Introducing GPUs for Electron/Photon Reconstruction

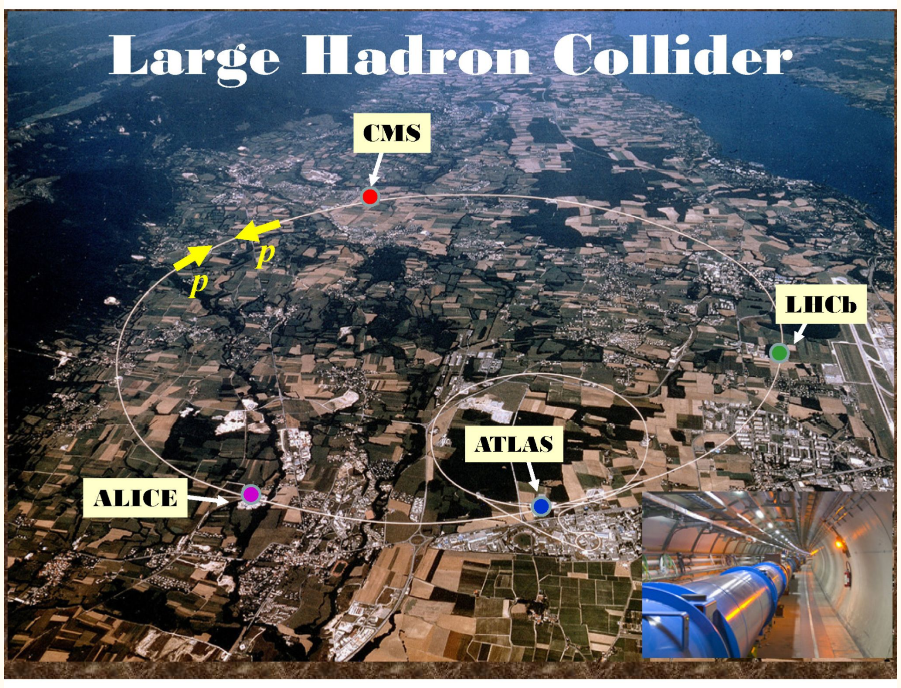
Gabriel Soto, Intern, Fresno State, and Charis Kleio Koraka, Mentor, University oF Wisconsin-Madison

Overview

- The LHC and the CMS detector
- Electron and photon reconstruction
- GPU vs CPU hardware
- Technical details
- Histograms
- Conclusion

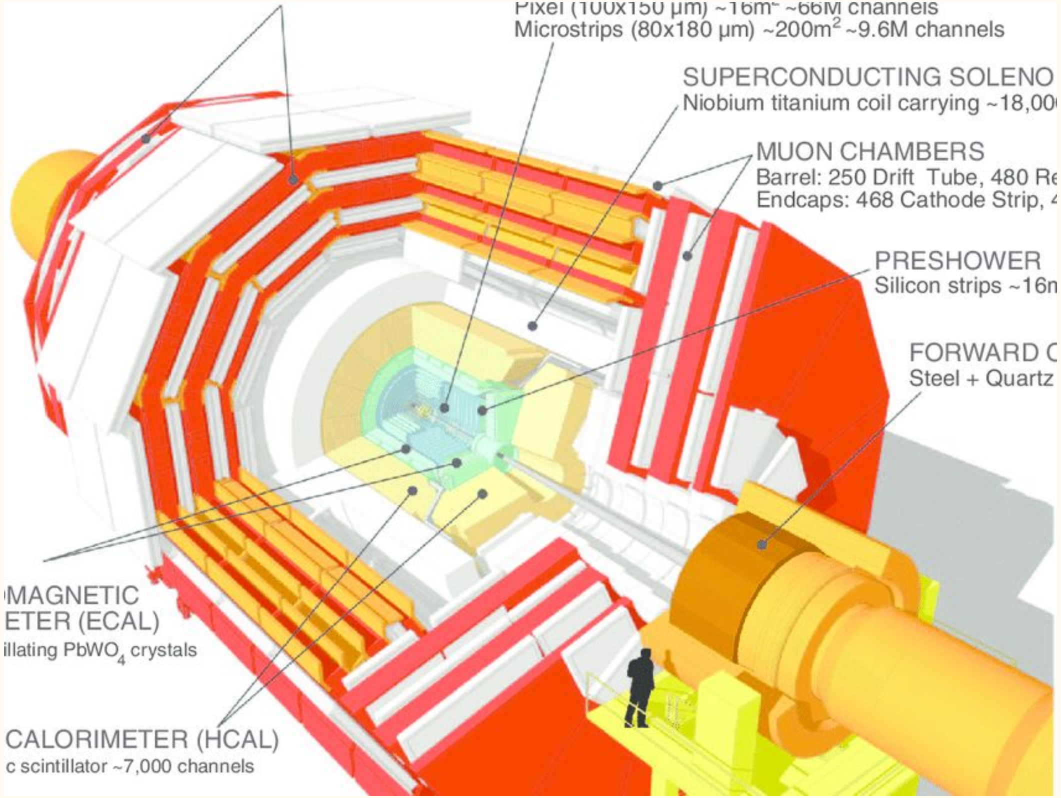
The LHC

- Worlds largest and most powerful particle collider.
- Consists of four main detectors: CMS, ATLAS, ALICE, and LHCb.
- Highest recorded beam energy is 13.6 TeV.



