GPU APPLICATION IN JUNO

Wuming Luo

May 30th 2019

高能物理计算和软件会议

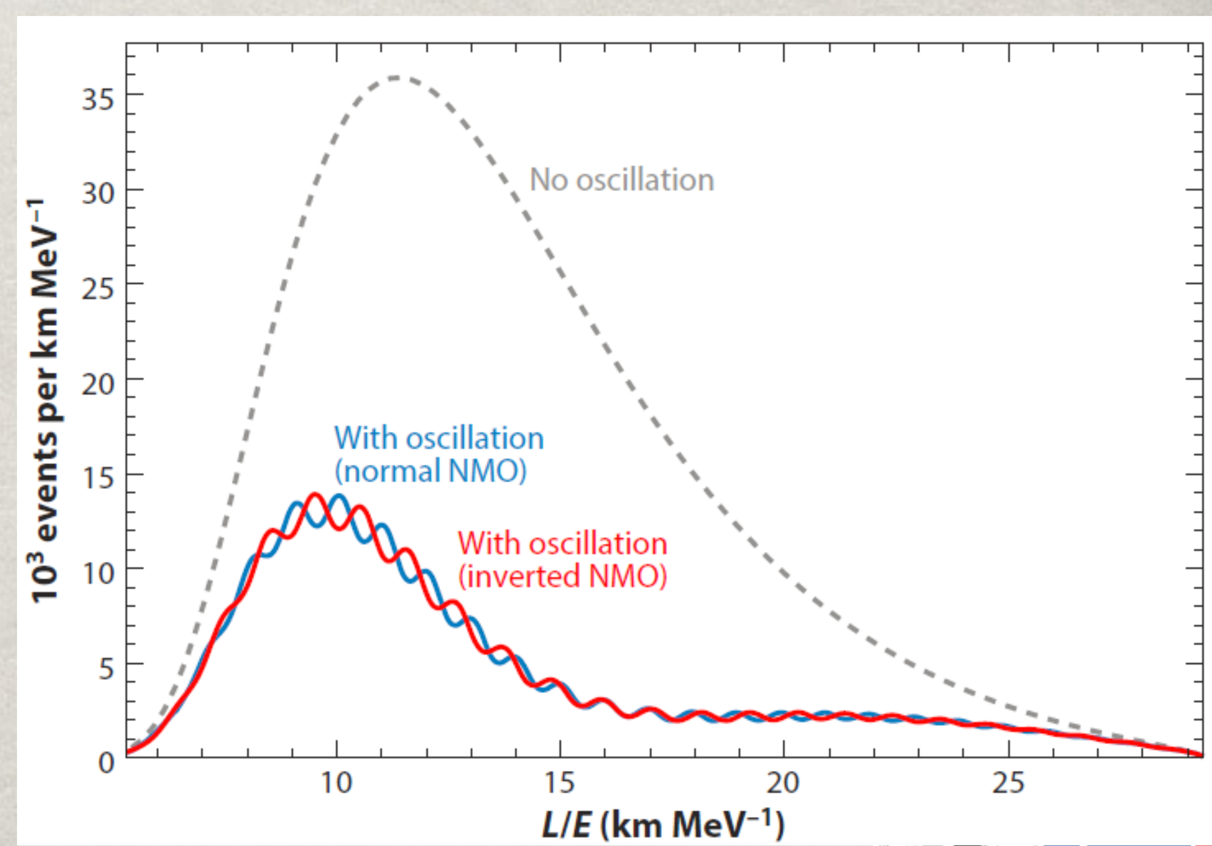
南京大学

OUTLINE

- Introduction to JUNO
- GPU vs CPU
- Applications
  - Vertex Reconstruction
  - Muon Simulation
  - Deep Learning
- Discussion
- Summary

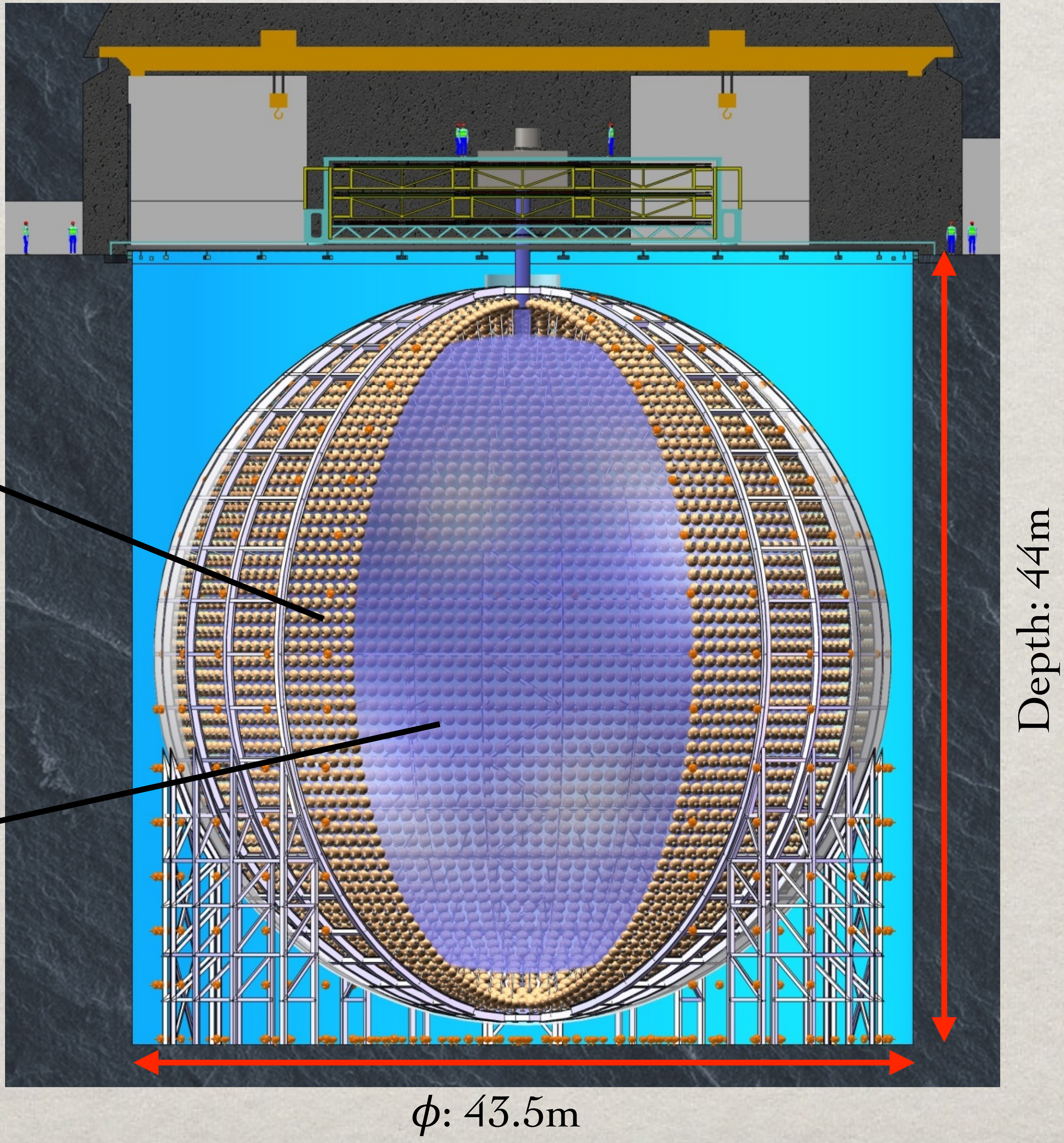
JUNO

- Jiangmen Underground Neutrino Observatory(JUNO):
  - Largest liquid scintillator detector (20 kton)
- Primary physics goals:
  - Determine the neutrino mass hierarchy
  - Measure neutrino oscillation parameters precisely



