Charged Particle Tracking with CUDA

Jacob Pomeranz
Governor's School for Science and Technology
2013 - 2014

Mentor Signature

David Lawrence, Ph.D.
Thomas Jefferson National Accelerator Facility
12000 Jefferson Ave, Newport News, VA 23606.

Abstract

The confinement mechanism of quantum chromodynamics (QCD) has not yet been explained. Lattice QCD calculations support the existence of exotic hybrid mesons that are not predicted by the quark model. The GlueX experiment at Thomas Jefferson National Laboratory will search for the existence of mesons in excited gluonic states. Because GPUs offer a low cost-method of performing many floating-point calculations and Jefferson Lab has a lattice QCD farm of GPUs available for research, a program was created using Compute Unified Device Architecture (CUDA) and C++ to reconstruct the tracks of charged particles in the GlueX experiment. The program was designed to use location and time data from the GlueX detector to simultaneously fit multiple tracks with multiple mass hypotheses to each particle using Nvidia GPUs. The track fitting uses the least squares method of curve fitting. Particles are identified by comparing the chi-squared per degree of freedom of the fit for each particle type. The program was compared to the existing CPU-based Kalman Filter code at Jefferson Lab with 830 single-track events generated by a particle gun simulation to determine whether parallel processing on GPU cores offers a faster running time while maintaining the same momentum resolution and efficiency. The GPU program reconstructed tracks at 32.5 events per second, while the CPU program ran at 80.0 events per second, indicating that the GPU program is not a viable alternative in its current state. Optimization and refactoring of the CUDA code could potentially improve performance, but the GPU program will not be used in the GlueX project due to limitations on the time available for development. Development of the GPU program revealed that the integration of CUDA with existing code requires major structural changes to maximize performance.

Introduction

The standard model of particle physics explains the basic interactions between known particles. The six quarks that make up all strongly interacting particles (hadrons) are the up, down, strange, charm, top, and bottom quarks. The theory of quantum chromodynamics (QCD) states that the strong force, one of the fundamental forces of nature, is mediated by gluons. Gluons are believed to act as force carriers upon quarks, antiquarks, and other gluons. When a pair of quarks is separated, gluons form a tube of constant energy density between them. As the distance between the quarks grows, the total energy of this gluonic flux-tube, or color force, increases. If the energy of this tube increases enough, the tube will split and a new set of quarks will be created. This concept, which is known as confinement in particle physics, prevents quarks from existing separately from each other. The mechanism behind this phenomenon is not fully understood (Dhamija 2010). According to the flux-tube model, the tube of energy that binds a meson (a quark and antiquark pair) together can be excited and caused to vibrate. These excited-state mesons would form a family of states analogous to ground state mesons (Underwood 2004). Mesons in this excited state are called hybrid mesons. Support for the existence of hybrid mesons has been increasing rapidly with recent experiments (Smith 2012). At specific levels of excitement, certain hybrid mesons can possess exotic quantum numbers (Papandreou 2007). These exotic quantum numbers are not possible under the quark model because they cannot be obtained from a simple quark and antiquark pair (Dhamija 2010). Therefore, the gluonic flux-tube must directly contribute to the quantum number of hybrid mesons in this state. Advances in computing power and algorithms have allowed scientists to make lattice QCD predictions of the masses and decays of exotic hybrid mesons (Papandreou 2007).

In the past, photon beams of a high enough energy, luminosity, and quality were not available to create hybrid mesons of sufficient quantity to detect. A 9 gigaelectronvolt (GeV) photon beam is needed to produce mesons in the predicted mass range of 2.0 GeV/c^2 (Dhamija 2010). Understanding how gluonic excitations and confinement occur will contribute to the existing body of knowledge on the nature of the strong force and how the basic particles of matter function. Previously, information gained from experimentation with the basic particles of matter has led to the creation of many new technologies in a variety of industries (Amaldi 2000). For example, several medical scanning technologies and cancer treatment methods have been created using particle beam techniques that were copied from particle accelerators. Accelerator technology has also been used in nuclear energy production, planetology, and ion implantation for semiconductor creation. The study of new types of matter helps to increase the knowledge of the scientific community regarding the origin of the universe.

The objective of the GlueX experiment at Thomas Jefferson National Accelerator Facility is to search for the existence of hybrid mesons, beginning with those that have exotic quantum numbers (Papandreou 2007). Mesons will be produced in this experiment by colliding high-energy photons with a liquid hydrogen target. Photons will be used instead of pions or kaons due to their supposed increased cross-section (probability) to create mesons with exotic numbers (Dhamija 2010). The 12 GeV upgrade to the electron beam accelerator will allow for a linearly-polarized photon beam of 8-9 GeV, which is believed to be the ideal energy range for the creation of exotic hybrid mesons

(Papandreou 2007). The photon beam will be created by passing the electron beam through a 20 μm diamond radiator in a process known as coherent bremsstrahlung (Lawrence 2009). The radiator will be tunable in order to collimate the beam to a desired polarization. The photon beam will be "tagged" by a device called a photon tagger, which identifies the energy and creation time of each photon in the photon beam (Underwood 2004). The photon tagger operates by directing electrons exiting the diamond regulator into fiber-optic scintillator arrays. The scintillators feed into silicon photomultipliers that allow for photon detection. These data are then used to tag each photon in the beam that is created through coherent bremsstrahlung. It is expected that numerous different hadrons, including many mesons, will be produced from the collision of the photon and the protons in the liquid hydrogen target. Any exotic hybrid mesons that are produced will decay within roughly 10^{-20} seconds. Because of this rapid decay, the resulting daughter particles must be identified and tracked.

The GlueX detector will feature two charged particle tracking systems, the Forward Drift Chambers and the Central Drift Chamber (Lawrence 2009). The Forward Drift Chambers will consist of four packages, each of which will contain six chambers. Each chamber will have 96 wires in addition to 2 cathode planes that will each contain 216 copper strips. The cathode planes allow for the detection of the avalanche location along the sense wires, or the point at which there are enough electrons present to register a charge. The avalanche locations from the cathode planes will be represented by three-dimensional points close to the track. The Central Drift Chamber will use 3522 straw tube wire sensors arranged in 28 layers to track passing particles. Particle types will be identified using time-of-flight information from the scintillator walls in the GlueX detector.

The program that is currently used for tracking charged particles is written in the C++ programming language and operates on the CPU. It is based on the Kalman Filter fitting technique and is highly optimized. The Kalman Filter technique achieves the same fits as the least squares method, but it is better at dealing with multiple scattering (Billoir 1984). This feature allows the Kalman Filter technique to produce a more accurate covariance matrix, which is used for error analysis later on. The program functions by iterating though possible curve fits for the wire hits from the detector in order to find the most accurate curve. However, GPUs have recently become much more common in the scientific computing community due to their low cost and high computing power (Ranjbar et al. 2010). Additionally, because Jefferson Lab frequently upgrades its supercomputers, many GPUs are available for use at very little cost. These GPUs could provide a low-cost alternative that may perform better than CPUs for charged particle tracking.

Unlike CPUs, GPUs are specifically designed to execute floating-point operations on large sets of data. Furthermore, GPUs have many more computing cores than CPUs, allowing them to perform many operations in parallel. Past studies have shown GPUs to be very effective for high energy physics event reconstruction (Al-Turany, Uhlig, and Karabowicz 2010). While GPU cores are slower than CPU cores individually, they can run simultaneously, allowing them to compute multiple data points at the same time.

Current tracking code must fit tracks for several different particle types to each track candidate. The new CUDA (Compute Unified Device Architecture) program uses the parallel processing ability of the GPU to fit multiple tracks for different particle types to a set of data points at the same time. Because these threads of code all run in parallel, divergences between the threads cause all of the threads to wait for the slowest threads to catch up (Ranjbar et al. 2010). Therefore, it is essential that all parallel threads remain synchronized in order to avoid bottlenecks in the routines being performed. Nvidia CUDA is used to run routines on the GPU. CUDA offers several benefits over other frameworks. It functions very well alongside the C and C++ programming languages, so the CUDA program was able to reuse code from the current particle tracking program (Al-Turany, Uhlig, and Karabowicz 2010). C and C++ were used for this program for their low-level memory management and high performance. In addition, most of the GPUs that Jefferson Lab uses for scientific computing are manufactured by Nvidia, so they perform best with CUDA as a result of their physical design. In order to take advantage of the GPU power available at Jefferson Lab, a program was created to simultaneously fit tracks of different particle types to each data set.

Methods and Materials

Dr. David Lawrence initially coded the program to the point where it swam potential tracks through the inhomogeneous magnetic field produced by the GlueX detector magnet. The magnetic field is stored as a texture map on the device memory. The program uses a constant value to calculate energy loss from particles traveling through the detector materials. The wire detector information is also stored on the device memory. Functions were created in order to read in position and time data from hits on the wire detectors. CUDA kernels were created to calculate the chi-squared per degree of freedom for a given set of initial track parameters. The tracking code takes into account five parameters, including particle charge divided by momentum, x-coordinate particle position, y-coordinate particle position, the x-direction the particle is traveling in, and the y-direction the particle is traveling in. Momentum is estimated for track candidates assuming a helical particle trajectory caused by the two-tesla solenoidal magnet the detector is built around (Lawrence 2009). Time is represented by an independent variable z. The least squares method is run on the CPU to find the best fit using the chi-squared per degree of freedom value returned by the CUDA kernels.

To be deemed working, the program must correctly calculate the least squares curve from a given set of wire detector hits, which includes the time and location at which each wire detector is triggered. For each data set, the program must return the difference between the simulated trajectory of the particle and the theoretical trajectory based on the wire sensor hits. This value takes into account the uncertainty of the calculations as per the least squares method.

The program was coded in C++ and CUDA. The free GNU Compiler Collection (GCC) and Nvidia CUDA Compiler Driver (NVCC) compilers were used to compile C and CUDA code, respectively. Nvidia Nsight, a free CUDA debugger, was used to debug the code during development. For data analysis and graphing of momentum resolution and other parameters, ROOT was used. ROOT is a free high energy nuclear physics data analysis program developed by the European Organization for Nuclear Research (CERN).

The program fits particles to curves for positively- and negatively-charged pions, positively- and negatively-charged kaons, protons, positrons, and electrons. The program can reconstruct the tracks of particles with momentums between 250 MeV/c and 4

GeV/c. Particles with higher momentum cannot accurately be identified using the wire sensors in the drift chambers due to their straighter, more constant trajectories. Once a set of curves has been generated for a given set of data, the chi-squared per degree of freedom values for each curve are used to generate statistical weights. These weights are used by other programs to pick the most likely curve candidate and identify the particle type.

The program was tested by calculating fit curves and identifying particle types for 830 events using a particle gun to simulate the behavior of charged particles inside of the GlueX detector. These events were selected by generating 1000 events and choosing only those with a single track candidate.

The program was tested on a machine with a Tesla K20M GPU with four gigabytes of memory and a 705 megahertz core clock. The machine had an Intel® E5-2650 Xeon® CPU clocked at 2.00 gigahertz. The program was tested against the existing CPU-based Kalman Filter charged particle tracking code that Jefferson Lab is using to determine if there a difference between the running times of these two programs. Compared to the least squares method, the Kalman Filter method offers a better estimate of uncertainty or error (Billoir 1984). Because of the advantages the Kalman Filter methods offers for charged particle tracking, Jefferson Labs has focused on optimizing its Kalman Filter code for its current particle tracking experiments instead of the older least squares code. For this reason, the GPU program, which uses the least squares method, was compared to the Kalman Filter code instead of CPU-based least squares code. The GPU program uses the least squares method because the GPU architecture is better suited to running the least squares method than the Kalman Filter technique. To compare the running times of these programs in milliseconds, the "itimer" facility in a UNIX-based operating system was used. The itimer facility provides time measurement with ten milliseconds of precision.

To determine if the program satisfies the precision requirements, the momentum resolution from the fitting of the events created by the particle gun was evaluated to ensure that the program achieves a momentum resolution equivalent to that of the existing Kalman Filter CPU code that Jefferson Lab uses.

Results

The program currently completes all stages of the tracking process. Nvidia Visual Profiler (NVVP) showed that ninety-five percent of the running time of the program is spent in the kernel that swims the track through the magnetic field, showing that this stage of the tracking process is bottlenecking the overall performance of the program. A timeline showing this bottleneck in NVVP is displayed in Figure 1.

Compared to the CPU code, the program ran much more slowly. Using 830 events generated from the particle gun with one track per event, the old CPU code fitted 80.0 events per second, while the GPU code fitted at 32.5 events per second. Figure 2 demonstrates how the event reconstruction rate was generated for the CPU. This calculation was needed since the old CPU code generated the track candidates in addition to reconstructing their tracks, while the CUDA program did not have to generate the track candidates.

Macros were created for the ROOT framework to graph the momentum, phi, and theta resolutions. The resolutions are defined as the width of a Gaussian curve and are calculated in the same way as standard deviation. The track candidates and fits generated by the CUDA program were compared to one another using the “thrown,” or true, values from the particle gun simulation that generated the tracks. As shown in Figure 3, the theta angle error and theta resolution of the fit tracks showed improvement over the track candidates prior to fitting. The graphs of the momentum and phi resolutions, shown in Figures 4 and 5, both showed worsened resolutions after the fitting process when compared to the candidates. However, the mean error for both the momentum and phi parameters decreased, suggesting that the GPU track fitter is functioning properly. A likely reason for the decreased resolution from the fitting process is that the track candidates were generated using information from the cathode planes alongside the wire arrays found in the Forward Drift Chamber. The GPU program only uses wire-based tracking and does not utilize information from these cathode planes due to the additional complications these calculations would add. Therefore, it was expected that the resolution of the GPU track fitter would be worse than that of the candidates.

ROOT was also used to create graphs of the efficiency for the GPU program and the old CPU program. The efficiency of a program was defined to be the number of tracks that were successfully fitted divided by the number of track candidates that were input into the fitter. The efficiency was calculated from the momentum resolution graph for each program. A three-σ cut, with σ equal to the respective momentum resolution of each program, was used to eliminate outlying fit tracks. The number of remaining fit tracks was divided by the number of original track candidates to calculate the efficiencies of the programs. The efficiencies of both programs were between eighty-eight and eighty-nine percent. This result shows that the GPU program had an efficiency equivalent to that of the CPU program.

Discussion and Conclusions

The results shown in Figures 3 through 5 suggest that the code is reconstructing the tracks of the test events successfully. In terms of reconstructing tracks faster than the CPU program, the GPU program failed. This result was expected since the CPU code was developed over a much longer period of time and has undergone a great deal of optimization. Due to time constraints, very little optimization was performed on the GPU program. The results show that the performance gains from the parallelism in the uncertainty calculations are not enough to offset the slower GPU core speed during the swimming process without major refactoring of the track swimming code to improve parallelism.

Further optimization with CUDA memory could significantly increase the speed of the tracking process. Using memory coalescing techniques for memory transfers could significantly decrease the time spent copying data to and from CUDA kernels. These techniques would add to the overall complexity of the code, but they could especially reduce the time spent copying data between host and device memory. Similar studies point to breaking up operations into smaller parts to increase parallelism (Al-Turany, Uhlig, and Karabowicz 2010). The GPU program currently uses many CUDA kernels to complete the track fitting process, but some kernels could be broken up further to possibly improve performance. Using different types of device memory was difficult with the existing track fitting framework due to limitations on the size of local device memory. Refactoring the track swimming code to employ memory types such as shared and local memory could offer major speed gains for the entire program.

Because the GPU program does not currently perform as fast as its CPU counterpart, it is not a viable option for use in the GlueX project. In addition, GPUs have some inherent disadvantages, such as higher power consumption. Also, CUDA code is not standardized for all models of GPUs due to differing levels of computing ability between GPU models. CPUs can generally perform identical tasks using the same code, so they have several innate advantages over GPUs.

Acknowledgements

I would like to thank Dr. Lawrence for all of his guidance throughout the course of this project. I would also like to thank the Thomas Jefferson National Accelerator Facility for allowing me to use the Nvidia graphics cards and computers necessary for this project.

Literature Cited

Al-Turany, M., F. Uhlig, and R. Karabowicz. 2010. GPU's for Event Reconstruction in the FairRoot Framework. *Journal of Physics: Conference Series* 219 (4): 1-8.

Amaldi, Ugo. 2000. The Importance of Particle Accelerators. *Proceedings of EPAC 2000, Vienna, Austria* 7: 3-7.

Billoir, Pierre. 1984. Track Fitting With Multiple Scattering: A New Method. *Nuclear Instruments and Methods In Physics Research* 225 (2): 352-366.

Dhamija, S. 2010. QCD Confinement and the GlueX Experiment at Jefferson Lab. *Chinese Physics C* 34 (9): 1390-1392.

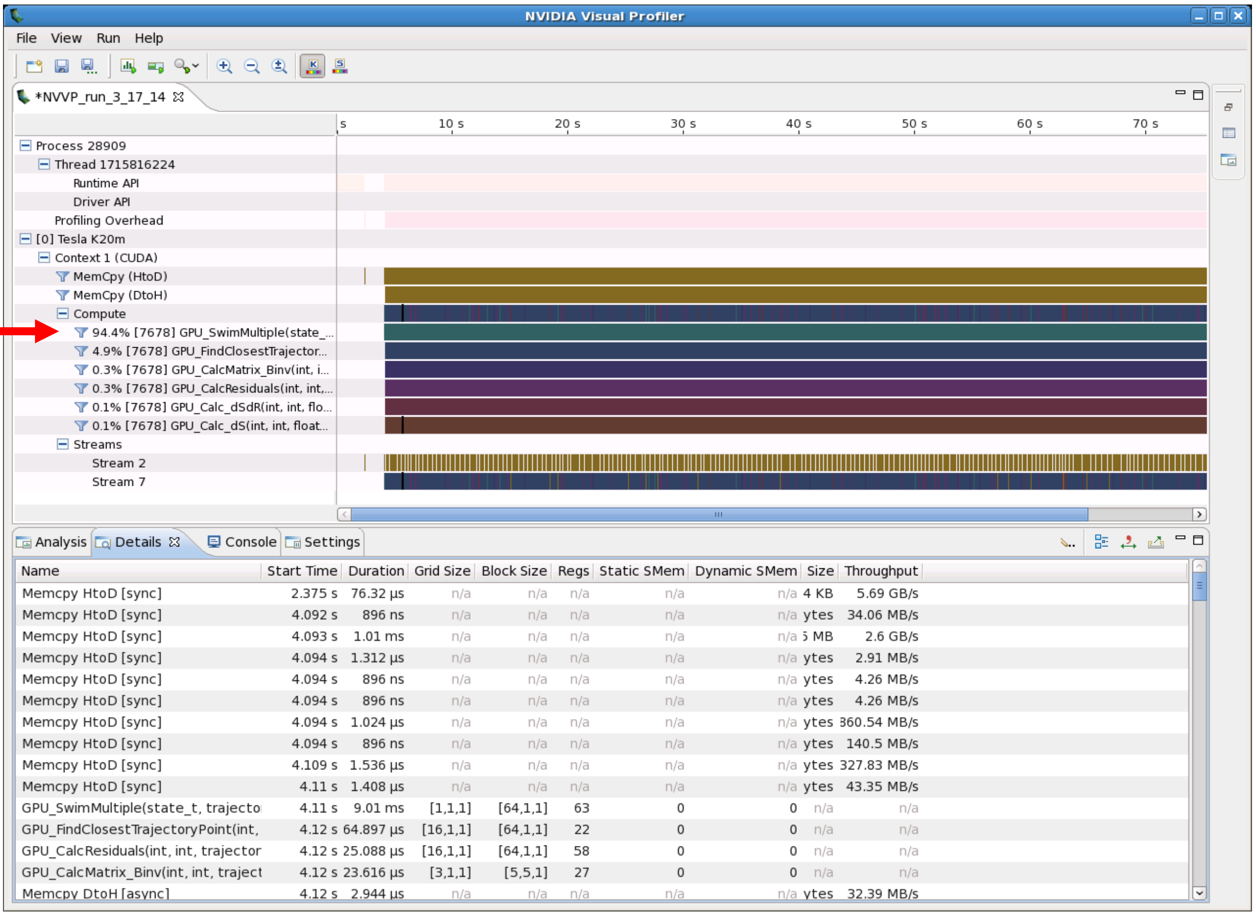
Lawrence, David. 2009. The GlueX Detector. *AIP Conference Proceedings* 1182: 811-815.

Papandreou, Zisis. 2007. The GlueX Project at Jefferson Lab. *Frascati Physics Series* 46: 000-000.

Ranjbar, V., I. Pogorelov, P. Messmer, and K. Amyx. 2010. Accelerated Particle Tracking Using GPULib. *Proceedings of HB2010, Morschach, Switzerland* 46: 286-289.

Smith, Elton S. 2012. "GlueX: Photoproduction of Hybrid Mesons." *EPJ Web of Conferences* 37, December 6. http://dx.doi.org/10.1051/epjconf/20123701026/ (accessed October 17, 2013).

Underwood, Mitchell. 2004. Design of Electronics for a High-Energy Photon Tagger for the GlueX Experiment. Honors Scholar Theses, University of Connecticut. In University of Connecticut Digital Commons Honors Scholars Theses Database [database on-line]; available from http://digitalcommons.uconn.edu/srhonors_theses/ (publication number 163 accessed October 16, 2013).