CMS Detector

- CMS detector is one of the largest particle detectors built.
- Detector itself consists of several different sub-detectors such as calorimeters and trackers, each with a specific job.
  - Pixel Tracker
  - ECAL
  - HCAL
  - Muon Detectors
  - Superconducting Solenoid
  - Preshower
- Electromagnetic calorimeter (ECAL) is the calorimeter within the detector that detects electrons and photons.
- The magnet between calorimeter and the muon chambers with a strength of 3.8 T.


Pixel (100x150 µm) ~16m^2 ~66M channels
Microstrips (80x180 µm) ~200m^2 ~9.6M channels
SUPERCONDUCTING SOLENO
Niobium titanium coil carrying ~18,00
MUON CHAMBERS
Barrel: 250 Drift Tube, 480 Re
Endcaps: 468 Cathode Strip, 4
PRESHOWER
Silicon strips ~16m
FORWARD C
Steel + Quartz
MAGNETIC
ETER (ECAL)
illating $PbWO_4$ crystals
CALORIMETER (HCAL)
c scintillator ~7,000 channels

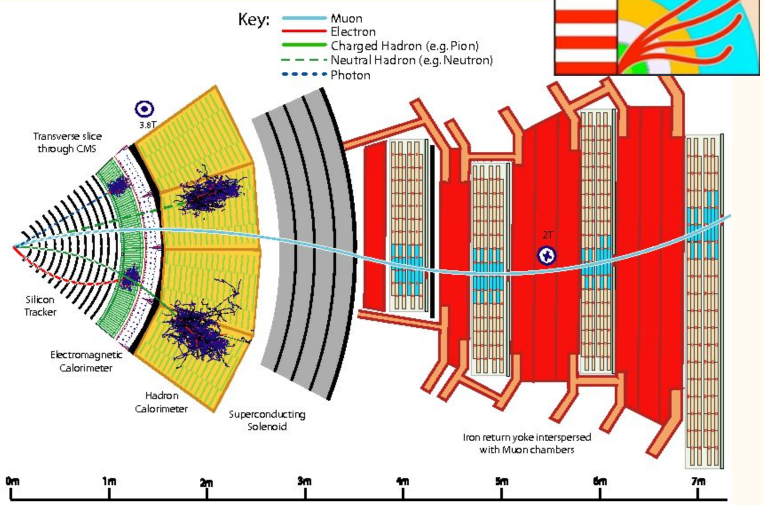
Event reconstruction

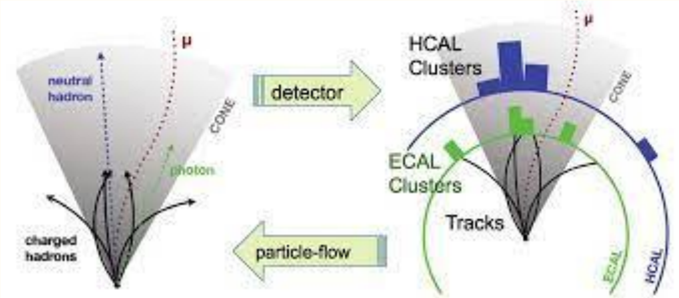
Different particles leave traces in different detectors:

- First, particles pass through the tracker which identifies charged particle trajectories.
- Second it passes through the ECAL. Electrons and photons are stopped here.
- If a particle is not a photon or an electron it will continue throughout the detector to the HCAL
- Muons are minimum ionizing particles and are detected using the muon chambers.

Information from all detectors is used to reconstruct the particles

- The software used is CMSSW.
- Different algorithms are employed (i.e. particle flow) to reconstruct different quantities

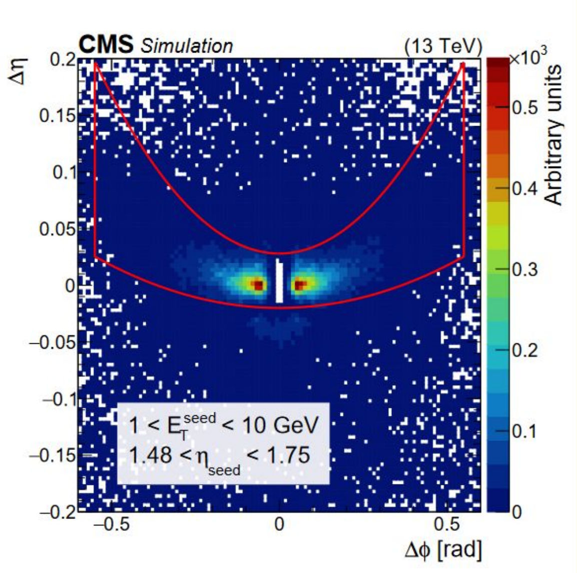

Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon
3.8T
2T
Transverse slice through CMS
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers
0m
1m
2m
3m
4m
5m
6m
7m



Electron reconstruction

For the electron reconstruction information from the tracker and the ECAL is used :

- **ElectroMagnetic Calorimeter (ECAL) :**
    - This is where electrons and photons deposit all of their energy.
    - The energy deposited within the detector crystals can form an ECAL cluster.
    - Then Super Clusters (SCs) are formed as a big group of clusters using the Mustache Algorithm.
- **Tracker:**
    - Is a silicon tracker that measures charged particles within $|\eta| < 2.5$.
    - Is composed by silicon pixels and strips.



