

Day 4

1st NextGen Hackathon



NextGen
Next Generation Triggers

```
template <typename T>  
void BM_TheBestTeam  
(benchmark::State &state, T &t);
```

```
BENCHMARK_CAPTURE(BM_TheBestTeam, SoABenchmark, {
```

```
    Jolly Chen, Leonardo Beltrame, Oliver Rietmann, Simone Balducci,  
    Simone Rossi Tisbeni, Luca Ferragina, Aurora Perego,  
    Davide Gadioli
```

```
});
```



NextGen
Next Generation Triggers

What happened today

- Continued creating a shared **SoA benchmark**:
<https://github.com/cern-nextgen/wp1.7-soa-benchmark>
 - Added benchmarks with 2, 10, 64 data members (int, float, double, Eigen vector3D/matrix3x3)
 - 10 and 10e4 number of elements
 - Leonardo's and Oliver's version
- See how parts of Oliver's template approach could be improved with *macros and reflection*
- Looked into **iterators**
 - Iterating is fine, but swapping and moving is a problem with reference views

Plan for tomorrow

- Continue what we did today...
 - Try to fix our current compilation errors
- Try to run SoA benchmarks as a github action for CI
- Add manual version of SoA to benchmarks for baseline performance

Running event generators on GPU

Daniele Massaro



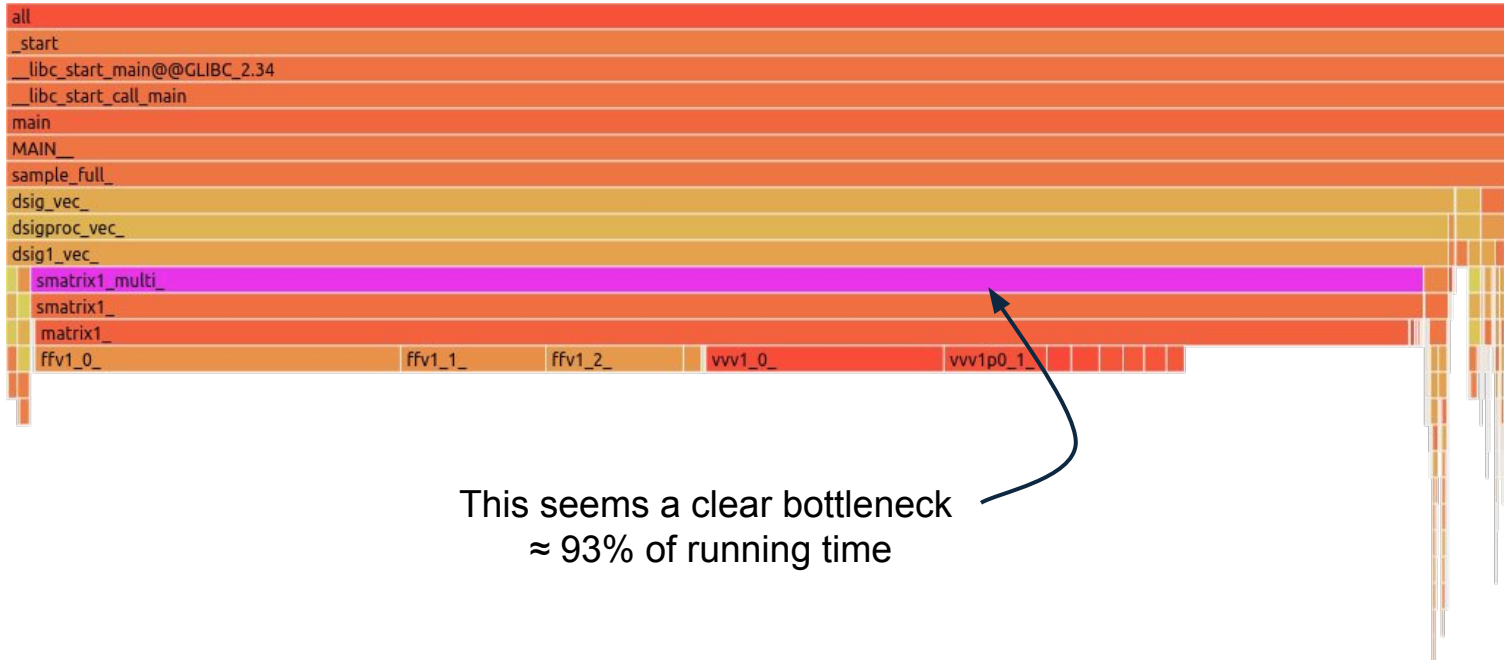
NexTGen
Next Generation Triggers

What happened today

- What is the bottleneck of the code?
 - run process $g g \rightarrow t \bar{t} g g$, around 200k events;
 - profile with Adaptyst, generate flamegraphs;
 - test both original FORTRAN version and the hardware-accelerated version.