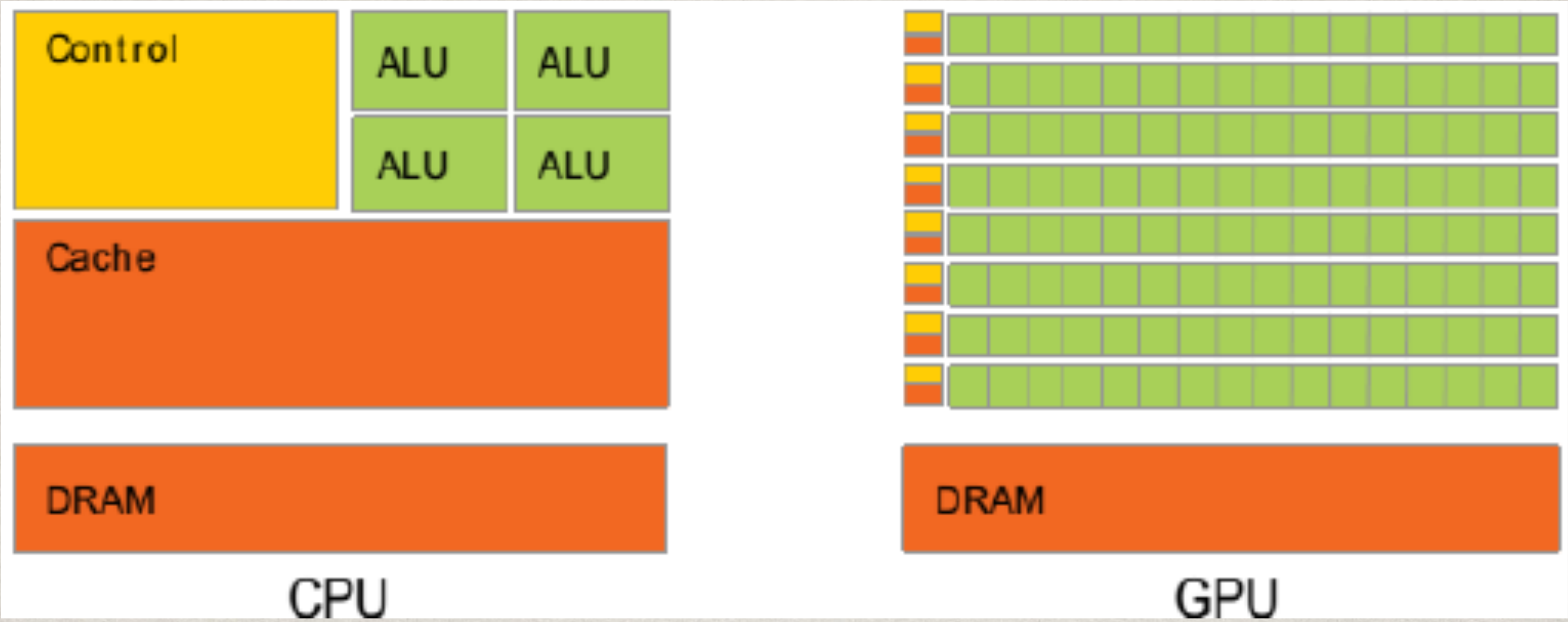
DETECTOR

Central Detector PMT
~18,000 20" PMTs
+ ~25,000 3" PMTs



Liquid Scintillator
20kton

GPU VS CPU



Large Cache

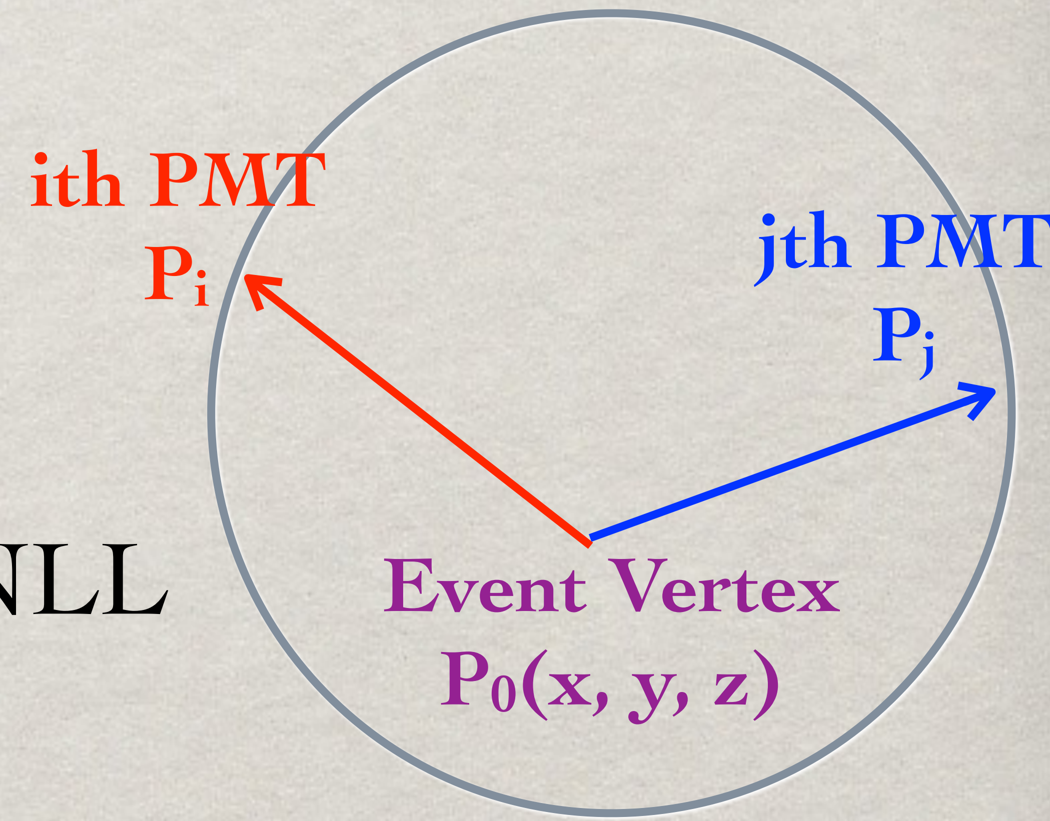
Many cores

Optimized for serial operations

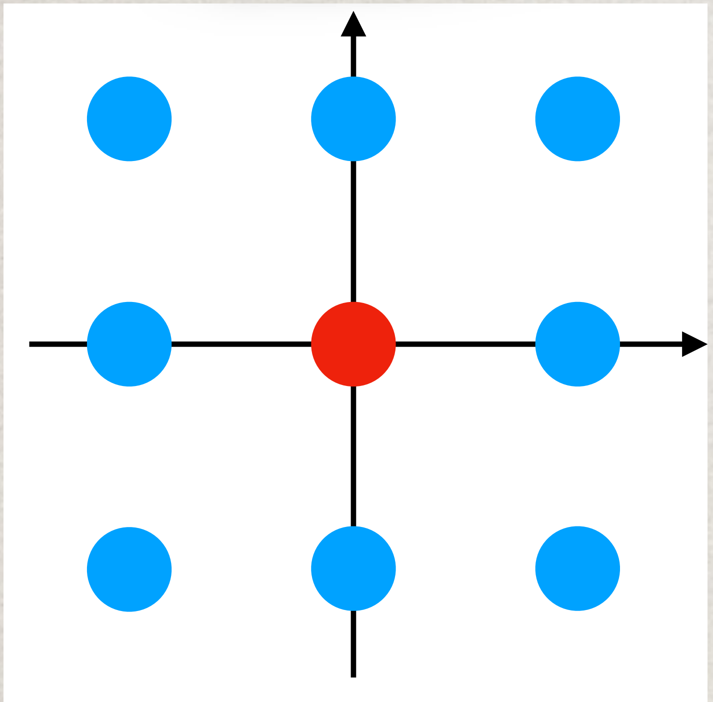
Built for parallel operations

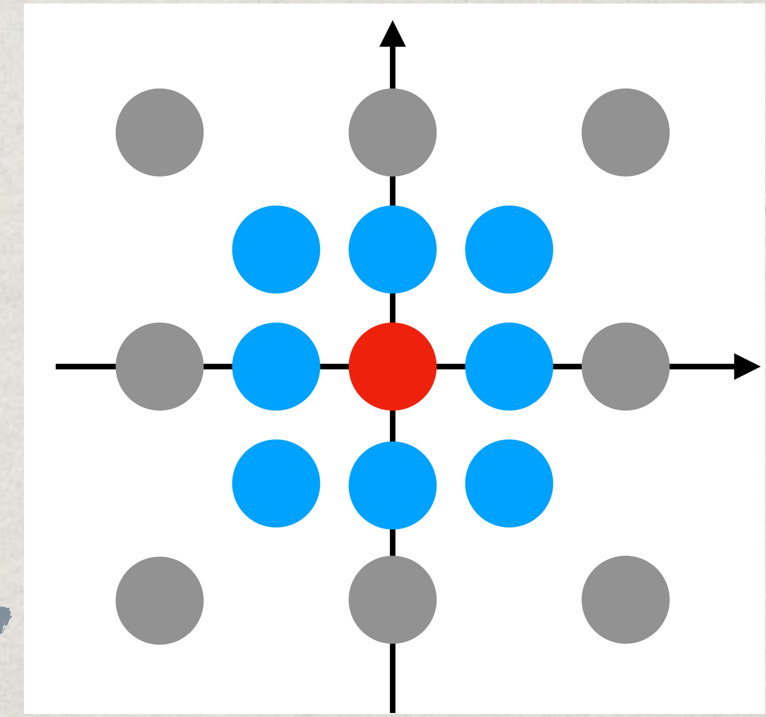
CASE 1: VERTEX RECONSTRUCTION

- Parameters to reconstruct: x, y, z, t_0
- Algorithm: $-\ln\mathcal{L} = -\sum \ln f_{res}(t_{i,res}) = -\sum \ln f_{res}(t_i - t_{i,tof} - t_0)$
  - t_i : *first hit time of ith fired PMT*
  - t_{tof} : *time of flight*
  - t_0 : *event start time*
  - f_{res} : *pdf of residual time*
- Scan 4D grid to minimize the NLL

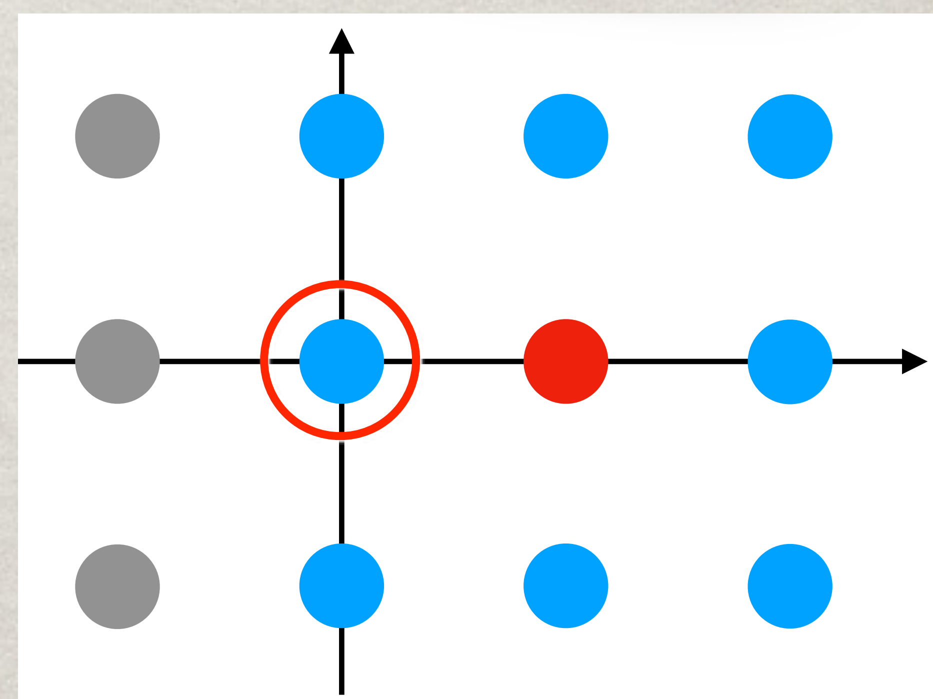


GRID SEARCH — 2D





```
if(Center is minimum){
   step /= 1/2
}
else{
   move to NEW center
}
```

