Partial wave analysis with OpenAcc

Yanjia Xiao[1,*], *Xiaobin* Ji[1], *Beijiang* Liu[1], and *Xi'an* Xiong[1]

[1]Institute of High Energy Physics, Chinese Academy of Science, China

Abstract. Partial wave analysis(PWA) is an important tool in hadron physics. Large data sets from the experiments in high precision frontier require high computational power. To utilize GPU cluster and the resource of super computers with various types of accelerator, we implement a software framework for partial wave analysis using OpenAcc, OpenAccPWA. OpenAccPWA provides convenient approaches for exposing parallelism in the code and excellent support for the large amount of existing CPU-based codes of partial wave amplitudes. It can avoid heavy workload of code migration from CPU to GPU. This proceeding will briefly introduce the software framework and performance of OpenAccPWA.

1 Partial wave analysis(PWA)

The generally accepted theory for the strong interaction, quantum chromodynamics (QCD), remains a challenging part of the standard model in the low energy regime. Hadron spectroscopy provide a validation of and valuable input to the quantitative understanding of QCD. Partial wave analysis(PWA) is an important tool in hadron spectroscopy.

In PWA, the full kinematic information is used and fitted to a model of the amplitude in a partial wave decomposition. The resonance's spin-parity, mass, width and decay properties are accurately measured.[1] The most common approach to the partial wave analysis in modern experiments is the event-by-event maximum likelihood fit. In a fit, a maximum of the logarithm of the likelihood, corresponding to the best set of parameters for the used model is searched for.

For example, the two-body decay amplitudes in the sequential decay process $J/\psi \to N_x\gamma, N_x \to K^+K^-$ are constructed using the relativistic covariant tensor amplitude formalism [2]. In $J/\psi \to N_x\gamma, N_x \to K^+K^-$, A_j is the jth partial wave amplitude, which is described as

$$A_j = A^j_{prod-X}(BW)_X A_{decay-X} \tag{1}$$

where A^j_{prod-X} is the amplitude describing the production of the intermediate resonance N_x, $(BW)_X$ is the Breit-Weigner propagator of N_x, and $A_{decay-X}$ is the decay amplitude of N_x. The total differential cross section $\frac{d\sigma}{d\Phi}$ is

$$\frac{d\sigma}{d\Phi} = |\sum_j c_j A_j + F_{phsp}|^2 \tag{2}$$

where F_{phsp} denotes the nonresonant contribution described by an interfering phase space term.

* Corresponding author: xiaoyanjia@ihep.ac.cn

The probability to observe the event characterized by the measurement ξ is

$$P(\xi) = \frac{\omega(\xi)\epsilon(\xi)}{\int d\xi \omega(\xi)\epsilon(\xi)} \quad (3)$$

where $\omega(\xi) = \frac{d\sigma}{d\Phi}$ and $\epsilon(\xi)$ is the detection efficiency. $\int d\xi \omega(\xi)\epsilon(\xi) = \sigma$ is the normalization integral calculated from the exclusive Monte Carlo(MC) sample. The joint probability density for observing n events in the data sample is $\prod_{i=1}^{n} P(\xi_i)$. For a given data set, the $\epsilon(\xi)$ is a constant and has no impact on the determination of the parameters of the amplitudes. So, the maximum likelihood function is

$$ln\, L = \sum_{i=1}^{n} ln\left(\frac{\omega(\xi_i)}{\int d\xi \omega(\xi)\epsilon(\xi)}\right) = \sum_{i=1}^{n} ln\left(\frac{d\sigma}{d\Phi}/\sigma\right) \quad (4)$$

According to the general form of the decay amplitude, $ln\, L$ can be defined as [3]

$$ln\, L = \sum_{n=1}^{N_{data}} \left(ln\left(\sum_{i,j}^{N_{wave}} P_{i,j} FU_{i,j}\right) - ln\left(\sum_{i,j}^{N_{wave}} P_{i,j} FUN_{i,j}\right)\right) \quad (5)$$

where $P_{i,j}$ are the fit parameters, $FU_{i,j} = \frac{1}{2}\sum_{\mu=1}^{2} U_i^{\mu} U_j^{*\mu}$, the expression of partial wave amplitude($U_{\mu\upsilon}^{i}$) varies with the decay channel, and $FUN_{i,j} = \sum_{m=1}^{N_{MC}} FU_{i,j}$

2 GPUPWA at BESIII

The Beijing Spectrometer III (BES-III) is an important particle physics experiment at the Beijing Electron–Positron Collider II (BEPC-II) at the Institute of High Energy Physics(IHEP). The pioneer approach of harnessing GPU parallel acceleration in PWA was performed in the framework of BES-III.[4] BES-III developed GPUPWA software framework based on OpenCL. GPUPWA uses the programming language of C + +, and its functions of fitting and drawing are realized by ROOT.[5] IHEP has established a GPU High Performance Computing Cluster.

