

Day 1

1st NextGen Hackathon



NextGen
Next Generation Triggers

[: ^^AoSoAoSoAoSoAoS :]

```
auto Names_of_people_in_your_group = ^^{
```

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

```
};
```



NextGen
Next Generation Triggers

Introduction to your topic

- Introduction to C++ **reflections** and token injection
- Exploring examples on Godbolt compiler explorer
- Discussing **feature requests** for NextGenTrigger's SoA library
 - Views of sub-ranges for SoA
 - Usefulness of AoSoA
 - Compatibility with the standard library (e.g. iterators)

Plan for the week

- **Tuesday:**
 - SoAs with C++ preprocessor macros
- **Wednesday:**
 - SoAs with C++ template metaprogramming
- **Thursday & Friday:**
 - How can we combine solutions/unify the implementations?

Plan for tomorrow

- Introducing how SoAs are implemented in CMSSW
- Using SoAs in CMSSW
- Discuss how to achieve new planned developments
- Discuss future possible developments

Running event generators on GPU

Daniele Massaro

Davide Di Croce

Johannes Junggeburth



NexTGen
Next Generation Triggers

Introduction to your topic

- MadGraph is a Monte Carlo event generator for high-energy physics.
- It is an automatic meta-code that writes the code for computing the cross-section and generating events for any process at colliders.
- It can:
 - generate all the feynman diagrams for a certain process;
 - automatically compute the cross section for that process;
 - generate events and store them in LHE format;
 - interface with other tools (like for shower and hadronisation - Pythia).
- Originally, MadGraph has been developed with only CPUs in mind, and the event generation can be parallelised on multiple cores.
- Recently, IT-FTI released a new plugin that allows MadGraph to use hardware acceleration for part of the event generation.
- This means using both GPUs and CPUs with vectorised instructions.

Plan for the week

- [MadGraph introduction](#);
- [Getting familiar with MadGraph](#):
 - how to install;
 - first event generation;
 - change the configuration;
 - install and link additional tools (e.g. Pythia);
 - AOB and any suggestions welcome.
- [The cudacpp plugin](#), a.k.a. running MadGraph on GPU/vectorized CPU:
 - the MadGraph plugin system and how to install;
 - details on hardware acceleration in MadGraph;
 - perform a MadGraph run using `cudacpp` plugin;
 - various possible settings.
- [Finding bottlenecks in the code](#):
 - profile MadGraph;
 - using [Adaptyst](#) to obtain flamegraphs;
 - discuss future updates.

Plan for tomorrow

- Finalize running MadGraph on GPUs:
 - inspect throughput and GPU usage.
- Explore profiling options, and create flamegraphs for the main executable.
- Hands-on with Adaptyst profiler.

Fast ML Inference

Lorenzo Moneta, Anastasiia Petrovych, Enrico Lupi,
Alessandro Crespi, Sanjiban Sengupta
Jamie Gooding, Jonas Rembser



NexTGen
Next Generation Triggers

Introduction to your topic

- Intro to ROOT, TMVA/SOFIE
- Building ROOT in Mac, AlmaLinux, Windows
- Building standalone SOFIE with ROOT binary
- Experimenting SOFIE with statistical inference model

Plan for the week

- Build ROOT!
- Add support for Automatic Differentiation through SOFIE's generated code
- Make SOFIE standalone
- Experiment with Python Interface of SOFIE
- SOFIE Memory Optimization
- Extend SOFIE Operator Support
- SOFIE Dynamic Computation Support

Plan for tomorrow

- SOFIE Memory Optimization
- Experiment SOFIE with Smart Pixels model
- SOFIE ALPAKA Integration
- SOFIE Dynamic Computation support

Fast ML Inference using hls4ml

Dimitrios Danopoulos, Roope Niemi



NextGen
Next Generation Triggers

Introduction to your topic

- Hls4ml is a package for ultra-low latency targeting FPGA inference
 - Parses a trained model in ONNX, Keras or PyTorch
 - Generates optimized inference code (HLS) for the target FPGA