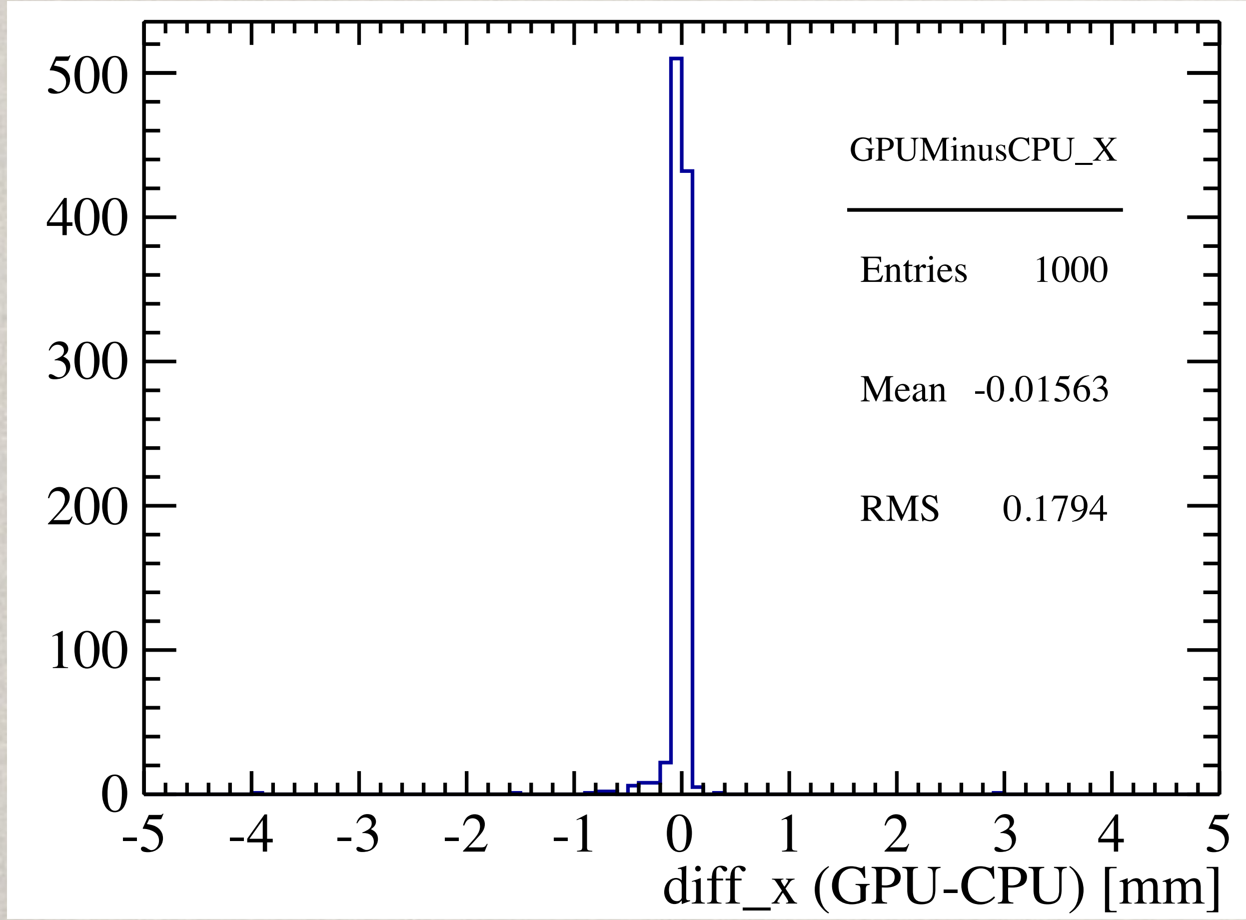


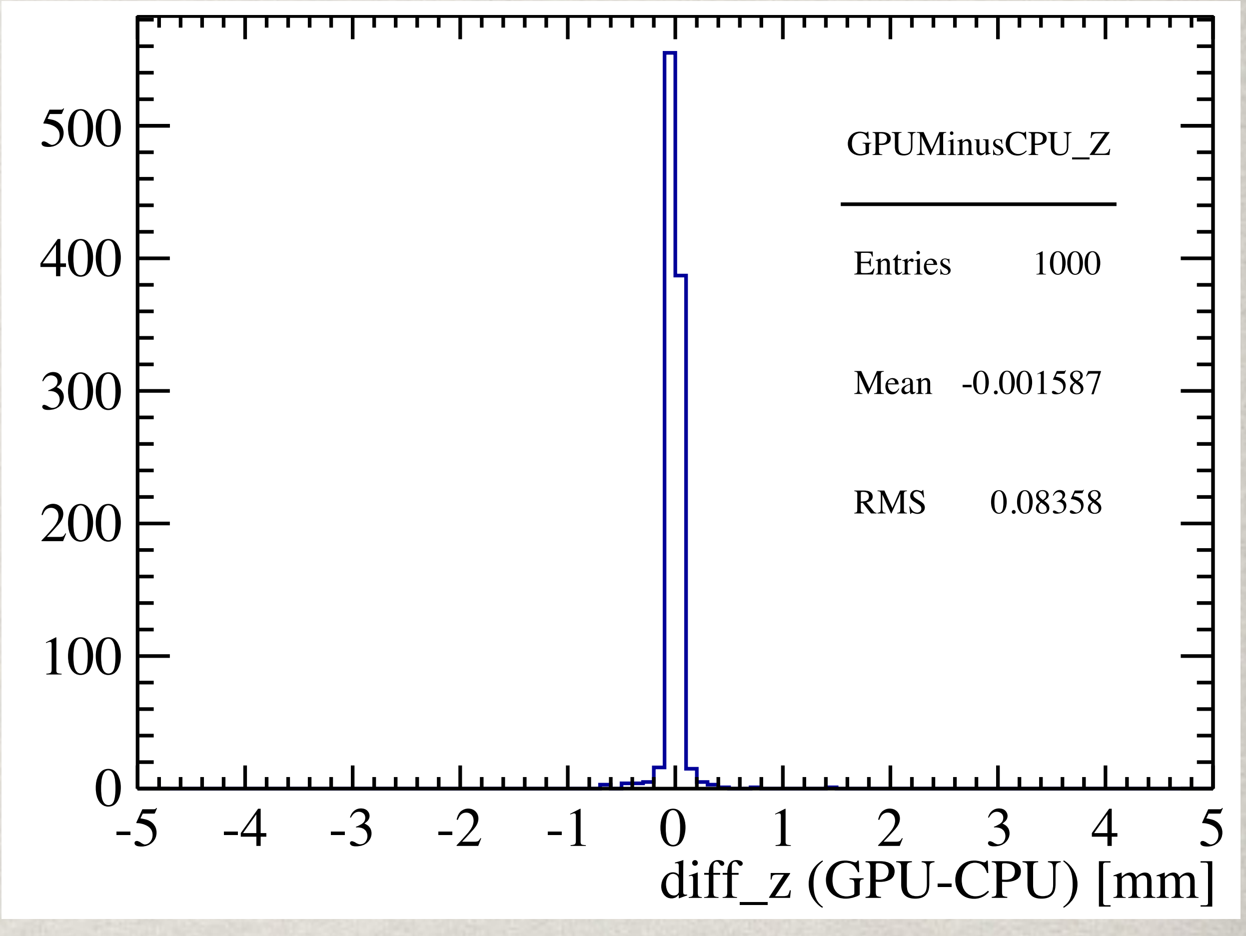
PARALLELIZATION ON GPU

```
for(t){
    for(x){
        for(y){
            for(z){
                for(ith PMT){
                    calc. NLLi
                }
    ...
}                   ON CPU
```

- 4D Grid Search
- Number of loops: x-dim*y-dim*z-dim*t-dim*n_fired_PMTs = 3*3*3*9*1200/MeV = $3*10^5$/MeV
- Parallelize the calculations on GPU