What happened today

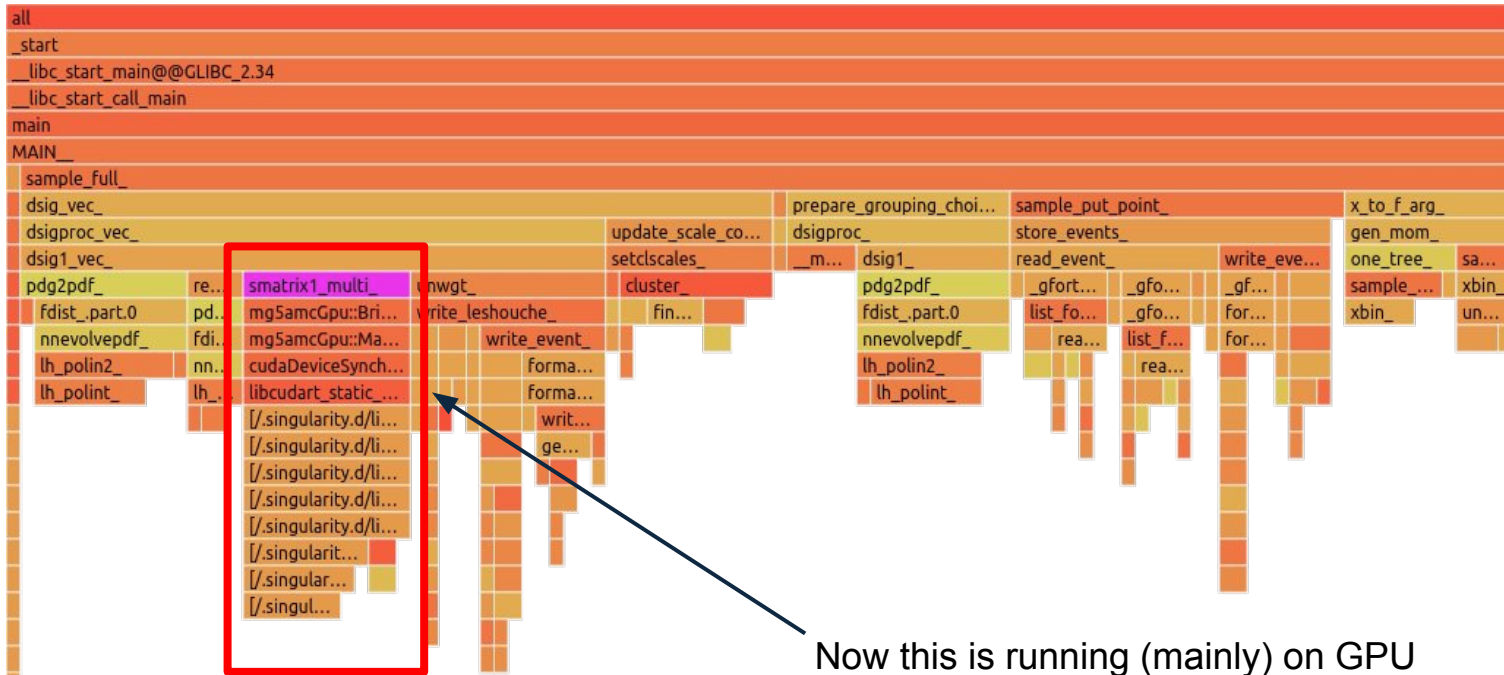
Total execution time: 134.83 s



This seems a clear bottleneck
≈ 93% of running time

What happened today

Total execution time: 10.94 s



Next steps

- Further tests with Adaptyst on other configurations.
- LHAPDF ships also a GPU version, how can it be included?

Fast ML inference

Alessandro Crespi, Jamie Gooding, Enrico Lupi,
Lorenzo Moneta, Anastasiia Petrovych,
Sanjiban Sengupta, [Jonas Rembser](#)



NexTGen
Next Generation Triggers

What happened today

- Good discussion on memory optimization
 - Plan to implement SoA based Memory Allocator which will be easier to extend in the future with GPU memory
- Fixing the reshape operators
- Extending support of broadcasting of dynamic tensors
- Added flag for optimization level of emitted SOFIE code, in particular to disable memory reuse because this prevents reverse mode AD (aka backpropagation) with Clad
- Experimented with SPANet architecture
 - Added support for missing operators for this architecture
- Started developing unit test for AD of gemm wrapper

AD on Neural Network code emitted by SOFIE

- Hacked TMVA-SOFIE to emit C++ code for the neural network inference that is differentiable with **Clad**
- Works for fully-connected neural networks like in our motivating example for an LHCb analysis with Simulation based inference
 - Conditional neural network with one Physics parameter that we want to get the derivative for

```
numr : -0.614012
clad  : -0.61405
```

```
void doInfer(const float *weights, // trained weights
            float *tensor_x, // input 1
            float *tensor_theory_params, // input 2
            float *tensor_linear_3) // output 1
{
    auto &tensor_val_2 = weights;
    auto &tensor_val_0 = weights + offset ...;
    ...

    //----- Gemm
    Gemm_Call(tensor_linear, false, false, 5, 1, 1, 1, tensor_val_0,
              tensor_theory_params, 1, tensor_theory_projectorbias);
    //----- Add
    for (size_t id = 0; id < 5; id++) {
        tensor_add[id] = tensor_x[id] + tensor_linear[id];
    }
    //----- Gemm
    Gemm_Call(tensor_linear_1, false, false, 128, 1, 5, 1,
              tensor_val_2, tensor_add, 1, tensor_base_model0bias);
    //----- Sigmoid -- 5
    for (int id = 0; id < 128; id++) {
        tensor_val_4[id] = 1 / (1 + std::exp(-tensor_linear_1[id]));
    }
    ...
}
```

Plan for tomorrow

- Progress with ALPAKA integration
- Implement memory allocator with SoA
 - With functionality to accept a memory handler from user
- Test more user models to extend SOFIE coverage
 - Make SPANet architecture work
- Finish work on gemm wrapper AD unit test
- Work on RooFit tutorial combining SOFIE and simulation-based inference

Fast ML Inference with hls4ml

Dimitrios Danopoulos, Roope Niemi, Vladimir Loncar, Francesco
Vaselli, Rimsky Rojas



NextGen
Next Generation Triggers

fAD: flows for anomaly detection

Continuous transformation ($t \in [0, 1]$)

$$f(0; z) = z = \text{Gaussian}$$

$$f(1; z) = \text{target p.d.f.}$$

$$f(t + dt) = f(t) + v(t) \cdot dt$$

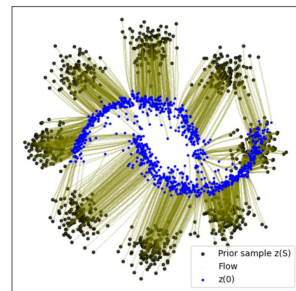
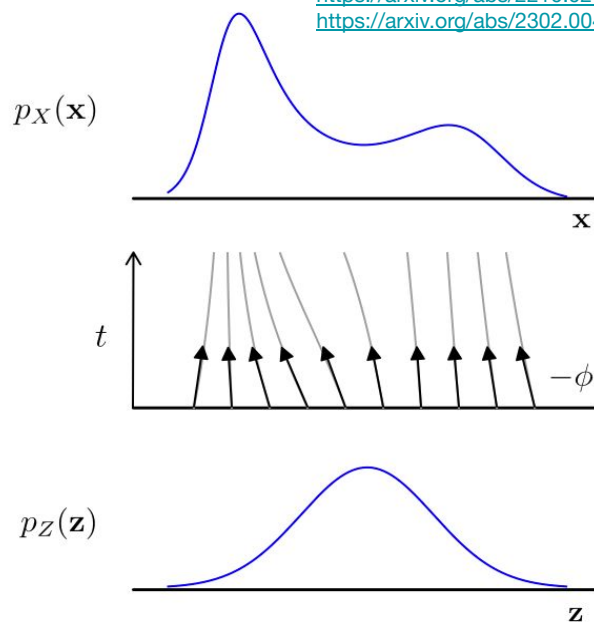
$$f(t + dt) = f(t) + DNN(f(t)) \cdot dt$$

The model is learning the vector field v_t , then we solve the ODE

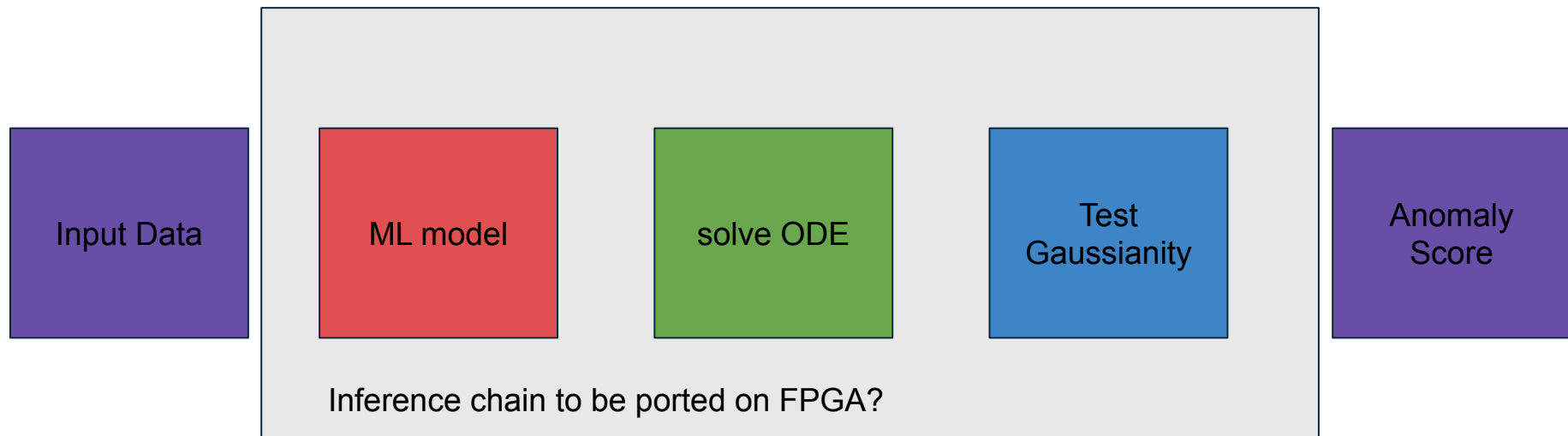
Idea: what if we train to send SM data to Gaussians? Then anomalies will fall in the Gaussian tails!

Works on CPU; how to port this to FPGA for 40Mhz Anomaly detection? Testing on the CMS-like pseudo realistic “Anomaly detection at 40Mhz data challenge” dataset:

<https://mpp-hep.github.io/ADC2021/>



Problem: complex inference chain

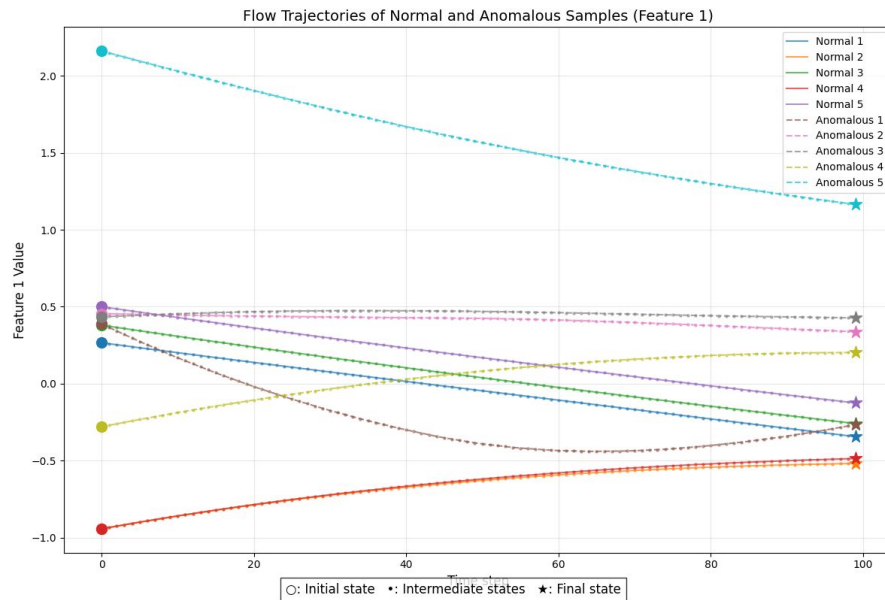
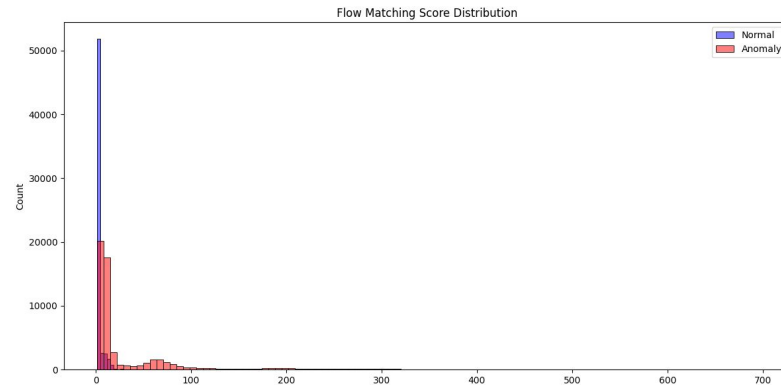


Idea: port only the actual model

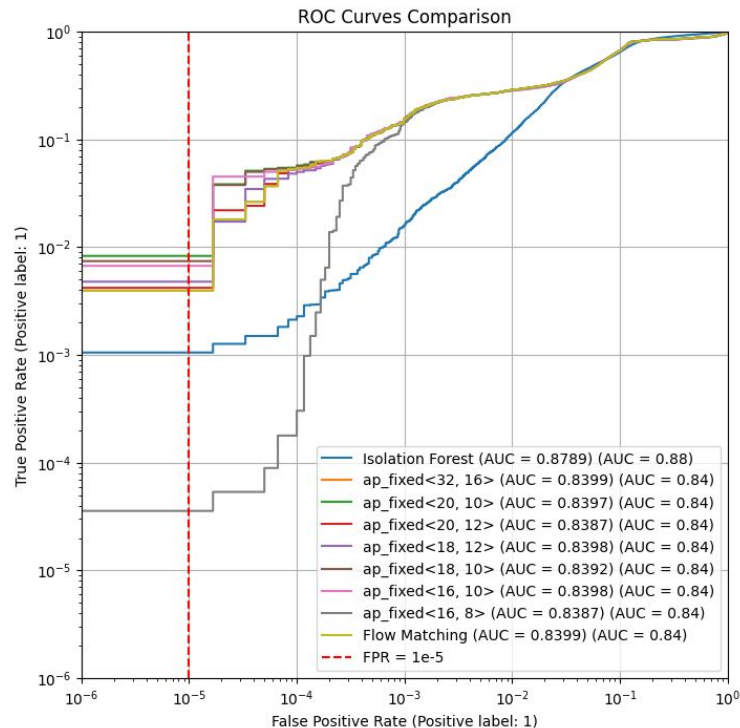
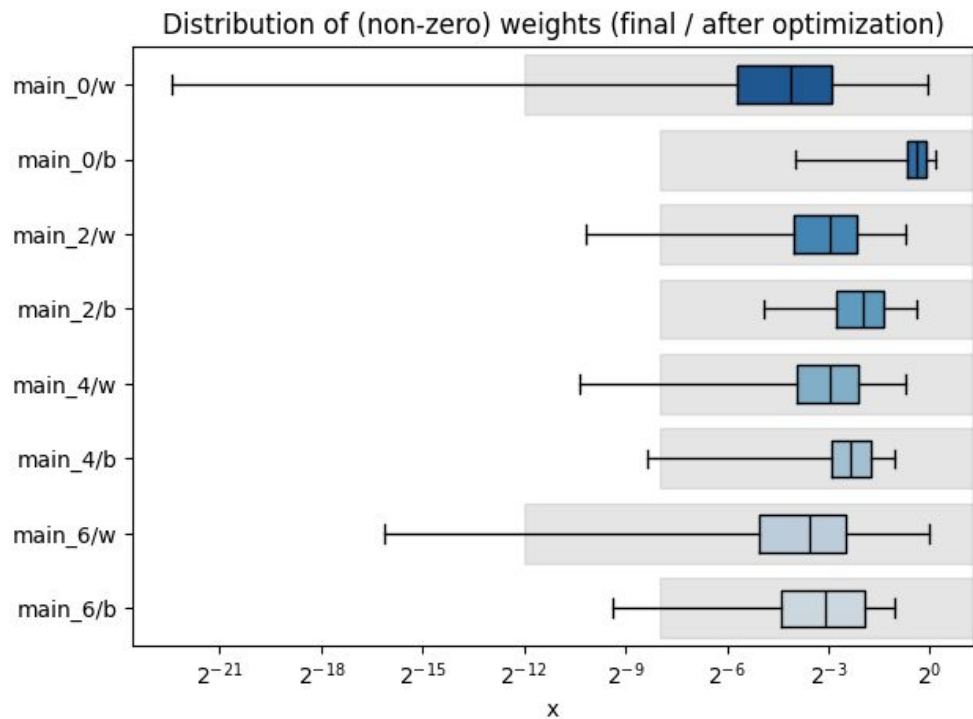
The anomalous data points will be displaced more to end up in the tails of the distribution

We can use the norm of the vector field at the first time step as the actual Anomaly metric!

Easier to port on FPGA!



We create the config and optimize. Then compile and test the results



The model synthesises successfully!

Could benefit from pruning, but good latency so far!

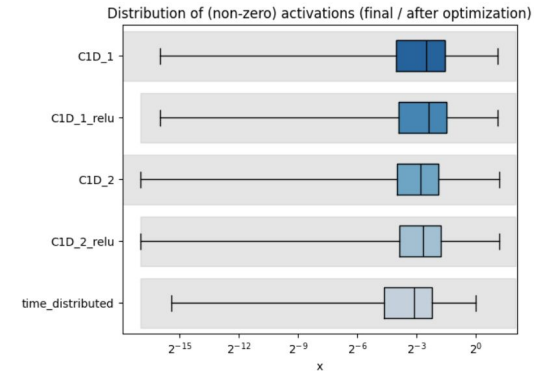
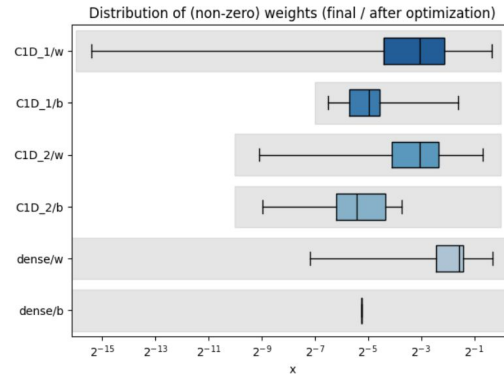
Modules & Loops	Issue Type	Slack	Latency (cycles)	Latency (ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT	URAM
+ myproject	II	0.02	15	75.000	-	1	-	yes	-	5048 (41%)	93820 (2%)	302716 (17%)	-
+ dense_latency_ap_fixed_18_10_5_3_0_ap_fixed_48_28_5_3_0_config2_s	-	0.02	2	10.000	-	1	-	yes	-	2019 (16%)	32255 (~0%)	103105 (5%)	-
+ relu_ap_fixed_48_28_5_3_0_ap_fixed_18_10_5_3_0_relu_config3_s	II	2.40	0	0.000	-	1	-	yes	-	-	-	2336 (~0%)	-
+ dense_latency_ap_fixed_18_10_5_3_0_ap_fixed_42_26_5_3_0_config4_s	-	0.06	2	10.000	-	1	-	yes	-	484 (3%)	12013 (~0%)	45783 (2%)	-
+ relu_ap_fixed_42_26_5_3_0_ap_fixed_18_10_5_3_0_relu_config5_s	II	2.39	0	0.000	-	1	-	yes	-	-	-	2144 (~0%)	-
+ dense_latency_ap_fixed_18_10_5_3_0_ap_fixed_42_26_5_3_0_config6_s	-	0.06	1	5.000	-	1	-	yes	-	475 (3%)	10725 (~0%)	44827 (2%)	-
+ relu_ap_fixed_42_26_5_3_0_ap_fixed_18_10_5_3_0_relu_config7_s	II	2.39	0	0.000	-	1	-	yes	-	-	-	2144 (~0%)	-
+ dense_latency_ap_fixed_18_10_5_3_0_ap_fixed_46_26_5_3_0_config8_s	-	0.02	2	10.000	-	1	-	yes	-	2070 (16%)	33008 (~0%)	102329 (5%)	-

What happened today NGT WP2.2

NGT WP2.2: Implementation of classification model for L0 MDT Trigger

- HLS4ML conversion and now synthesizing using per layer quantization based on profiling information

```
for layer in config['LayerName'].keys():
    print(layer)
    config['LayerName'][layer]['Trace'] = True
    if 'softmax' in layer or 'sigmoid' in layer or 'relu' in layer or 'tanh' in layer:
        config['LayerName'][layer]['table_t'] = f'ap_fixed<25,3>'
    if 'C1D_1' in layer and 'relu' not in layer:# or 'output' in layer:
        config['LayerName'][layer]['Precision']['weight'] = f'ap_fixed<17,1>'
        config['LayerName'][layer]['Precision']['bias'] = f'ap_fixed<8,1>'
        config['LayerName'][layer]['Precision']['result'] = f'ap_fixed<23,3>'
    if 'C1D_2' in layer and 'relu' not in layer:# or 'output' in layer:
        config['LayerName'][layer]['Precision']['weight'] = f'ap_fixed<11,1>'
        config['LayerName'][layer]['Precision']['bias'] = f'ap_fixed<11,1>'
        config['LayerName'][layer]['Precision']['result'] = f'ap_fixed<25,3>'
```



Fully Convolutional Network for Jet Reconstruction

Combination of Conv2d, Avg/MaxPooling2d, ConvTranspose2d, BatchNorm2d and ReLU.

