A Fast Jet Finder Algorithm Using Graphic Processing Unit

Raman Sehgal and A. K. Mohanty
Nuclear Physics Division, Bhabha Atomic Research Centre, Mumbai - 400085, INDIA
** email: sc.ramansehgal@gmail.com*

Introduction

A collimated emission of hadrons usually called Jet is the experimental counterparts of the partons (quarks and gluons) which are not observed separately. The CMS detector at LHC is ideally designed to study jet tomography which is an important probe to investigate the hot and dense medium formed during the heavy ion collisions. Although CMS analysis package incorporates various techniques for jet finding, the algorithm of successive recombination of particles based on k_t jet-clustering has been widely used for jet reconstruction. This algorithm is simple and infrared safe, however it takes a large computing time of the order of N^3 where N is the total multiplicity in a given event and the computation becomes extremely difficult or even impossible when multiplicity increases particularly in the environment of LHC heavy ion colliders. Recently Cacciari and Salam [1] have proposed a fastjet algorithm which uses geometrical approach to find the closest pair and reduces its running time from N^3 to NlogN. Here we are trying to implement the variant of fastjet algorithm where processing is done parallely over multiple threads on the Graphics Processing Unit.

Graphics Processing Unit (GPU)

GPU's are earlier designed to be dedicated processor optimized for accelerating graphics display. Recently the GPU's are available for general purpose computing having support for floating point arithmetic. Modern GPUs are massively parallel, and fully programmable. Using GPU, it is now possible to develop programs based on specialized language like CUDA or openCL which combines the computing power of both CPU and GPU, thus giving a large throughput. In this work, we have tried to implement the jet finding algorithm which can execute on GPU kernel. Although the work is very preliminary the results as discussed below are highly encouraging and suggest that GPU based programming can be used for Jet finding using multi-thread architecture which otherwise takes huge computation time if implemented only on a single CPU. For demonstration purpose, we have used NVIDIA Geforce GPU card (GT520) having 48 cores, compute capability of 2.1, (supports floating point), global memory of 1 gigabyte and 48 kilobytes per block of share memory.

The kt Jet Finder Algorithm

The sequential clustering algorithm for k_t jet finder can be formulated as follows:

1)For each pair of particles i, j, find out the distance

$$d_{ij} = min\,(k^2_{ti}, k^2_{tj})\, R^2_{ij}$$

with $R^2_{ij} = (\eta_i - \eta_j)^2 + (\Phi_i - \Phi_j)^2$

where k_{ti} , η_i , Φ_i are the transverse momentum, rapidity and azimuth of particle. Also for each particle find out the beam distance

$$d_{iB} = k^2_{ti}$$

2) Find the minimum d_{min} of all the d_{ij}, d_{iB}. If d_{min} is d_{ij} , merge particles i and j into a single particle, summing their four-momenta. If it is d_{iB} , then declare particle i to be a final jet and remove it from the list.

3) Repeat from step 1 until no particles are left.

This algorithm is easy to understand and implement but has a drawback of its high computational complexity of $O(N^3)$

Implementation on GPU

In this work, we are using CUDA (Compute Unified Device Architecture) which is an API in C language to program the kernel of the GPU [2]. Execution is carried out on each thread which is the smallest computing unit. Threads within the same block can synchronize with each other and can share data using shared memory. Before kernel execution begins, data is transfered from host (CPU) memory to device

(GPU) global memory. This is the process which takes longer time. However, once data is transferred, the kernel execution is extremely fast as each thread executes in parallel. For general-purpose computation on the GPU, an essential requirement is that the data structure can be arranged in arrays. The input to the k_t jet algorithm is a set of particles having four momentum components Px, Py, Pz & E. In the present implementation this input is casted as an one dimensional array of 4N elements. While implementing the sequential clustering algorithm for jet finder, the input array is divided into n equal parts, where n is the number of threads that execute on a given GPU. The initialization work is done on CPU and the computational work is performed on GPU. This results in considerable reduction of execution time.

Results

The algorithm when executed on a system having dual core CPU of 2.8 GHz frequency and 1 GB RAM takes 526.56 ms. The same algorithmic when implemented using CUDA library to run on NVIDIA GT 520 GPU having 48 core takes considerably less time about 32.17 ms when thread number increase to 6 (see figure 1).

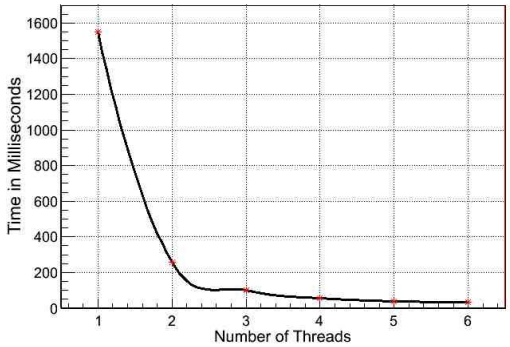

Fig 1: Execution time v/s number of threads

Figure 2 shows input data which contains 180 particles comprising of ten jets. Figure 3 shows the reconstructed jets which are computed using jet finder algorithm implemented in GPU

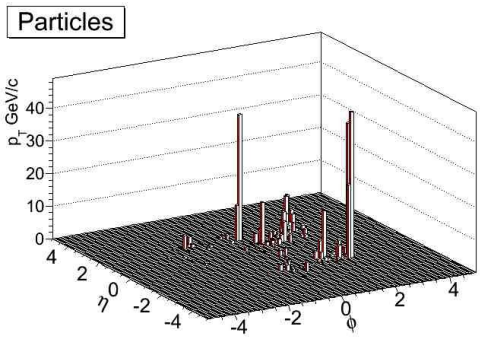

Fig 2: Real data with 180 particles

.

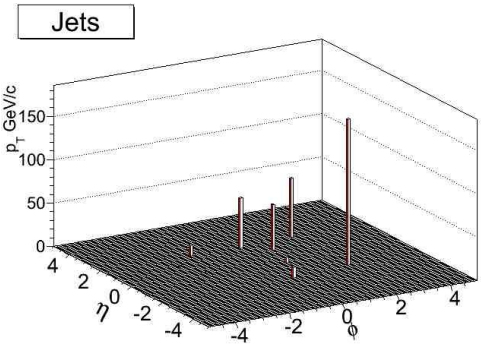

Fig 3: 10 Detected jets

Conclusion

In this paper we have presented an implementation of jet finder algorithm using NVIDIA GT 520 GPU card. It is observed that GPU can speed up the jet finding algorithm depending on how many threads are used. This work is in progress where the algorithm will be improved using Delaunay triangulation as was used in the original work [1] to search the nearest neighbours. Once this is achieved, the algorithm will parallelize the jet finding codes particularly for heavy ion jet reconstruction.

Reference

[1] Dispelling the N^3 myth for the k_t jet-finder, M. Cacciari and G. P. Salam, Phys. Lett. B 641, 57, 2006.

[2] http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf