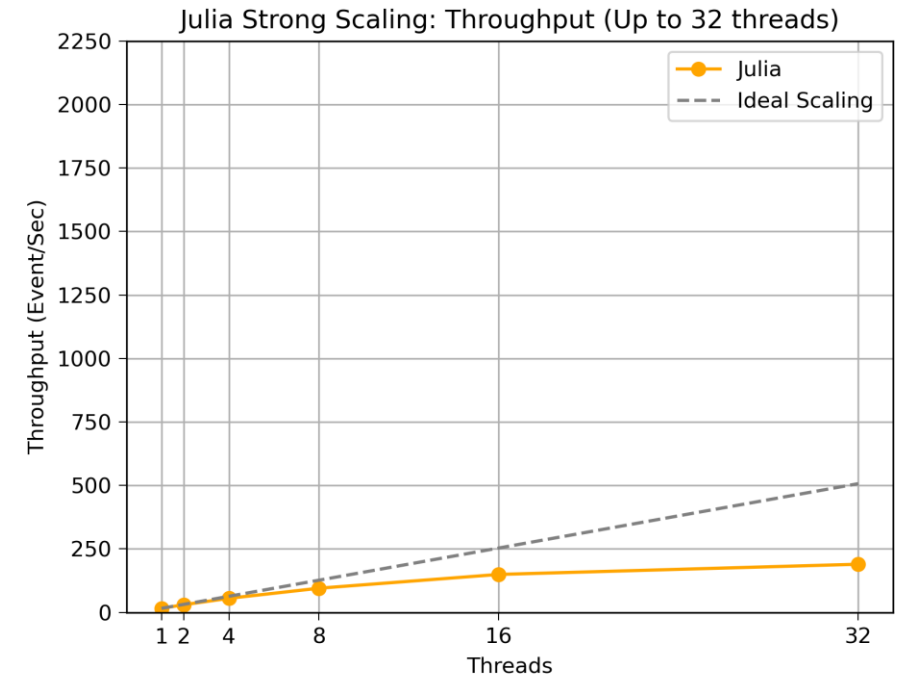
# Benchmark Setup

- **CPU: 2 x AMD EPYC 9534 64-Core Processor (125 cores available)**
- **Operating System: AlmaLinux 9.6 (Sage Margay)**
- **Execution Mode: Multi-threaded**
- **Warm-up run: 1000 events**
- **Main performance test: 20,000 events C++, 10,000 events Julia**
- **Repetitions: 4**