Seeds

What are the electron seeds?

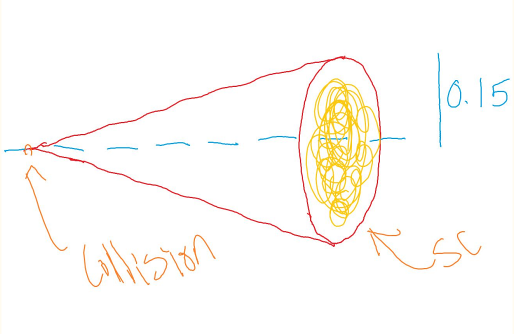
- Seeds are hits in a given area within the tracker that are used as input in tracking algorithms. Electron seeds are hits that could potentially be originating from an electron and lie on the electron trajectory.

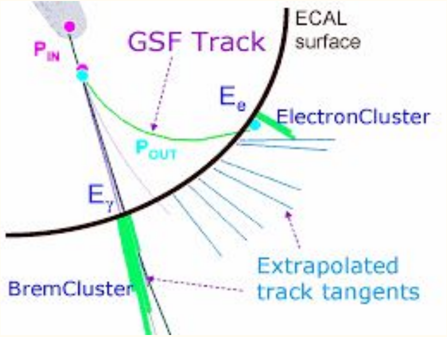
Why do we need seeds?

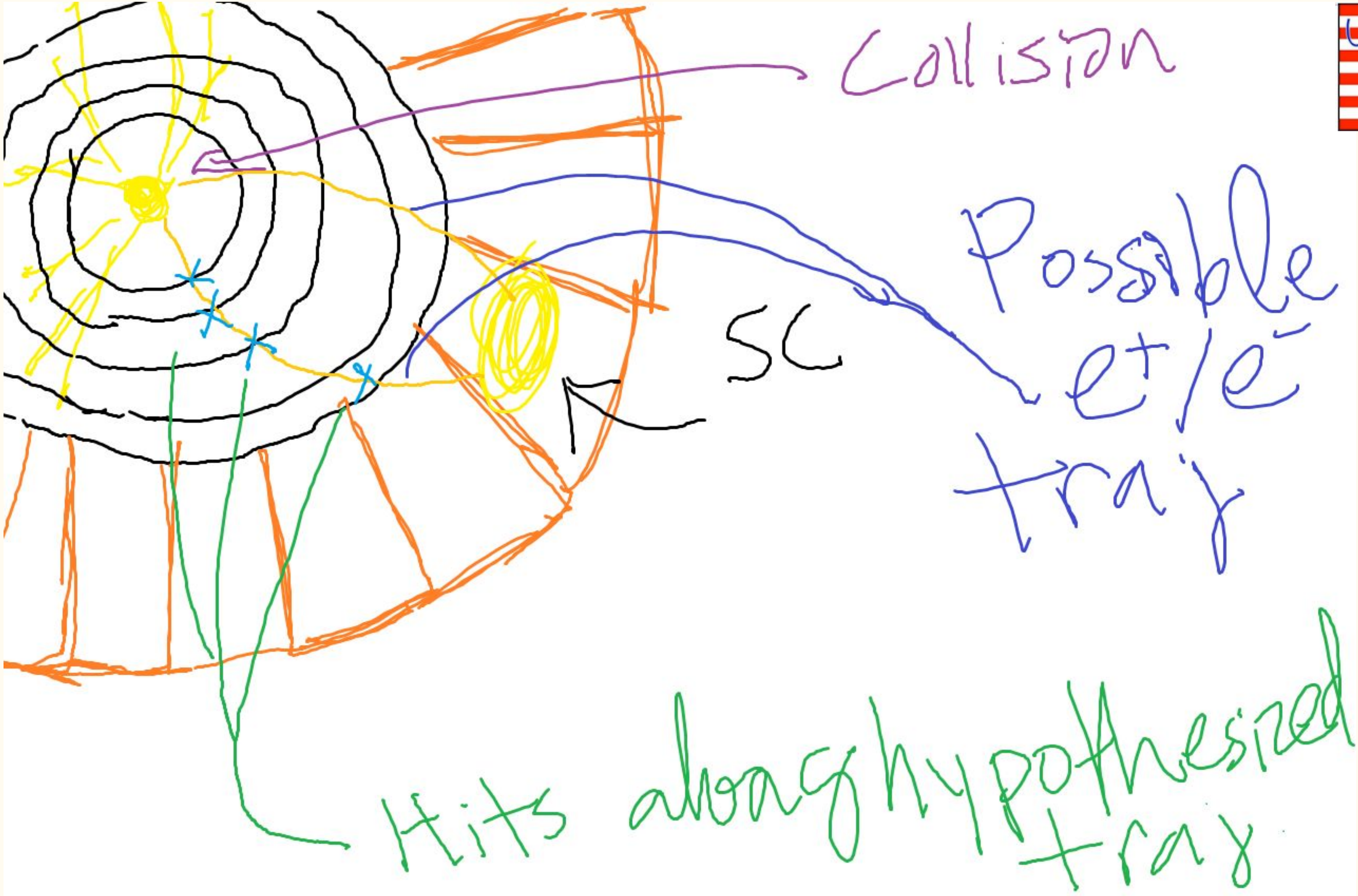
- Tracking algorithms are computationally intensive(GSF) so we cannot use every single tracker hit.

