Porting the CMS pixel reconstruction to Julia: preliminary results

Maya Ali
Mohamad Ayman Charaf
Mohamad Khaled Charaf
Ruba El Houssami
Dr. Andrea Bocci
Dr. Phillipe Gras

October 1st , 2024

The Team

Maya Ali

4^{th} year Computer and Communications Engineering Student at AUB

Khaled Charaf

4^{th} year Computer Science Engineering Student at AUB

Ayman Charaf

4^{th} year Computer Science Engineering Student at AUB

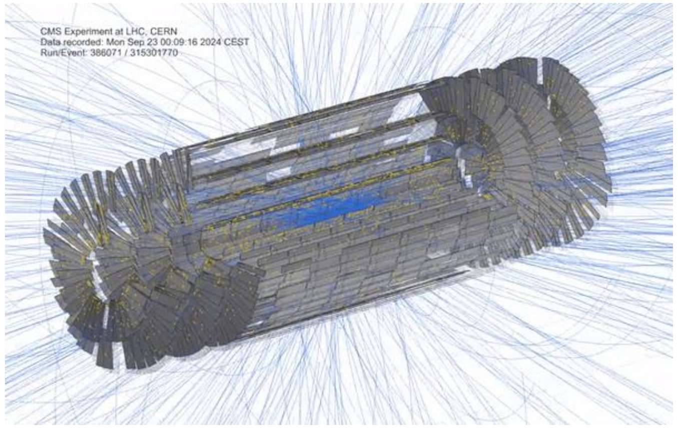
Ruba El Housami

3^{rd} year Computer Science Engineering Student at AUB

Outline

Project Context

- Evaluation of Julia as a language for High Energy Physics:
  - <u>General evaluations</u>
  - <u>Jet clustering evaluations</u>



- 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 on multiple CPU and GPU technologies (OpenMP, CUDA, HIP, SYCL, Kokkos, etc.).

Main Goal

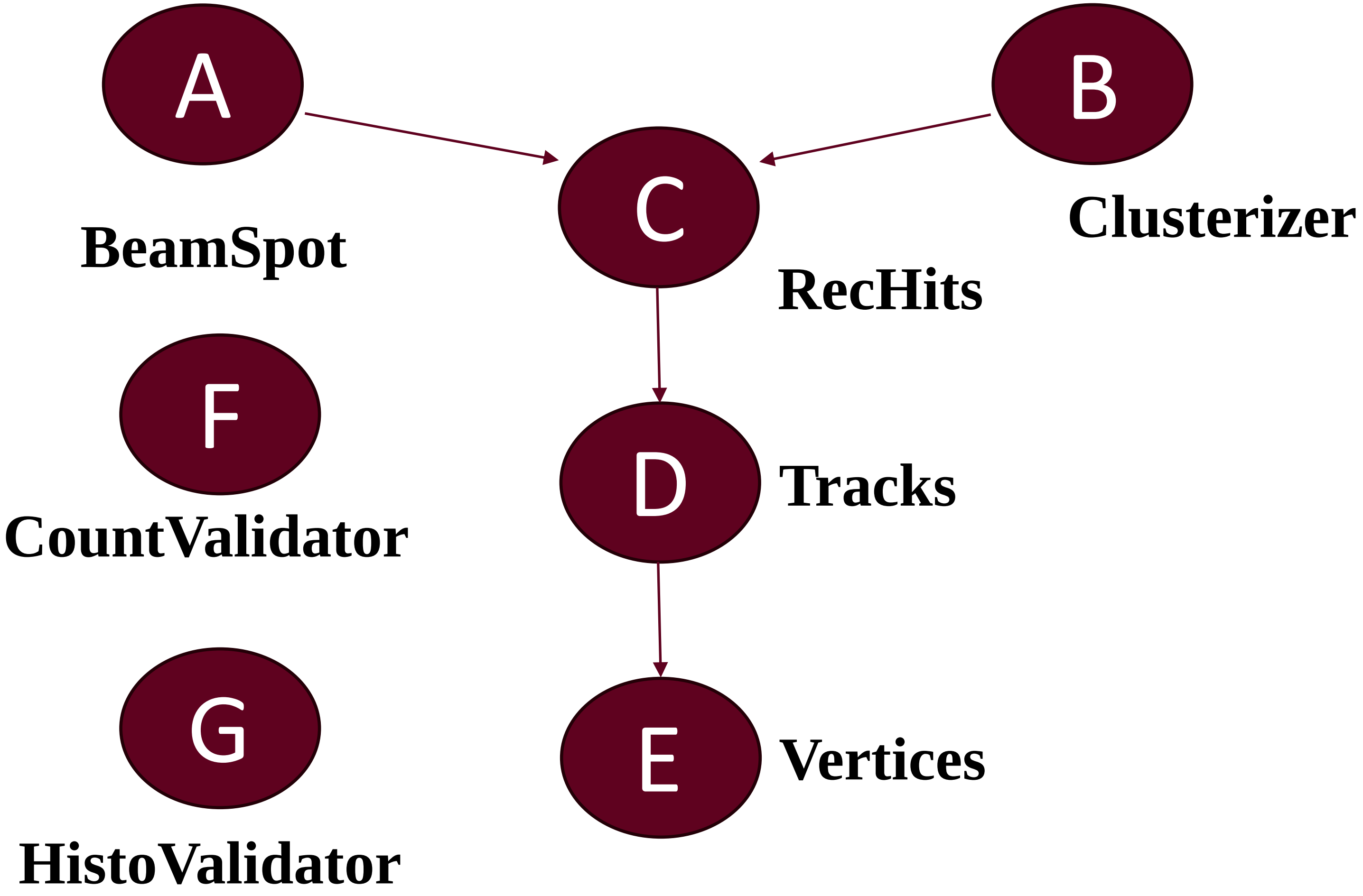
To evaluate the feasibility of using Julia for large-scale HEP applications by:

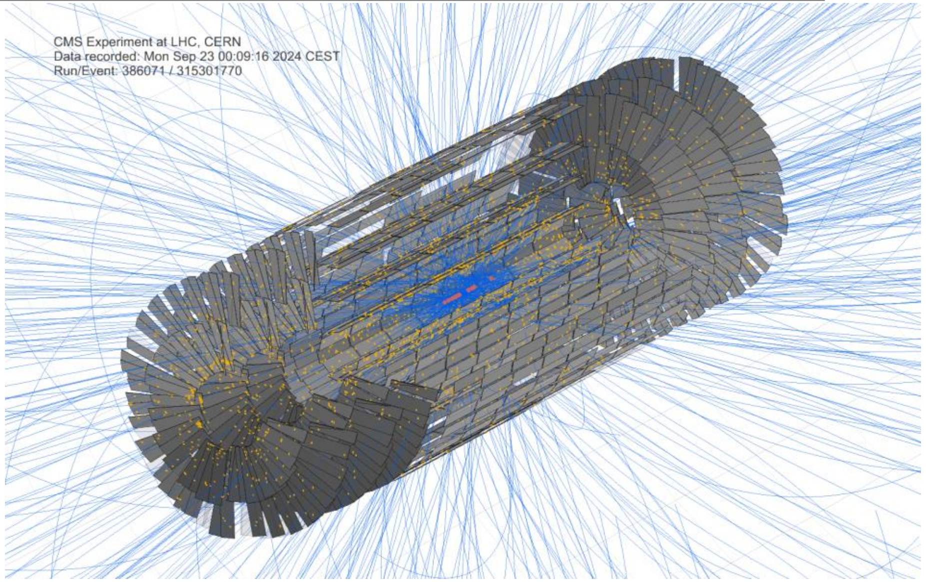
1. Re-writing pixel track reconstruction algorithms, and comparing it with existing C++ application.
2. Test Julia's ability to handle multithreading, GPU acceleration, and overall flexibility in a realistic HEP environment.

Intermediate Steps

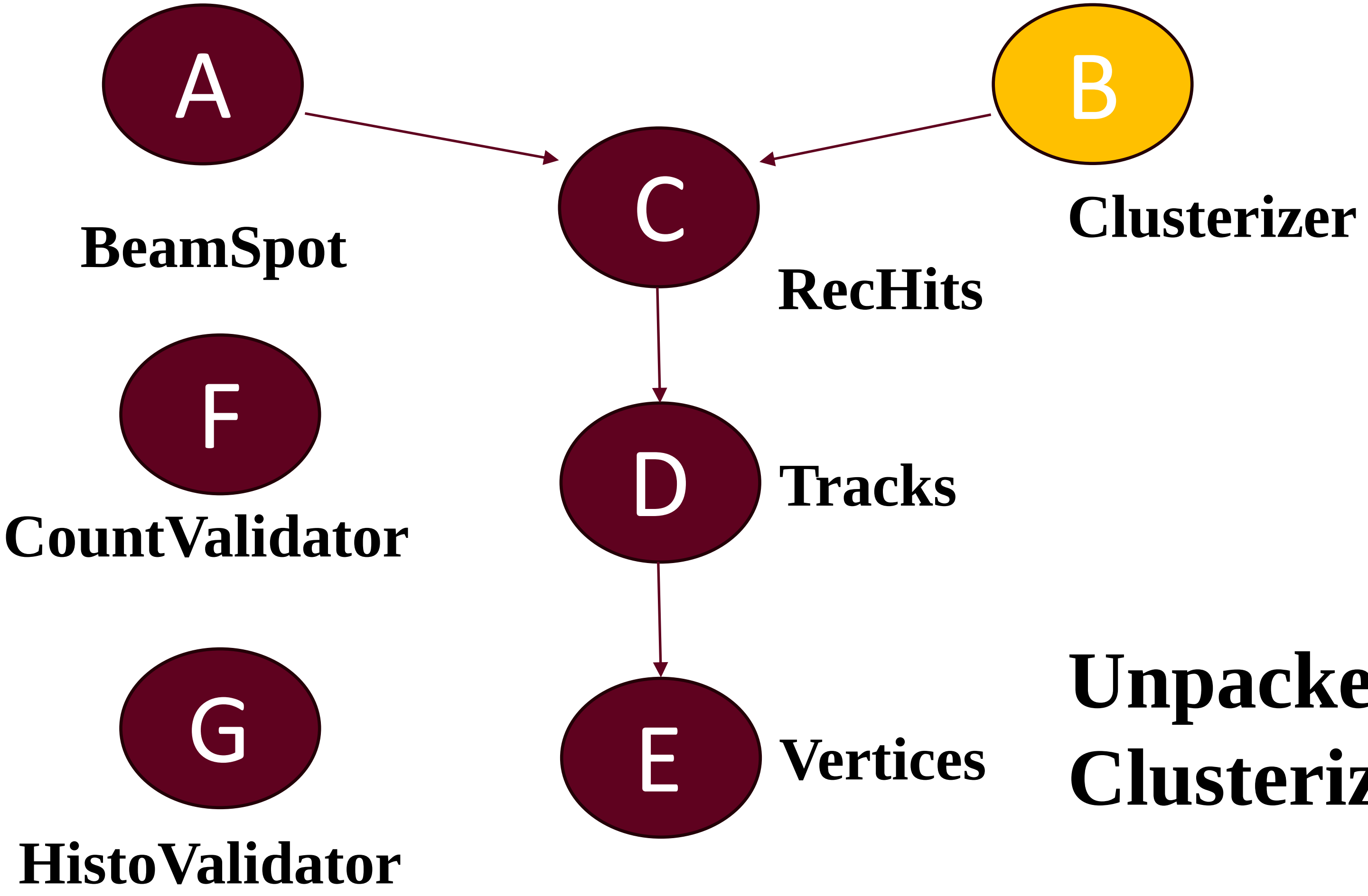
1. Port the Patatrack application to Julia.
2. Ensure its structure and output are similar to the C++ applications.
3. Monitor and optimize performance of the Julia implementation.