# Results: Throughput

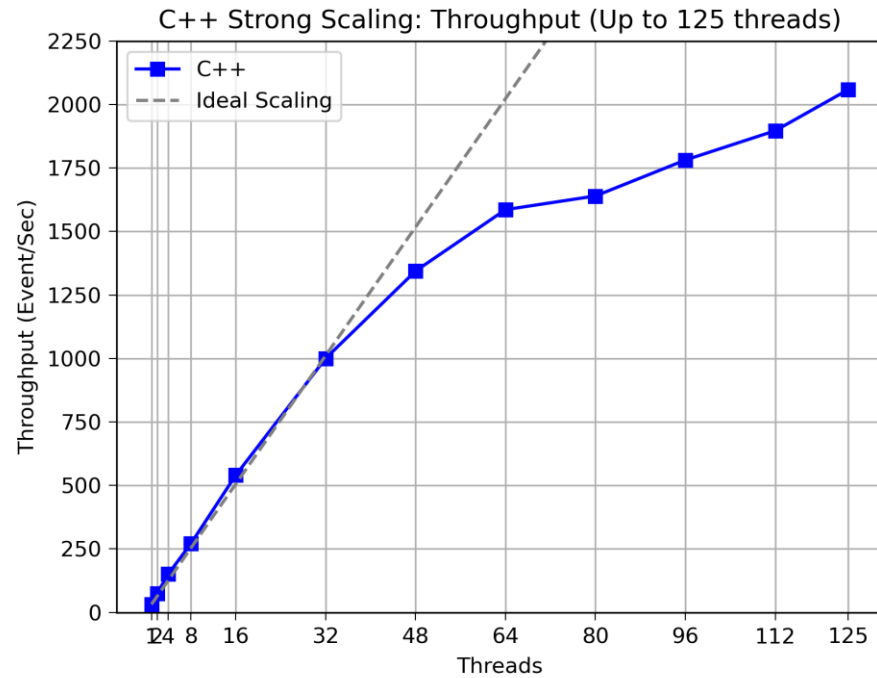


C++

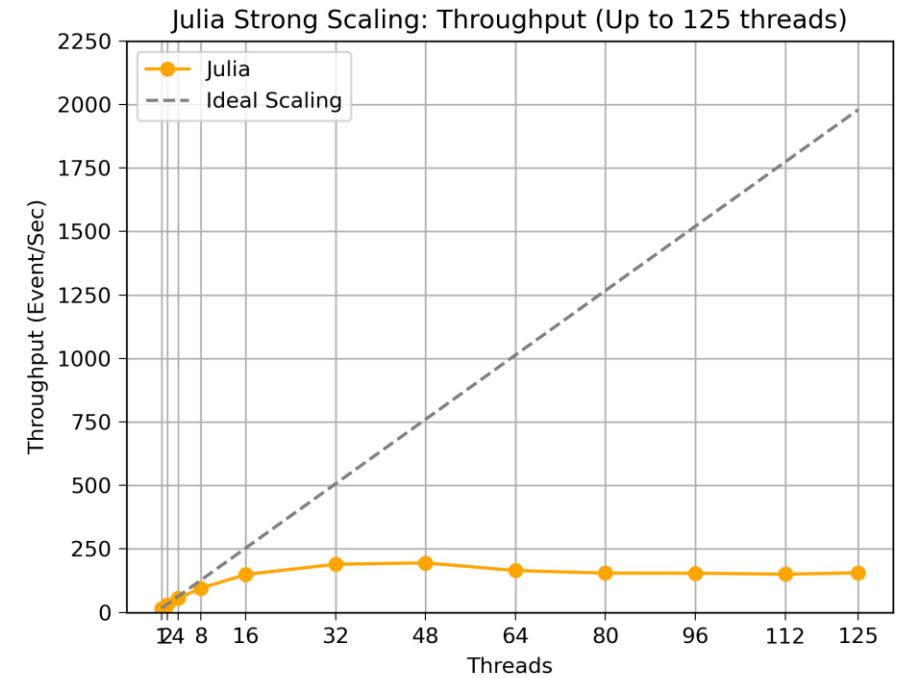


Julia