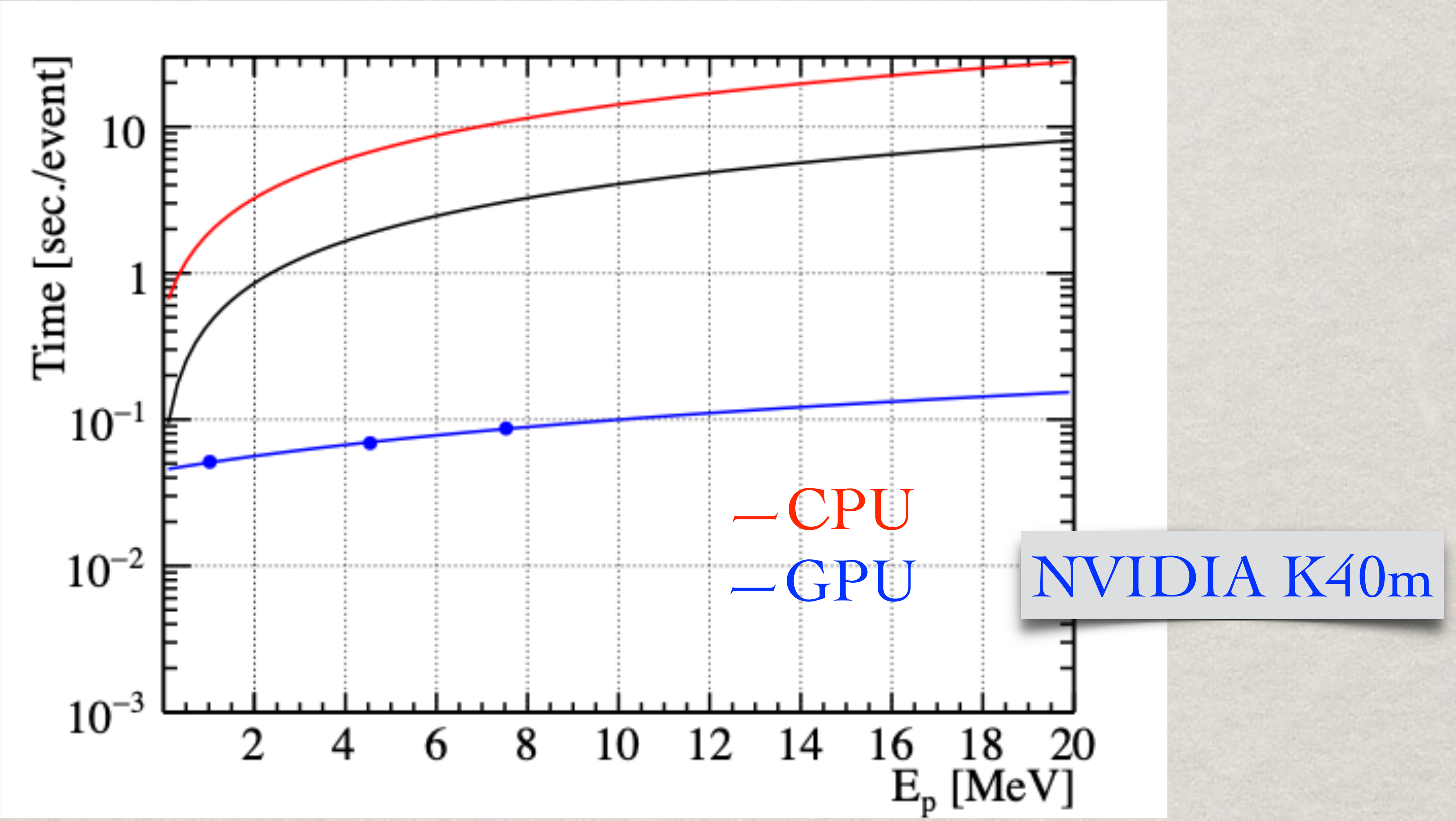
VALIDATION





- GPU Rec was able to reproduce the CPU Rec results
- Tiny difference (<0.5mm), negligible w.r.t. vertex resolution (60mm)

PERFORMANCE

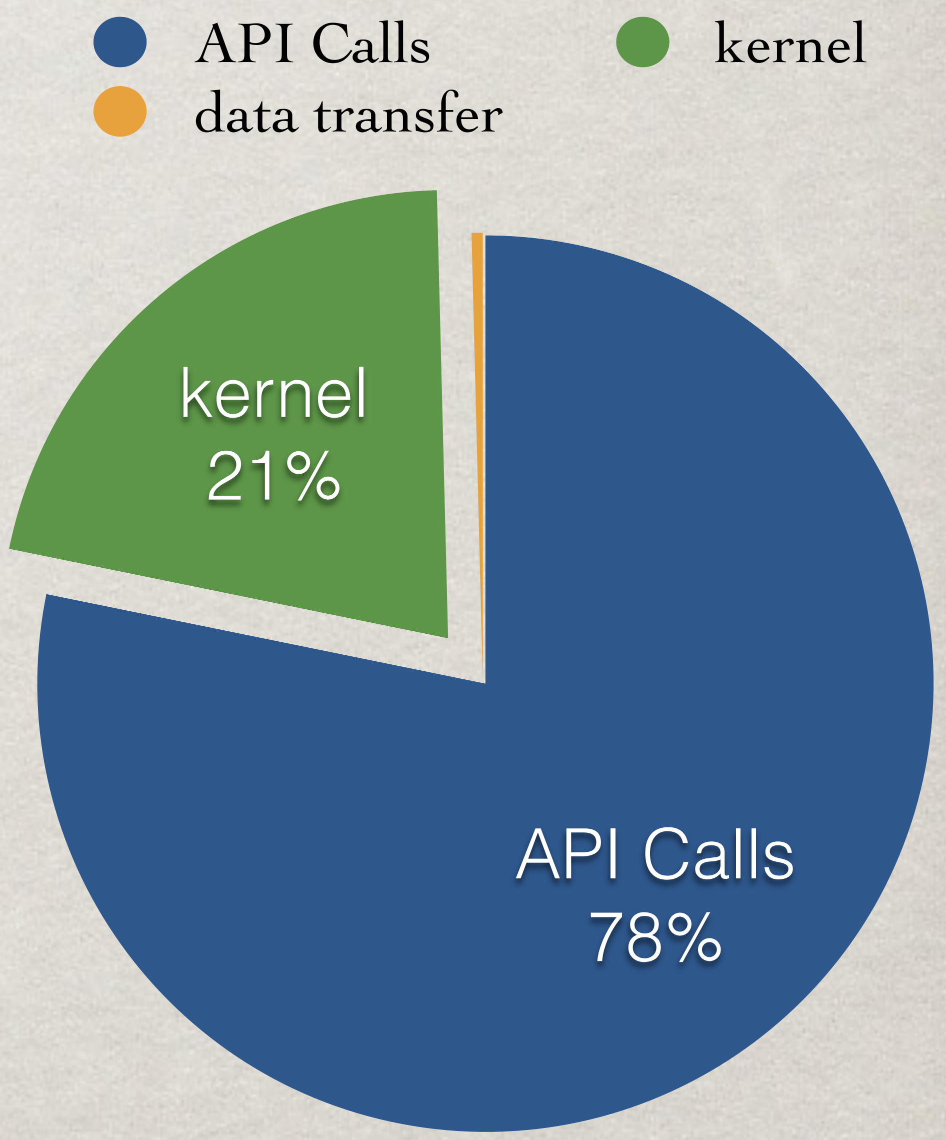


	CPU	GPU	Ration: CPU/GPU
Time@1MeV(s)	1.88	0.05	~40
Time@10MeV(s)	14.19	0.095	~150
Gradiant	1.37	0.005	—

DISCUSSION

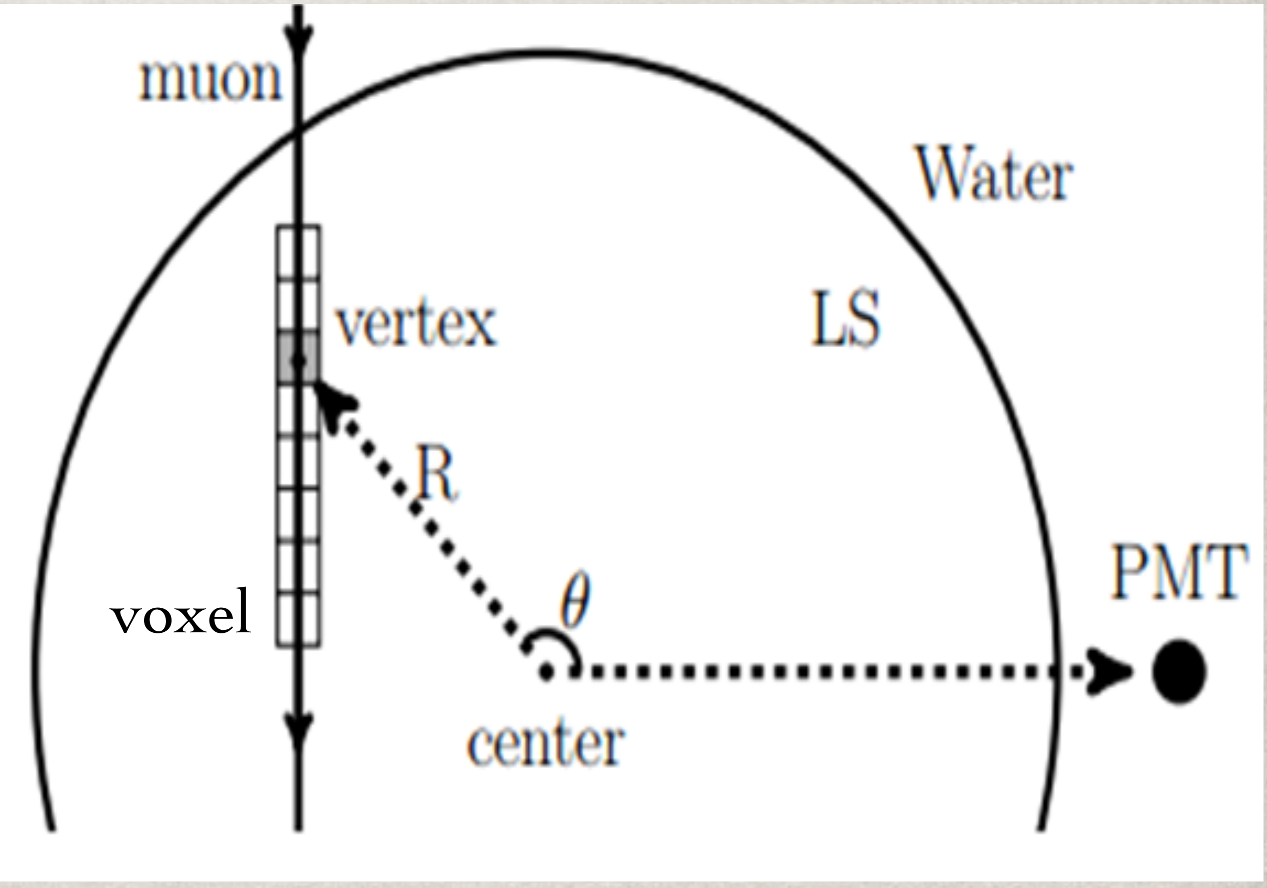
- Memory allocation and free, Synchronization etc… take up most of the time, room for future optimization
- Potential improvement with multiple GPUs
- Instead of Grid Search, divide the detector ROI to tiny units and parallelize with GPU(s)