Name	Start Time	Duration	Grid Size	Block Size	Regs	Static SMem	Dynamic SMem	Size	Throughput
Memcpy HtoD [sync]	2.375 s	76.32 µs	n/a	n/a	n/a	n/a	n/a	4 KB	5.69 GB/s
Memcpy HtoD [sync]	4.092 s	896 ns	n/a	n/a	n/a	n/a	n/a	ytes	34.06 MB/s
Memcpy HtoD [sync]	4.093 s	1.01 ms	n/a	n/a	n/a	n/a	n/a	5 MB	2.6 GB/s
Memcpy HtoD [sync]	4.094 s	1.312 µs	n/a	n/a	n/a	n/a	n/a	ytes	2.91 MB/s
Memcpy HtoD [sync]	4.094 s	896 ns	n/a	n/a	n/a	n/a	n/a	ytes	4.26 MB/s
Memcpy HtoD [sync]	4.094 s	896 ns	n/a	n/a	n/a	n/a	n/a	ytes	4.26 MB/s
Memcpy HtoD [sync]	4.094 s	1.024 µs	n/a	n/a	n/a	n/a	n/a	ytes	360.54 MB/s
Memcpy HtoD [sync]	4.094 s	896 ns	n/a	n/a	n/a	n/a	n/a	ytes	140.5 MB/s
Memcpy HtoD [sync]	4.109 s	1.536 µs	n/a	n/a	n/a	n/a	n/a	ytes	327.83 MB/s
Memcpy HtoD [sync]	4.11 s	1.408 µs	n/a	n/a	n/a	n/a	n/a	ytes	43.35 MB/s
GPU_SwimMultiple(state_t, trajecto	4.11 s	9.01 ms	[1,1,1]	[64,1,1]	63	0	0	n/a	n/a
GPU_FindClosestTrajectoryPoint(int,	4.12 s	64.897 µs	[16,1,1]	[64,1,1]	22	0	0	n/a	n/a
GPU_CalcResiduals(int, int, trajector	4.12 s	25.088 µs	[16,1,1]	[64,1,1]	58	0	0	n/a	n/a
GPU_CalcMatrix_Binv(int, int, traject	4.12 s	23.616 µs	[3,1,1]	[5,5,1]	27	0	0	n/a	n/a
Memcpy DtoH [async]	4.12 s	2.944 µs	n/a	n/a	n/a	n/a	n/a	ytes	32.39 MB/s

Figure 1. NVVP timeline for the execution of the CUDA program. The timeline shows that almost 95% of the time was spent swimming the tracks through the magnetic field. This CUDA kernel is the main bottleneck of the performance of the GPU program.

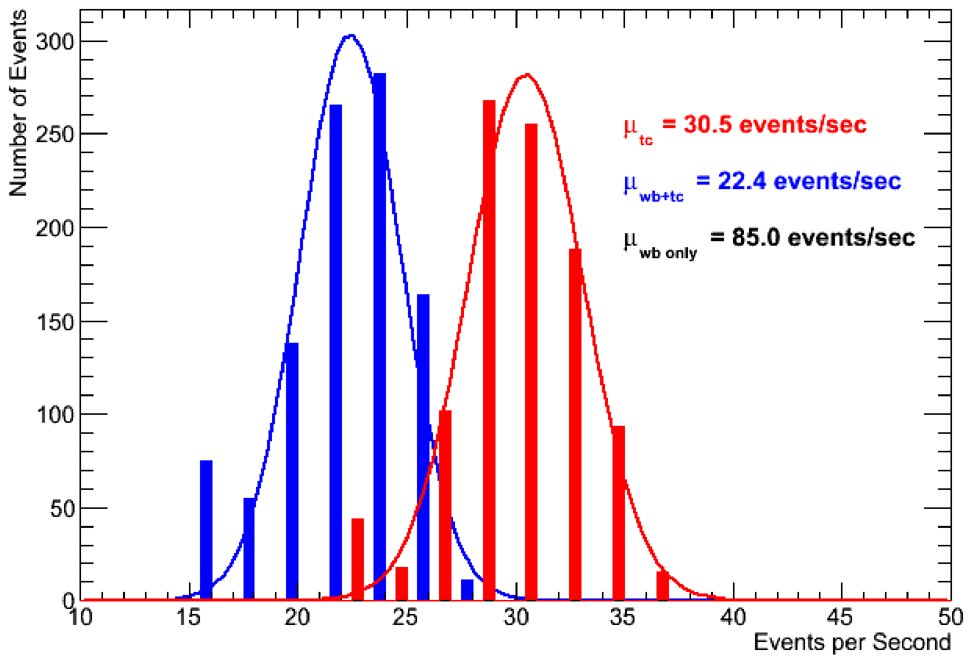


Figure 2. ROOT graph of track reconstruction rate for CPU code. The mean values for the track reconstruction rate alone and track-finding combined with reconstruction were used to find the mean of the reconstruction rate alone in events per second.

