Распределённые и параллельные вычисления в науке и технике

УДК 004.93'1:519.237:51-72 PACS 02.50.Sk; 02.70.Rr; 29.85.-c; 07.05.-t

Векторизация и распараллеливание алгоритмов селекции и реконструкции распадов $J/\psi \to e^+e^-$ в реальном времени эксперимента CBM

О. Ю. Дереновская, В. В. Иванов

Лаборатория информационных технологий
Объединённый институт ядерных исследований
ул. Жолио-Кюри, д. 6, Дубна, Московская область, Россия, 141980

Измерения распадов $J/\psi \to e^+e^-$ относятся к ключевой задаче эксперимента CBM. Для их регистрации разработана методика, которая включает в себя цепочку методов и алгоритмов, предназначенных для реконструкции траекторий и импульсов заряженных частиц с помощью детектора STS, их идентификации с помощью детекторов RICH, TRD и TOF, формирования кандидатов в J/ψ-мезоны и определения их характеристик с помощью пакета KFParticle. Принимая во внимание тот факт, что отбор и реконструкцию распадов $J/\psi \to e^+e^-$ планируется проводить в реальном времени эксперимента, используемые методы и алгоритмы должны быть не только эффективными, но и быстрыми.

В настоящей работе проведена оценка временных затрат существующих алгоритмов с учётом их ускорения за счёт векторизации программного кода посредством SIMD-инструкций и распараллеливания между ядрами процессора, реализованное с помощью программных сред OpenMP, OpenCL и библиотеки TBB. Проведённый анализ позволил установить «слабые» места в этой цепочке, над которыми предстоит дальнейшая работа по их ускорению, а также предложить быстрый и эффективный параллельный алгоритм для идентификации заряженных частиц с помощью детектора TRD на основе критерия ω_n^k.

Ключевые слова: многомерные методы анализа данных, эксперимент CBM, отбор распадов $J/\psi \to e^+e^-$, параллельные вычисления, SIMD-инструкции.

1. Введение

Детальные экспериментальные исследования распадов чармония в плотной и горячей среде, такой, как кварк-глюонная плазма, могут предоставить новые данные для изучения проблемы восстановления киральной симметрии и объяснения механизма возникновения массы адронов [1]. Измерения процессов, связанных с рождением чармония, выполненные в экспериментах NA50 [2] и RHIC [3] при высоких температурах и низких барионных плотностях, до сих пор не дали чёткого объяснения существующим моделям. Определение с высокой точностью выходов, функции возбуждения, прямых и эллиптических потоков J/ψ-мезонов при относительно низких энергиях и высоких барионных плотностях должно предоставить дополнительную экспериментальную информацию, необходимую для понимания происходящих процессов.

В настоящее время в ГСИ (Дармштадт, Германия) ведутся работы по созданию ускорительного комплекса антипротонов и тяжёлых ионов FAIR (Facility for Antiproton and Ion Research). На этом комплексе планируется проведение экспериментов на установке CBM (Compressed Baryonic Matter) [4], создаваемой большой международной коллаборацией с участием ОИЯИ [5]. Физическая программа эксперимента CBM нацелена на изучение свойств сверхплотной барионной материи, образующейся в ядро-ядерных соударениях при энергии пучка $2 \div 45$ ГэВ/нуклон [4, 6].

Статья поступила в редакцию 19 августа 2014 г.

Схема экспериментальной установки CBM для изучения электронных и адронных распадов приведена на рис. 1.

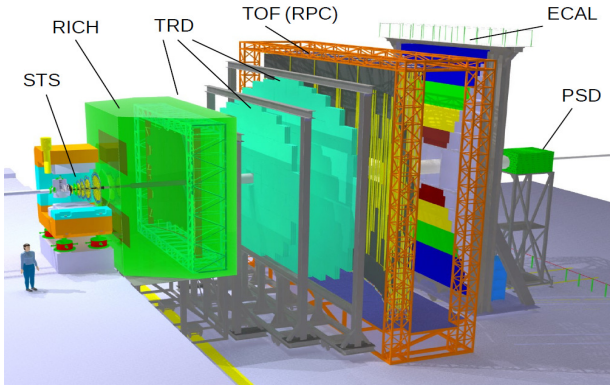

Рис. 1. Схема экспериментальной установки CBM для изучения электронных и адронных распадов

Непосредственно за мишенью между полюсами сверхпроводящего дипольного магнита располагается система координатных трековых детекторов STS (Silicon Tracking System), состоящая из восьми двухсторонних кремниевых микростриповых детекторов. Детекторы STS предназначены для реконструкции траекторий и определения импульсов заряженных частиц, а также для восстановления первичных и вторичных вершин. Система идентификации электронов/позитронов включает детекторы черенковского RICH (Ring Imaging CHerenkov) и переходного TRD (Transition Radiation Detector) излучений. TRD также используется для реконструкции траекторий заряженных частиц, регистрируемых координатными детекторами TRD. Детектор измерения времени пролёта TOF (Time-Of-Flight) предназначен для идентификации адронов. Электромагнитный калориметр ECAL (Electromagnetic CALorimeter) служит для идентификации фотонов и электронов, а калориметр PSD (Projectile Spectator Detector) используется для определения центральности соударений и плоскости реакций[1].

Измерения чармония (J/ψ, ψ') по диэлектронному каналу распада относятся к ключевым задачам эксперимента CBM. Редкие распады $J/\psi \to e^+e^-$ можно будет регистрировать в условиях интенсивных потоков (до 10^7 ядро-ядерных соударений в секунду) и высокой множественности вторичных частиц (от 100 до 1000 частиц в одном соударении). С учётом вышесказанного потребуется не только применение надёжных и эффективных математических методов и алгоритмов, но и максимальное использование потенциала современных высокопроизводительных компьютеров.

В работе [7] рассмотрена методика, используемая нами для реконструкции распадов $J/\psi \to e^+e^-$ в установке CBM. Она включает в себя цепочку методов и алгоритмов, предназначенных для реконструкции траекторий и импульсов заряженных частиц с помощью детектора STS, их идентификации с помощью детекторов RICH, TRD и TOF, формирования кандидатов в J/ψ-мезоны и определения их характеристик с помощью пакета KFParticle [8].

