The parallel framework for the partial wave analysis

V. A. Tokareva[a], I. I. Denisenko

Department of Colliding Beam Physics
Dzhelepov Laboratory of Nuclear Problems, Joint Institute for Nuclear Research,
6, Joliot-Curie st., Dubna, Moscow region, 141980, Russia

E-mail: [a] tokareva@jinr.ru

Partial wave analysis is a fundamental technique for extracting hadron spectra and studing hadron scattering properties. It is employed in some current experiments in particle physics like BES-III, LHCb, COMPASS, and is planned to be employed in future ones like PANDA.

The analysis is typically performed using the event-by-event maximum likelihood method for which the calculations for contributions of different amplitudes can be done independently.

Large amount of already collected data and planned increase of data flow in future make it especially important to develop the software capable of analysing large sets of information at small times, since currently existing software is either not designed to be scalable for growing data amount, or has significant limitations in its capabilities.

The development of high performance computing technologies employing multi-core CPUs, co-processors and GPUs allow to decrease the time needed for data processing.

To accomplish this task the software framework for the partial wave analysis is being developed. It is intended to accelerate the calculations by using the parallel computing technologies, and to get rid of limitations of currently existing analogs.

At present time the architecture and performance of this framework is being optimized using the J/ψ to pions and kaons decay channel.

In this work the development of software realizations employing OpenMP and MPI high performance computing technologies is described. The OpenMP realization was tested both on multi-core CPUs and on multi-core Intel Xeon Phi co-processors in the native execution mode.

These realizations were studied on various hardware architectures taking into account distinctive features of the task and external software used in the realization.

They were tested using the resources of the heterogeneous cluster HybriLIT. The results on calculation speedup and efficiency as well as a comparative analysis of the developed parallel implementations are presented.

Keywords: data analysis, high performance computing, high energy physics, statistics and probability

Introduction

Partial wave analysis (PWA) is a key method used to determine spectra of hadrons in particle physics. This method has been successfully used in the analysis of number of previous experiments, but huge statistics already collected by experiments like COMAPASS, BESIII and LHCb or is expected to be collected in future experiments like PANDA is challenging from the point of analysis.

In particular, the BES-III experiment [1] has accumulated about 1.225 billion J/ψ decays in hope to enrich the spectroscopy of ordinary light hadrons and search for exotic states like tetraquarks, pentaquarks, or glueballs. These aims can be accomplished only by correct handling of mixing and interference phenomena, and PWA is the only tool for that. Due to the complexicity of the task lots of various initial PWA assumptions need to be tested, so fast fitting software becomes crucial for the success of data analysis.

Typical data fit is performed in the framework of the unbinned maximum likelihood method, which results in huge and not often acceptable computation time. On the other hand the task can be naturally parallelized, since amplitudes for each event are computed independently.

Currently available PWA tools are either single-threaded or multi-threaded with limited capabilities not sufficient for all physical cases [2; 3]. Evolution of Intel Xeon Phi co-processors gives us promising practical tool for high perfomance computing different from commonly used methods employing GPU-cards.

Here we study scalability of a self-written PWA code prototype. An OpenMP and MPI realizations are presented. The OpenMP realization was tested both on multicore CPU and Intel Xeon Phi co-processors using native mode.

PWA basics

Currently the PWA program prototype is limited to the $J/\psi \to K^+K^-\pi^0$ decay (typical process that can be observed in BESIII experiment). Momenta of the final particles are considered as observables. The program consists of minimizer (currently MINUIT [4]) and a part that provides the objective function

$$s = -\ln L. \tag{1}$$

Here L is a likelihood to observe experimental events with measured momenta ($L = \prod_i P_i$, where P_i is the probability for each collected event i). Finally, the probability P_i is proportional decay amplitude squared: $P_i \propto |A_i|^2$. The proportionality factor will be discussed later.

The decay is considered in the isobar model: $J/\psi \to R_{KK}\pi^0(R_{KK} \to KK)$ and $J/\psi \to R_{K\pi^0}K(R_{K\pi^0} \to K\pi^0)$, where R_{KK} $(K\pi)$ is the intermediate resonance in the KK $(K\pi^0)$ kinematic channel. Each resonance is characterized by product of complex production and decay couplings, mass, width and decay radius. Contributions from all intermediate states are summed to the decay amplitude. The resonances can have different quantum numbers, which leads to different wave functions for subsequent J/ψ and resonance decays. These wave functions are constructed using covariant tensor formalism [5], each decay vertex is supplemented with Blatt-Weisskopf form-factors (their explicit form can be found in [6]). The resonance lineshape is given the mass-dependent Breit-Weigner formula:

$$A_m^{BW} = \frac{1}{M^2 - s_m - iM\Gamma(s_m, J)}.$$

Here M, J and s_m are the resonance mass, spin and the invariant mass squared of its daughter particles (m refers to the kinematic channel).

The width is parameterized as follows:

$$\Gamma(s_m, J) = \frac{\rho_J(s_m)}{\rho_J(M^2)}\Gamma, \quad \rho_J(s) = \frac{2q}{\sqrt{s}} \frac{q^{2J}}{F^2(q^2, r, J)},$$

where Γ_a is the resonance width, q is the relative momentum of resonance daughter particles calculated in the resonance rest frame, $F(q^2, r, J)$ is the mentioned Blatt-Weisskopf form factor, which also depends on the resonance decay radius r.

Note, that no normalization condition has been introduced. To obtain the probability P_i from the amplitude squared $|A_i|^2$ a normalization factor, defined as integral of the $|A|^2$ over the phase space is introduced. This integration is approximated as Monte-Carlo integral over generated Monte-Carlo phase space (PHSP) sample.

Performance has been measured using a toy generated MC sample of "data" ($K^*(892)$ and $K_2(1430)$ resonances were introduced) of 5000 events and a PHSP sample of data for normalization consisting of 10000 events.

Parallel realizations and their features

The framework is designed to consist of the modules common for any studied decay channel: some code responsible for the mathematical formalism, an interface for the minimizer, and a purely virtual parent class providing interface for interacting with the particular isobar model. The model-dependent code is to be realized in subclasses corresponding to the processes being analyzed.

An important feature of the framework is intensive use of caching. When intermediate results do not need to be recalculated for at least some subsequent calls of minimization function, they are stored in memory. So caching means that memory saving is traded in favour of faster computations.

The part being parallelized is the objective function (1) that is implemented as a part of the minimizer interface. The parallelization has been done on an event-by-event basis using either OpenMP version with multiple threads or MPI realization with parallel computational processes. The OpenMP realization has been tested both on multi-core CPUs and on multi-core Intel Xeon Phi co-processors in the native execution mode. For the OpenMP version the realization is straightforward because of shared memory model and implicit thread communication mechanism. In the MPI realization the root process sends the corresponding parts of data array to the child processes. When the objective function is called, it instructs the child processes to calculate their parts of the sum, then summarizes their results into a single value and passes it back to the minimizer.

Computations were executed on the heterogeneous cluster HybriLIT [7].

Comparative analysis of realization's quality was performed using the following quality indicators: 1) calcutation time vs. the number of threads/processes; 2) speedup T_1/T_n, where T_1 is calcutation time for one thread (or process) and T_n is calcutation time for n threads (processes); 3) efficiency $T_1/(nT_n)$, where n is a number of threads or processes, and T_1 and T_n are defined above.

The indicator values for different realizations are given onwards in figures 1–3. It can be seen that they considerably fluctuate depending on the number of threads. There are several reasons for such behaviour, one of them being interplay of Intel hyper-threading technology with assignment of virtual CPU cores for different threads/processes, resulting in some drop of efficiency when two virtual cores are assigned to the same physical one. Another reason contributing to fluctuations of quality indicators is non-optimal caching realization. In current

version of the code each iteration requires unpredictable calculation time differing much from each other. That's why it is difficult to split the cycle on parallel subtasks with equal computing time. Real-time cluster conditions may also contribute to fluctuations.

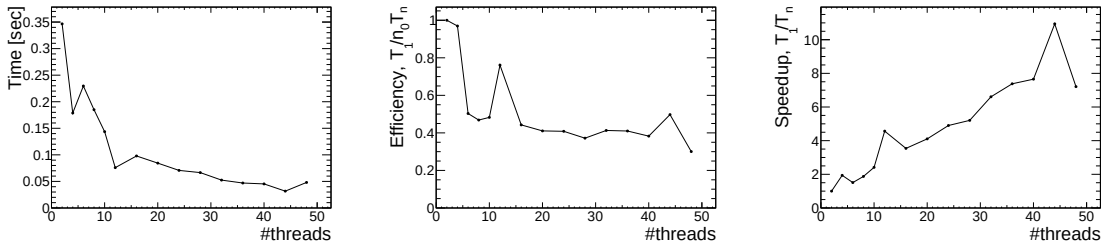

Рис. 1. Quality indicators of OpenMP realization for multicore CPU vs. the number of threads, from left to right: calcutation time; efficiency; speedup.

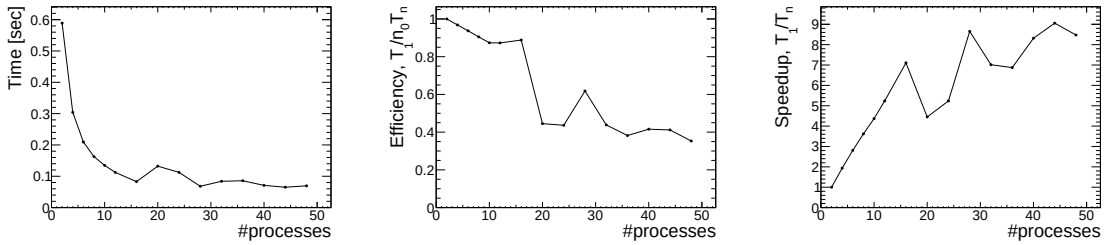

Рис. 2. Quality indicators of MPI realization for multicore CPU vs. the number of processes, from left to right: calcutation time; efficiency; speedup.

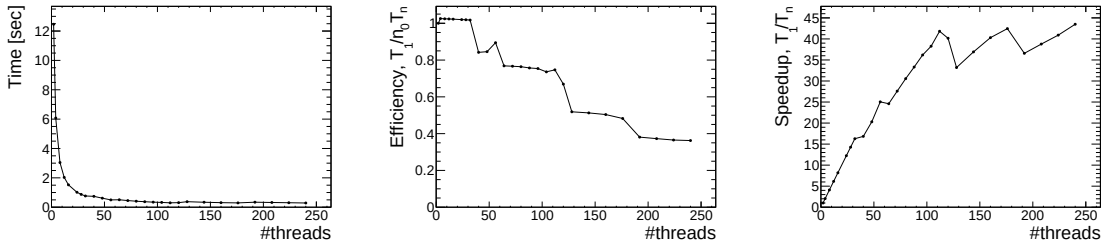

Рис. 3. Quality indicators of OpenMP for Xeon Phi co-processors native mode realization vs. the number of threads, from left to right: calcutation time; efficiency; speedup.

The main results of calculations are joined in the table 1. Here we can see that the OpenMP realization outperforms the MPI one in terms of both computation time and speedup, so it was chosen as a primary one for the future development of the framework. The application executed on Xeon Phi shows the best speedup due to the largest number of available cores, as expected, but for Xeon Phi to compete with CPU in terms of computation time the code crucially needs to be vectorized by refactoring the main program structures.

Outlook

The near-term plans of improving the framework include several steps. We are going to add the possibility to use the FUMILI minimizer [8] in addition to MINUIT one. The caches organization needs to be reworked: the currently used array of structures should be changed

Таблица 1. Comparing computation time for different parallel realizations; $t_{\min}$ means computation time for 2 threads/processes, and $t_{\max}$ for maximum employed number of threads/processes, that was 48 for CPU and 240 on Xeon Phi co-processors.

Technology	$t_{\min}$ [sec]	$t_{\max}$ [sec]	Speedup
OpenMP	0.35	0.03	11
MPI	0.59	0.06	9
OpenMP on Xeon Phi	12.4	0.28	44

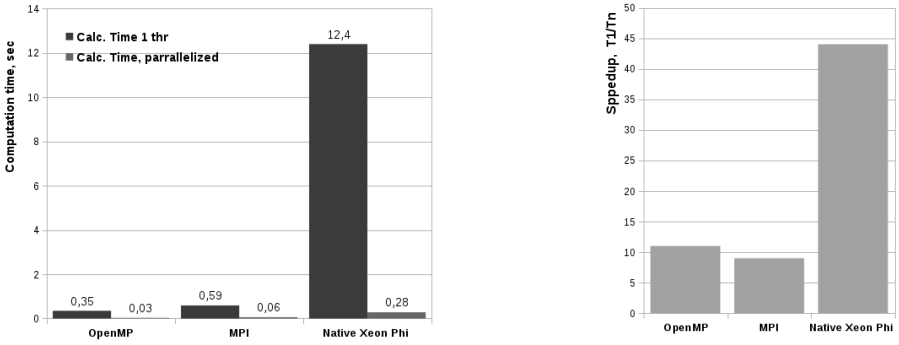

Рис. 4. Comparing quality indicators for different parallel realizations, from left to right: calcutation time; speedup.

to structure of simple arrays. This will provide a necessary environment for code vectorization, allow to use Xeon Phi offload mode, that is more user-friendly than currently employed native one, and make computation times more predictable for better splitting of work between threads. User-friendly interface and extending analysis capabilities are planned as well.

Acknowledgments

Authors would like to express thier gratitude to HybriLIT heterogeneous cluster team for providing access to the cluster and for knowledge sharing.

References

BES-III experiment [Electronic resource]: http://bes3.ihep.ac.cn/ (accessed 31.10.2016)

Berger N. Partial wave analysis using graphics cards [Electronic resource]: https://arxiv.org/abs/1108.5882v1 (accessed 31.10.2016)

ComPWA framework [Electronic resource]: https://github.com/ComPWA/ComPWA (accessed 31.10.2016)

James F., Roos M. MINUIT — a system for function minimization and analysis of the parameter errors and correlations // Comp. Phys. Communications. — 1975. — Vol. 10. — P. 343–3675.

Zou B. S., Bugg D. V. Covariant tensor formalism for partial wave analyses of φ decay to mesons // Eur. Phys. J. A. — 2003. — Vol. 16. — P. 537–547

Anisovich A. V., Klempt E. et al. Partial wave decomposition of pion and photoproduction amplitudes // Eur. Phys. J. A. — 2005. — Vol. 24. — P. 111–128

HybriLIT cluster, oc̄ial site [Electronic resource]: http://hybrilit.jinr.ru/ (accessed 31.10.2016)

Dymov S. N., Kurbatov V. S., Silin I. N., Yaschenko S. V. Constrained minimization in the C++ environment // Nucl. Instrum. Methods Phys. Res. A. — 2000. — Vol. 440. — P. 431–437