# Results: Throughput



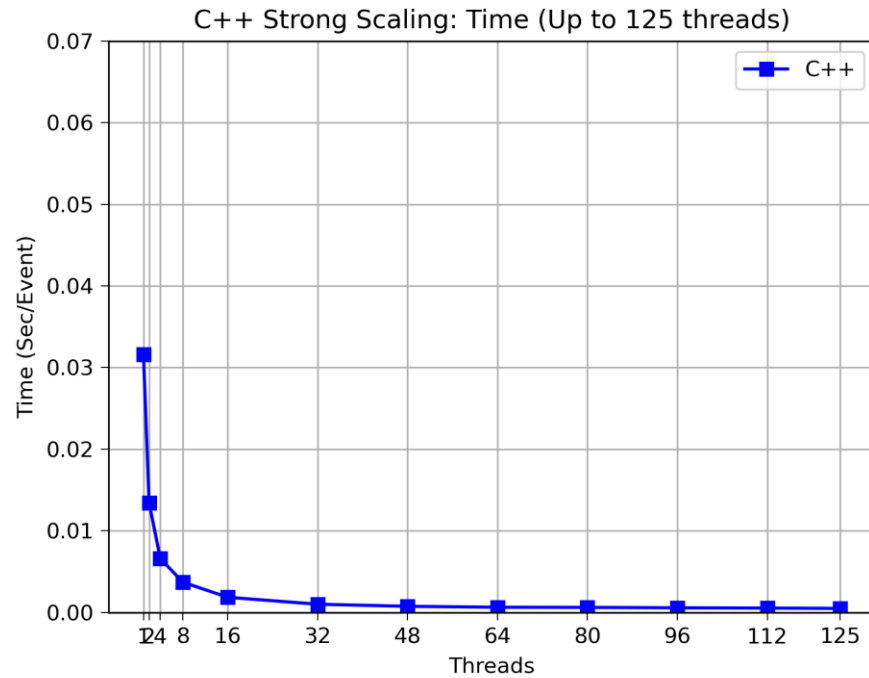
C++



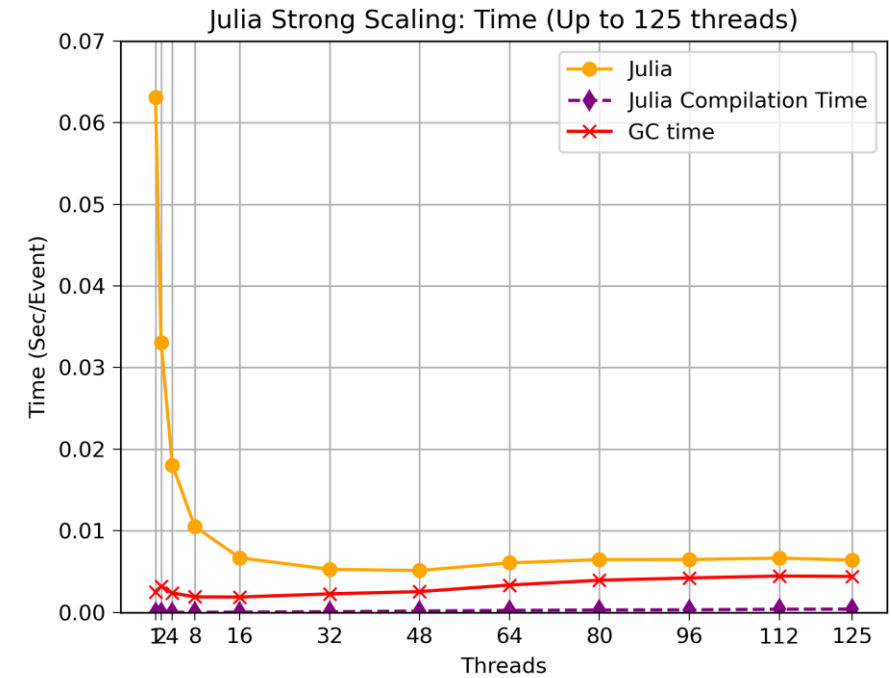
Julia