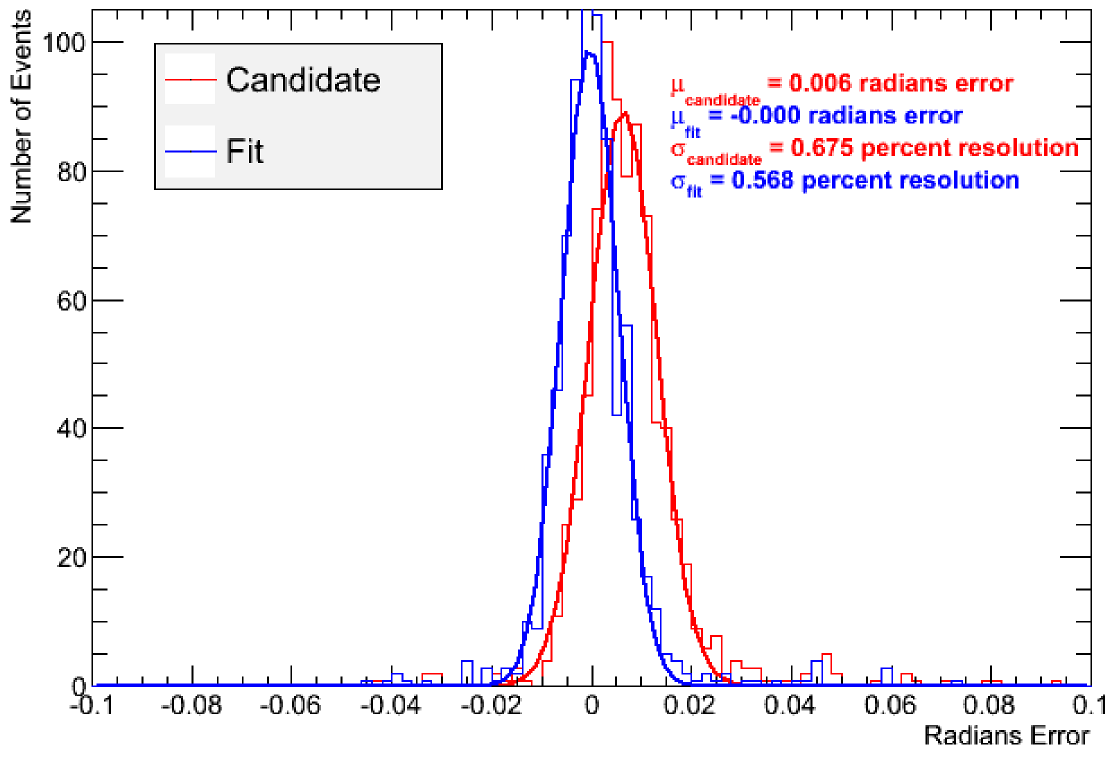


Figure 3. ROOT graph of theta resolution of unfitted track candidates vs. tracks fit with the CUDA program. The mean of the fit tracks was closer to zero and the resolution was better, suggesting that the fit is working for the theta angle of the fitting.