On an Intel Core 2 Quad 2.4 GHz workstation with 2 GB RAM and an ATI Radeon 4870 GPU with 512 MB RAM, a $J/\psi \to \gamma K^{+}K^{-}$ analysis with four partial waves runs more than 100 times faster than the reference FORTRAN implementation for sufficiently large numbers of events.

3 Introduction to OpenAcc

The OpenACC Application Program Interface describes a collection of compiler directives to specify loops and regions of code in standard C, C++ and Fortran to be offloaded from a host CPU to an attached accelerator device, providing portability across operating systems, host CPUs and accelerators.

At its core OpenACC supports offloading of both computation and data from a host device to an accelerator device. In the case that the two devices have different memories the OpenACC compiler and runtime will analyze the code and handle any accelerator memory management and the transfer of data between host and device memory.

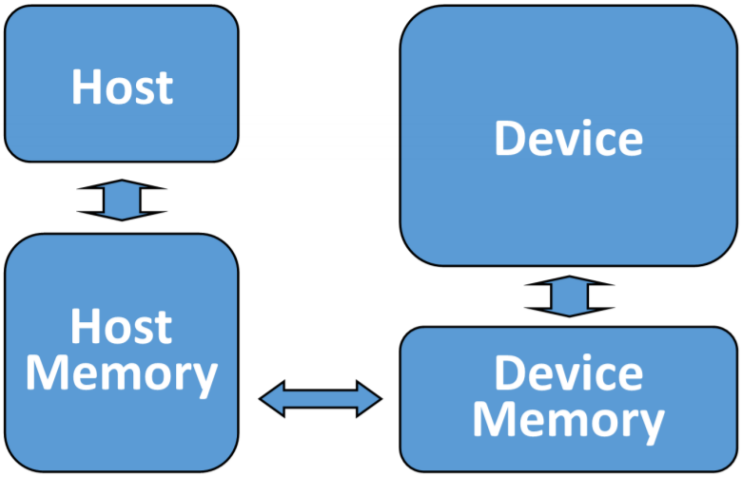

Fig. 1. OpenACC's Abstract Accelerator Model

OpenACC uses high-level compiler directives to expose parallelism in the code and parallelizing compilers to build the code for a variety of parallel accelerators. OpenACC allows parallel programmers to provide simple hints to the compiler identifying which areas of code to accelerate, without requiring programmers to modify or adapt the underlying code itself. It reinforces the ability of code transplant.

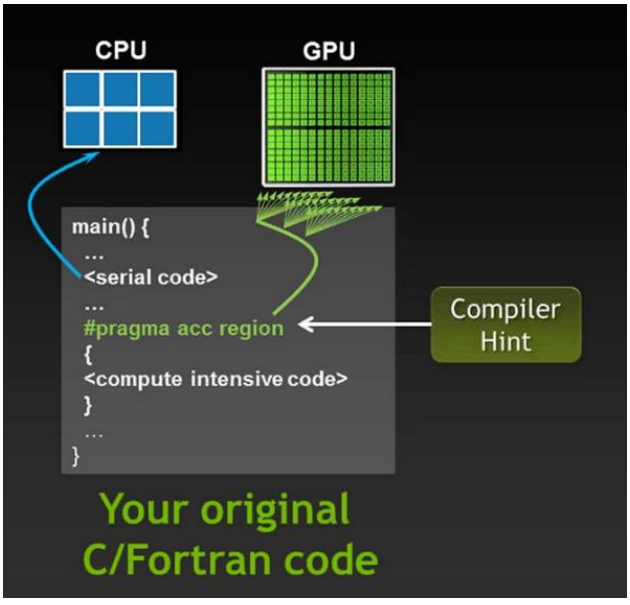

Fig. 2. Compiler hint

OpenACC compiler can generate parallel code on different platforms through this high-level programming model, so that the application written by OpenACC has excellent cross platform performance. It is more convenient to use supercomputing resources. On the other hand, covariant tensor amplitudes of baryon spectroscopy are very complicated. The corresponding codes are difficult to be ported to GPUPWA (OpenCL).

4 OpenAccPWA

To utilize GPU cluster and the resource of super computers with various types of accelerator, we implement a software framework for partial wave analysis using OpenAcc, OpenAccPWA based on GPUPWA.

In OpenAccPWA, users need to provide the calculation formula of $FU_{i,j}$, data sets of Data, MC and initial parameters according to their own physical analysis at first. OpenAccPWA can keep the original partial wave amplitude calculation code from C++.

```
struct double4 p12 = {(p1.x+p2.x),(p1.y+p2.y),(p1.z+p2.z),(p1.w+p2.w)};
double mx2 = p12.w*p12.w-(p12.x*p12.x+p12.y*p12.y+p12.z*p12.z);

double2 U_BW1 = BW(mx2,M_bw1,W_bw1);
double2 U_BW2 = BW(mx2,M_bw2,W_bw2);
double2 P_1 = {Mag_bw1*cos(Pha_bw1),Mag_bw1*sin(Pha_bw1)};
double2 P_2 = {Mag_bw2*cos(Pha_bw2),Mag_bw2*sin(Pha_bw2)};

double2 U11 = Multi(U_BW1,U_BW1);
double2 U12 = Multi(U_BW1,U_BW2);
double2 U22 = Multi(U_BW2,U_BW2);
double2 P11 = Multi(P_1,P_1);
double2 P12 = Multi(P_1,P_2);
double2 P22 = Multi(P_2,P_2);

double2 P11U11=Multi(P11,U11);
double2 P22U22=Multi(P22,U22);

dcs = 4*(P11U11.x*P11U11.x+P22U22.x*P22U22.x+2*(P12.x*U12.x-P12.y*U12.y));
```