- PQuant is a library for training compressed ML models (public from today)
 - Helps optimizing the models (pruning, quantization) before deploying them to hls4ml
 - Currently supports PyTorch (TF under development)

Plan for the week

- Install the dependencies and build hls4ml and PQuant libraries
- Profile the models, identify bottlenecks/problematic layers
- Perform optimization on the SW level (use PQuant, HGQ)
- Perform optimization on the HW level (refining operators, bit level optimization, etc)

Project title / ID	Description	DL framework	Type of model + params	Plan
QDips	Optimize jet-tagging model, as an exploration of Deep Sets architecture	QKeras / TF	Conv1D -> custom sum -> dense	Make model definition more hls4ml friendly and remove manual quantization definition in model hls4ml config (and use QKeras one).
NGTWP2.2	Enhance L0 MDT Trigger	Keras	Conv1D, Conv2D, RNN, LSTM, Dense	1. Test optimization on Convolutional models. 2. test LSTM layers
2 FCNs	Convert 2 FCN models for Jets Reconstruction	PyTorch/Brevitas/QONNX	Conv2D, MaxPool2D, AvgPool2D, Upsample/ConvTranspose2D, BatchNorm2D, ReLU	1. Identify problematic layers 2. Resolve issues 3. Begin optimization
CaloJetSSD	Calorimeter Jet Single Shot Multibox Detector for ATLAS Trigger System	PyTorch/Brevitas	Too Many	Continue to try and convert it with HLS4ML
fAD	flow-based Anomaly Detection for triggers	PyTorch	Dense; but custom forward?	1. get a decently performing model 2. understand if I want a custom forward to solve the ODE or I should switch to a FF model 3. distill it to a good complexity level 4. test the actual export to FPGA

Torch Alpaka Inference in CMSSW

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



NextGen
Next Generation Triggers

Introduction to your topic

Torch + Alpaka Fast ML Inference

- **Memory Optimizations:**
 - Interface for direct conversion of SoA data into PyTorch tensors without explicit data movement.
 - Leverages GPU memory and uses memory-stealing to reduce unnecessary copy operations.
 - Supports optimized memory layout, computation of tensor strides, and flexible column/tensor list ordering.
 - Simplifies integration with ML models, especially for users of PortableCollection concepts in CMSSW.
- **CMSSW Integration:**
 - Fully integrated into the CMSSW framework, ensuring controlled execution without hidden multithreading or process spawning by PyTorch
 - Prevents use of Torch's internal thread pool, preserving CMSSW resource management strategy.
 - Enables fast and seamless use of industry-standard ML libraries inside the framework.
 - Aligned with ongoing improvements in Alpaka, promoting future-proof, portable module development.

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.

Plan for tomorrow

- **Dig a bit more deeply into AOT compilation**
- Close open topics regarding PR status

C++ Coroutines

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



NexTGen
Next Generation Triggers

Introduction to your topic

- C++20 introduced “coroutines”
 - <https://en.cppreference.com/w/cpp/language/coroutines>
- They are **super generic**
 - Hopefully they could be used in our offline software for efficient asynchronous task scheduling. But also possibly other things.
- Spent practically the entire day having technical discussions
 - Learning about coroutines, from the very helpful examples prepared by Mateusz

Plan for the week

- Continue discussions...
- Add coroutine usage to an old test project of Axel's
 - <https://github.com/Axel-Naumann/asyncforest/blob/master/asyncforest.cxx>
 - See what performance overhead / improvement we would get out of using coroutines
- Add naive coroutine usage to Gaudi
 - <https://gitlab.cern.ch/gaudi/Gaudi>
- Have a more detailed look at coroutine related features in Boost
 - Try to compare them with what's available in C++20

Plan for tomorrow

...?



NextGen
Next Generation Triggers