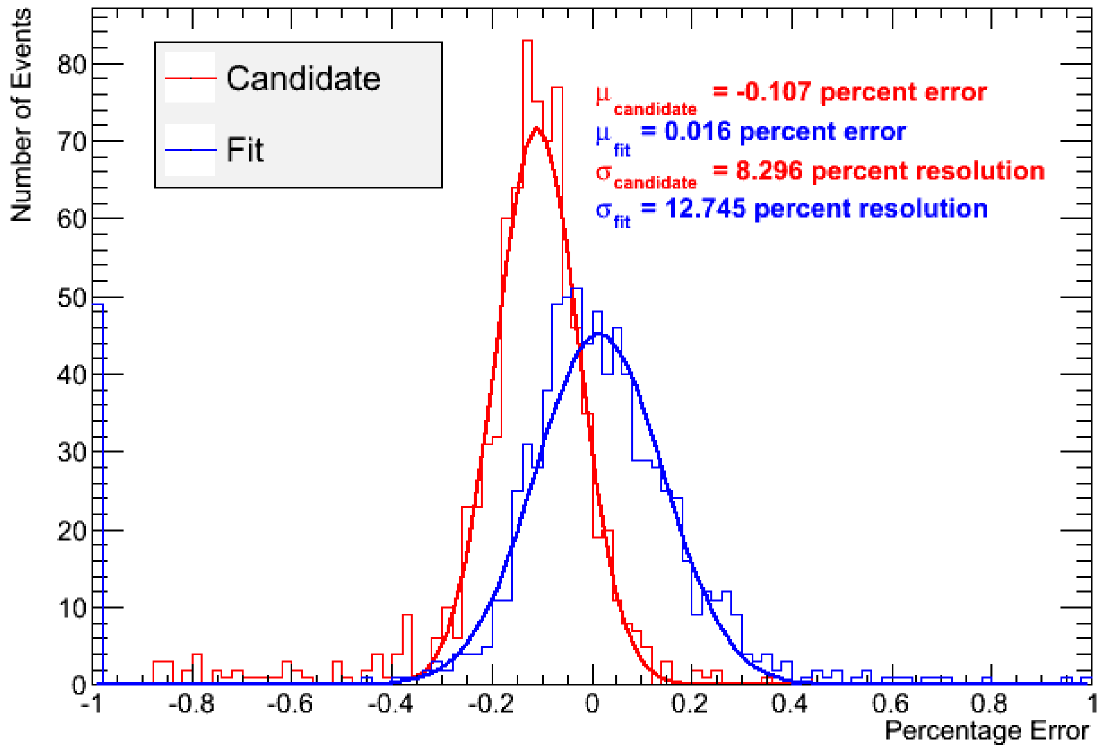


Figure 4. ROOT graph of momentum resolution of unfitted track candidates vs. tracks fit with the CUDA program. The mean error shifted towards zero, suggesting the track fitting is working. The worsened resolution is most likely a result of the GPU track fitter not using information from the cathode planes in the GlueX detector, while the candidate generation process did.

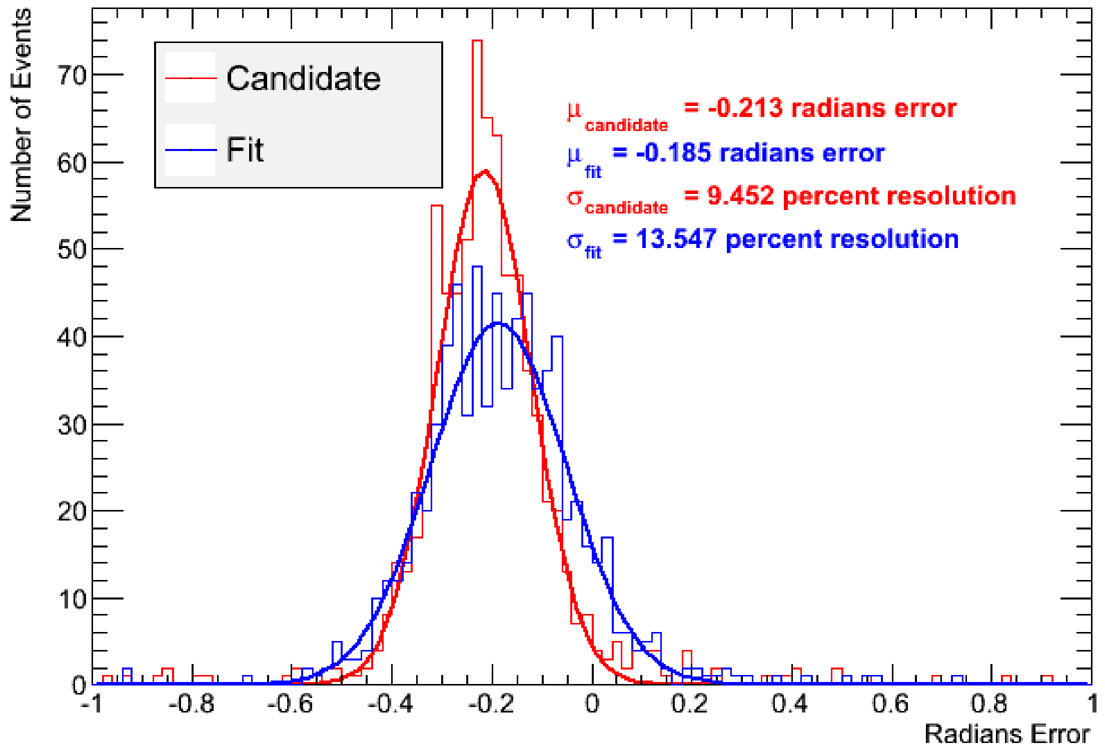


Figure 5. ROOT graph of phi resolution of unfitted track candidates vs. tracks fit with the CUDA program. The equivalent mean error suggests the track fitting is working. The worsened resolution is most likely a result of the GPU track fitter not using information from the cathode planes in the GlueX detector, while the candidate generation process did.