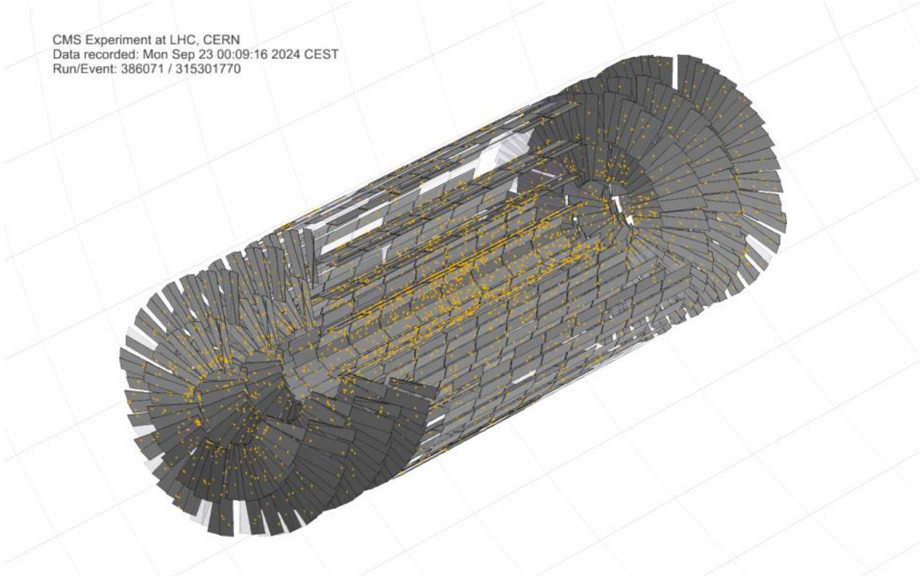
Overview of the Patatrack Application: **Modules**





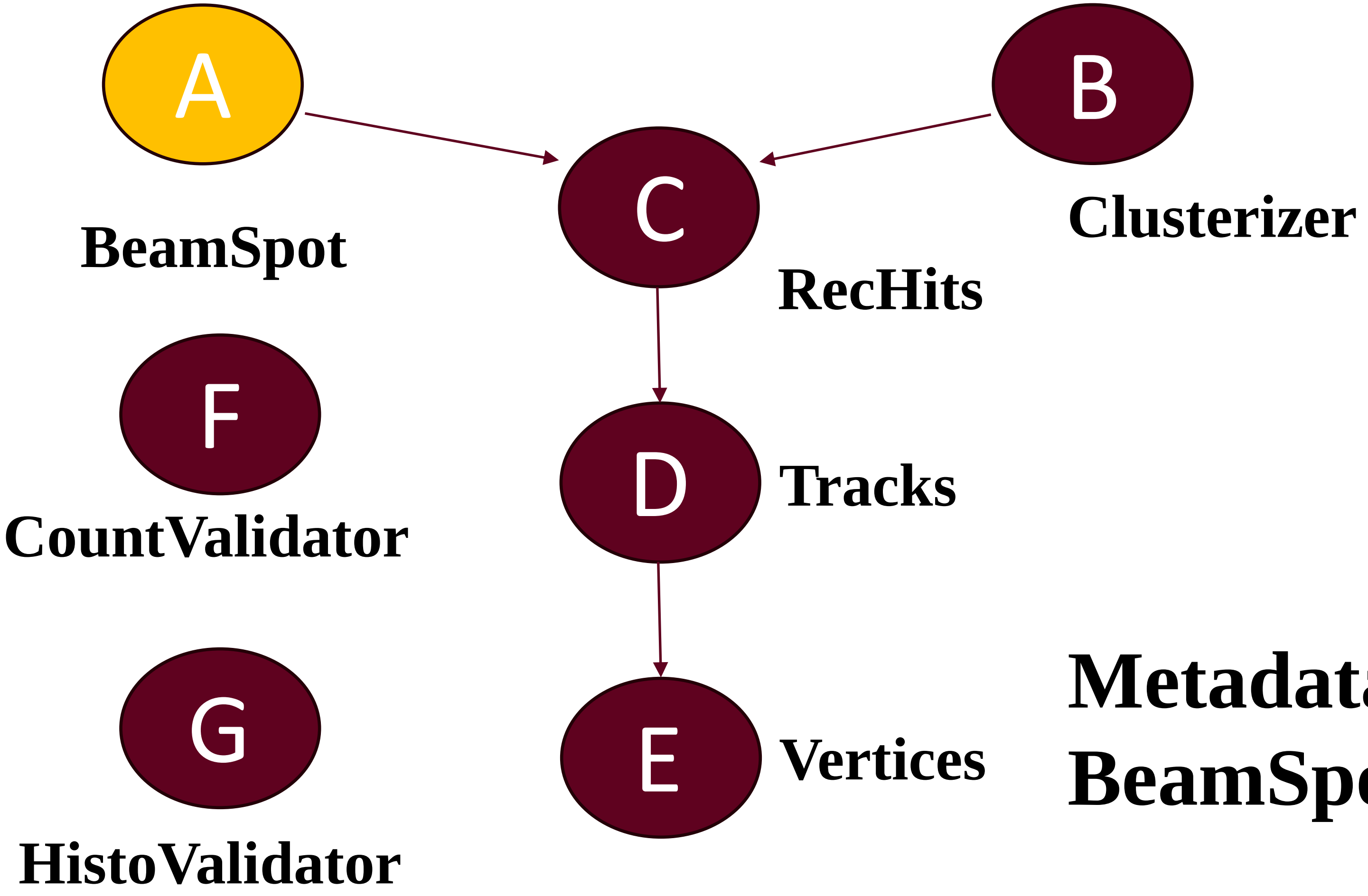
Overview of the Patatrack Application: **Modules**

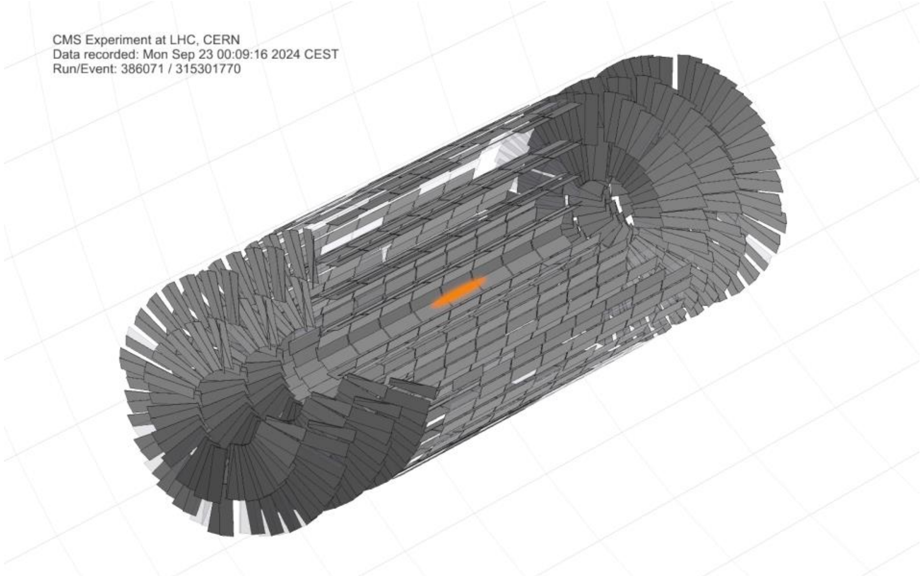




Unpacker and Clusterizer

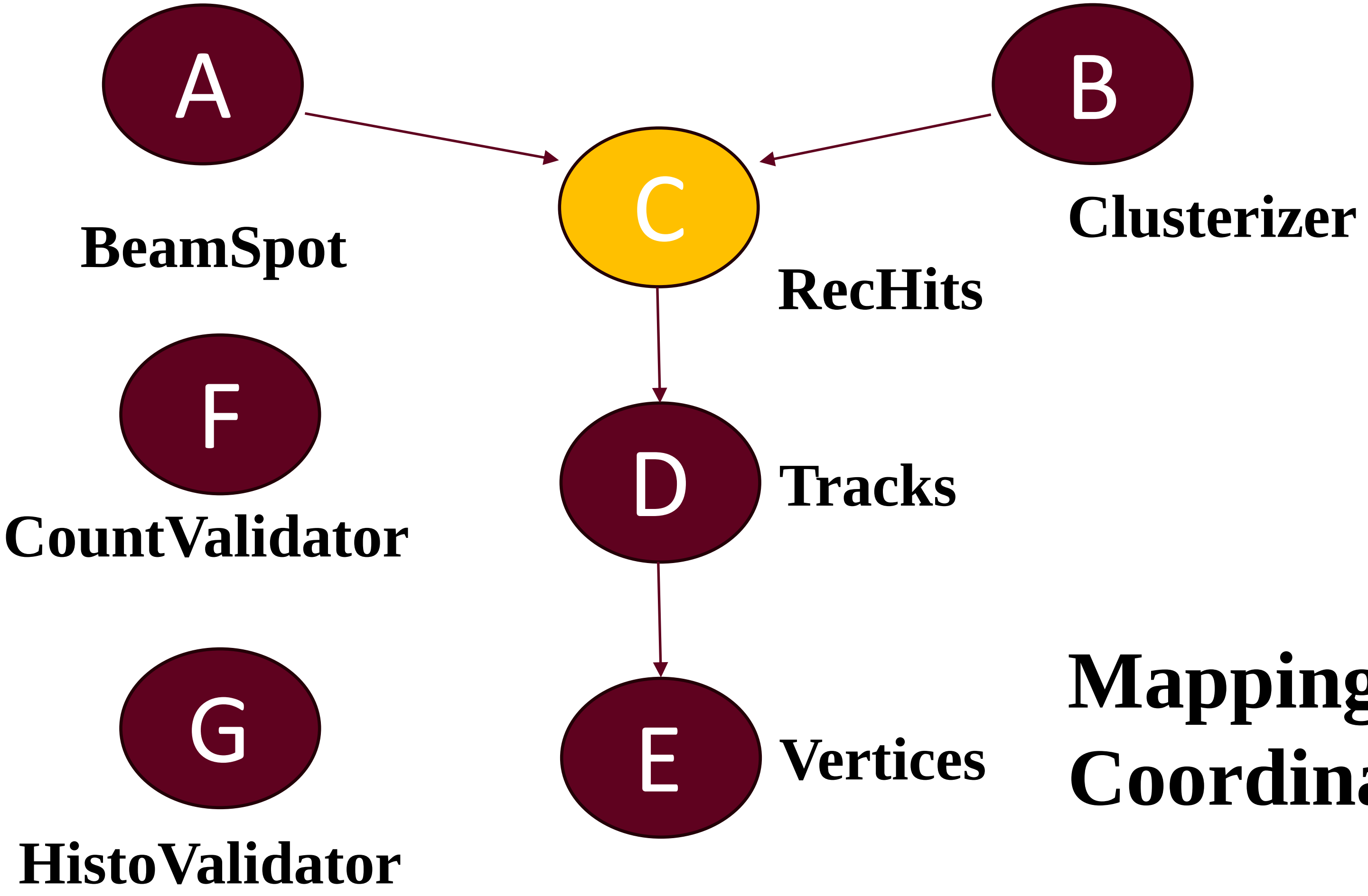
Overview of the Patatrack Application: **Modules**