Electron Seeding and How it is Done

- Reconstruction of an electron track begins with the identification of a hit pattern that may lie on an electron trajectory.
- Electron trajectory seeds can be ECAL or tracker-driven (ECAL approach is better at high energy).
- ECAL driven seeds select mustache SC's with energy $>$ 4GeV and with a cone of ΔR=0.15 centered at the SC position.
- For each SC a trajectory is hypothesized, for either a electron or a positron:
  - If we find a tracker hit along the hypothesized trajectory, then these hits that are identified as electron seeds and are used as inputs in the GSF tracking algo.





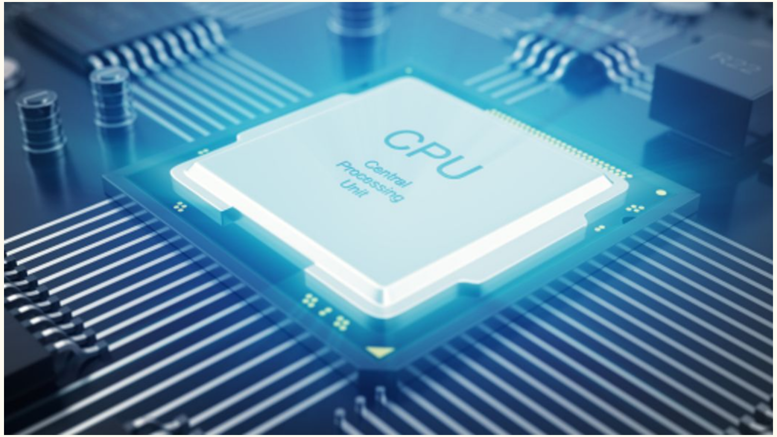

Collision
Possible e^+/e^- traj
SC
Hits along hypothesized traj

Graphics Processing Unit (GPU)

CPUs vs GPUs

CPU

- Uses only a few cores
- Can run complex tasks
- Cannot run many tasks in parallel
- Low latency



GPU

- Uses thousands of cores
- Runs simple tasks
- Can run many tasks in parallel
- High throughput

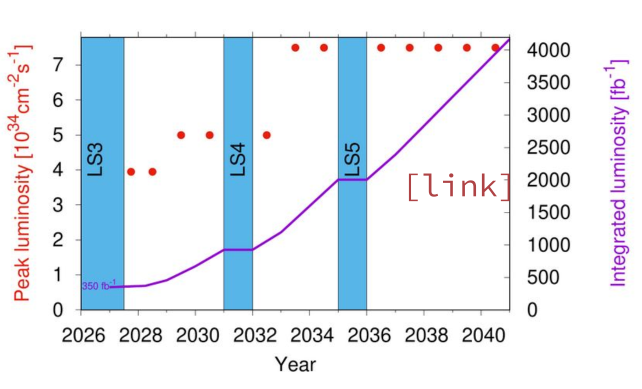
Computing at the High Luminosity-LHC

What we expect:

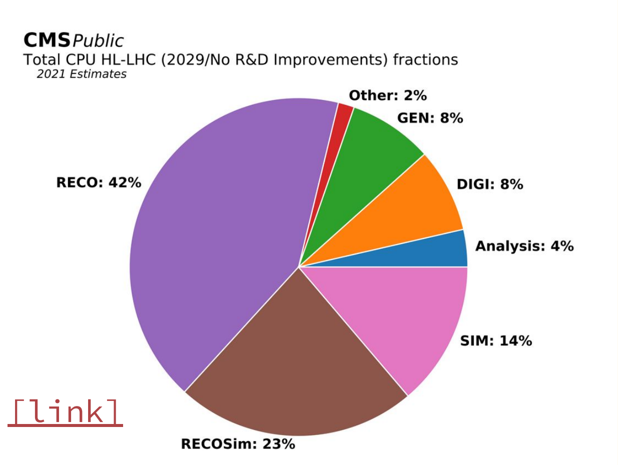
- 2-3 times greater instantaneous luminosity compared to Run-2.
- Much larger event processing rate.
- Unique challenge for online and offline event reconstruction.

How to mitigate this:

- Use of different co-processors designed to handle specific tasks in parallel.
- Current software should be adapted to run on such systems.



[link]



[link]

Technical Details

How to Get Environment Setup

Connect to lxplus

Create a directory to work in:

```
mkdir myDirectory
cd myDirectory
```

Download a CMSSW release and add the relevant packages in working area:

```
cmsrel CMSSW_12_4_0
cd CMSSW_12_4_0/src
cmsenv  # This commands sets the environment variables in order to run cmssw code
git cms-addpkg RecoEgamma/EgammaElectronProducers
git cms-addpkg DataFormats/EgammaReco
git cms-addpkg RecoEgamma/EgammaElectronAlgos/
git cms-checkdeps -a
```

Running

Copy in directory:

https://cernbox.cern.ch/index.php/s/rfAU9SLQy4JRVap

Run the config:

```
cmsRun step1.py #this returns a new root file called step0_HTL.root which starts from a raw file and runs the High Level Trigger (HLT) step.
```

Using Resulting Files

Clone repository:

```
git clone git@github.com:ckoraka/egProducer.git
mv step1_HLT.root  egProducer/Producer/test/
```