Так как регистрацию и реконструкцию событий, связанных с рождением чармония, планируется проводить в режиме реального времени эксперимента, то рассматриваемые методы и алгоритмы должны быть не только эффективными, но и быстрыми. В настоящей работе нами представлены результаты по оценке временных затрат используемых алгоритмов, в том числе с применением векторизации программного кода посредством SIMD-инструкций [9] и распараллеливания

[1]Плоскость реакции – это плоскость, проходящая через ось пучка и вектор прицельного параметра, соединяющий центры провзаимодействовавших ядер мишени и пучка.

задач между ядрами процессора, реализованных с помощью программных сред OpenMP (Open Multi-Processing) [10], OpenCL (Open Computing Language) [11] и библиотеки TBB (Threading Building Blocks) [12].

2. Компьютерные технологии и средства для высокопроизводительных вычислений

В последние годы интенсивно развиваются компьютерные технологии и средства для проведения высокопроизводительных вычислений на современных вычислительных архитектурах. Ниже приведено краткое описание тех технологий и средств, которые используются для ускорения алгоритмов, предназначенных для селекции и реконструкции распадов $J/\psi \to e^+e^-$.

2.1. SIMD-инструкции

Для организации параллельной обработки данных на современных процессорах (CPU) используются так называемые SIMD-инструкции [9]. SIMD — это аббревиатура от Single Instruction Multiple Data — одна команда для многих данных. Иногда такую операцию называют *векторной* обработкой, т.к. основной элемент SIMD-инструкций — это векторный регистр, позволяющий проводить арифметические операции параллельно (одновременно) над всеми скалярными данными, занесёнными в регистр.

SIMD-инструкции поддерживаются практически всеми существующими процессорами. В частности, в процессорах Intel они реализованы посредством технологий SSE (Streaming SIMD Extensions) и AVX (Advanced Vector Extensions).

Прежде чем проводить какую-либо операцию над числами, используя SIMD-инструкции, их нужно занести (упаковать) в векторный регистр. В SSE используются специальные регистры процессора с разрядностью в 128 бит. Эти регистры могут содержать данные любого типа, которые могут быть размещены в 128 битах. Например, четыре числа с плавающей точкой одинарной точности (float): x= $\{x_0, x_1, x_2, x_3\}$. Новейшие CPU уже имеют 256- и 512-битные регистры (набор инструкций AVX), в каждый из которых можно разместить, соответственно, 8 или 16 чисел.

На рис. 2 схематично показано, как четыре последовательные операции умножения (scalar) можно заменить с помощью одной SIMD-инструкции (SIMD).

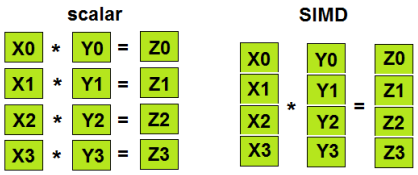

Рис. 2. Выполнение четырёх последовательных операций умножения с помощью одной SIMD-инструкции

Применение заголовочных файлов позволяет перегружать арифметические и логические операторы, используемые SIMD-инструкциями, и делает код программы компактным и легко читаемым. К примеру, процедура умножения векторов x и y (см. рис. 2) с использованием SIMD-инструкции на языке C++ будет выглядеть следующим образом:

```
_m128 z = _mm_mul_ps(x,y);
```

в то время как код функции, использующей заголовочный файл

```
fvec z = x*y;
```

с перезагружаемым оператором умножения

```
friend fvec operator*(const fvec &x, const fvec &y) {
  return _mm_mul_ps(x,y); }
```

смотрится и читается очень просто.

Вследствие простоты реализации данный подход обеспечивает гибкость по отношению к различным архитектурам центральных процессоров. Она достигается путём подключения соответствующих заголовочных файлов к неизменяемому кода программы. В частности, на этапе отладки программного кода удобно использовать заголовочный файл для скалярной версии программы.

Так как время выполнения операций над скалярными и векторными величинами на CPU одинаково, то, используя SIMD-инструкции, можно получить ускорение вычислений, пропорциональное длине вектора, т.е. 4 для SSE-технологии и 8 или 16 для технологии AVX.

2.2. Многопоточность

Многопоточность (multithreading) — модель программирования и исполнения кода программы, позволяющая нескольким потокам выполняться в рамках одного процесса [13]. Она предоставляет разработчикам удобную абстракцию параллельного выполнения процесса (программы) на компьютерных системах, имеющих несколько процессоров, на процессоре с несколькими ядрами, или на кластере машин.

Дальнейший этап в развитии данного подхода — это технология гиперпоточности (Hyper-Threading Technology, HTT), разработанная компанией Intel [13], которая поддерживается практически на всех современных многоядерных процессорах. В технологии HTT каждое физическое ядро может хранить состояние сразу двух потоков, что для операционной системы выглядит как наличие двух логических ядер. Это позволяет более эффективно использовать ресурсы отдельного физического ядра, добиваясь тем самым уменьшения времени, затрачиваемого на выполнение конкретной программы.

2.3. Распараллеливание на уровне инструкций

Следует также упомянуть ещё об одной технологии ускорения выполнения программы — это технология ILP (Instruction Level Parallelism) [14] — распараллеливание на уровне инструкций. Обычно инструкции в программе выполняются последовательно и в том порядке, как написал их разработчик. Технология ILP позволяет менять порядок выполнения инструкций, распределять их по группам, которые будут обрабатываться процессором параллельно, без изменения результатов работы программы. При этом расположение инструкций в наиболее удобной для процессора последовательности выполняет компилятор, а не сам программист.

2.4. Используемые среды и библиотека

Для распараллеливания рассмотренных в настоящей работе алгоритмов использовались библиотека TBB и программные среды OpenMP и OpenCL.

TBB (Intel Threading Building Blocks) — кросс-платформенная библиотека шаблонов C++ [12]. Предлагая богатый функционал для распараллеливания задач, она позволяет реализовывать параллельные алгоритмы на языке высокого уровня, абстрагируясь от деталей архитектуры конкретной машины. Библиотека TBB скрывает низкоуровневую работу с потоками, упрощая тем самым процесс разработки параллельного кода. Все операции трактуются как «задачи», которые динамически распределяются между ядрами процессора. При этом достигается эффективное использование многопоточности, а также кэш-памяти.

OpenMP (Open Multi-Processing) — это программная среда, включающая совокупность директив компилятора, библиотечных процедур и переменных окружения, которые предназначены для программирования многопоточных приложений на многопроцессорных системах с общей памятью [10]. Участки кода (программы), выполняемые потоками параллельно, выделяются с помощью специальных директив препроцессора – прагм. Количество создаваемых потоков может

регулироваться как в самой программе путём вызова библиотечных процедур, так и извне при помощи переменных окружения.

OpenCL (Open Computing Language) — программная среда для написания компьютерных программ, связанных с параллельными вычислениями на различных графических (GPU) и центральных процессорах (CPU) [11]. OpenCL предоставляет разработчикам программ доступ ко всем ресурсам гетерогенных вычислительных платформ, позволяет создавать универсальный код, избавляя от надобности поддерживать разные версии программы для различных вычислительных процессоров. OpenCL также позволяет использовать обе опции современных процессоров: векторизацию и распараллеливание между ядрами.

3. Реконструкция траекторий и восстановление импульсов заряженных частиц с помощью STS

Для реконструкции траекторий и восстановления импульсов заряженных частиц используется система координатных детекторов STS [15]. С её помощью регистрируются координаты (хиты) мест пересечения заряженной частицей плоскостей станций STS, расположенных на расстояниях z_i от мишени: $i = 1, \ldots, 8$ — порядковый номер станции. В эксперименте CBM используется декартова система координат с началом в центре мишени, ось OZ совпадает с направлением падающего пучка, ось OX лежит в горизонтальной плоскости, а ось OY ориентирована вертикально вверх.

Траектория частицы в каждой точке z_i характеризуется пятью параметрами, образующими вектор состояния

$$\mathbf{r}(x_i, y_i, t_{xi}, t_{yi}, q/p), \tag{1}$$

где (x_i, y_i) — координаты и (t_{xi}, t_{yi}) — тангенсы углов наклона трека при заданной z-координате, соответственно, в плоскостях XOZ и YOZ, а q/p — отношение заряда q частицы к её импульсу p. Задача реконструкции треков заключается в поиске хитов, отвечающих траекториям отдельных частиц, и оценке параметров найденных траекторий.

3.1. Алгоритм распознавания треков

В основу алгоритма распознавания треков положена концепция клеточного автомата (КА) [16]. Алгоритм включает два этапа. Вначале формируются наборы элементов треков — это так называемые трек-сегменты (триплеты), содержащие хиты трёх соседних станций STS. На следующем этапе из наборов элементов треков строятся трек-кандидаты. Данный алгоритм обеспечивает быстрое нахождение треков со средней эффективностью, близкой к 90%. При этом эффективность восстановления треков, отвечающих частицам от распадов $J/\psi \to e^+e^-$, превышает 95% [16].

3.1.1. Масштабируемость алгоритма распознавания треков

Для оценки производительности алгоритма были подготовлены два набора модельных данных, отвечающих соударениям Au+Au при энергии 25 ГэВ/нуклон:

1) центральные события (им отвечают ядро-ядерные соударения, в которых параметр соударения равен нулю): из-за большого количество образующихся вторичных заряженных частиц (в среднем 720 треков в одном соударении) такие события соответствуют максимально сложному сценарию в задаче распознавания треков;
2) так называемые MB (Minimum Bias) события (в основном такие события будут регистрироваться в эксперименте CBM): в MB-событиях параметр соударения

сталкивающихся ядер разыгрывается случайным образом; в этих событиях наблюдается в среднем 150 вторичных заряженных частиц.

Ввиду большого числа заряженных частиц, регистрируемых системой STS в одном ядро-ядерном соударении, распознавание треков на основе КА относится к сложной комбинаторной задаче, которая плохо поддаётся распараллеливанию на уровне SIMD-инструкций. Усилия по оптимизации алгоритма на этом уровне позволили добиться его ускорения всего в 2 раза.

Следующий шаг по ускорению алгоритма – это распараллеливание вычислений на многоядерных высокопроизводительных серверах. Для решения данной задачи использовался высокопроизводительный сервер cuda.jinr.ru ЛИТ ОИЯИ. Его структура представлена на рис. 3.

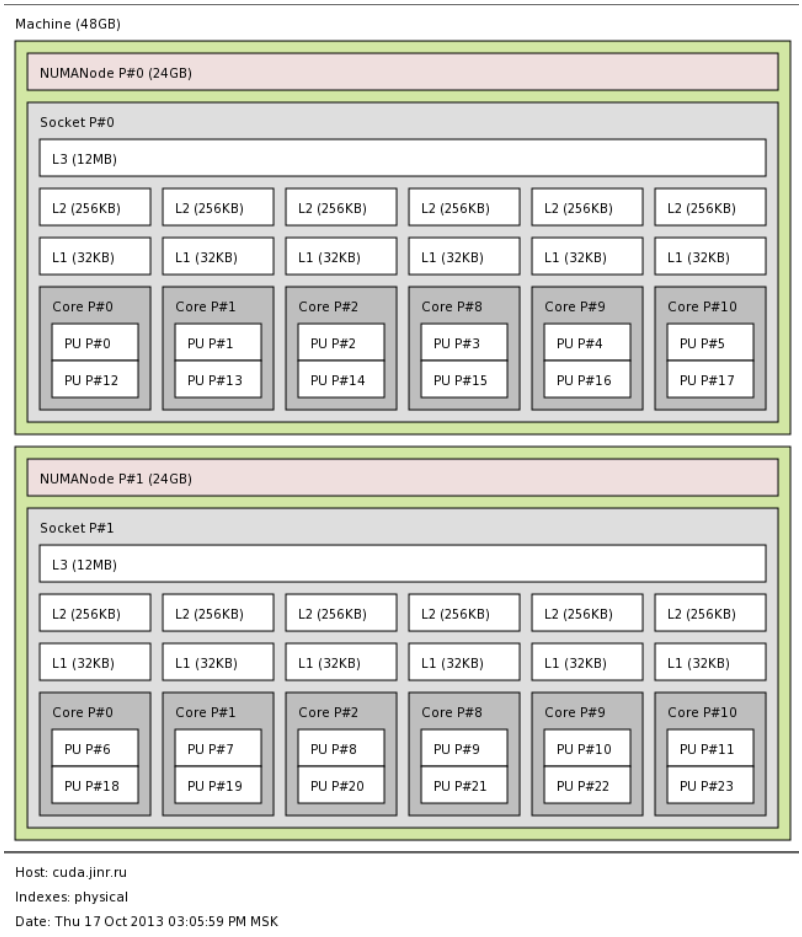

Рис. 3. Структура сервера cuda.jinr.ru

Сервер оснащён двумя процессорами Intel Xeon X5660, каждый процессор включает 6 ядер с частотой 2,8 ГГц, 12 Мбайт кэш-памяти третьего уровня, которая делится между ядрами, и 24 Гбайт оперативной памяти. Каждому физическому ядру отвечает два логических ядра, 32 Кбайт кэш-памяти первого уровня и 256 Кбайт второго. За счёт применения технологии гиперпоточности на сервере можно одновременно запускать 24 потока.

При запуске вычислений на одном ядре сервера среднее время, затрачиваемое алгоритмом на распознавание треков в одном событии, составило 220 мс и 25 мс для центральных и МВ-событий соответственно [16].

Для организации обработки событий параллельно на разном числе логических ядер сервера использовалась библиотека TBB. Распараллеливание алгоритма проводилось на уровне событий: на каждое ядро отправлялось на обработку

данные о 100 МВ-событиях. На рис. 4 приведён результат масштабируемости алгоритма (И. Кулаков, частное сообщение от 14.11.2013) — зависимость числа обработанных за 1 сек событий от числа включаемых в обработку ядер. Из рисунка видно, что эта зависимость близка к линейной.

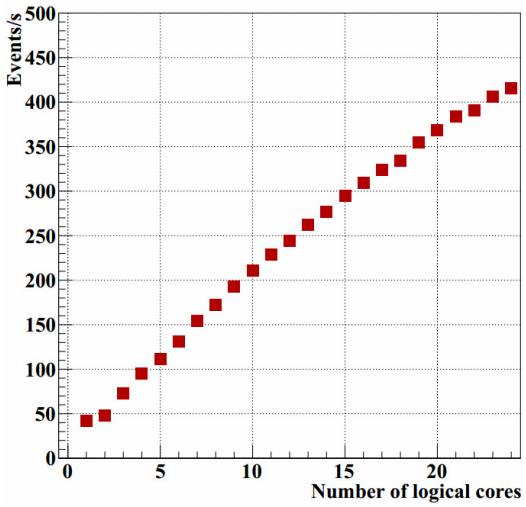

Рис. 4. Масштабируемость алгоритма распознавания треков для МВ-событий

3.2. Определение характеристик найденных треков с помощью фильтра Калмана

На следующем шаге проводится аппроксимация измерений найденных треков с целью уточнения их пространственных характеристик и восстановления импульсов заряженных частиц. Эта процедура выполняется с помощью итерационного процесса, реализованного на основе фильтра Калмана [17]. Стартуя с некоторого начального приближения для вектора состояния $\mathbf{r}$ и последовательно переходя от хита к хиту, происходит корректировка вектора $\mathbf{r}$ с учётом положения каждого хита. При этом принимаются во внимание измерительные ошибки координатных детекторов, многократное рассеяние в веществе станций и неоднородность магнитного поля. По завершению вычислений на выходе алгоритма получаем оптимальную оценку для координат и направления траектории частицы и для её импульса. Точность восстановления импульса составляет величину порядка 1 % [18].

3.2.1. Масштабируемость алгоритма определения характеристик найденных треков

Так как вектора состояния $\mathbf{r}$, отвечающие отдельным трекам, не зависят друг от друга, использование SIMD-инструкций на этом этапе позволяет добиться максимального возможного ускорения вычислений, равного 4 [16]. При этом скорость работы алгоритма на одном ядре сервера cuda.jinr.ru составила 0,5 мкс/трек.

Результаты масштабируемости алгоритма, используя для распараллеливания вычислений между ядрами CPU среду OpenCL, представлены на рис. 5 [18]. При работе с OpenCL у пользователя нет возможности управлять загрузкой ядер процессора, поэтому ядра заполняются согласно системной нумерации. В этой связи на сервере cuda.jinr.ru вначале запускается по одному потоку на каждом из физических ядер CPU (ядра от 1 до 12: см. рис. 5), а затем, используя технологию гиперпоточности, к вычислениям подключаются вторые логические ядра. Из

рис. 5 видно, что на обоих стадиях подключения ядер CPU производительность алгоритма в зависимости от числа задействованных в обработку логических ядер растёт линейно. При этом максимальное число треков, которое удаётся обработать за 1 мкс, составило 27 [18].

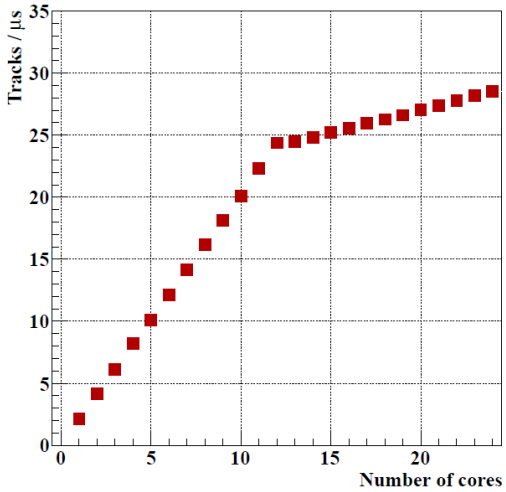

Рис. 5. Производительность алгоритма определения характеристик треков заряженных частиц в зависимости от числа задействованных логических ядер на двух процессорах Intel Xeon X5660