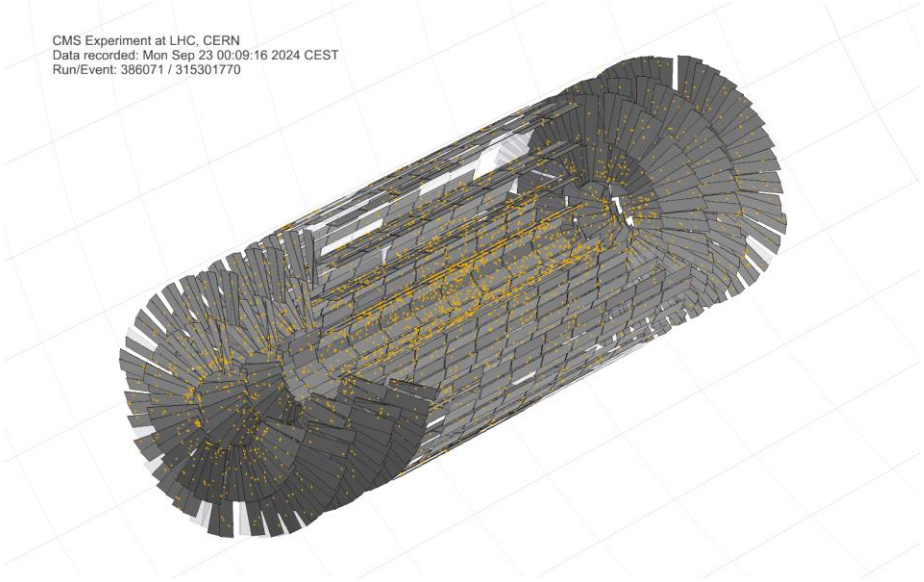




Metadata of BeamSpot

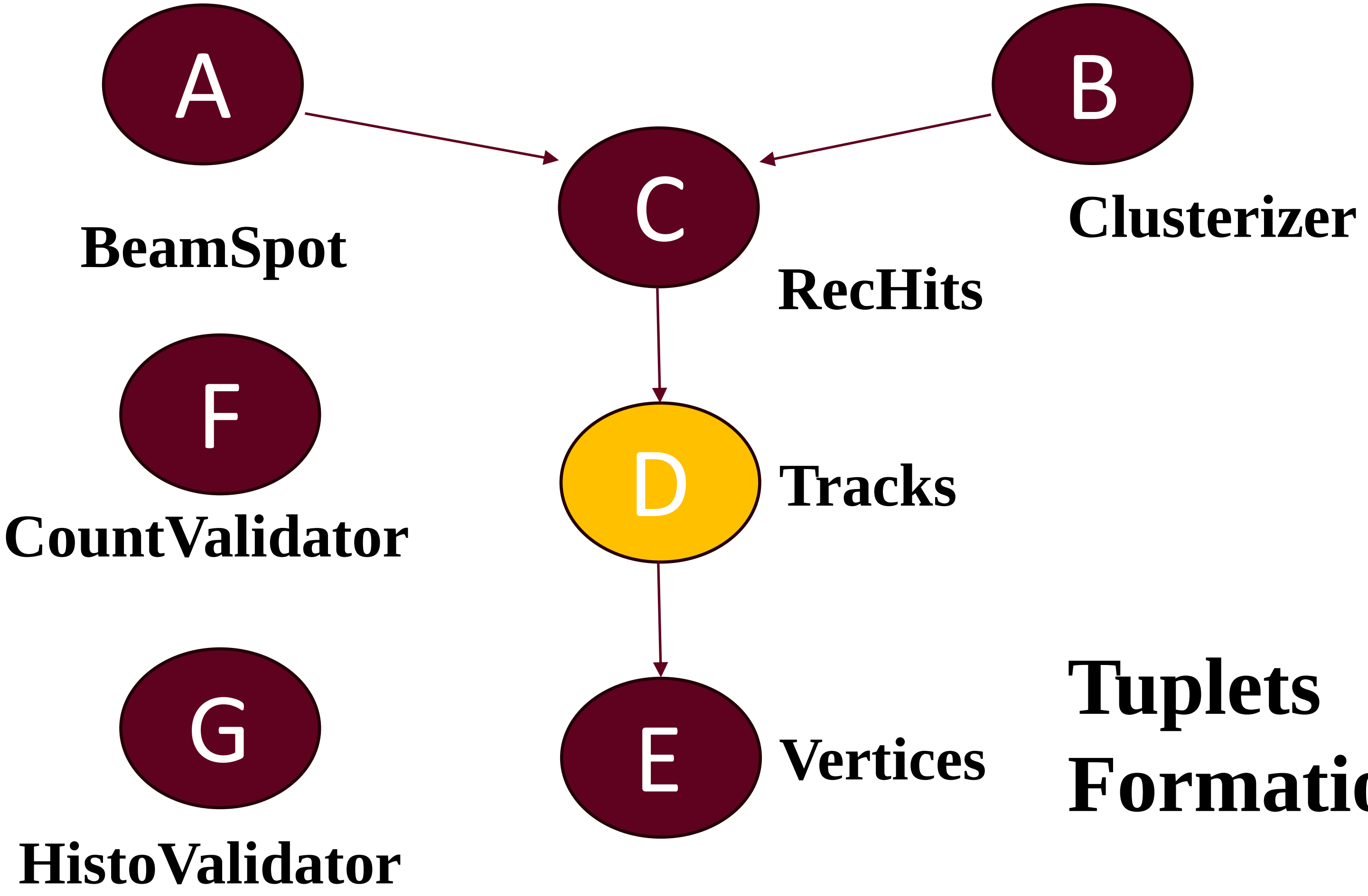
Overview of the Patatrack Application: **Modules**

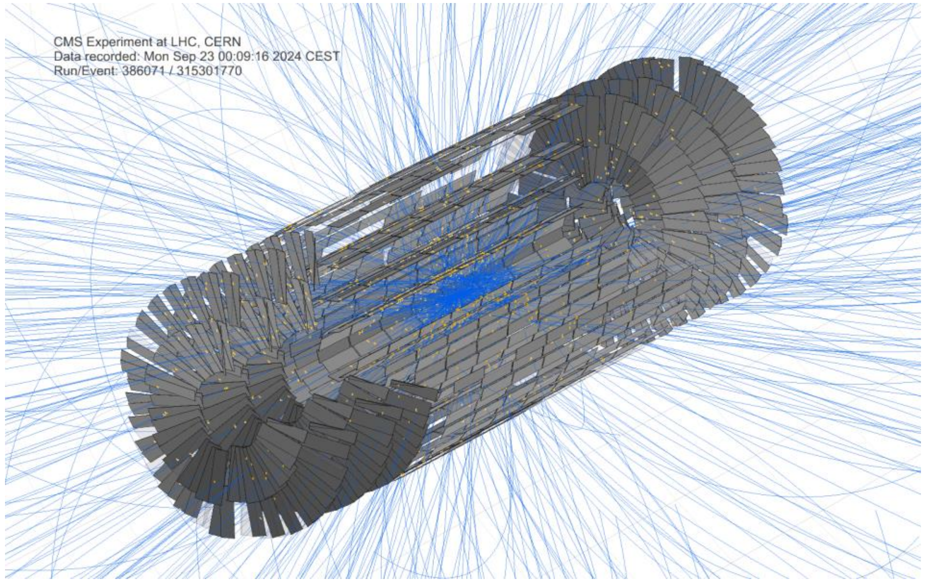




Mapping of Coordinates

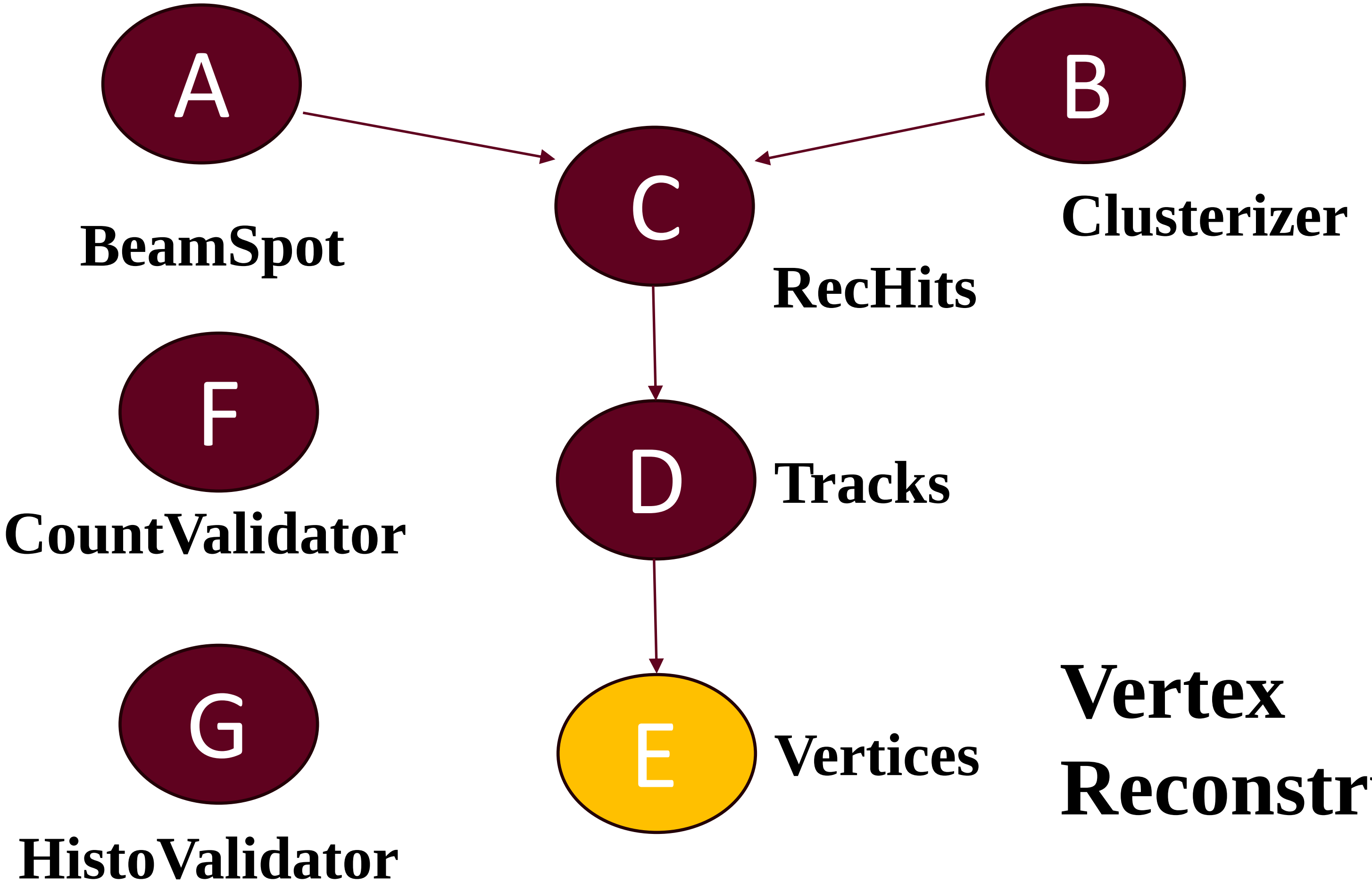
Overview of the Patatrack Application: **Modules**