Recompile:

```
scram b -j 8
```

Run Producer:

```
cd egProducer/Producer/test
cmsRun eganalyzer_cfg.py #This should output a root file. This runs over the output of the previous step.
```

Histograms

Simulations

Three ROOT simulations were used:

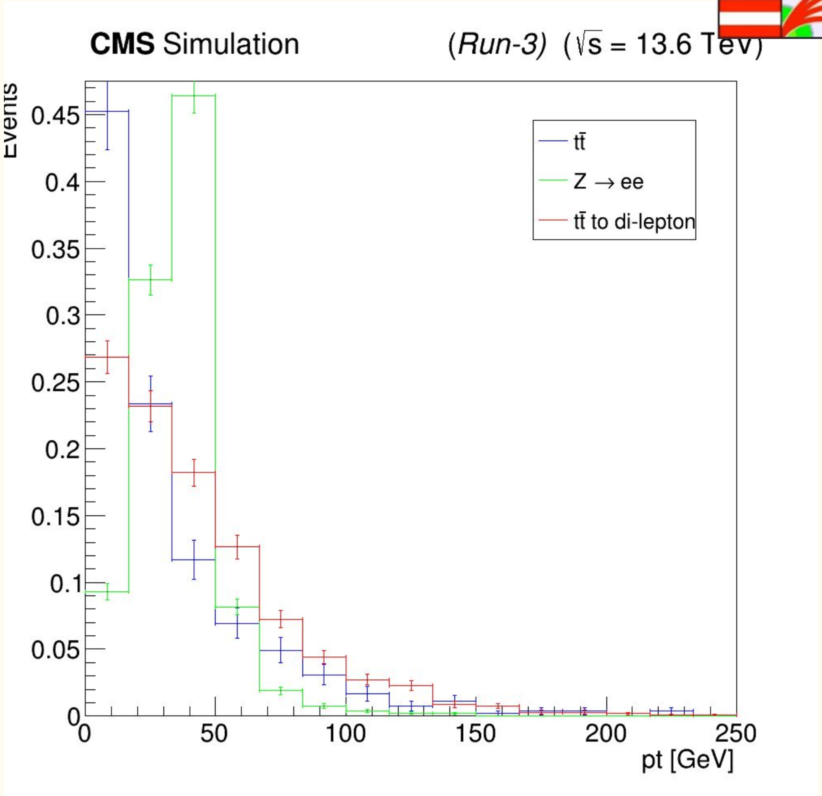
- ttbar
- Z → e e
- ttbar → Di-Lepton

Number of Events:

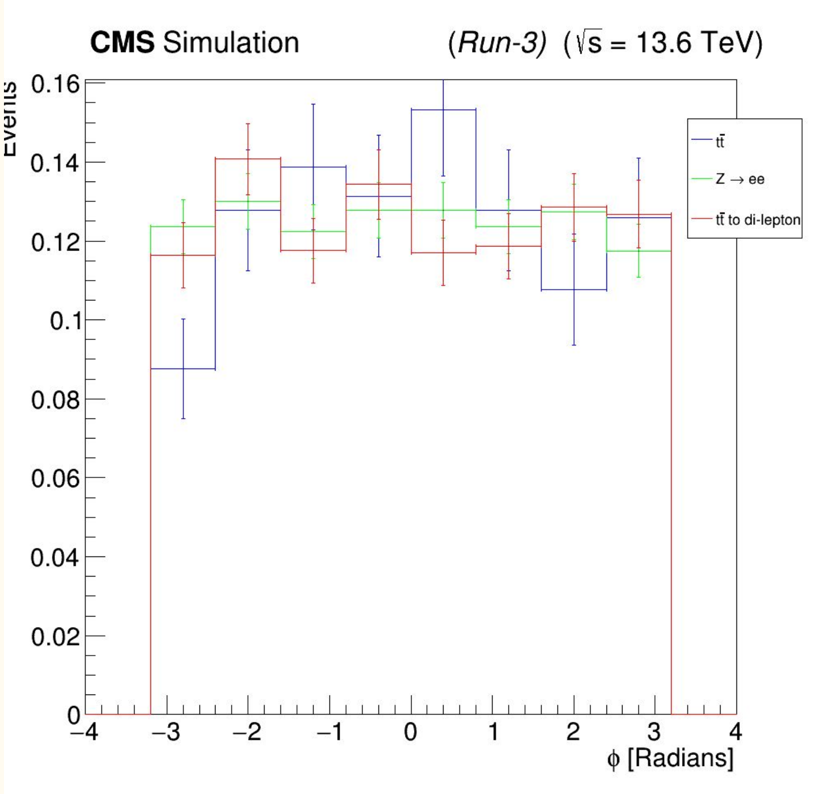
- 2000

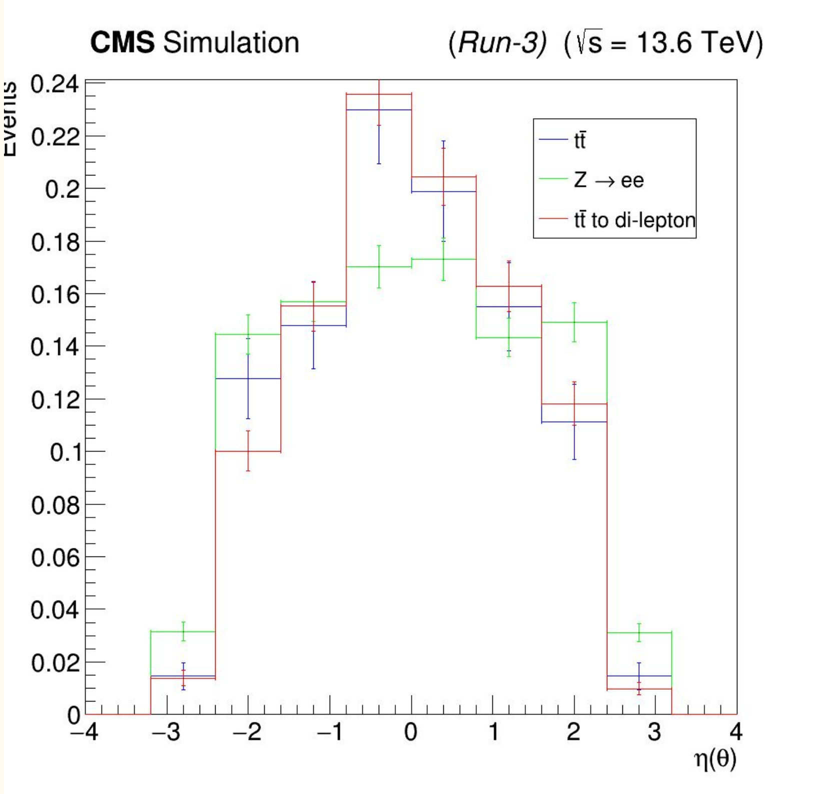
Electron pt Normalized

- The electron pt spectrum is harder for the Z → ee simulation compared to the ttbar simulations.
- In the tt simulation, this is due to electrons originating from the W boson, which is lighter than the Z boson, that are produced along with an electron neutrino.
- Histograms are normalized to area.



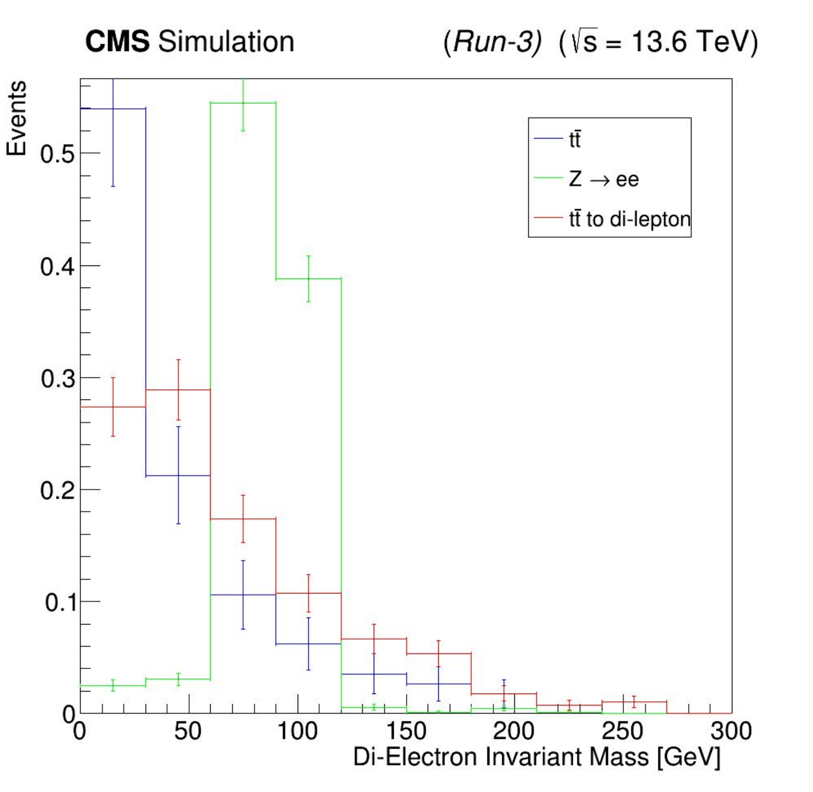
Electron eta and phi Normalized





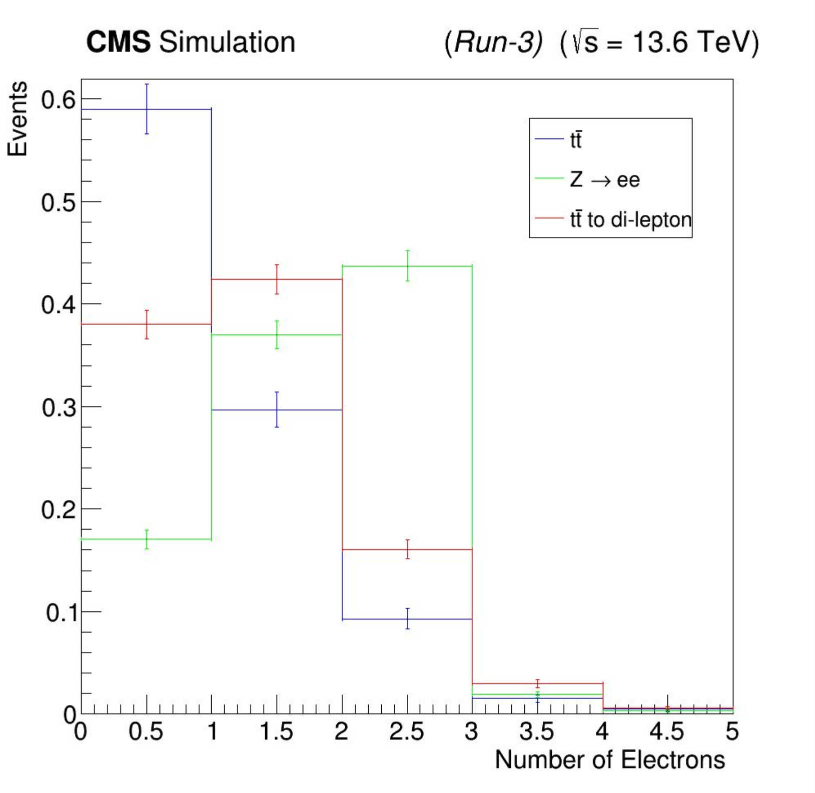
Di-Electron Invariant Mass Normalized

- For the Z → ee simulation, we expect a peak at 90 GeV, which is the Z boson mass. For the ttbar simulation, we do not expect any peak.



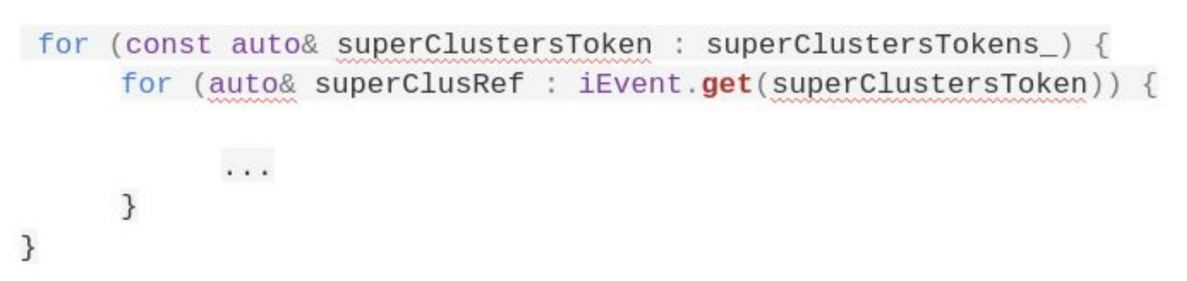
Number of Electrons per Event Normalized

- For ttbar, it is know that 99.9% of the time it decays into a W boson and a b quark. The W boson does not always decay into a lepton and lepton neutrino pair, so there is a difference between ttbar and ttbar → dilepton.
- The Z → ee simulation, for each event we expect 2 electron. This is why the peak is shifted with respect to the ttbar simulations.



Improving Electron Seeding via Parallelization

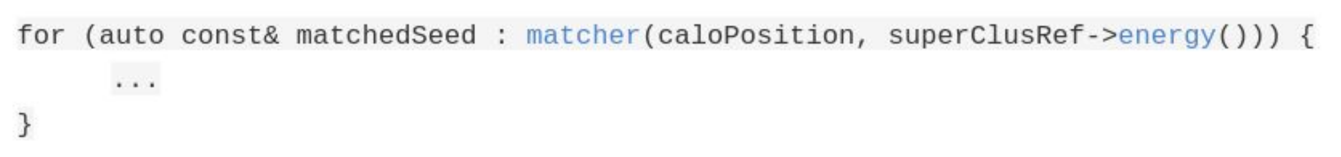
- Inside the script that performs the electron seeding, we can write a Kernel that would assign a tracker hit and ECAL supercluster to a different GPU thread.
- The first part we identified that can be parallelized is the loop over the different ECAL SCs as shown below :



```
 for (const auto& superClustersToken : superClustersTokens_) {
      for (auto& superClusRef : iEvent.get(superClustersToken)) {

           ...
      }
}
```

- Additionally, for each SCs, a loop over the collection of tracker hits is performed. Therefore each hit, is checked whether it geometrically agrees with the hypothesized electron trajectory. This part can be parallelized as well :



```
for (auto const& matchedSeed : matcher(caloPosition, superClusRef->energy())) {
      ...
}
```

Conclusions

- Electron reconstruction is a major part of the CMS reconstruction software.
- With the expected increase in luminosity, more efficient algorithms and new types of hardware, such as GPUs, should be explored.
- A description of the electron seeding algorithm and its potential to be adapted to run on GPUs was described.
- Implementation of this will hopefully result to a speed up of the reconstruction software.

Results of Internship

- Worked remotely on daily basis with CMS physicists, PhD students, and post-docs.
- Learned new methods of critical thinking, technical details of GPUs vs CPUs, and a new coding language called CUDA.
- Strengthen my knowledge on particle detectors, ROOT, C++, python, troubleshooting code, and github.
- Gained a good relationship with Dr. Charis Kleio Koraka.
- Paid a total of $6000.

Citation

- *Electron and photon reconstruction and identification with the CMS experiment at the CERN LHC*. (2021). IOPscience. https://iopscience.iop.org/article/10.1088/1748-0221/16/05/P05014.

BACK - UP

Algorithm Version-How do we match the trajectories and the hits?

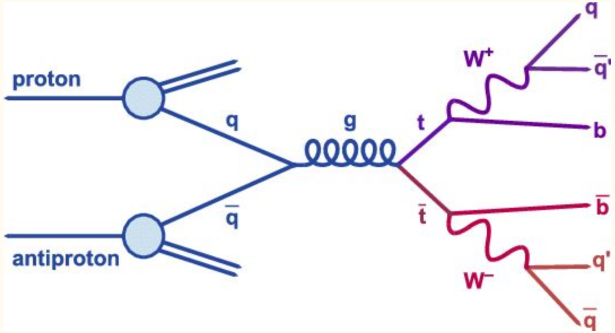
- Tracks are evaluated using the Kalman Filter(KF) algorithm to create an electron trajectory.
- KF proceeds iteratively from the seed layer, starting from a rough estimate of the track parameters provided by the seed. It also includes the information of the succeeding detection layers one by one.
- While taking into account the electron energy loss using Bethe-Heitler Distribution.
- If the algorithm finds hits that satisfy the track, it creates candidate trajectories by using a Chi-squares fit.
- Penalties are applied to the Chi-squared when there are missed hits.
- This helps minimize the inclusion of bremsstrahlung photons in hits of the electron trajectory.

Physics Version(ECAL)

- When high energy particles are flying towards the ECAL, if the particle is an electron, positron, or a photon the particle is stopped by the ECAL detector.Why?
- A calorimeter measures the energy a particle loses as it passes through. It is usually designed to stop entirely or “absorb” the particles coming from a collision.
- They are usually made up of very dense material.
- This forces them to deposit all of their energy within the detector, thus measuring their full energy.
- Electromagnetic calorimeters measure the energy of electrons and photons as they interact with the electrically charged particles in matter.

ttbar simulation

- t quark first viewed by Kadanoff and Ceva in 1971.
- Most massive quark.
- Interacts via with all forces, but can only decay through weak interactions.
- Top decays ~99.9% into W boson and b quark (rarely a strange quark, or a down quark).
- Lifetime ~5×10^{-25} s.
- W can decay into many different combinations, lepton and antilepton or quark and antiquark.



Altering Prebuilt Code

There were a few instances where the code was not being built or compiled correctly. This resulted in having to manually modify the code.

- Changing certain paths in egammaProducer.cc that were called on.
- Changing the number of events in step1_HLT.py to add better better statistics.

Troubleshooting

When running the code I ran into issues with the building code. It would save and modify the current build information.

Terminal commands:

rm -rf * (after entering the CMSW_12_4_0/python directory)

scram b clean

Initializing Variables from the Output ROOT File

Create python file:

```
nano output.py
```

Input this code:

```
import ROOT

import numpy as np

f1 = ROOT.TFile("output.root", "OPEN")

NBin = 13

binning =
np.array((0.,15.,30.,45.,60.,75.,90.,105.,120.,135.,150.,165.,180.,195.))
```

```
h0 = ROOT.TH1F("h0","NUmber of Elecrons", 5, 0, 5)

h1 = ROOT.TH1F("h1",";pT [GeV];410/15 ",NBin,binning)

h2 = ROOT.TH1F("h2","Electron phi",10,-4,4)

h3 = ROOT.TH1F("h3","Electron eta",10,-4,4)

h5 = ROOT.TH1F("h5","Di-Electron invariant
mass",10,0,100)

electron1 = ROOT.TLorentzVector()

electron2 = ROOT.TLorentzVector()

eTot = ROOT.TLorentzVector()

h5.Fill(eTot.M())
```

Filling Histograms

```
for evt in f1.Get("egammaReconstruction/tree"):
    h0.Fill(len(evt.electron_pt))
    for i in range(0,len(evt.electron_pt)):
            h1.Fill(evt.electron_pt[i])
            h2.Fill(evt.electron_phi[i])
            h3.Fill(evt.electron_eta[i])
    if(len(evt.electron_pt) < 2):
        continue
    electron1.SetPtEtaPhiE(evt.electron_pt[0], evt.electron_eta[0], evt.electron_phi[0], evt.electron_energy[0])
    electron2.SetPtEtaPhiE(evt.electron_pt[1], evt.electron_eta[1], evt.electron_phi[1], evt.electron_energy[1])
    eTot = electron1 + electron2
```

Drawing Histogram and Canvas then Saving it

```
c0 = ROOT.TCanvas("c0","c0",1000,1000)
h0.SetTitle("CMS Simulation         Run3 (13.6 TeV)")
h0.GetXaxis().SetTitle("Number of ELectrons")
h0.GetYaxis().SetTitle("Events")
h0.Draw("HIST")
h0.Draw("C9E1 SAME")
l0 = ROOT.TLegend(0.8, 0.7, 0.95, 0.75)
l0.AddEntry(h0, "ttbar", "l")
l0.Draw()
c0.Update()
c0.Draw()
c0.SaveAs("Number_Electron.png")
```

Finishing up Histograms and Closing File

This process was used repeatedly used for all histograms. Up until we finished and closed the code as shown in this block of code.

```
c5 = ROOT.TCanvas("c5","c5",1000,1000)
h5.SetTitle("CMS Simulation        Run3 (13.6 TeV)")
h5.GetXaxis().SetTitle("Di-Electron Invariant Mass [GeV]")
h5.GetYaxis().SetTitle("Events")
h5.Draw("HIST")
h5.Draw("C9E1 SAME")
l5 = ROOT.TLegend(0.8, 0.7, 0.95, 0.75)
l5.AddEntry(h5, "ttbar", "l")
l5.Draw()
c5.Update()
c5.Draw()
c5.SaveAs("Electron_Di-Electron.png")
f1.Close()
```