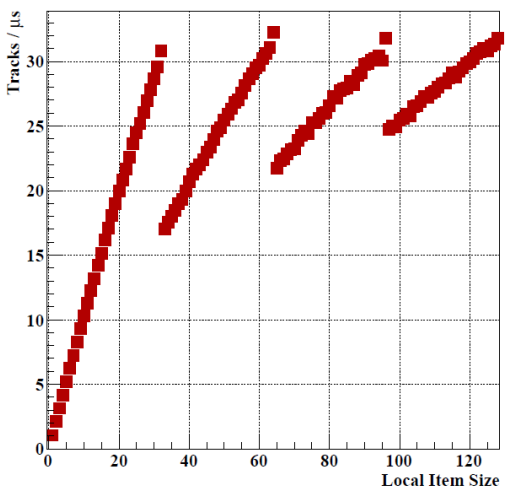

Рис. 6. Производительность алгоритма определения характеристик треков заряженных частиц в зависимости от числа треков в рабочей группе на графическом процессоре NVidia GTX 448

Для дополнительного увеличения производительности рассматриваемого алгоритма также использовалась установленная на сервере cuda.jinr.ru графическая карта NVidia GTX 448 [19]. Процессор данной карты содержит 448 CUDA-ядер. При запуске программы на графической карте в среде OpenCL весь набор обрабатываемых треков распределяется между рабочими группами. Каждая такая группа обслуживается одним потоковым мультипроцессором, содержащим 32 физических ядра.

На рис. 6 показана производительность алгоритма в зависимости от числа треков в рабочей группе [18]. Видно, что максимальная производительность, равная 33 трека/мкс, достигается в том случае, когда число треков, содержащихся в группе, кратно числу ядер в мультипроцессоре. В противном случае ресурсы, выделяемые на одну рабочую группу, используются неоптимально.

Таким образом, суммарная производительность алгоритма определения характеристик треков заряженных частиц для существующей комплектации сервера cuda.jinr.ru может составить около 60 треков/мкс.

4. Алгоритмы селекции электронов/позитронов с помощью детектора RICH

Детектор RICH используется для селекции электронов/позитронов в диапазоне импульсов от 0,5 ГэВ/с до 15 ГэВ/с [20]. При движении заряженной частицы в радиаторе детектора со скоростью, превышающей скорость распространения света в данной среде, образуется черенковское излучение. Оно регистрируется фотодетектором в виде колец.

Для селекции электронов/позитронов необходимо решить следующие задачи:

– реконструировать кольца черенковского излучения,

– связать восстановленные кольца с треками, реконструированными STS-детектором,
– идентифицировать заряженную частицу, зарегистрированную детекторами STS и RICH, используя информацию об импульсе и радиусе кольца.

4.1. Реконструкция колец черенковского излучения

Поиск колец, определение их центра и радиуса выполняется с помощью разработанного в коллаборации CBM алгоритма распознавания колец [21]. Данный алгоритм включает два этапа:

1) поиск всех колец-кандидатов, используя локальное преобразование Хафа, и
2) отбор из них с помощью искусственной нейронной только таких колец, которые удовлетворяют определённым критериям отбора.

Отметим, что из-за особенностей конструкции детектора RICH и неизбежных оптических искажений кольца имеют форму эллипса. Рассматриваемый алгоритм позволяет реконструировать до 93% колец.

После распознавания и реконструкции всех колец необходимо каждое кольцо «связать» с конкретным треком, найденным в STS. Для этого трек из STS экстраполируется до зеркал RICH-а и отражается на плоскость фотодетектора. Определённая таким образом точка пересечения треком плоскости фотодетектора используется для связывания трека с одним из колец. При этом используется алгоритм, который для данного кольца ищет ближайший к нему трек. Применение ограничения на расстояние от центра кольца до хита от трека не более 1 см позволяет существенно сократить ошибочные связи колец с треками.

4.2. Масштабируемость алгоритма реконструкции колец

Алгоритм реконструкции колец содержит большое число условных операций $if-then-else$, которые не поддерживаются SIMD-инструкциями. В этой связи, за счёт векторизации данного алгоритма скорость его выполнения удаётся повысить всего лишь в 2 раза.

В работе [21] распараллеливание алгоритма на уровне событий проводилось с помощью библиотеки TBB. Для этого использовался компьютер с двумя процессорами Intel Core i7, каждый из которых содержал 4 ядра с тактовой частотой 2,66 ГГц. Используя технологию гиперпоточности, на компьютере можно было одновременно запускать до 16 потоков. Для тестирования алгоритма был подготовлен набор модельных данных, отвечающих центральным соударениям Au+Au при энергии 25 ГэВ/нуклон. Причём в каждое такое событие добавлялось 10 электронов. Среднее число колец в одном событии равнялось 80.

На рис. 7 представлены полученные результаты масштабируемости алгоритма реконструкции колец [21]. Видно, что в случае обработки небольшого числа событий имеют место дополнительные расходы времени (*overhead*) на распределение процессов между ядрами. С увеличением числа событий, посылаемых для обработки на одно ядро CPU, наблюдается линейный рост производительности алгоритма в зависимости от числа ядер, включаемых в обработку. Среднее время, затрачиваемое алгоритмом на обработку одного центрального события на одном ядре компьютера, составляет 6,25 мс (около 80 мкс/кольцо), т.е. при максимальной загрузке всех ядер компьютера можно обрабатывать до 1800 центральных событий в секунду.

4.3. Селекция электронов/позитронов

Для селекции электронов/позитронов используются зависимости большой и малой полуосей эллипса от импульсов регистрируемых частиц (рис. 8). Все заряженные частицы, давшие одновременно отсчёты в обоих коридорах, отмеченных красными линиями на рис. 8, идентифицируются RICH-м как электроны/позитроны (см. детали в [22]).