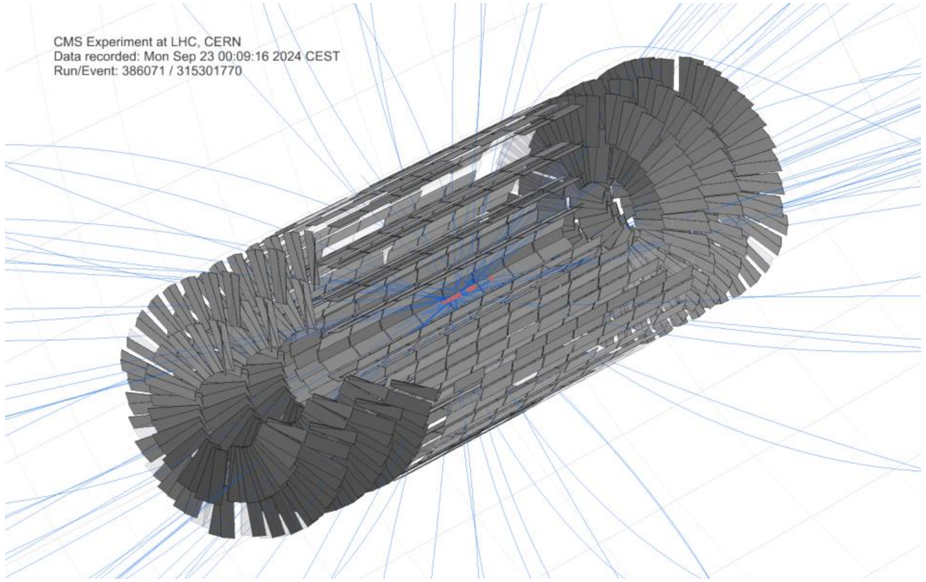




Tuplets Formation

Overview of the Patatrack Application: **Modules**



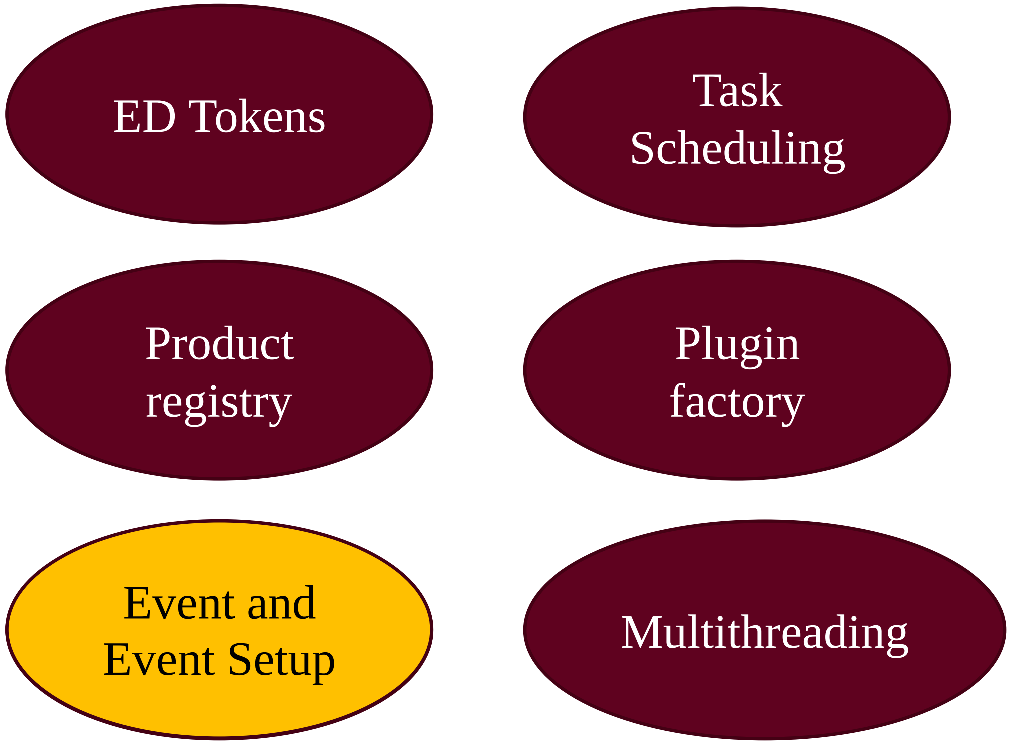


Vertex Reconstruction

Overview of the Patatrack Application: **Framework**

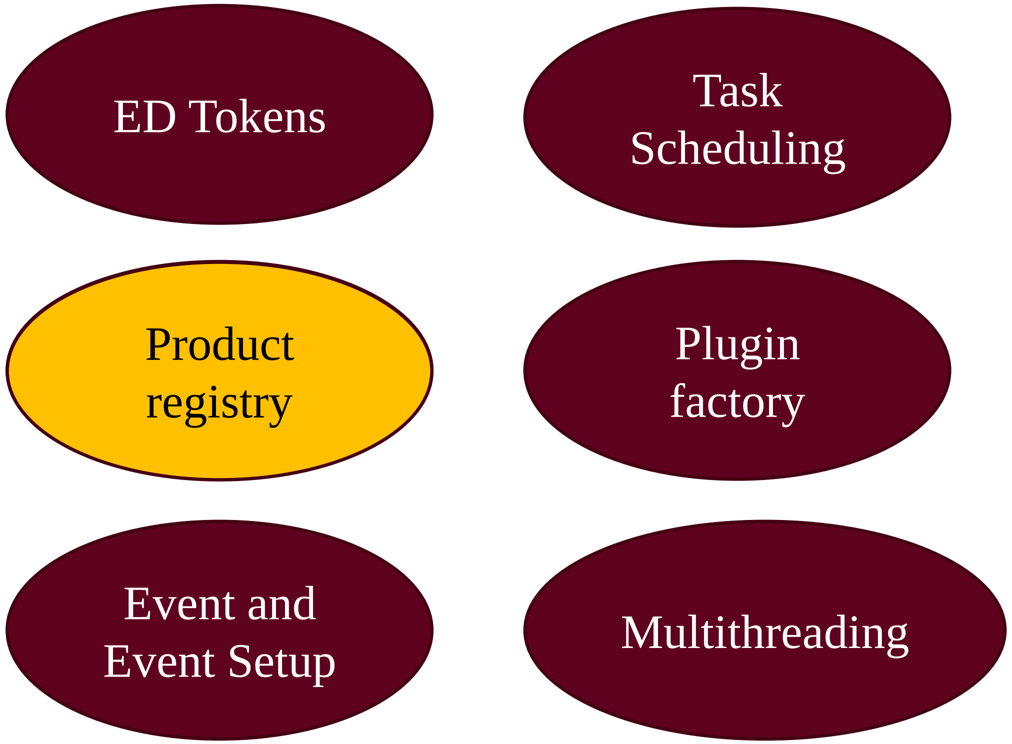
- ED Tokens
- Task Scheduling
- Product registry
- Plugin factory
- Event and Event Setup
- Multithreading