NVIDIA K40m



CASE 2: MUON SIMULATION

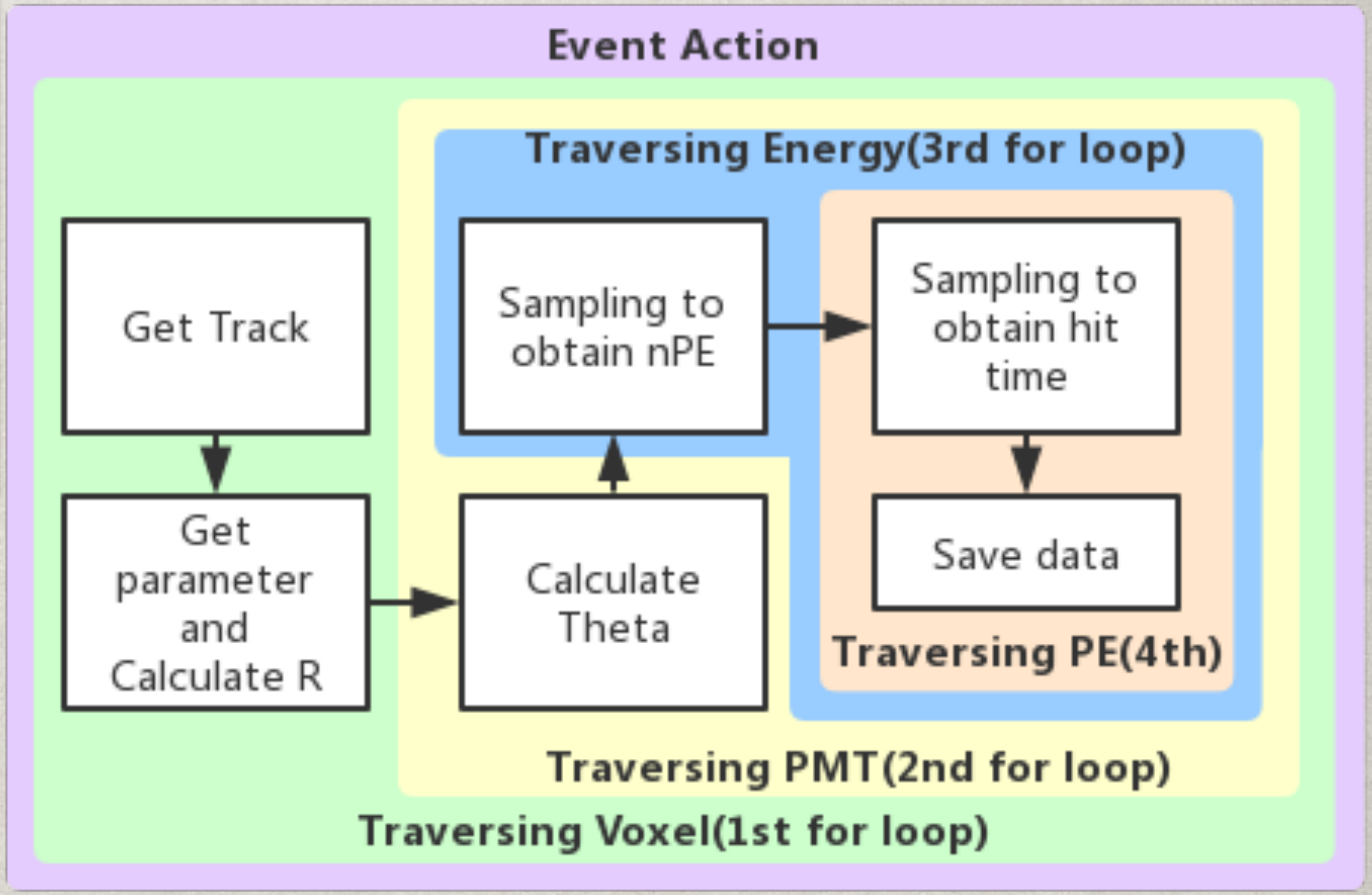
- Simulate the number of photons (nPE) and the corresponding hit time($\{t_i\}$) collected by each PMT for a traversing Muon
- Voxel: segments along the muon track
- For fixed (R, θ), sampling nPE and $\{t_i\}$ from templates



COMPUTATION FLOW

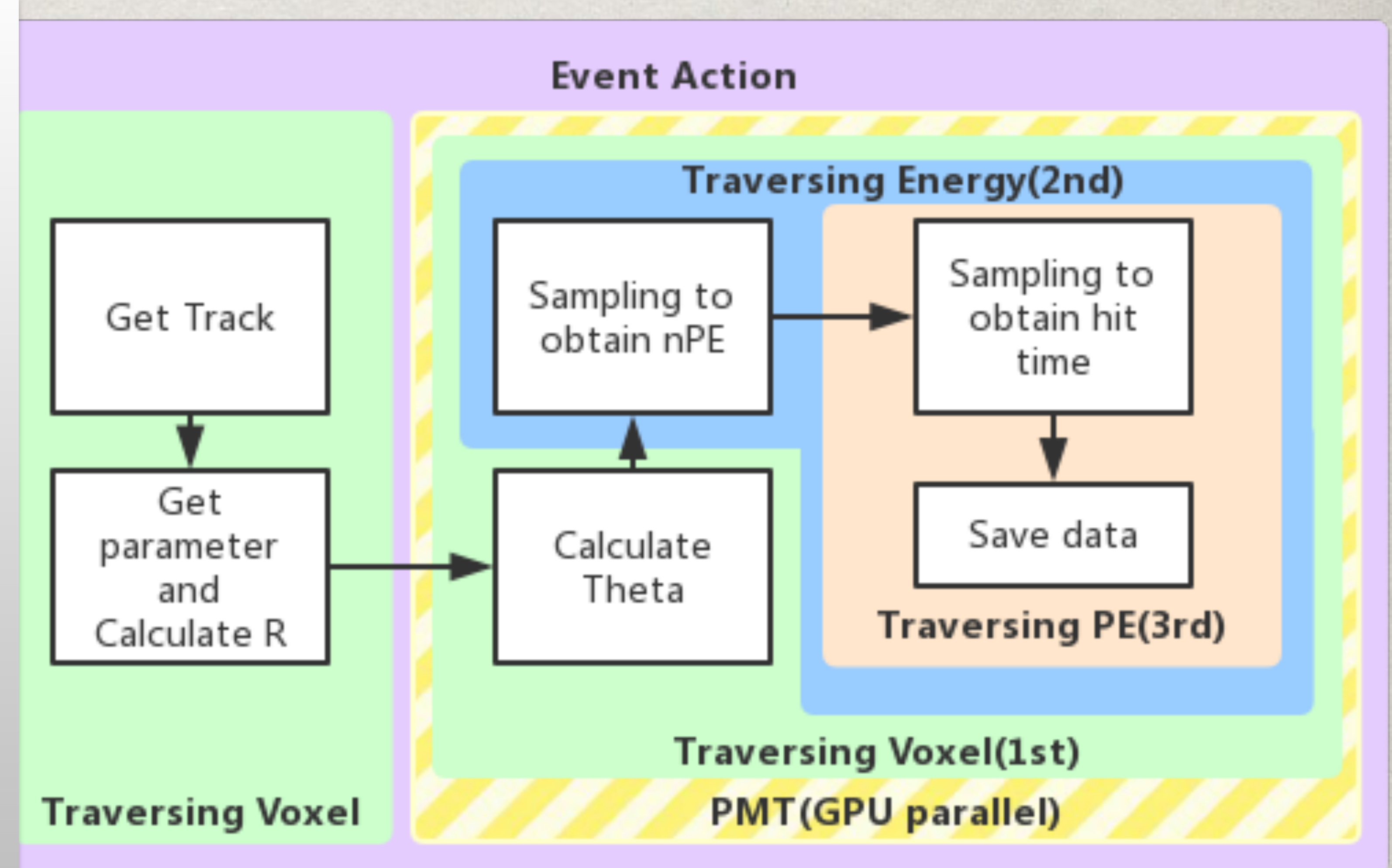
for(R){   // Voxel loop
    for(θ){   // PMT loop
        for(E){   // E loop
            for(nPE){
                sample t_i
            }
    …
}   ON CPU

~18,000 PMTs



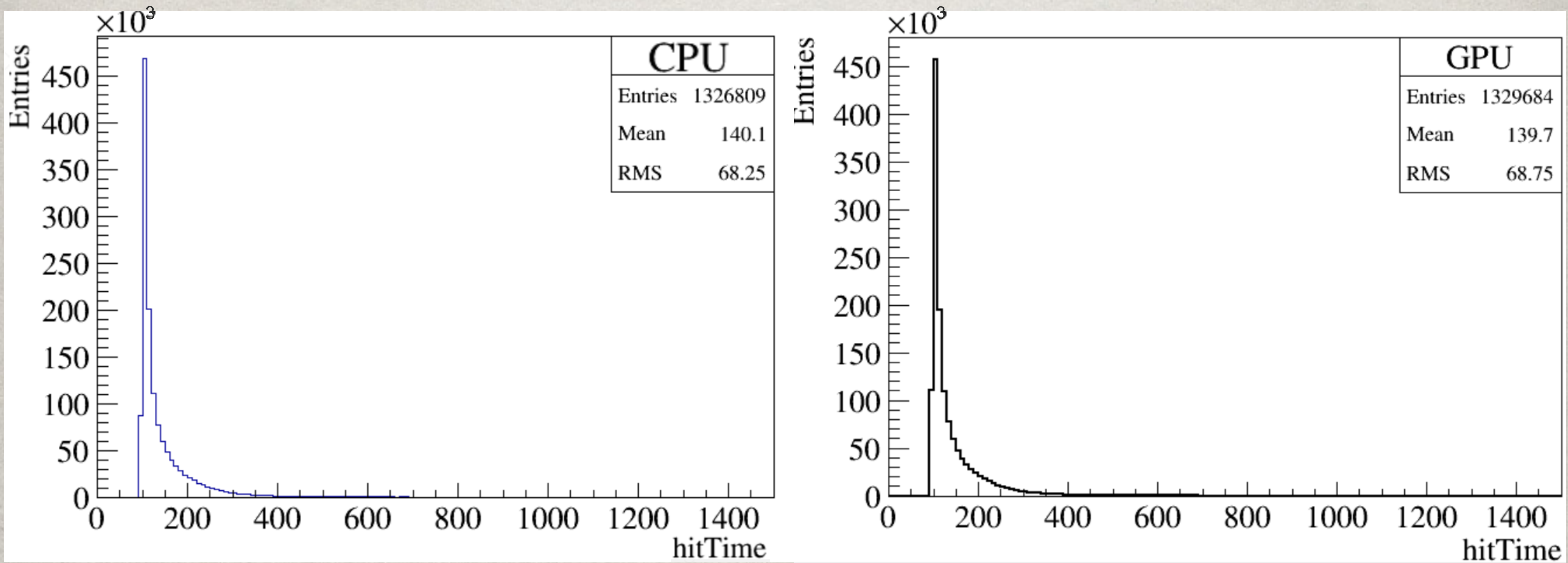
COMPUTATION FLOW

for(θ){ // PMT loop
    for(R){ // Voxel loop
        for(E){ // E loop
            for(nPE){
                sample t_i
            }
  ...
}