# Results: Execution Time

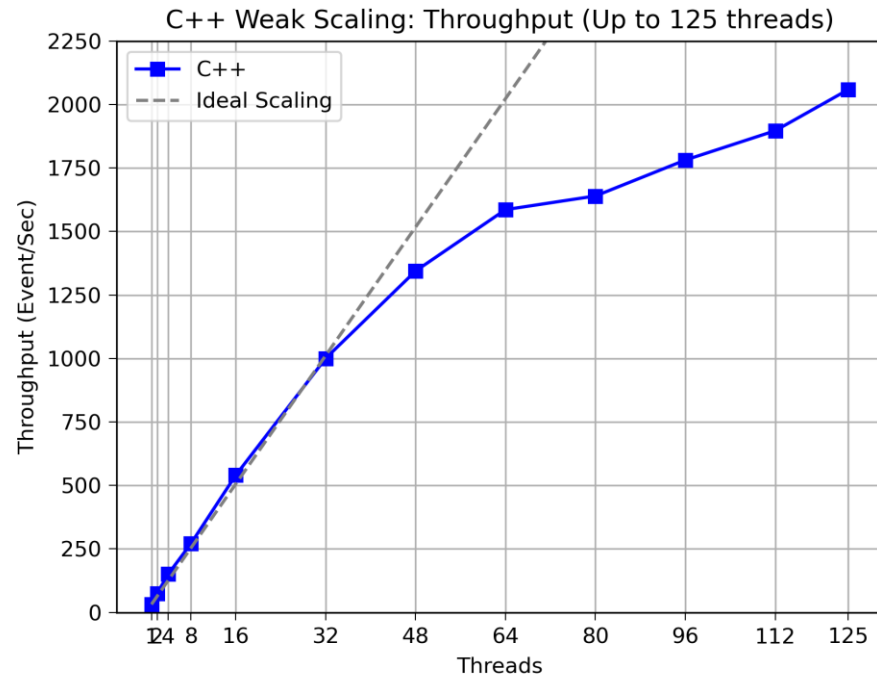


C++

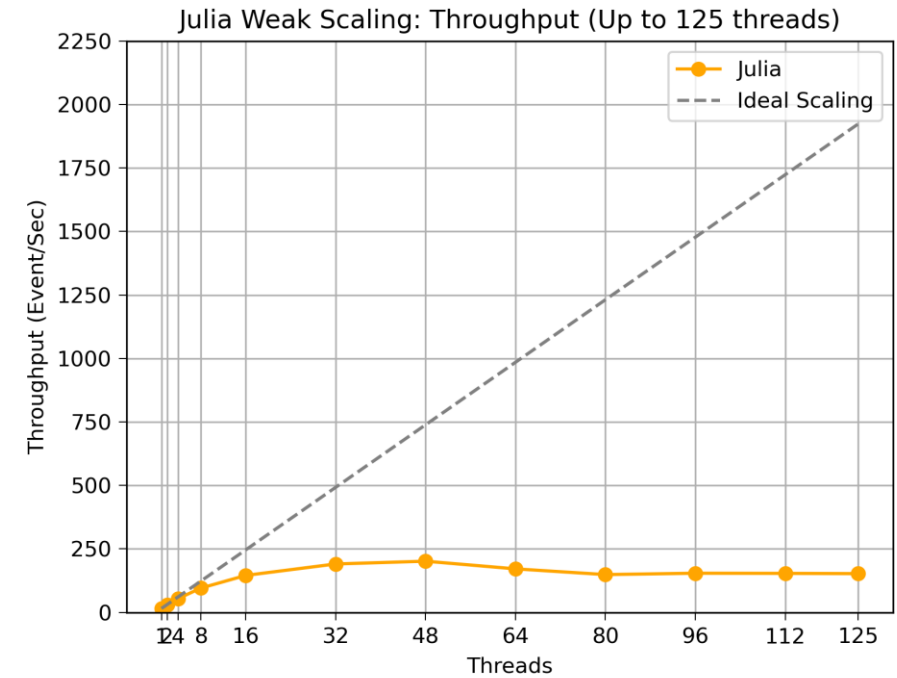


Julia

# Results: Throughput