Overview of the Patatrack Application: **Framework**



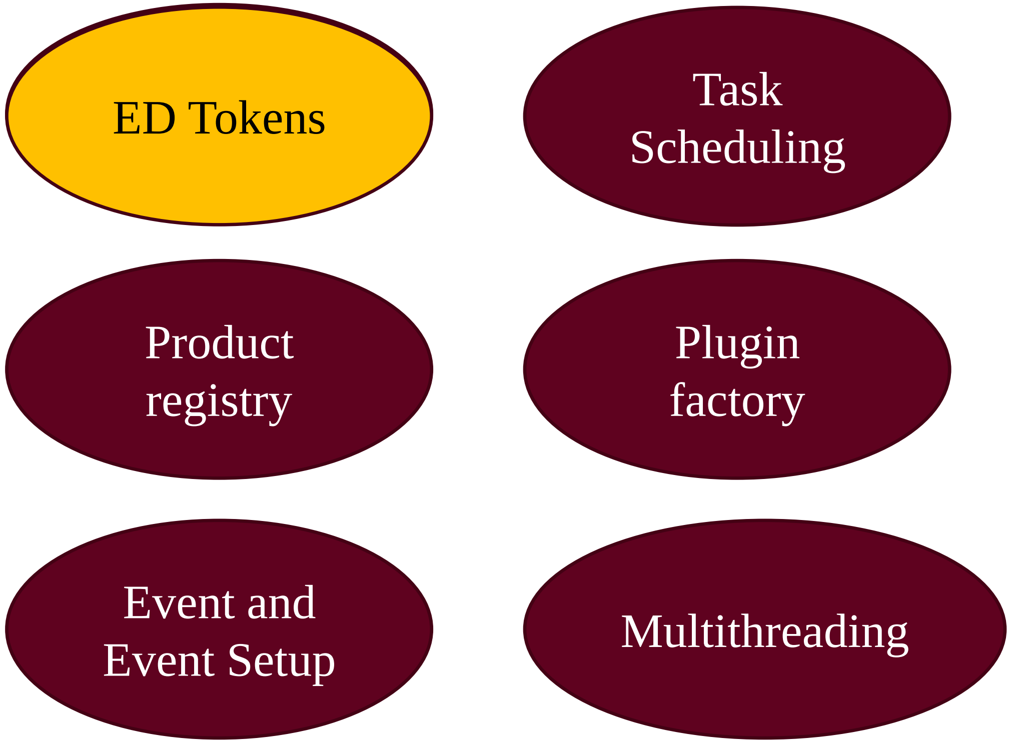
Placeholder of Data and Metadata

Overview of the Patatrack Application: **Framework**



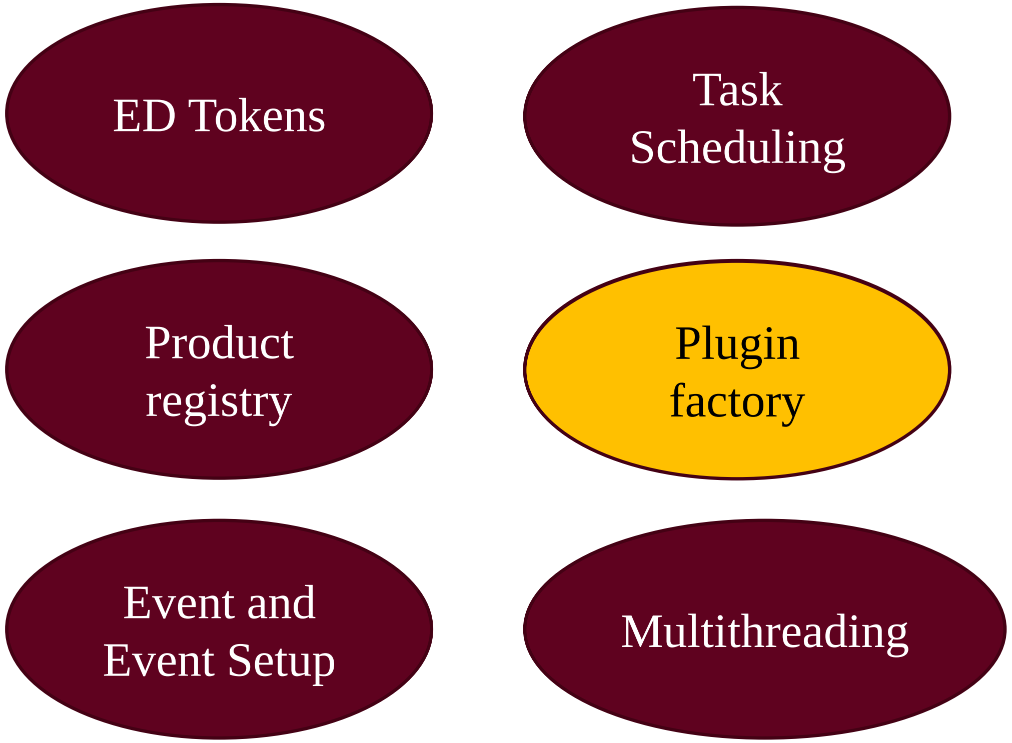
Interface between Modules and the Event

Overview of the Patatrack Application: **Framework**



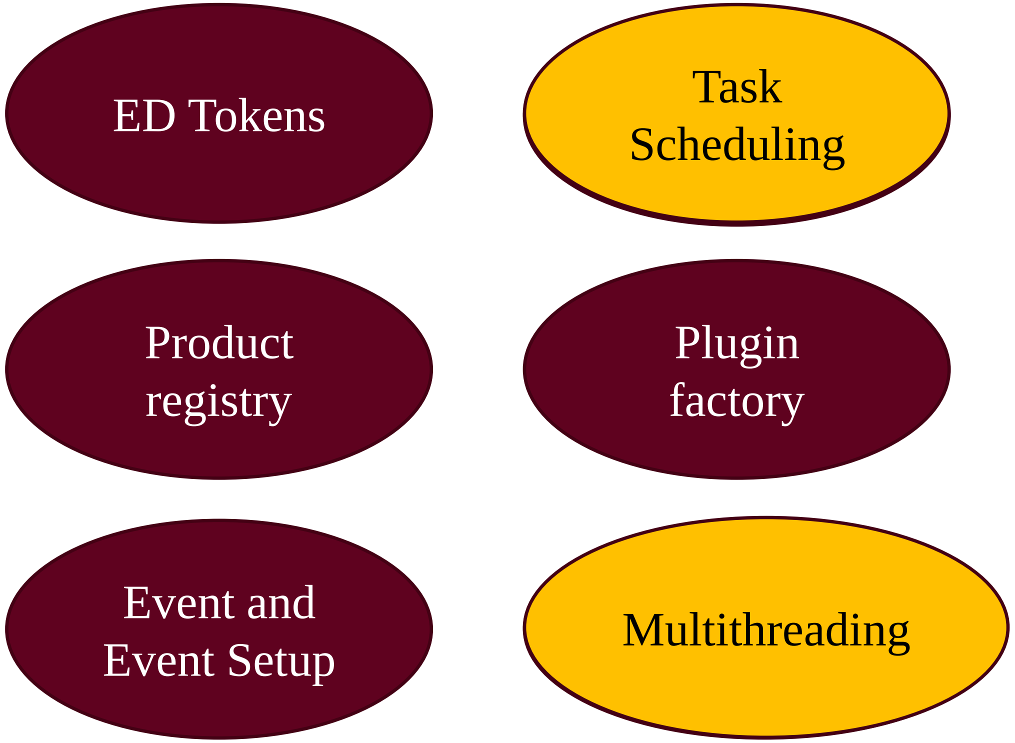
Used to access Data from the Event

Overview of the Patatrack Application: **Framework**



Interface to instantiate Modules

Overview of the Patatrack Application: **Framework**



Currently exploring this in Julia

Porting C++ to Julia

1. Understand the execution of the Patatrack reconstruction software
2. C++ : #ifndef include(“macro”) module reconstruction
3. Zero Index, One Index
4. Type casting
5. Pointers No pointer
6. ES data reading
7. Reinterpret for 32 bit words
8. Performance tips

Module Inclusion and Management in Julia

No Preprocessor: Julia doesn't use `#include` or `#ifdef` directives like C++

File Inclusion: The `include("file.jl")` function reads and executes files at runtime. Including the same file multiple times can cause errors.

Managing Modules in Patatrack:

Single Point of Inclusion: Each module is included once in the Patatrack package to prevent multiple inclusions.

Controlled Access: Other parts of the program access these modules through Patatrack, ensuring no duplicate file inclusions.

Patatrack.jl M ● main.jl

src > Patatrack.jl

```julia
export set_x_global
export set_y_global
export set_z_global

export toGlobal_special

export CAHitNtuplet

include("serial/bin/EventProcessor.jl")
include("serial/bin/source.jl")
include("serial/CUDACore/vec_array.jl")
include("serial/CUDACore/simple_vector.jl")
include("serial/CondFormats/si_pixel_fed_cabling_map_gpu.jl")
include("serial/CondFormats/si_pixel_fed_cabling_map_gpu_wrapper.jl")
include("serial/CondFormats/si_pixel_fed_ids.jl")
include("serial/CUDACore/cuda_assert.jl")
include("serial/CondFormats/si_pixel_gain_for_hlt_on_gpu.jl")
include("serial/CondFormats/si_pixel_gain_calibration_for_hlt_gpu.jl")
include("serial/CUDACore/cudaCompat.jl")
include("serial/CUDACore/cudastdAlgorithm.jl")
include("serial/CUDACore/prefix_scan.jl")
include("serial/CUDACore/hist_to_container.jl")
```

Zero Indexing to One Indexing

All detector metadata (e.g., module indices) are zero-indexed in C++, but Julia uses 1-based indexing, requiring careful adjustment by incrementing indices by one during the transition.

```julia
module gpuClusterCharge

    include("../CUDACore/cuda_assert.jl")
    # using .gpuConfig
    include("../CUDACore/prefix_scan.jl")
    using .prefix_scan:block_prefix_scan
    include("../CUDADataFormats/gpu_clustering_constants.jl")
    using .CUDADataFormatsSiPixelClusterInterfaceGPUClusteringConstants.pixelGPUConstants:INV_ID, MAX_NUM_CLUSTERS_PER_MODULES, MAX_NUM_MODULES
    using Printf
    function cluster_charge_cut(id, adc, moduleStart, nClustersInModule, moduleId, clusterId, numElements)
        charge = fill(0,MAX_NUM_CLUSTERS_PER_MODULES) # m
        ok = fill(0, MAX_NUM_CLUSTERS_PER_MODULES) # m
        newclusId = fill(0, MAX_NUM_CLUSTERS_PER_MODULES) # m

        firstModule = 1
        endModule = moduleStart[1]
        for mod ∈ firstModule:endModule
            firstPixel = moduleStart[1 + mod]
            thisModuleId = id[firstPixel]
            @assert thisModuleId < MAX_NUM_MODULES
            @assert thisModuleId == moduleId[mod]
            nClus = nClustersInModule[thisModuleId+1]
            if nClus == 0
                continue
            end
            if nClus > MAX_NUM_CLUSTERS_PER_MODULES
                 @printf("Warning too many clusters in module %d in block %d: %d > %d\n",
                thisModuleId,
                0,
                nClus,
                MaxNumClustersPerModules)
            end

            first = firstPixel
```

Module Error Toy Example

CarFactory.jl > ...

```julia
module CarFactory
export Ferrari

struct Ferrari
    l::Int
end

end
```

CarFunctions.jl > ...

```julia
module CarFunctions
include("CarFactory.jl")
using .CarFactory

export TurnOn

function TurnOn(obj::Ferrari)
    return obj
end

end
```

Main.jl > ...

```julia
include("CarFactory.jl")
include("CarFunctions.jl")

using .CarFactory
using .CarFunctions

FerrariV1 = Ferrari(1)

TurnOn(FerrariV1)
```

```
ERROR: MethodError: no method matching TurnOn(::Ferrari)

Closest candidates are:
  TurnOn(::Main.CarFunctions.CarFactory.Ferrari)
```

Module Error Toy Example

CarFactory.jl > ...

```julia
module CarFactory
export Ferrari

struct Ferrari
    l::Int
end

end

```

CarFunctions.jl > ...

```julia
module CarFunctions
include("CarFactory.jl")
using ..CarFactory

export TurnOn

function TurnOn(obj::Ferrari)
    return obj
end

end
```

Main.jl > ...

```julia
include("CarFactory.jl")
include("CarFunctions.jl")

using .CarFactory
using .CarFunctions

FerrariV1 = Ferrari(1)

TurnOn(FerrariV1)

```

```
ERROR: MethodError: no method matching TurnOn(::Ferrari)

Closest candidates are:
  TurnOn(::Main.CarFunctions.CarFactory.Ferrari)
```

Avoiding Pointers

```cpp
// check CRC bit
const uint64_t* trailer = reinterpret_cast<const uint64_t*>(rawData.data()) + (nWords - 1);
if (not errorcheck.checkCRC(errorsInEvent, fedId, trailer, errors_)) {
  continue;
}

// check headers
const uint64_t* header = reinterpret_cast<const uint64_t*>(rawData.data());
header--;
bool moreHeaders = true;
while (moreHeaders) {
  header++;
  bool headerStatus = errorcheck.checkHeader(errorsInEvent, fedId, header, errors_);
  moreHeaders = headerStatus;
}

// check trailers
bool moreTrailers = true;
trailer++;
while (moreTrailers) {
  trailer--;
  bool trailerStatus = errorcheck.checkTrailer(errorsInEvent, fedId, nWords, trailer, errors_);
  moreTrailers = trailerStatus;
}

const uint32_t* bw = (const uint32_t*)(header + 1);
const uint32_t* ew = (const uint32_t*)(trailer);

assert(0 == (ew - bw) % 2);
wordFedAppender_->initializeWordFed(fedId, wordCounterGPU, bw, (ew - bw));
wordCounterGPU += (ew - bw);
```