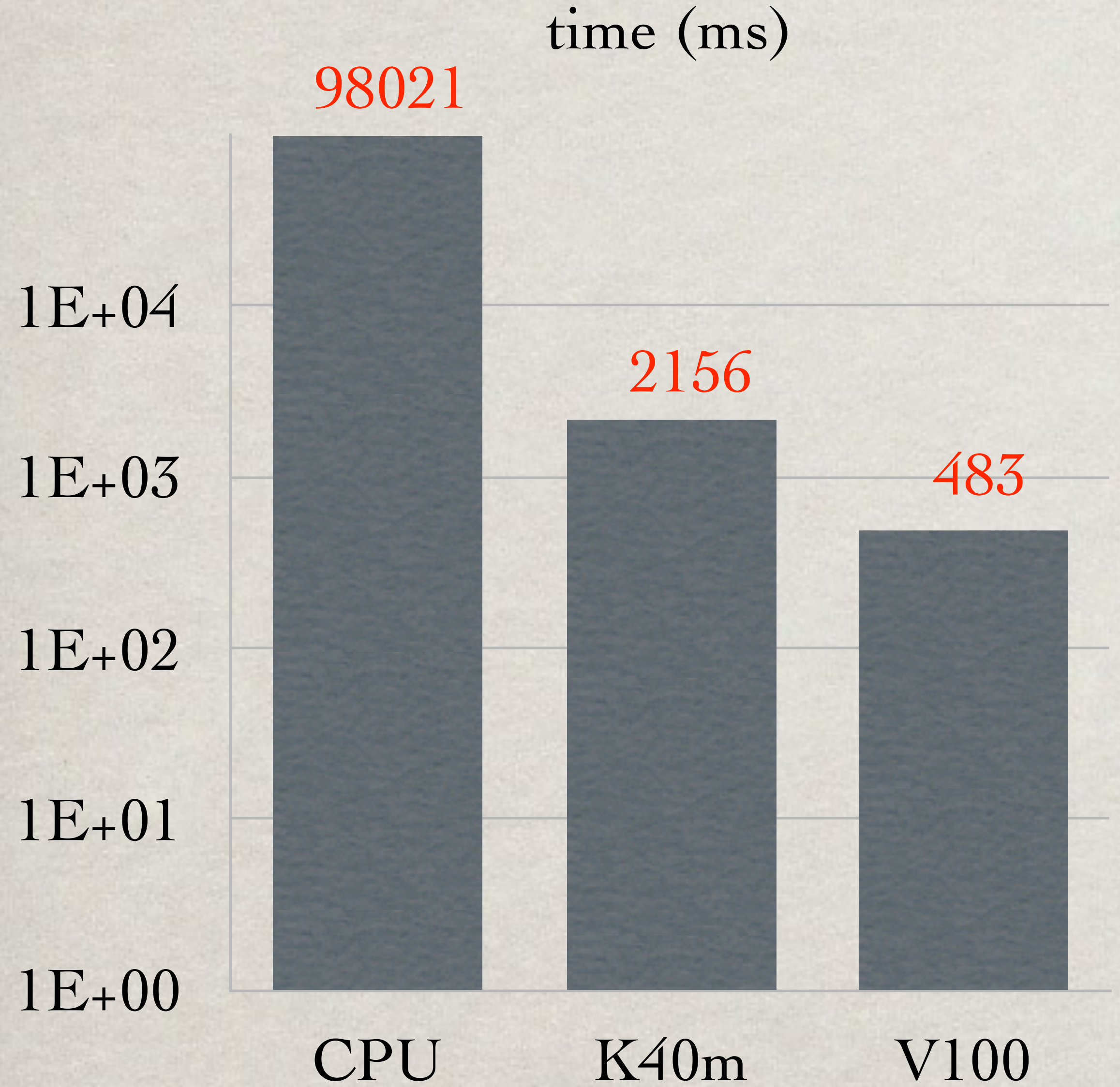
- Switch the Voxel loop and PMT loop levels
- Parallelize the PMT loop with GPU