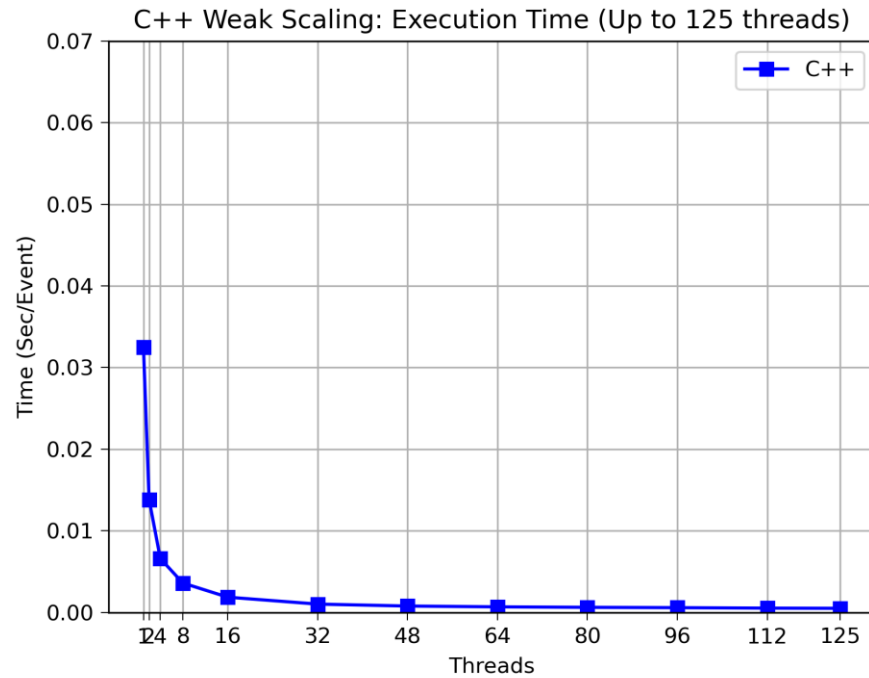


C++



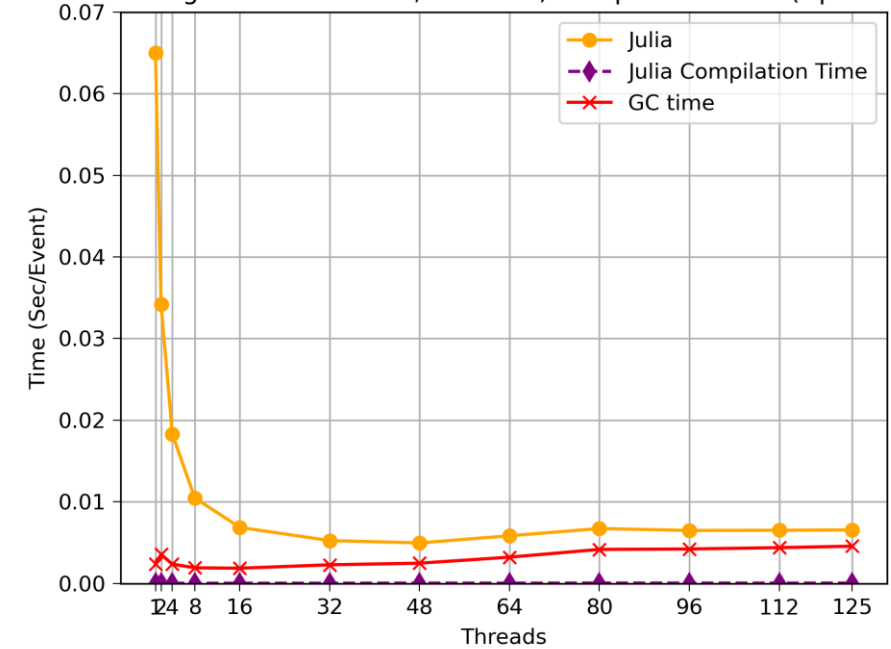
Julia

# Results: Execution Time