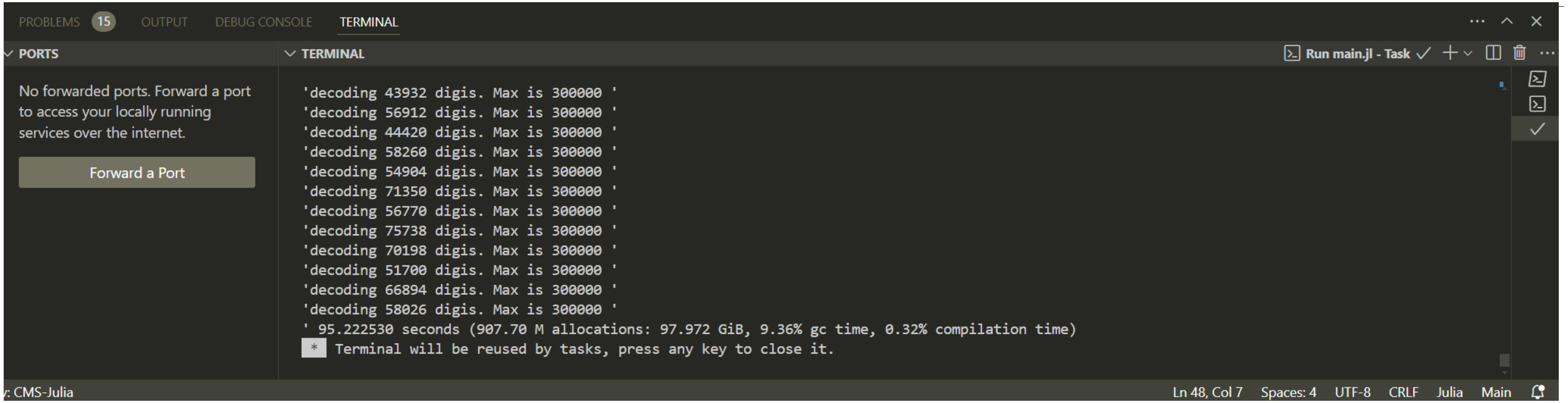
Reinterpret Function

```julia
"""
    counter takes the values from 1 to length
    Every Consecutive 4 bytes are reinterpreted as one word UInt32
    the fed_ids array is filled with the fed_id value in the range ceiling((word_counter + 1) / 2) up to (wod_counter + length) ÷ 2
"""
function initialize_word_fed(word_fed_appender::WordFedAppender, fed_id::Integer , src::AbstractArray, word_counter_gpu::Integer)
    len = length(src) ÷ 4
    for index ∈ (word_counter_gpu+1):(word_counter_gpu + len)
        counter = index-word_counter_gpu
        start_index_byte = 4*(counter-1) + 1
        word_32= view(src,start_index_byte:start_index_byte+3)
        get_word(word_fed_appender)[index] = reinterpret(UInt32,word_32)[1]
    end
    get_fed_id(word_fed_appender)[(cld((word_counter_gpu+1),2):(word_counter_gpu + len) ÷ 2)] .= (fed_id - 1200)
end
```

C++20 std::bit_cast : https://en.cppreference.com/w/cpp/numeric/bit_cast

Performance

Initial Results



1000 Events / 95.2 Seconds = 10.5 Events / Second

Initial Results - Cont'd

1000 Events / 95.2 Seconds = 10.5 Events / Second only on Clusterizer

Heap allocations reducing time performance !!

Julia Documentation

```
julia> @time sum_global()
  0.011539 seconds (9.08 k allocations: 373.386 KiB, 98.69% compilation time)
523.0007221951678

julia> @time sum_global()
  0.000091 seconds (3.49 k allocations: 70.156 KiB)
523.0007221951678
```

“Unexpected memory allocation is almost always a sign of some problem with your code, usually a problem with type-stability or creating many small temporary arrays. Consequently, in addition to the allocation itself, it's very likely that the code generated for your function is far from optimal. Take such indications seriously and follow the advice below.”

Vector to SVector

```julia
"""
function to find floor(log2(n)) in loglog(32)
"""
function i_log_2(v::UInt32)::UInt32
    b::Vector{UInt32} = [0x2,0xC,0xF0,0xFF00,0xFFFF0000]
    s::Vector{UInt32} = [ 1,2,4,8,16]
    r::UInt32 = 0

    for i ∈ 5:-1:1
        if (v & b[i]) != 0
            v >>= s[i]
            r |= s[i]
        end
    end
    return r
end
```

```julia
"""
function to find floor(log2(n)) in loglog(32)
"""
function i_log_2(v::UInt32)::UInt32
    b = SVector(0x2,0xC,0xF0,0xFF00,0xFFFF0000)
    s = SVector{5,UInt32}(1,2,4,8,16)
    r::UInt32 = 0

    for i ∈ 5:-1:1
        if (v & b[i]) != 0
            v >>= s[i]
            r |= s[i]
        end
    end
    return r
end
```

Before

```
decoding 47758 digis. Max is 300000 '
'  0.023656 seconds (145.55 k allocations: 21.667 MiB)
decoding 46898 digis. Max is 300000 '
'  0.050435 seconds (142.97 k allocations: 21.447 MiB)
decoding 46596 digis. Max is 300000 '
'  0.053570 seconds (142.07 k allocations: 21.395 MiB)
decoding 66534 digis. Max is 300000 '
'  0.079951 seconds (201.94 k allocations: 26.329 MiB)
```

After

```
decoding 47758 digis. Max is 300000 '
'  0.004317 seconds (2.45 k allocations: 10.022 MiB)
decoding 46898 digis. Max is 300000 '
'  0.004396 seconds (2.45 k allocations: 10.011 MiB)
decoding 46596 digis. Max is 300000 '
'  0.004373 seconds (2.46 k allocations: 10.034 MiB)
decoding 66534 digis. Max is 300000 '
'  0.005116 seconds (2.50 k allocations: 10.099 MiB)
```

Views For Slices

In Julia, an array "slice" expression like array[1:5, :] creates a copy of that data (except on the left-hand side of an assignment, where array[1:5, :] = ... assigns in-place to that portion of array).

Before

```
moreHeaders = true
while moreHeaders
    headerStatus =  check_header(error_check,errors_in_event, fed_id, header
    moreHeaders = headerStatus
    if moreHeaders
        header_byte_start += 8
        header =  (data(rawData)[header_byte_start:header_byte_start+7])
    end
end

moreTrailer = true
while (moreTrailer)
    trailerStatus = check_trailer(error_check,errors_in_event, fed_id, n_wor
    moreTrailer = trailerStatus
    if moreTrailer
        trailer_byte_start -= 8
        trailer =  (dataFormats.data(rawData),trailer_byte_start:trailer_byt
    end
end
```

Before

After

```
moreHeaders = true
while moreHeaders
    headerStatus =  check_header(error_check,errors_in_event, fed_id, header,
    moreHeaders = headerStatus
    if moreHeaders
        header_byte_start += 8
        header = @views (data(rawData)[header_byte_start:header_byte_start+7])
    end
end

moreTrailer = true
while (moreTrailer)
    trailerStatus = check_trailer(error_check,errors_in_event, fed_id, n_words
    moreTrailer = trailerStatus
    if moreTrailer
        trailer_byte_start -= 8
        trailer = @views (dataFormats.data(rawData),trailer_byte_start:trailer
    end
end
```

After

Abstract Type Within Struct

```julia
"""
Pixel Struct to store local coordinates inside ROC or global coordinates after mapping the local coordinates into its global coordinates within
the module
"""
mutable struct Pixel
    row::Integer
    col::Integer
```

```
decoding 48316 digis. Max is 300000 '
'  0.158097 seconds (176.75 k allocations: 17.217 MiB, 94.04% compilation time)
decoding 67188 digis. Max is 300000 '
'  0.004766 seconds (67.80 k allocations: 8.988 MiB)
 *  Terminal will be reused by tasks, press any key to close it.

  Activating project at `c:\Users\khale\OneDrive\Desktop\CernProjectLocal\CMS-Julia`
decoding 48316 digis. Max is 300000 '
'  0.152073 seconds (128.44 k allocations: 15.742 MiB, 94.07% compilation time)
decoding 67188 digis. Max is 300000 '
'  0.002882 seconds (667 allocations: 6.939 MiB)
 *  Terminal will be reused by tasks, press any key to close it.
```

before

after

Type Instability: Type of member Variable not known at compile time, Compiler Allocates Extra memory on Heap. Dynamic Dispatch due to runtime type check slows down performance

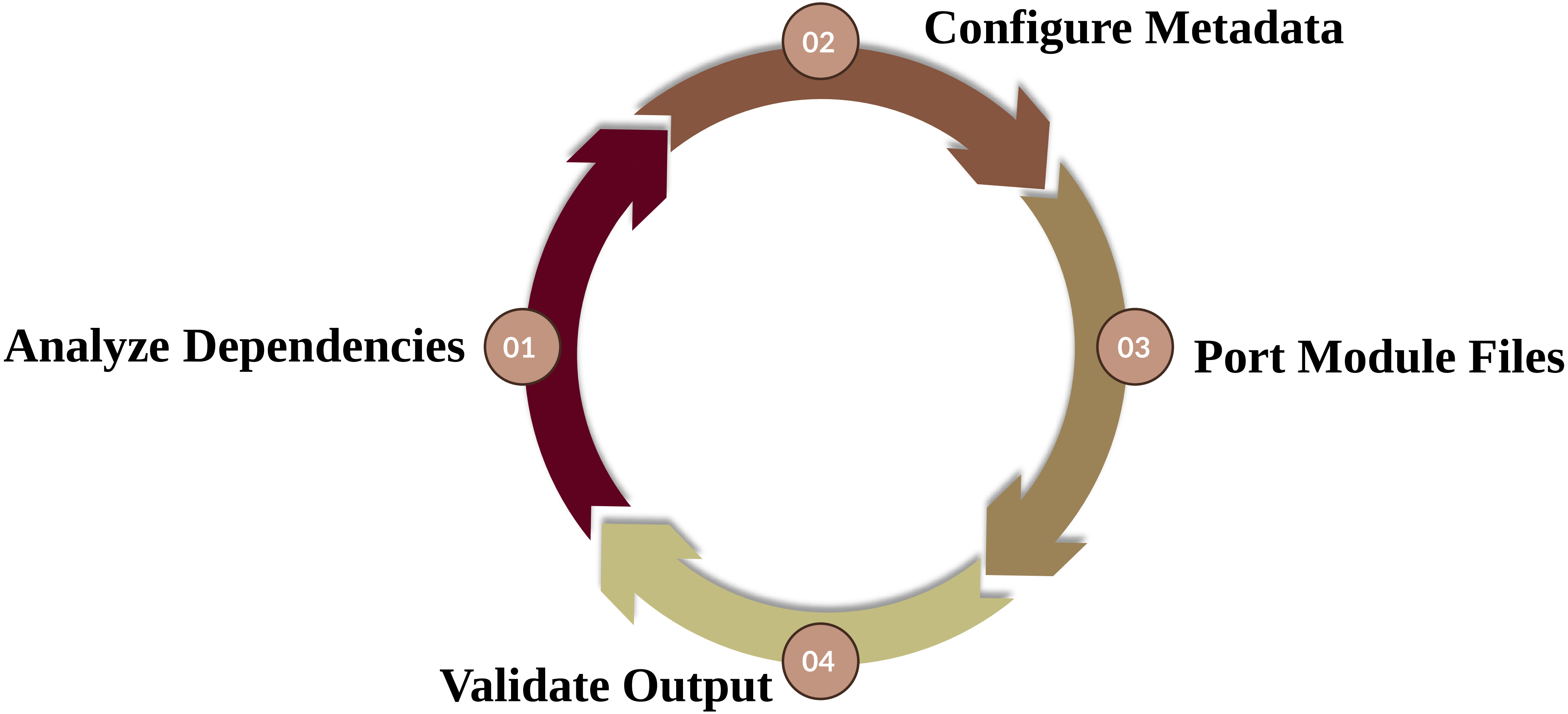
After Other Optimizations…

```
ERMINAL

 TERMINAL

locally
 'decoding 58260 digis. Max is 300000 '
 'decoding 54904 digis. Max is 300000 '
 'decoding 71350 digis. Max is 300000 '
 'decoding 56770 digis. Max is 300000 '
 'decoding 75738 digis. Max is 300000 '
 'decoding 70198 digis. Max is 300000 '
 'decoding 51700 digis. Max is 300000 '
 'decoding 66894 digis. Max is 300000 '
 'decoding 58026 digis. Max is 300000 '
 '  19.703 s (8002529 allocations: 43.64 GiB)
 *  Terminal will be reused by tasks, press any key to close it.
```

```
1828
decoding 58026 digis. Max is 300000
1807
Processed 1000 events in 1.461950e+01 seconds, throughput 68.4018 events/s, CPU usage per thread: 68.5%
khaled47@khaled47-virtual-machine:~/Desktop/cernProject/pixeltrack-standalone$
```