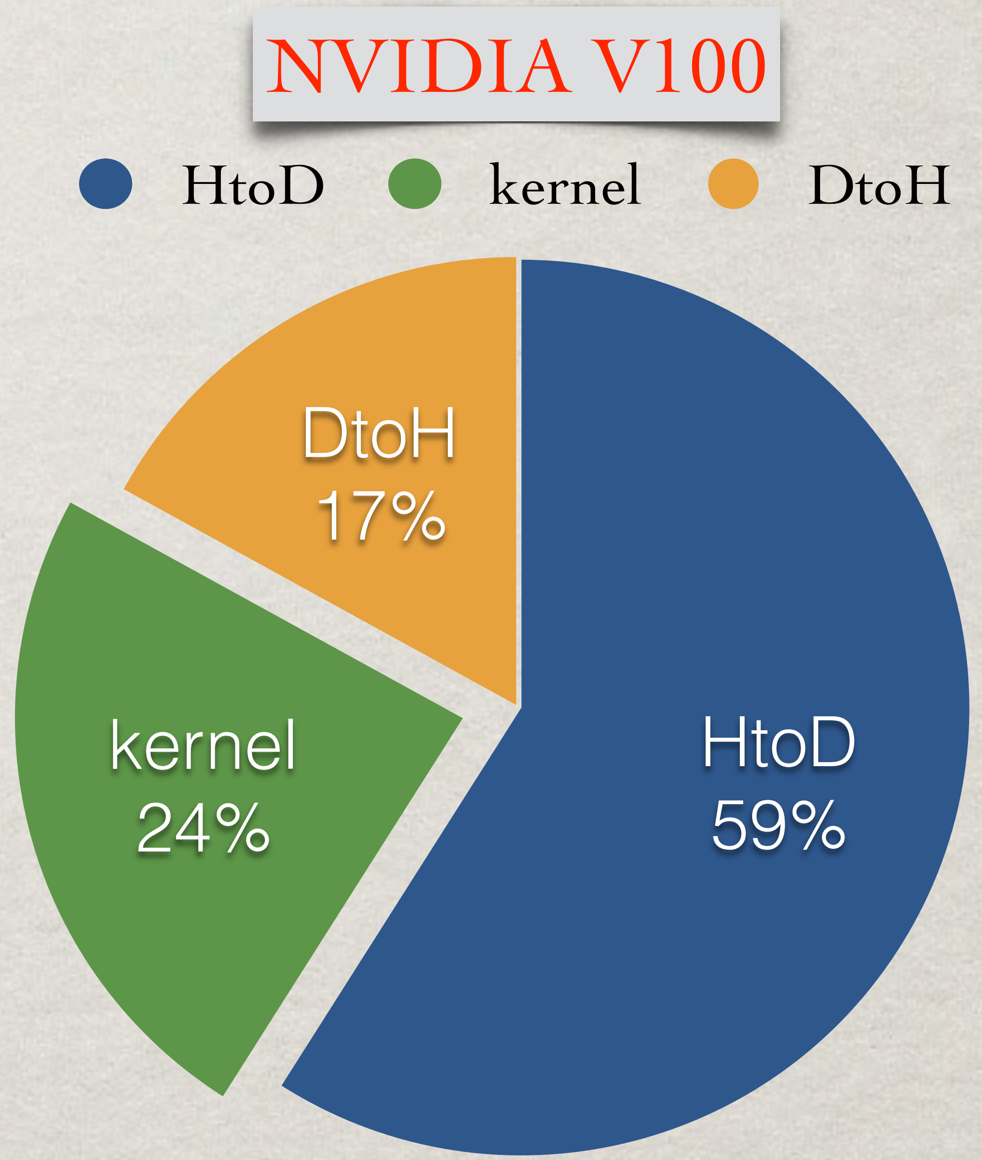
VALIDATION



- GPU Sim was able to reproduce the CPU Sim results
- Negligible difference

PERFORMANCE

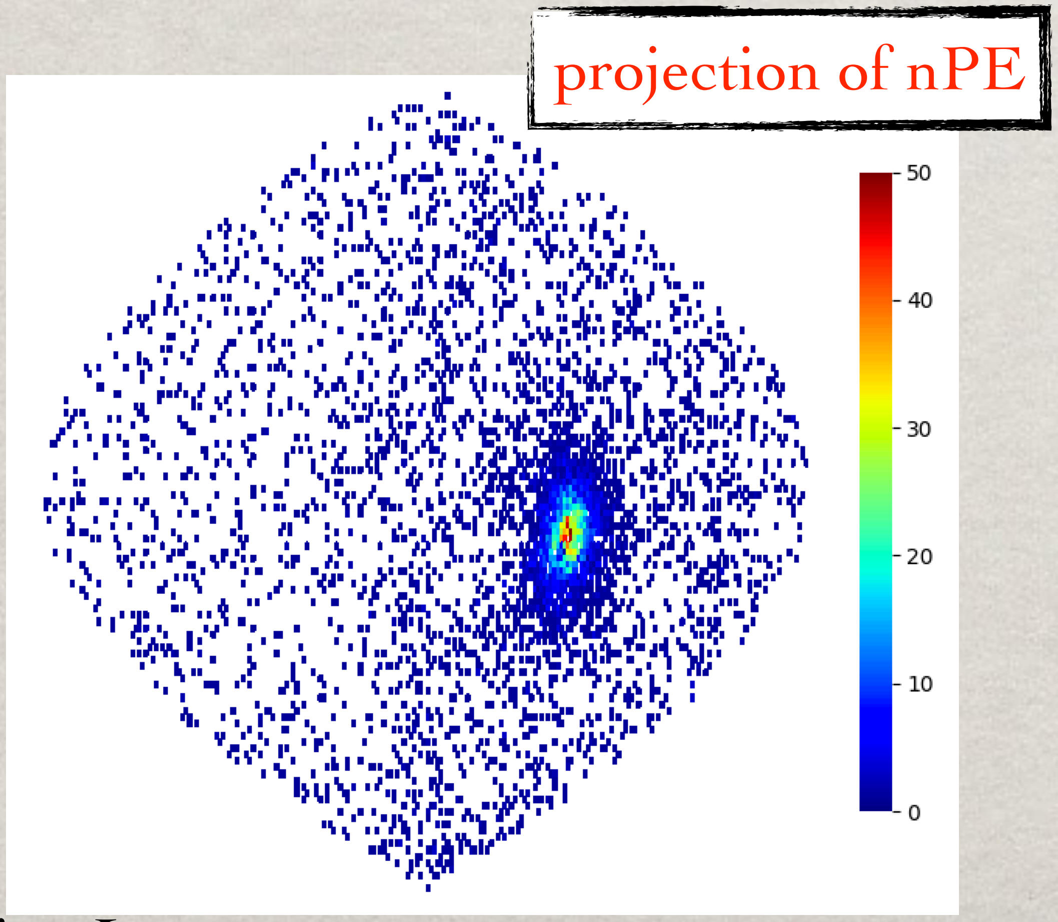


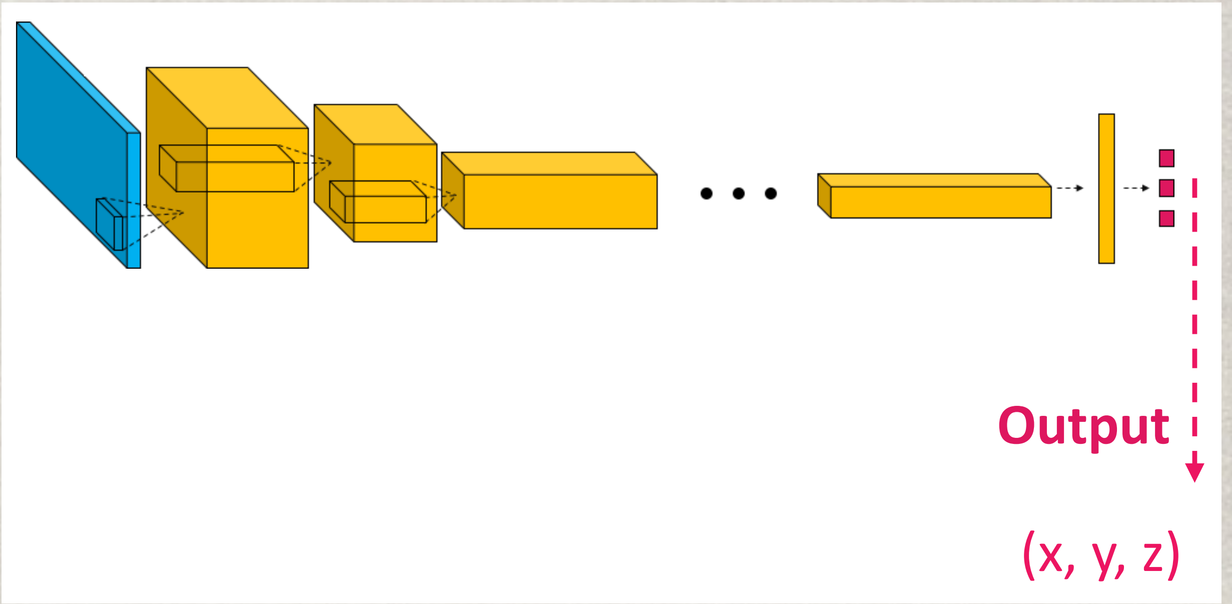


- O(10²) improvement with V100
- Future optimization: data transfer, more levels, multi-GPUs,

CASE 3: DEEP LEARNING

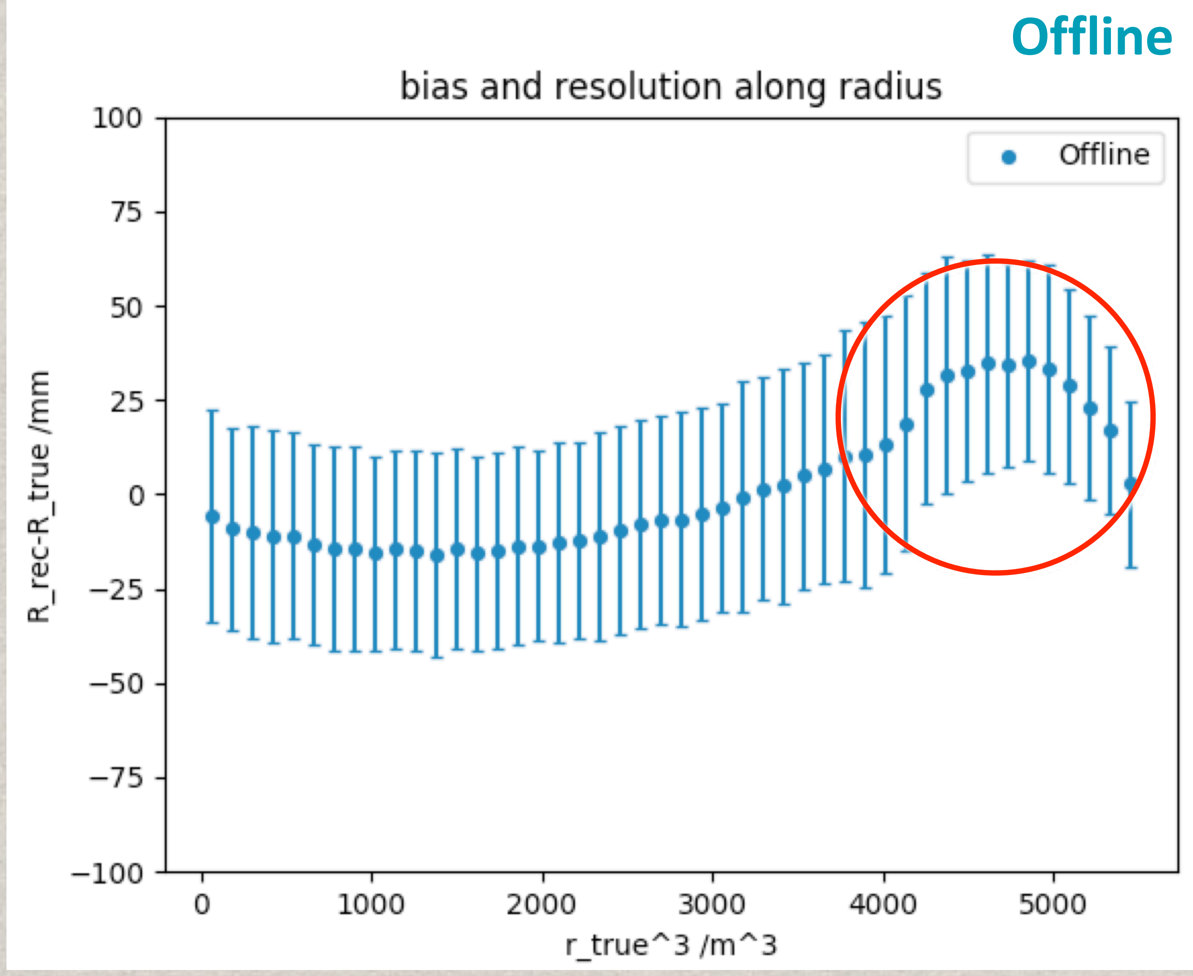
- GPU is widely used for DL
- Try Vertex Reconstruction with CNN in JUNO
- Input: hit time **{t_i}**, number of photoelectrons **{nPE_i}**
- Output: event vertex **(x, y, z)**