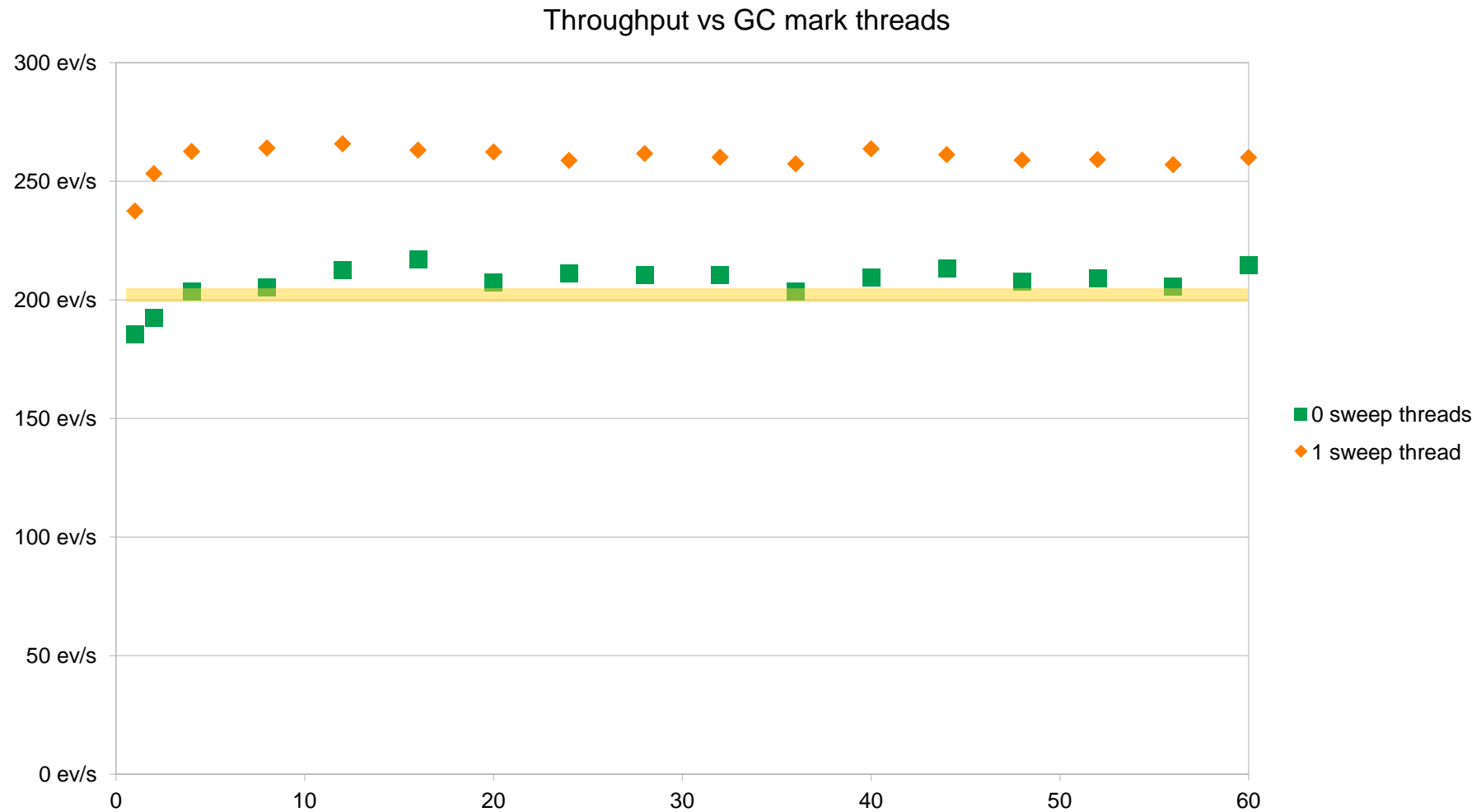
C++

Julia Weak Scaling: Execution Time, GC Time, Compilation Time (Up to 125 threads)

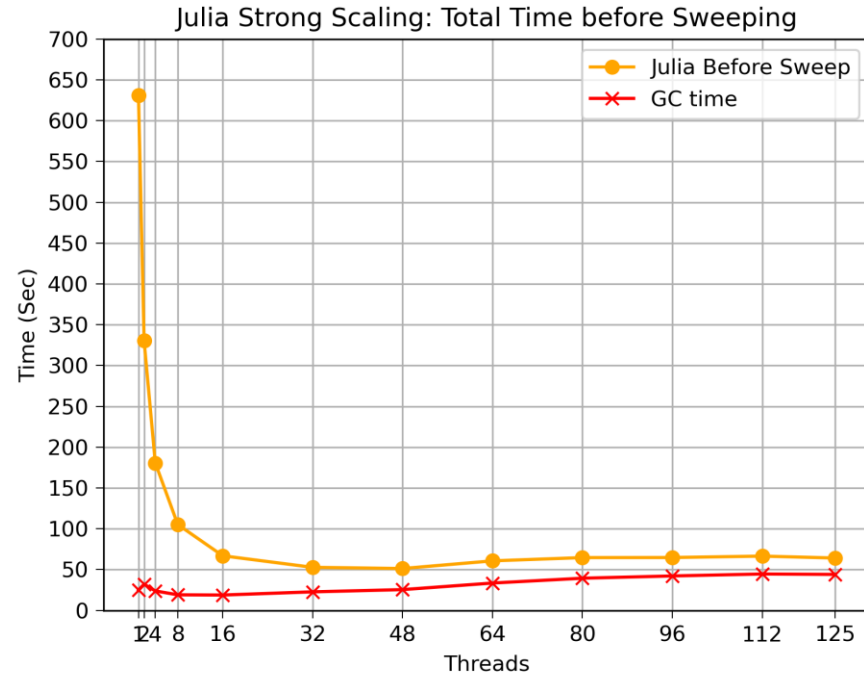


Julia

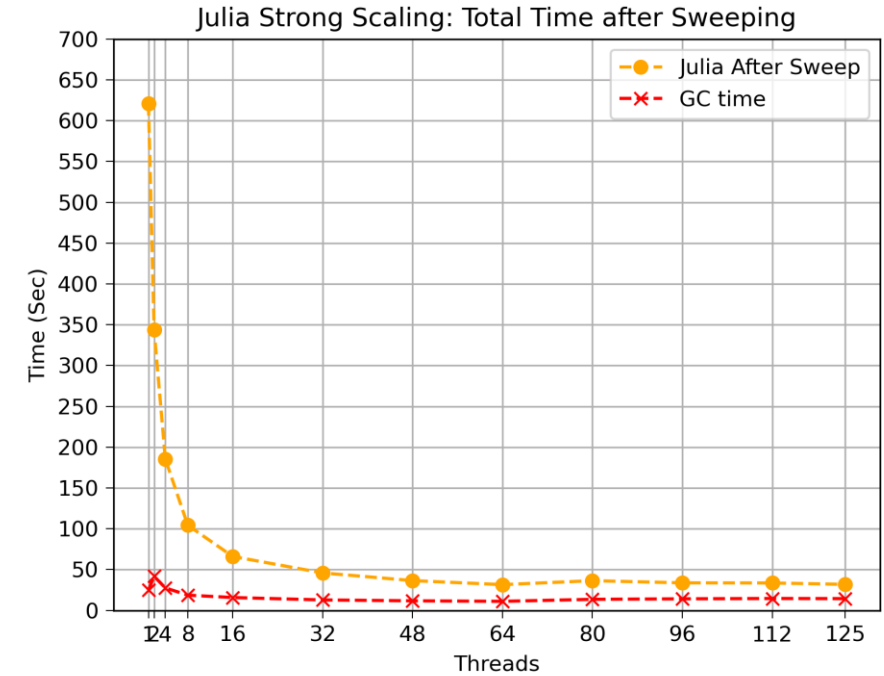
# GC Marking Threads: Performance



# Results: Total Execution Time

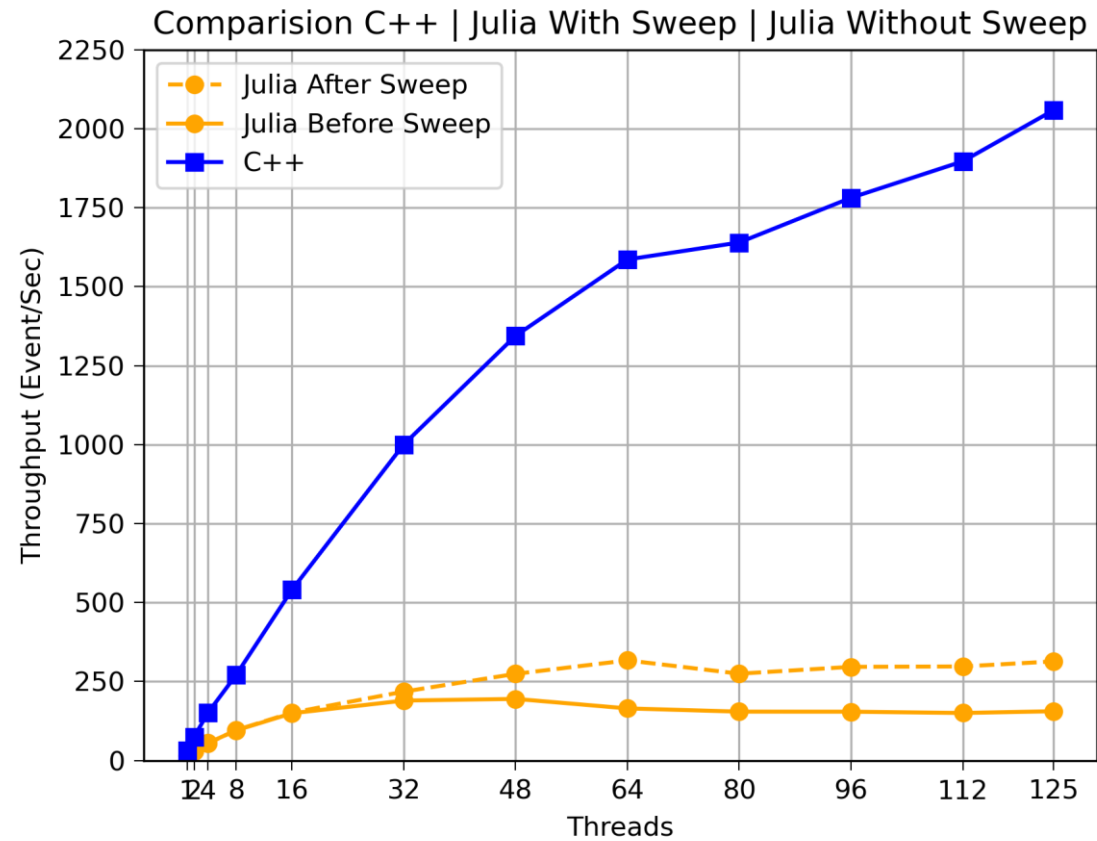


Before



After

# Comparison





# Ahead of Time Compilation

## ➤ Why?

Cut JIT time & stabilize runs for benchmarks/deployment.

## ➤ Tried tools

PackageCompiler.jl (1.12.0-rc1) → broken (jl\_get\_binding\_wr arg mismatch)

PackageCompiler.jl (1.11) → works

juliac (1.12.0-rc1, --trim=no) → works, smaller & faster compile than PC

# Ahead of Time Compilation - Results

## Results (AMD 5700G, 1000 events, 1 thread)

Compiler / Mode	Julia ver	Compile time	Binary size	Runtime (evt/s)	GC time (main)
<b>JIT (baseline)</b>	1.12-rc1	—	—	54.80 s (18.25 evt/s)	1.8 s (~3.3%)
<b>PackageCompiler</b>	1.11.5	~630 s	~24 KB exe + 544 MB deps	57.28 s (17.46 evt/s)	3.1 s (~5.4%)
<b>juliac --trim=no</b>	1.12-rc1	~120 s	301 MB	58.37 s (17.13 evt/s)	4.4 s (~7.5%)

# Ahead of Time Compilation – Next Steps

- Investigate why GC grows with juliac image
- Re-test when 1.12 final + PC fix lands; retry --trim=safe
- Integrate make target to build & warm-up automatically

# Next Steps and Future Work

- Finalize and validate local reconstruction on GPU
- Optimize track fitting (Bottleneck for CPU Single Threaded Version)
- Investigate GC overhead with increasing threads
- Test AOT compilation with future release of juliac



# Conclusion

---

- Ported serial version to Julia
- Implemented multithreading using Julia's native `@threads` macro
- Implemented AOT compilation using `juliac`



**NextGen**  
Next Generation Triggers