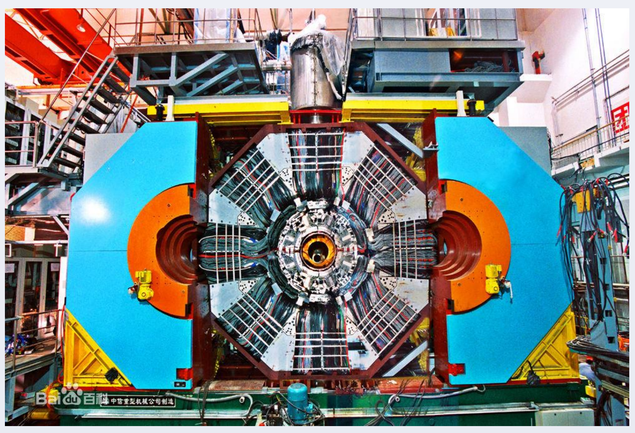
Partial wave analysis with OpenAcc

Beijiang Liu, Xiaobin Ji, Yanjia Xiao, Xi'an Xiong

Institute of High Energy Physics, Chinese Academy of Science

INTRODUCTION

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]



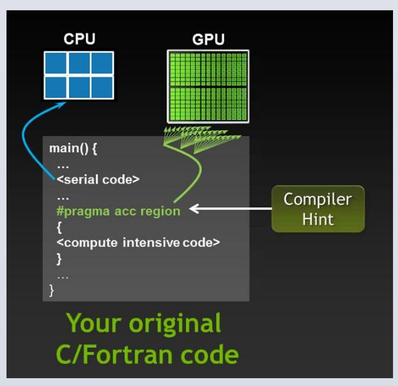
The Beijing Spectrometer III (BES-III) is an important particle physics experiment at the Beijing Electron–Positron Collider II (BEPC-II) at the Institute of High Energy Physics(IHEP). The pioneer approach of harnessing GPU parallel acceleration in PWA was performed in the framework of BES-III.[2] BES-III 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.[3] IHEP has established a GPU High Performance Computing Cluster.

OpenACC is a programming model that uses high-level compiler directives to expose parallelism in the code and parallelizing compilers to build the code for a variety of parallel accelerators.[4]

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.

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).

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.



OpenAcc HPC Cluster at IHEP Shenwei SC

OpenAccPWA Framework

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. The likelihood function can be constructed using this kind of formula:

$$\boldsymbol{ln\,L} = \sum_{n=1}^{N_{data}} \boldsymbol{ln}(\boldsymbol{Prob}(\overrightarrow{x_n}, \vec{p})) = \sum_{n=1}^{N_{data}} \boldsymbol{ln}\frac{\omega(\xi)\epsilon(\xi)}{\int d\xi\, \omega(\xi)\epsilon(\xi)}$$

$$= \sum_{n=1}^{N_{data}} \boldsymbol{ln}\left(\frac{d\sigma}{d\Phi}/\sigma\right) = \sum_{n=1}^{N_{data}} \boldsymbol{ln}\left(\sum_{i,j}^{N_{wave}} \boldsymbol{P_{i,j}FU_{i,j}}\right) - \boldsymbol{ln}\left(\sum_{i,j}^{N_{wave}} \boldsymbol{P_{i,j}} \sum_{m=1}^{N_{MC}} \boldsymbol{FU_{i,j}}\right)$$

$\overrightarrow{x_n}$, ξ : 4-momentum of events; $\vec{p}$, $P_{i,j}$: parameters to be fitted

$FU_{i,j}$: function related to $\overrightarrow{x_n}$, user definition

$\omega(\xi)$: differential cross section of each event ; σ : total cross section

OpenAcc Framework

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 FUij is only related to four momentum in some cases. Calculating FUij once will improve performance effectively.

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

Performance

The performance of OpenAccPWA with GPU are about 50~75 times than with CPU.

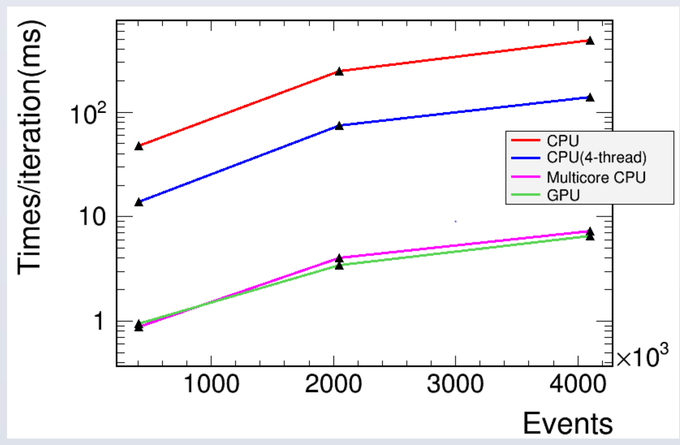
The acceleration effect is more significant as the number of events increases.

Using Multicore CPU as accelerator is faster than using GPU in the case of low statistics.

Hardware environment:
CPU: Xeon(R) 2X8 cores
GPU: Tesla K80 (8)

Software environment:
CUDA-9.0; ROOT v6.12;
PGI Community Edition v19.4

PWA: $J/\psi \to \gamma K^+ K^-$
2 partial waves are added



Time(ms)	CPU	CPU(4-thread)	Multicore CPU	GPU
Data:10k MC:400k	16440	4772	302	325
Data:50k MC:2000k	85906	26624	1437	1174
Data:100k MC:4000k	169119	50330	2630	2346

CONCLUSIONS

We have implement the parallelization of OpenAccPWA based on GPUPWA.

OpenAccPWA will be further improved :

Optimize performance and improve program reliability

Develop a friendly user environment to meet different PWA needs

To deploy OpenAccPWA on Sunway Taihulight, another version out of ROOT environment is under development

REFERENCES

[1] 1]Berger N. Partial Wave Analysis using Graphics Cards. // Proceedings of the XIV International Conference on Hadron Spectroscopy (hadron2011), Munich, 2011, edited by B. Grube, S. Paul, and N. Brambilla, eConf C110613 (2011) [arXiv:1108.5882v1], pp. 810-817

[2] M. Battaglieri et al., Analysis tools for next-generation hadron spectroscopy experiments, Acta Phys. Polon. B 46 (2015) 257 [arXiv:1412.6393] [INSPIRE].

[3] http://sourceforge.net/projects/gpupwa N. Berger, B. J. Liu, and J. K. Wang, J. Phys. Conf. Ser. 219, 042031(2010)

[4] http://www.openacc.org/

CONTACT

Mail: xiaoyanjia@ihep.ac.cn; liubj@ihep.ac.cn; jixb@ihep.ac.cn; xiongxa@ihep.ac.cn;