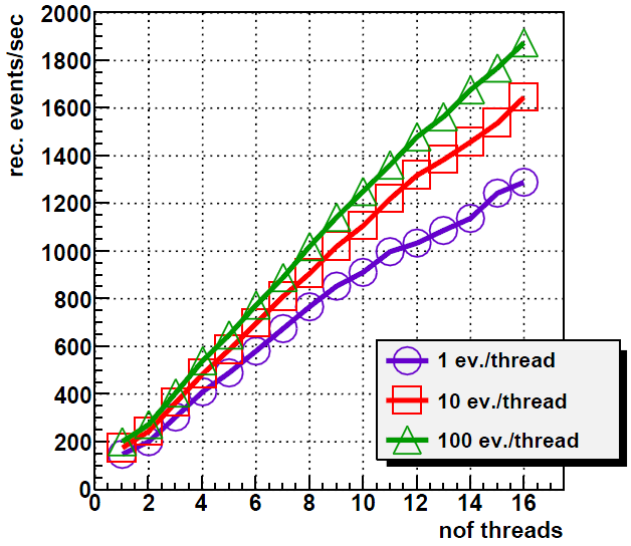

Рис. 7. Масштабируемость алгоритма реконструкции колец

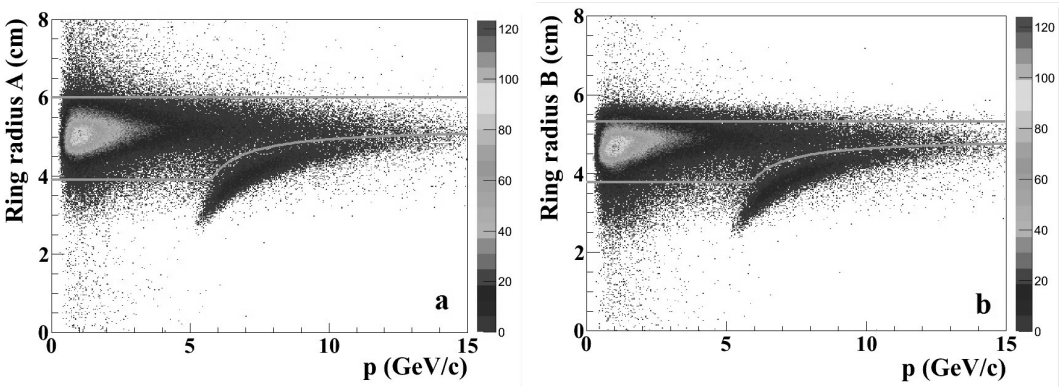

Рис. 8. Зависимости большой (а) и малой (б) полуосей эллипса от импульсов регистрируемых частиц

Ввиду простоты рассматриваемой процедуры её распараллеливание между ядрами процессора нецелесообразно, так как время, затрачиваемое на формирование потоков и распределение вычислений между ними, сопоставимо со временем выполнения самого алгоритма.

5. Алгоритмы селекции электронов/позитронов с помощью TRD

Для селекции электронов/позитронов и последующего подавления пионов, оставшихся в анализируемых данных после применения RICH, дополнительно используется многослойный детектор переходного излучения TRD. С его помощью решаются следующие задачи:

– распознать и реконструировать траектории заряженных частиц, зарегистрированных координатными плоскостями TRD,
– связать восстановленные треки с теми треками из STS-детектора, которые были отобраны с помощью RICH,
– используя потери энергии в станциях TRD, давших вклад в реконструированный трек, идентифицировать зарегистрированную заряженную частицу.

5.1. Реконструкция траекторий заряженных частиц

Для поиска и реконструкции треков используются метод слежения по треку и фильтр Калмана, позволяющие находить до 95% треков [23]. В качестве целеуказаний для поиска треков в TRD используются треки, найденные в детекторе STS. В этой связи все треки, восстановленные в TRD, оказываются однозначно связанными с конкретными STS-треками.

Рассматриваемый алгоритм характеризуется большой комбинаторикой и сложностью. На обработку одного центрального события, содержащего в среднем более 500 треков, требуется около 0,8 с, что очень много. К сожалению, векторизация и распараллеливание данного алгоритма пока не проводились[1].

5.2. Идентификация заряженных частиц

С каждым из реконструированных треков ассоциируется набор потерь энергий $\{\Delta E_{i=1}^{n}\}$, оставленных заряженной частицей в n модулях TRD: в стандартной версии детектора $n = 12$. Задача идентификации частицы состоит в определении к какому из распределений (в нашем случае электронов/позитронов или пионов) эти потери относятся.

В работе [22] были рассмотрены методы решения задачи идентификации регистрируемых частиц на основе искусственной нейронной сети (ИНС) и непараметрического критерия согласия ω_n^k, а также детально исследованы преимущества и недостатки данных методов. Проведённый нами анализ показал, что оба метода обладают одинаковой мощностью; при этом метод на основе ИНС имеет ряд ограничений, характерных для нейронных сетей рассматриваемого типа. Критерий ω_n^k лишён таких недостатков и имеет простую программную реализацию. Поэтому ему было отдано предпочтение для регистрации распадов $J/\psi \to e^+e^-$ в режиме реального времени эксперимента.

Для применения критерия ω_n^k требуется вычислить статистику:

$$\omega_n^k = -\frac{n^{\frac{k}{2}}}{k+1} \sum_{j=1}^{n} \left\{ \left[\frac{j-1}{n} - \phi(\lambda_j) \right]^{k+1} - \left[\frac{j}{n} - \phi(\lambda_j) \right]^{k+1} \right\}, \tag{2}$$

где k — степень критерия, n — объем выборки, $\phi(\lambda_j)$ — значения функции распределения Ландау от упорядоченной выборки величин λ: $\lambda_1 \leqslant \cdots \leqslant \lambda_j \leqslant \cdots \leqslant \lambda_n$. Переменная λ_i связана с потерей энергии частицей ΔE_i в i-ом радиаторе TRD следующим соотношением:

$$\lambda_i = \frac{\Delta E_i - \Delta E_{mp}^i}{\xi_i} - 0.225, \quad i = 1, \ldots, n, \tag{3}$$

где ΔE_{mp}^i — наиболее вероятная потеря энергии пионов, а $\xi_i = \dfrac{1}{4,02}\,\mathrm{FWHM}$ (Full Width on Height Medium — полная ширина на половине высоты) для потерь энергии пионов в i-ом радиаторе TRD.

Для вычисления значений ΔE_{mp}^i и ξ_i распределение ионизационных потерь энергии пионов фитировалось функцией плотности логнормального распределения [22].