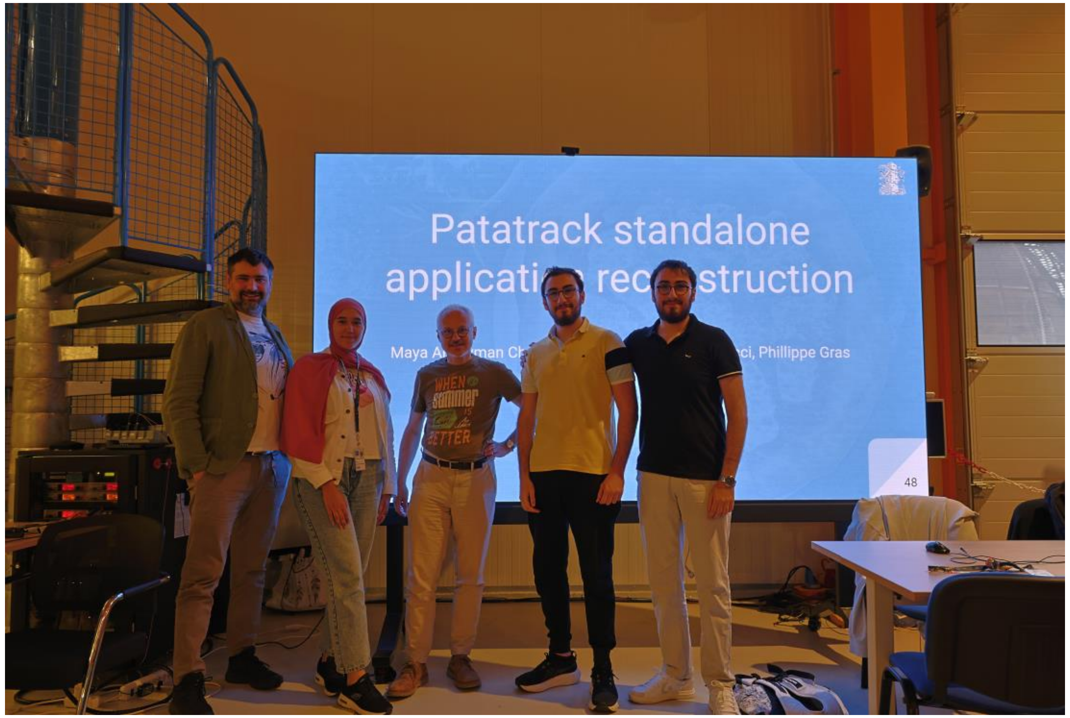
Run Time Drops to 19.7 seconds (9 seconds on Ayman's Macbook M1 Processor)
Run Time C++ 14.6 seconds (6 seconds on Ayman's Macbook M1 Processor)

The Process


02
Configure Metadata
01
Analyze Dependencies
03
Port Module Files
04
Validate Output

Achievements

Patatrack 16th Hackathon Results

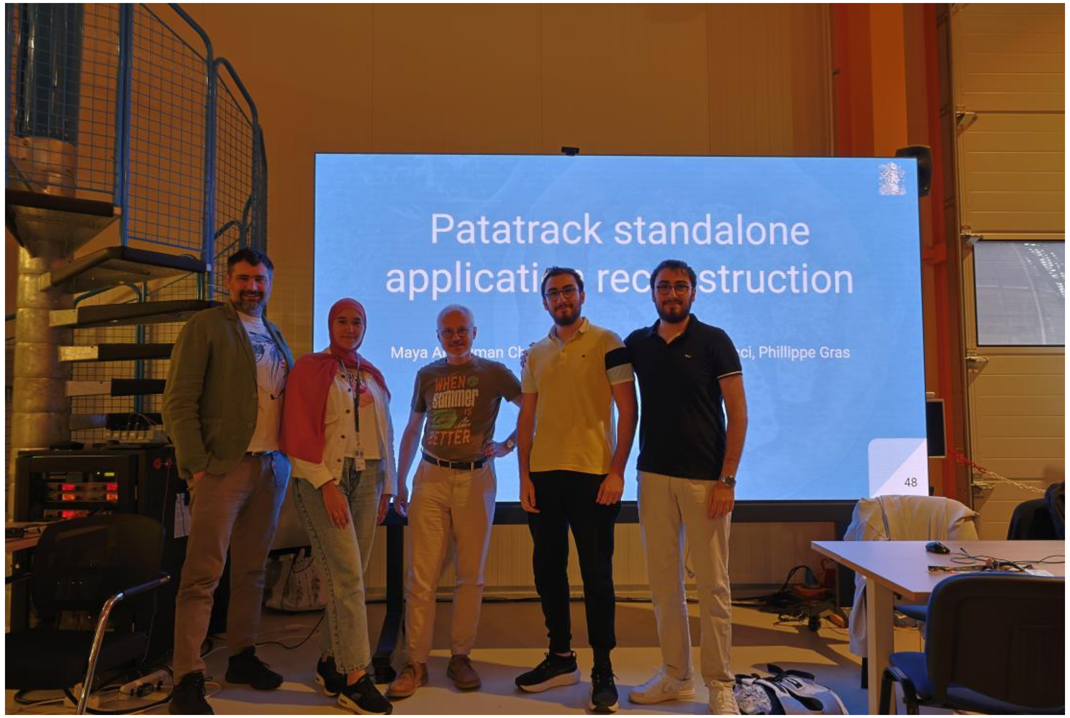


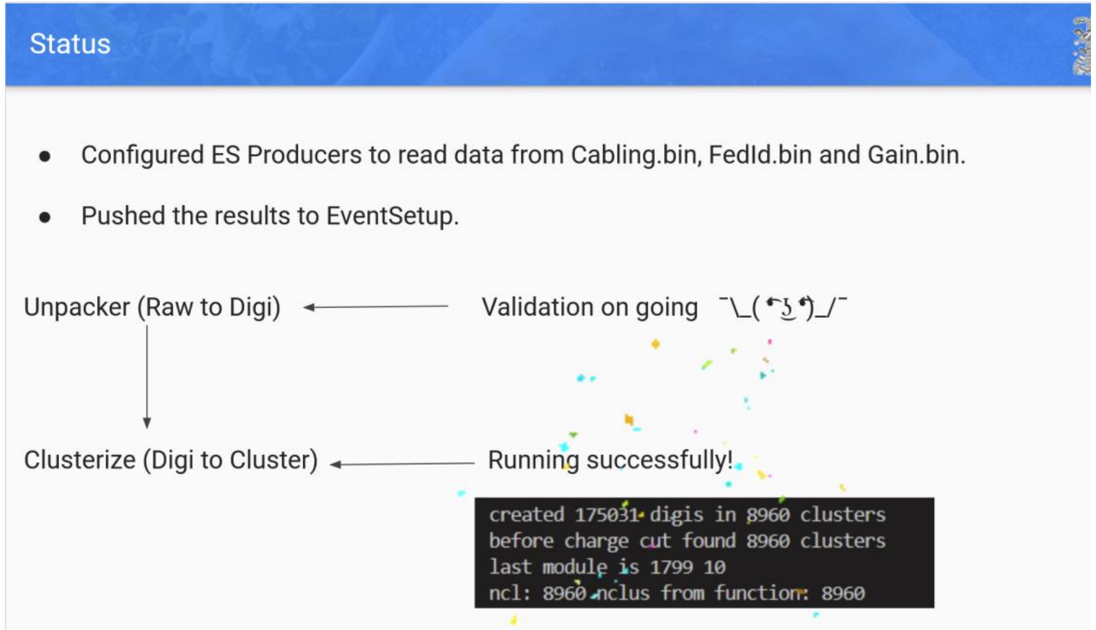
Patatrack.jl M ● main.jl

src > Patatrack.jl

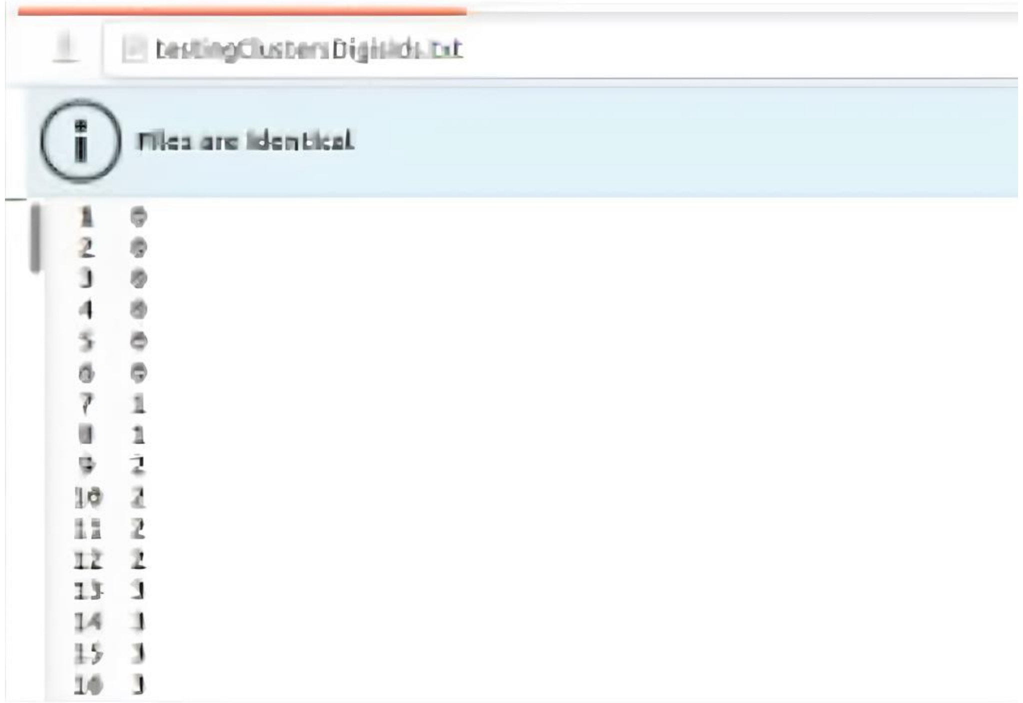
```
export set_x_global
export set_y_global
export set_z_global

export toGlobal_special

export CAHitNtuplet

include("serial/bin/EventProcessor.jl")
include("serial/bin/source.jl")
include("serial/CUDACore/vec_array.jl")
include("serial/CUDACore/simple_vector.jl")
include("serial/CondFormats/si_pixel_fed_cabling_map_gpu.jl")
include("serial/CondFormats/si_pixel_fed_cabling_map_gpu_wrapper.jl")
include("serial/CondFormats/si_pixel_fed_ids.jl")
include("serial/CUDACore/cuda_assert.jl")
include("serial/CondFormats/si_pixel_gain_for_hlt_on_gpu.jl")
include("serial/CondFormats/si_pixel_gain_calibration_for_hlt_gpu.jl")
include("serial/CUDACore/cudaCompat.jl")
include("serial/CUDACore/cudastdAlgorithm.jl")
include("serial/CUDACore/prefix_scan.jl")
include("serial/CUDACore/hist_to_container.jl")
```