- Using the PyTorch -> hls4ml workflow initially
- Managed to convert to an HLS project by substituting the unsupported ConvTranspose2d with an Upsample layer.

Using the Brevitas -> QONNX -> hls4ml workflow for quantization

Identified problematic layers:

- Upsample (HLS error -> type issue)
- AvgPool (channel last/first issue)

Solved some issues:

- Latest hls4ml update fixed a parsing issue
- Upsample (workaround for empty ROI proposed by Nicolò Ghielmetti)

Investigating solutions:

- Replace AvgPool with MaxPool or fixed-weight Conv ?
- Manually fix HLS type issue or find the precise source of the problem ?

Plan for tomorrow

NGT WP2.2 model → port the model to torch and try PQuant

fAD: test models and experiment with quantized-aware training+pruning

Jet reconstruction net:

- Continue investigating solutions for ongoing issues
- Set up a test to compare PyTorch model performance
- Complete a full synthesis

FPGA



NexTGen
Next Generation Triggers

Torch inference in heterogenous CMSSW software

Leonardo Beltrame, [Lukasz Michalski](#), Davide Valsecchi, Christine Zeh



NextGen
Next Generation Triggers

Plan for the week

- **Integrate Torch-Alpaka Inference** in cmssw, open PR to collect feedback
- **Improve interface for SoA to torch::Tensor** improvements possible with merged PR: <https://github.com/cms-sw/cmssw/pull/47306>
- **Profiling & Optimization:**
 - Ensure compatibility with CMSSW's execution model.
 - Ensure we are able to lock Torch internal optimizations (multithreading, Alpaka GPU queues).
- **Explore Ahead-of-Time (AOT) Compilation:** Investigate how AOT can improve model execution instead of relying on just-in-time (JIT) compilation.

TODAY

What was achieved today

- Exploring **torch Ahead-Of-Time compiler (AOTinductor)** in the context of the CMS Software.
 - Running AOT compiled models for both **CPU and CUDA** support in alpaka code
- Investigate how to use cmssw **CachingAllocator** to eliminate cudaMalloc's inside torch.

Plan for tomorrow

- Come up with a **demo version** of AOT compilation models running with Alpaka producers
- Benchmark **JIT vs. AOT** approaches
- Set up our **CachingAllocator** and profile to check if the problem is gone.

C++ Coroutines

Mateusz Jakub Fila, Eric Cano, Axel
Naumann, Attila Krasznahorkay



NexTGen
Next Generation Triggers

What happened today

- Distilled the [Gaudi](#) example for the NGT repo
 - <https://github.com/cern-nextgen/wp1.7-coroutine-tests/blob/main/examples/gaudi.cpp>
 - Successfully implement lateral transfer in a coroutine algorithm with multilevel delegation to helper tools
 - All the plumbing is done in the awaitable classes (not shown)

```
Gaudi::CoroutineT<int> algorithm()
{
    std::cout << "Starting algorithm" << std::endl;
    co_yield 42;

    std::cout << "Launching tool1" << std::endl;
    co_await tool1("algorithm");

    std::cout << "Continuing algorithm" << std::endl;
    co_yield 84;

    std::cout << "Launching tool2" << std::endl;
    co_await tool2("algorithm");

    std::cout << "Finishing algorithm" << std::endl;
    co_return 126;
}

int main()
{
    std::cout << "Starting main" << std::endl;
    auto alg = algorithm();
    while (!alg.done())
    {
        std::cout << "Main: " << alg.value() << std::endl;
        alg.resume();
    }
    std::cout << "Main: " << alg.value() << std::endl;
    std::cout << "Main finished" << std::endl;
    return EXIT_SUCCESS;
}
```

```
Gaudi::CoroutineT<int> tool1(std::string_view parent)
{
    std::cout << std::format("Starting {}.tool1", parent) << std::endl;
    co_yield 1;
    std::cout << std::format("Continuing {}.tool1", parent) << std::endl;
    co_yield 2;
    std::cout << std::format("Finishing {}.tool1", parent) << std::endl;
    co_return 3;
}

Gaudi::CoroutineT<int> tool2(std::string_view parent)
{
    std::cout << std::format("Starting {}.tool2", parent) << std::endl;
    co_yield 11;

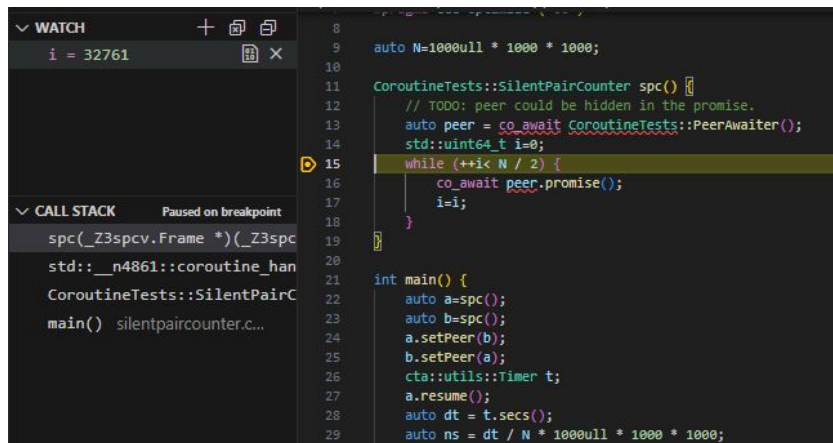
    std::cout << std::format("Launching tool1 from {}.tool2", parent) << std::endl;
    co_await tool1(std::format("{}.tool2", parent));

    std::cout << std::format("Continuing {}.tool2", parent) << std::endl;
    co_yield 12;
    std::cout << std::format("Finishing {}.tool2", parent) << std::endl;
    co_return 13;
}
```

```
Starting main
Starting algorithm
Main: 42
Launching tool1
Starting algorithm.tool1
Main: 1
Continuing algorithm.tool1
Main: 2
Finishing algorithm.tool1
Main: 3
Continuing algorithm
Main: 84
Launching tool2
Starting algorithm.tool2
Main: 11
Launching tool1 from algorithm.tool2
Starting algorithm.tool2.tool1
Main: 1
Continuing algorithm.tool2.tool1
```