В работе [24] была предложена модификация критерия ω_n^k, учитывающая особенности распределения потерь энергии электронов/позитронов в радиаторах

[1] В настоящее время ведётся разработка альтернативного подхода по поиску треков в TRD, основанного на модели КА. Ожидается, что новый алгоритм, не уступая существующему в эффективности, будет более надёжным и быстрым.

TRD и позволившая существенно повысить мощность критерия. Было также показано, что максимальная мощность по идентификации электронов/позитронов достигается при использовании статистики ω_6^k со степенью k, равной 4 [22].

5.3. Масштабирование алгоритма идентификации частиц

Среднее время работы скалярных версий алгоритмов на основе ИНС и критерия ω_n^k на одном ядре сервера cuda.jinr.ru составляет соответственно $2,4$ мкс/трек и $1,7$ мкс/трек. Таким образом, алгоритм на основе критерия ω_n^k оказался в $1,4$ раза быстрее, чем алгоритм с использованием ИНС.

Для ускорения алгоритма ω_n^k вначале была проведена частичная векторизация кода с использованием SIMD-инструкций. Заметим, что при вычислении статистики (2) требуется упорядочить значения переменной λ (3). На данный момент процедура сортировки не векторизована. Несмотря на это, за счёт оптимизации кода удалось добиться ускорения алгоритма в 3,5 раза.

Распараллеливание алгоритма между ядрами CPUs проводилось на сервере cuda.jinr.ru, используя среду программирования OpenMP. При этом загрузка ядер двух процессоров сервера происходит в следующем порядке: вначале загружаются ядра первого процессора, а затем второго.

На рис. 9 представлена зависимость производительности алгоритма идентификации заряженных частиц на основе критерия ω_n^k от числа запущенных потоков. Видно, что эта зависимость носит линейный характер; при этом максимальная производительность составила 31 трек/мкс.

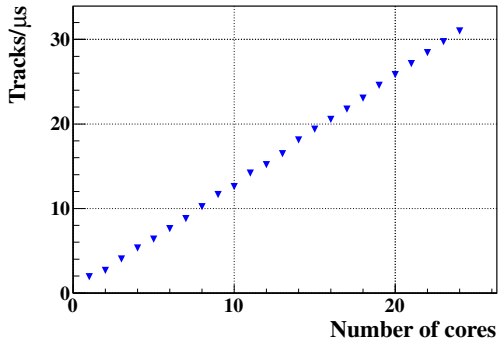

Рис. 9. Зависимость производительности алгоритма идентификации заряженных частиц на основе критерия ω_n^k от числа запущенных потоков

6. Алгоритмы селекции электронов/позитронов с помощью детектора TOF

В дополнение к RICH и TRD для селекции электронов/позитронов используется детектор измерения времени пролёта TOF [25]. С помощью этого детектора измеряется время t, за которое частица (идентифицированная ранее детекторами RICH и TRD как электрон/позитрон) пролетает расстояние l от мишени до плоскости RPC: зная импульс частицы p, можно вычислить её массу m [22].

Для идентификации частицы с помощью TOF используется зависимость квадрата её массы m^2 от импульса p: см. рис. 10 [22]. В качестве электронов/позитронов принимаются такие частицы, которые дают отсчёт ниже показанной на рис. 10 сплошной пороговой линии.

Аналогично рассмотренному выше методу порогов, используемому для идентификации электронов/позитронов с помощью RICH, распараллеливание данной процедуры нецелесообразно.

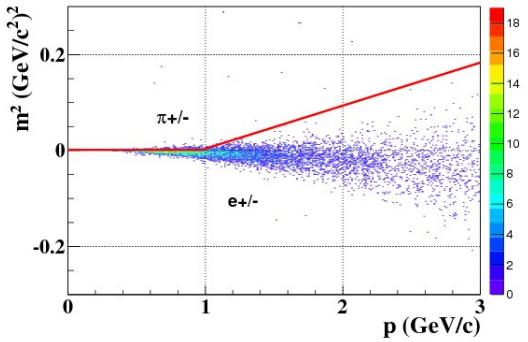

Рис. 10. Зависимость квадрата массы от импульса для частиц, идентифицированных с помощью детекторов RICH и TRD как электроны/позитроны

7. Формирование кандидатов в J/ψ-мезоны

Для реконструкции J/ψ-мезонов, распадающихся по диэлектронному каналу, используется специализированный пакет KFParticle [8]. Этот пакет предназначен для поиска и реконструкции короткоживущих частиц по продуктам их распада.

В основу пакета KFParticle положен фильтр Калмана. В рамках этого подхода распадную частицу можно описать вектором состояния $\mathbf{r}_{J/\psi}(x, y, z, p_x, p_y, p_z, E)$: (x, y, z) – координаты вершины распада J/ψ-мезона, (p_x, p_y, p_z) – три компоненты его импульса, а E – полная энергия. Для реконструкции вектора $\mathbf{r}_{J/\psi}(\cdot)$ на вход пакета KFparticle подаются наборы векторов состояния треков заряженных частиц, восстановленных с помощью детектора STS (смотри Главу 3) и идентифицированных с помощью детекторов RICH, TRD и TOF как электроны и позитроны. При этом рассматриваются только такие дочерние частицы, которые испускаются из окрестности первичной вершины и имеют поперечный импульс больший 1 ГэВ/с [26]. Путём комбинирования всех электронов со всеми позитронами из диэлектронных пар, удовлетворяющих определённым критериям отбора, формируется набор кандидатов в J/ψ-мезоны [8].

Пакет KFparticle был оптимизирован и частично векторизован, что позволило получить коэффициент ускорения $2,5$. Это можно считать неплохим результатом, если учесть комбинаторику, связанную с перебором очень большого количества дочерних частиц. Скорость реконструкции J/ψ-мезонов при использовании одного ядра процессора Intel Xeon E7-4860 составила $1,4$ мс для одного MB и $10,5$ мс для одного центрального AuAu-соударения при энергии 25 ГэВ/нуклон [27].