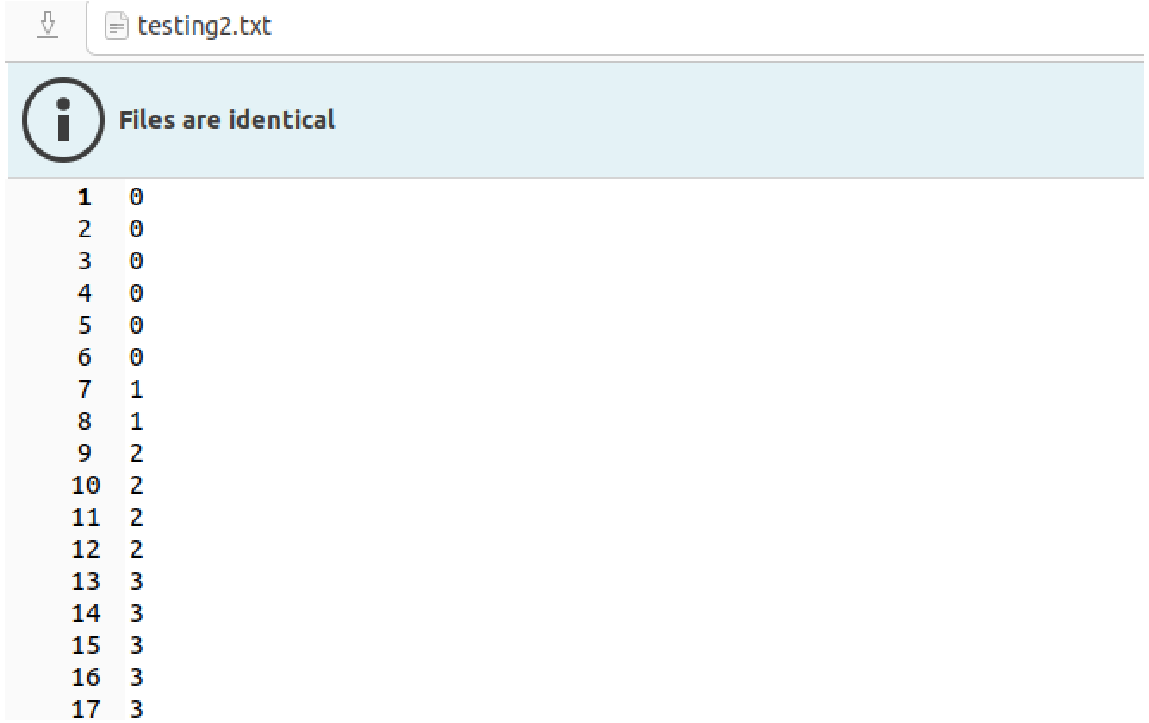
Patatrack 16th Hackathon Results- Cont'd





Clusterizer Validation





Cluster Ids of digis Validated

RecHits Validation

```
[mcharaf@lxplus928 PROJECT]$ diff RecHitsJulia.txt RecHitsC++.txt
[mcharaf@lxplus928 PROJECT]$
```

Doublets Validation

```
[mcharaf@lxplus928 PROJECT]$ diff DoubletsJulia.txt DoubletsC++.txt
[mcharaf@lxplus928 PROJECT]$
```

Running Time: Up to Doublets

C++:

```
Processed 1000 events in 3.180145e+01 seconds
```

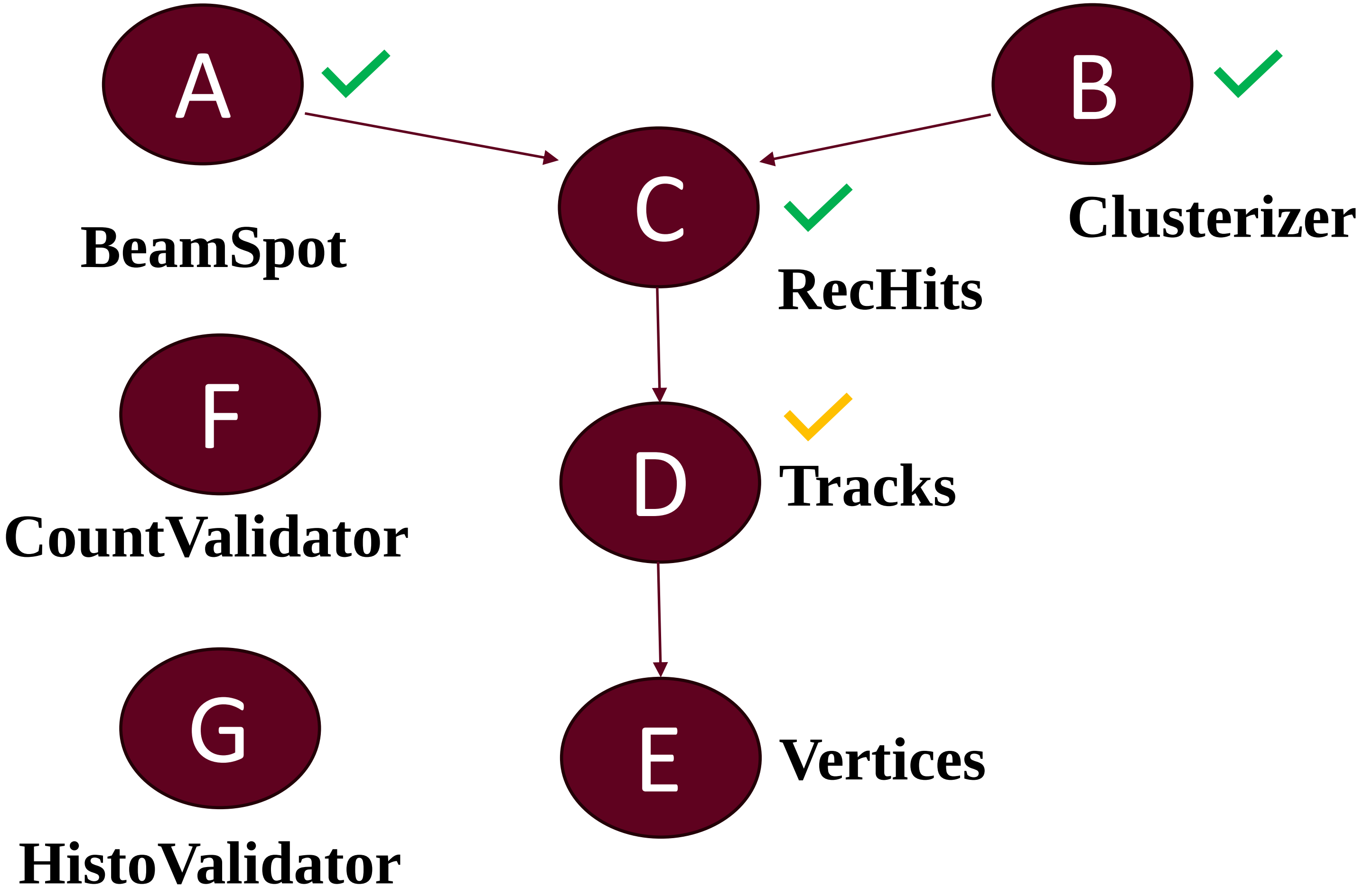
31.8 Seconds

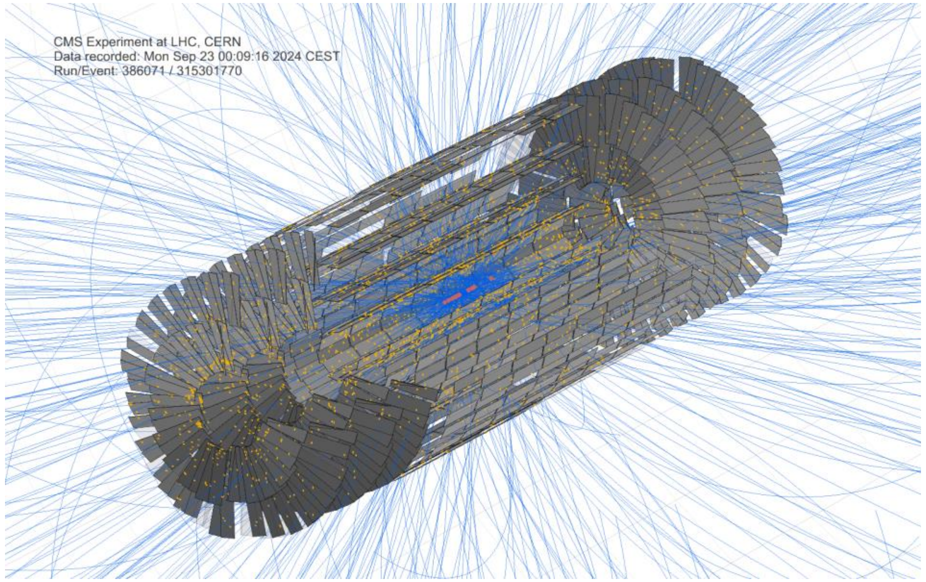
Julia:

```
35.081 s (235105557 allocations: 76.85 GiB)
 *  Terminal will be reused by tasks, press any key to close it.
```

35.1 Seconds

Ported 100% of the local reconstruction





Ongoing Activities:

- **Precompilation and Distribution**
  - **PackageCompiler.jl**
- **Multithreading**
  - **Threads.@threads**
  - **FLoops.jl**
  - **ThreadPools.jl**
- **GPU Integration**
  - **CUDA.jl**

Next Steps and Future Work

- Integrate GPU acceleration into the entire application.
- Optimize Performance.
- Share findings to encourage more adoption of Julia in scientific computing!

Conclusion

- **Advantages gained from using Julia**
  - Syntax that is easier to read and write
  - Automatic memory management
  - Large Ecosystem
- **Key Accomplishments**
  - Significant strides in initial implementations and testing.
- **Remaining Challenges**
  - Address multithreading, pre-compilation, and GPU integration.
- **Future Vision**
  - Enhance both development speed and runtime performance.

Thank You!

Contact Info:

maya.ali@cern.ch
mohamad.ayman.charaf@cern.ch
mohamad.khaled.charaf@cern.ch