What happened today

- Investigated call stack of nested coroutines with ping-pong example
 - Just a [dubious behavior](#) of GCC in low optimisation levels
 - Now works as expected with ≥ 02 , also with clang
 - Measured the coroutine overhead vs single threaded counter and multithreaded semaphore synchronized counted (time per iteration)
 - Single thread 0.460 ns, with coroutine switching: 1.8 ns, with semaphore based multithreading: 3.6 μ s



```
8
9 auto N=1000ull * 1000 * 1000;
10
11 CoroutineTests::SilentPairCounter spc() {
12     // TODO: peer could be hidden in the promise.
13     auto peer = co_await CoroutineTests::PeerAwaiter();
14     std::uint64_t i=0;
15     while (++i < N / 2) {
16         co_await peer.promise();
17         i=i;
18     }
19 }
20
21 int main() {
22     auto a=spc();
23     auto b=spc();
24     a.setPeer(b);
25     b.setPeer(a);
26     cta::utils::Timer t;
27     a.resume();
28     auto dt = t.secs();
29     auto ns = dt / N * 1000ull * 1000 * 1000;
```

What happened today

- Investigated `conurrencpp`, the C++ concurrency library\
 - <https://github.com/David-Haim/conurrencpp>
 -

Table of contents

- `conurrencpp` overview
- Tasks
 - `conurrencpp` coroutines
- Executors
 - `executor` API
 - `Executor` types
 - Using executors
 - `thread_pool_executor` API
 - `manual_executor` API
- Result objects
 - `result` type
 - `result` API
 - `lazy_result` type
 - `lazy_result` API
- Parallel coroutines
 - Parallel Fibonacci example
- Result-promises
 - `result_promise` API
 - `result_promise` example
- Shared result objects
 - `shared_result` API
 - `shared_result` example
- Termination in `conurrencpp`
- Resume executors

- Utility functions
 - `make_ready_result`
 - `make_exceptional_result`
 - `when_all`
 - `when_any`
 - `resume_on`
- Timers and Timer queues
 - `timer_queue` API
 - `timer` API
 - Regular timer example
 - Oneshot timers
 - Oneshot timer example
 - Delay objects
 - Delay object example
- Generators
 - `generator` API
 - `generator` example
- Asynchronous locks
 - `async_lock` API
 - `scoped_async_lock` API
 - `async_lock` example
- Asynchronous condition variable
 - `async_condition_variable` API
 - `async_condition_variable` example

Plan for tomorrow

- Further investigate concurenCpp
 - Might provide some of the other targets
- Explore executing coroutines on a thread-pool to achieve asynchronicity
- Write example utility functions:
 - Aggregating await to resume when all operations are done(`co_await when_all(task1, task2)`)
 - Waiting synchronously for async coroutine (`task1.run(thread_pool).wait()`)
- How to properly integrate callback based asynchronous calls