Fig. 3. Two partial wave amplitudes from the decay channel: $J/\psi \to \gamma K^+K^-$ as OpenAccPWA code

Then multi iterations are needed to complete maximum likelihood fitting, the input parameters of the next iteration are dependent on the results of the previous iteration. So the iteration process run serially. Due to events are independent of each other, parallel computing can be used to speed up the process of calculating likelihood function. In addition, the matrix element $FU_{i,j}$ is only related to four momentum in some cases. Calculating $FU_{i,j}$ once will improve performance effectively.

At last, in order to find the real optimal solution, the fit has to be repeated over and over with new input.

Table 1. Workflow of OpenAccPWA framework.

workflow	Input	Output	Function
Data reading	Data, MC File(root) Parameters File		Get the parameters to be fitted Get the 4-momentum of events
Cal. FUNij	4-momentum of events	FUNij_data FUNij_mc	Parallel computing once Store FUNij to GPU memory
Cal. mctcs	Parameters FUNij_mc	mctcs(total cross section of MC)	Parallel computing
Cal. likelihood	Parameters FUNij_data, mctcs	likelihood	Parallel computing
Fit	Parameters likelihood	Fit state Fitted parameters Final likelihood	Fit algorithm running in CPU Multiple iterations

5 Performance

On an Intel Xeon(R) 2X8 cores CPU and a NVIDIA Tesla K80(8) GPU, a $J/\psi \to \gamma K^+K^-$ analysis with two partial waves has been run. Several acceleration schemes are used to accelerate this example, including CPU, CPU(4-thread), multicore CPU(16 cores) and GPU.

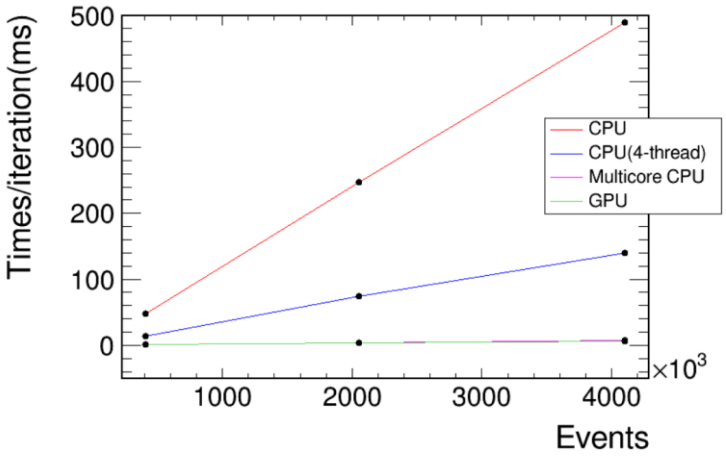

Fig. 4. Performance of OpenAccPWA

The performance of OpenAccPWA with GPU are about 50~75 times than with CPU. The acceleration effect is more significant as the number of events increases. Using Multicore CPU as accelerator is faster than using GPU in the case of low statistics. The program speed is limited by data transfers between the CPU and GPU and GPU memory accesses.

Predictably, in computationally expensive PWA, OpenAccPWA with GPU will have a better performance.

6 Conclusions

We have implement the parallelization of OpenAccPWA based on GPUPWA. The next step is to optimize the performance of the framework and improve program reliability. A friendly user environment to meet different PWA needs is also necessary. In addition, to deploy OpenAccPWA on Sunway Taihulight, another version out of ROOT environment is under development.

This work is supported in part by the CAS Large-Scale Scientific Facility Program; National Key Research and Development Program of China No.2017YFB0203200; National Natural Science Foundation of China(NSFC) under Contracts No. 11775245,11275210; Joint Large-Scale Scientific Facility Funds of the NSFC and CAS under Contract No. U1732103.

References

1. M. Ablikim et al. (BES Collaboration), Phys. Rev. D **98**, 072003 (2018)
2. J. X. Wang, Nucl. Instrum. Methods Phys. Res., Sect. A **534**, 241 (2004)
3. S.U. Chung, CERN Yellow Report, No. CERN 71-8, Geneva,Switzerland(1971)
4. M. Battaglieri et al., Acta Phys. Polon. B 46 (2015) **257**
5. N. Berger, B.J. Liu, and J.K. Wang, J. Phys. Conf. Ser. **219**, 042031 (2010).