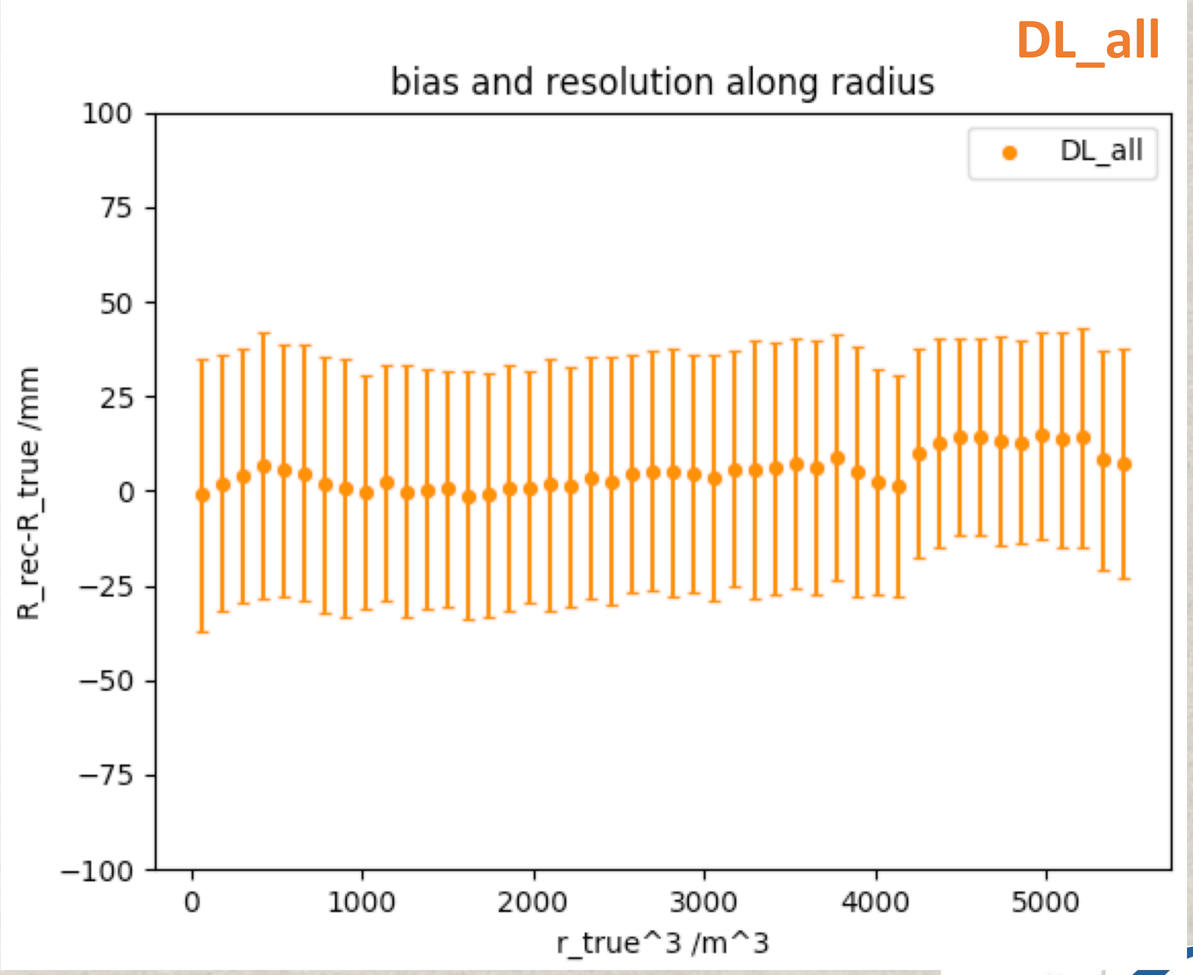




PERFORMANCE

- Smaller Bias for DL w.r.t. traditional method
- Better uniformity for DL w.r.t. traditional method

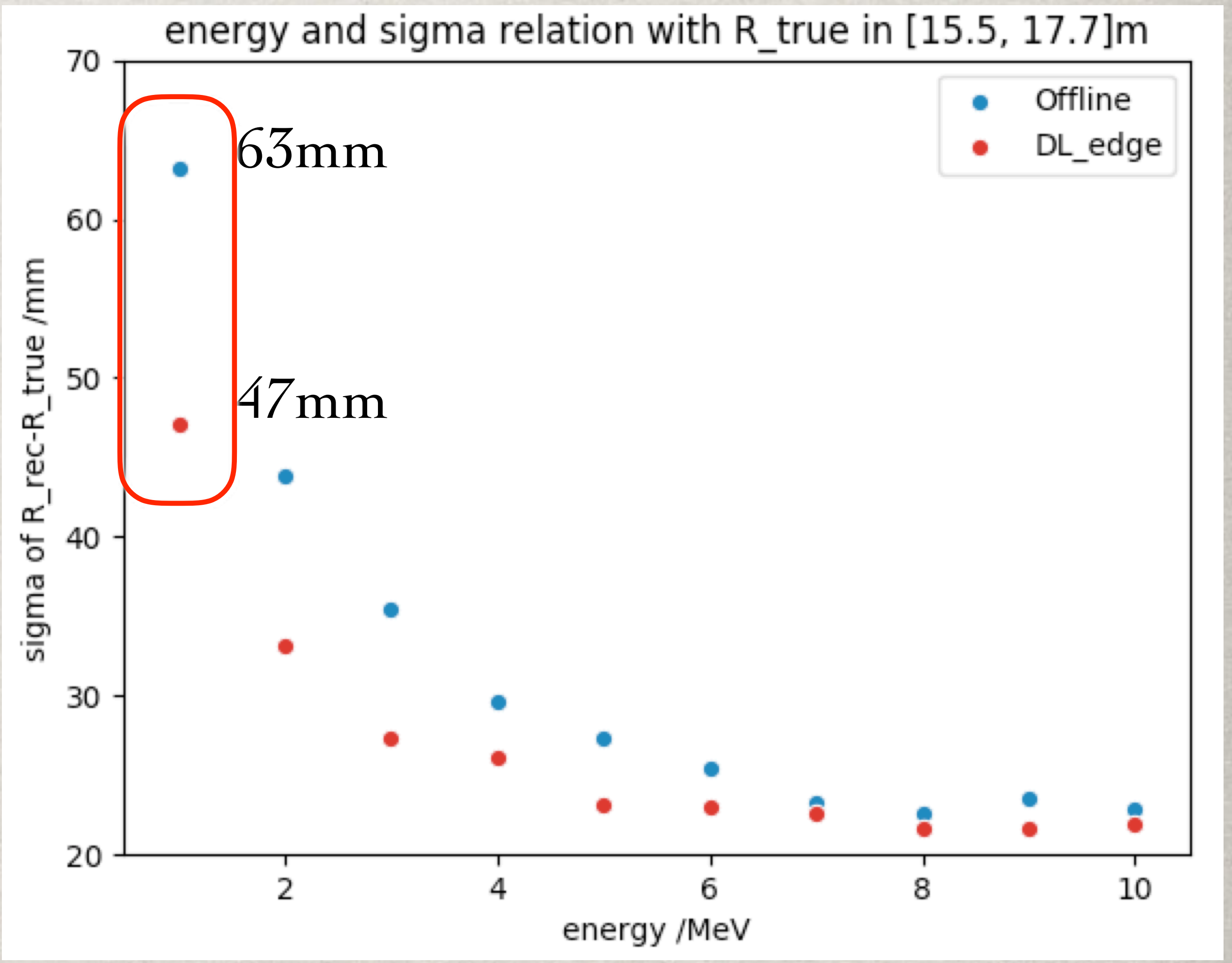




PERFORMANCE

- For events near the detector edge, total refraction complicates the optical model
- Better resolution for DL w.r.t. traditional method using a specialized model

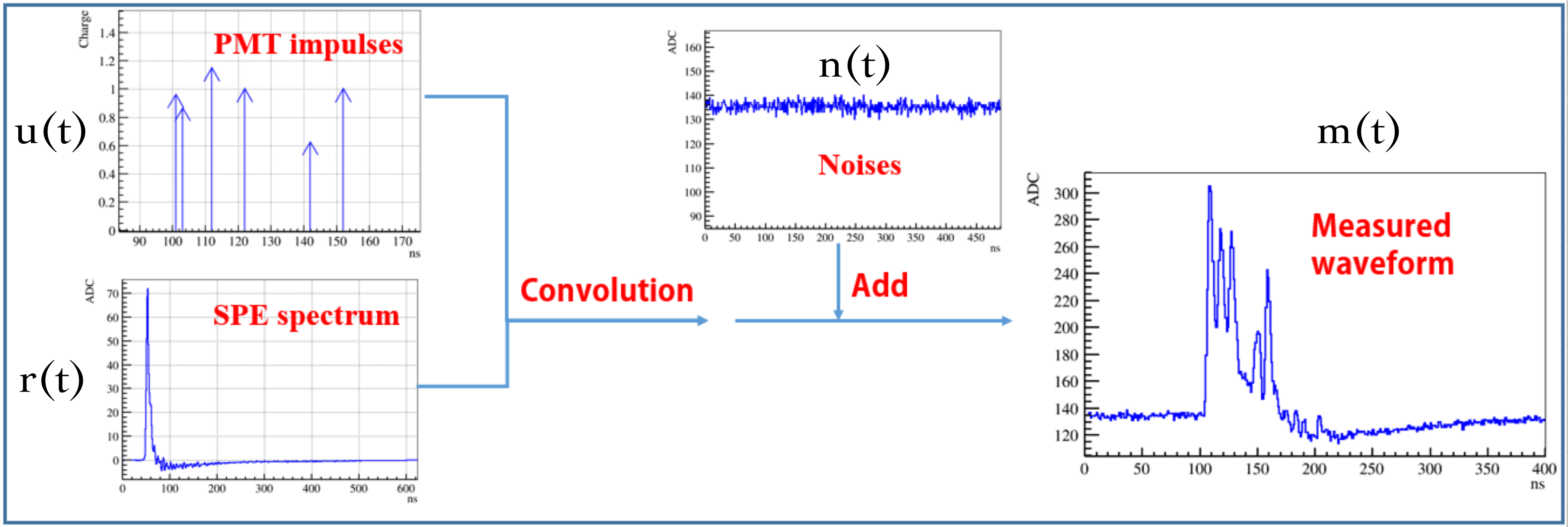
Resolution vs E



DISCUSSION FOR DL

- Pros:
  - fast speed, energy independent
  - avoid the complex optical model
- Cons:
  - rely heavily on GOOD Monte Carlo simulation
- Training samples
  - MC: large statistics, might be different w.r.t. real data
  - Calibration data: close to real data, limited stats.
- Possible solutions?

PMT WAVEFORM REC



- $m(t) = s(t) + n(t) = r(t) * u(t) + n(t)$
- We need to reconstruct $\{t_j\}$ and $\{charge_j\}$ or ideally $\{nPE_j\}$

DL FOR WAVEFORM REC?

- FADC raw waveform —> Time series
- We know roughly what the feature looks like —> sPE response template
- We want to know $\{t_j, Q_j(nPE_j)\}$ for all pulses
- We have PMT testing data —> real waveform
  - Issue: unsupervised, real labels unknown
- Analogies? Voice recognition? Suggestions?
- Try to answer simpler questions:
  - Q1: what is the first hit time?
  - Q 2: classify waveform to [0, 1, ≥2]PE three categories

SUMMARY

- JUNO has ~O(10^5) PMTs, perfectly suitable for utilizing GPU
- Showed a few simple applications of GPU in JUNO
  - Vertex reconstruction, Muon simulation, Deep Learning
  - Large room for further improvements
- Could be used in other aspects of JUNO
- Huge potential for experiments with lots of PMTs

BACKUP

TOOLS

- CUDA
- Thrust
- TensorFlow

	multi-processors	CUDA cores	ram(GB)
K40m	15	2880	12
V100	80	5120	32