8. Обсуждение результатов и Заключение

В настоящей работе проведён анализ производительности алгоритмов, используемых в эксперименте CBM для реконструкции распадов $J/\psi \to e^+e^-$. Цель данного анализа – это оценить возможность проведения такой реконструкции в реальном времени эксперимента. Для ускорения работы алгоритмов использовались два подхода: векторизация кода и распараллеливание алгоритмов.

В табл. 1 представлены коэффициенты ускорения алгоритмов, полученные путём векторизации кода, т.е. за счёт использования SIMD-инструкций. Заметим, что имеются определённые резервы на этом пути ускорения вычислений, так как некоторые алгоритмы, такие как алгоритм поиска и реконструкции траекторий

Таблица 1
Коэффициенты ускорения алгоритмов, полученные за счёт векторизации программного кода

STS: КА поиск треков	STS: фильтр Калмана	RICH: рек. колец	TRD: критерий ω_n^k	пакет KFParticle
2	4	2	3,5	2,5

заряженных частиц с помощью координатных детекторов TRD, вообще не подвергались векторизации, а другие, в частности пакет KFParticle, были векторизованы не полностью. В случае максимально возможной оптимизации и векторизации всех алгоритмов суммарный фактор ускорения вычислений может быть заметно выше.

Согласно предварительным оценкам доля центральных соударений в реальном эксперименте не должна превышать 1%. В этой связи для определения производительности алгоритмов использовались события, отвечающие смеси из центральных (вклад 1%) и MB (вклад 99%) AuAu-соударений при энергии 25 ГэВ/нуклон. Кроме того, чтобы исключить зависимость алгоритмов от множественности частиц, рождающихся в одном AuAu-соударении, подсчитывалось среднее время Δt, затрачиваемое конкретным алгоритмом на обработку одной траектории. Для этого использовалась следующая формула:

$$\Delta t = \frac{t_{\mathrm{mbias}}}{N_{\mathrm{mbias}}} \cdot 0,01 + \frac{t_{\mathrm{centr}}}{N_{\mathrm{centr}}} \cdot 0,99, \tag{4}$$

где t_{mbias} — среднее время, затрачиваемое алгоритмом на обработку одного MB-события, а t_{centr} — одного центрального события; N_{mbias} — среднее число реконструированных треков в одном MB-событии, а N_{centr} — в одном центральном событии.

В табл. 2 приведены средние времена Δt, затрачиваемые разными алгоритмами (в мкс/трек или мкс/кольцо), используемых для реконструкции распадов $J/\psi \to e^+e^-$. Заметим, что приведённые результаты относятся к SIMD-версиям алгоритмов (исключая алгоритм поиска и реконструкции треков в TRD) и получены с использованием одного логического ядра CPU.

Таблица 2
Средние времена Δt (в мкс/трек или мкс/кольцо), затрачиваемые SIMD-версиями алгоритмов, используемых для реконструкции распадов $J/\psi \to e^+e^-$

STS: КА поиск треков	STS: фильтр Калмана	RICH: рек. колец	TRD: поиск, рек. треков	TRD: кр. ω_n^k	пакет KFParticle
164,5	0,5	49	1390	0,5	9,15

Из приведённой таблицы видно, что время, затрачиваемое алгоритмом TRD на поиск и реконструкцию траекторий заряженных частиц, во много раз превышает суммарное всех остальных алгоритмов. В разделе 5.1 нами отмечалось, что в настоящее время ведётся разработка нового алгоритма, основанного на модели КА. Ожидается, что скалярная версия этого алгоритма будет существенно более быстрой. Кроме того, проведение векторизации кода позволит добиться дополнительного ускорения нового алгоритма. Следует заметить, что результаты работы рассматриваемого алгоритма используются в процедуре идентификации электронов/позитронов с помощью критерия ω_n^k [22].

Все рассмотренные выше алгоритмы были адаптированы в программных средах OpenMP, OpenCL и с использованием библиотеки TBB для проведения параллельных вычислений на высокопроизводительных гибридных серверах, построенных на основе многоядерных CPU и GPU. Для всех алгоритмов был получен линейный рост производительности в зависимости от числа включённых в обработку ядер.

Фактор ускорения, получаемый за счёт распараллеливания отдельного алгоритма между ядрами CPU, можно вычислить по следующей формуле:

$$F = F_{\text{simd}} \cdot HW \cdot N_{\text{socets}} \cdot N_{\text{cores}}, \tag{5}$$

где F_{simd} — фактор ускорения, получаемый за счёт векторизации кода; N_{socets} — число процессоров; N_{cores} — количество физических ядер в одном процессоре; HW — ускорение, получаемое за счёт технологии гиперпоточности:

$$HW = \frac{P_{all}}{P_1 \cdot N_{\text{socets}} \cdot N_{\text{cores}}},$$

где P_{all} — максимальная производительность алгоритма с учётом технологии гиперпоточности; P_1 — производительность алгоритма на одном логическом ядре процессора.

Вычислим, для примера, фактор ускорения F для критерия ω_n^k. Соответствующий алгоритм тестировался нами на сервере cuda.jir.ru ЛИТ ОИЯИ, содержащем два процессора Intel Xeon X5660: каждый CPU имеет 6 физических ядер, каждое из которых включает два логических ядра. Так как производительность алгоритма на одном логическом ядре составила $1,94$ трек/мкс (см. рис. 9), то коэффициент $HW = 31/(1,94 \cdot 2 \cdot 6) \simeq 1,33$. В результате для фактора ускорения F получим:

$$F = 3,5 \cdot 1,33 \cdot 2 \cdot 6 \simeq 56,$$

т.е. за счёт векторизации кода и распараллеливания обработки на многоядерном сервере cuda.jinr.ru, по сравнению со скалярной версией алгоритма, удалось ускорить обработку в 56 раз.

Таким образом, проведённый анализ производительности алгоритмов позволил оценить возможность ускорения обработки данных за счёт использования средств высокопроизводительных вычислений, а также выявить «слабые» места в цепочке методов (алгоритм поиска и реконструкции треков в TRD), над которыми ещё предстоит работать.

Литература

1. *Matsui T., Satz H.* J/ψ Suppression by Quark-Gluon Plasma Formation // Physics Letters B. — 1986. — Vol. 178, No 14. — Pp. 416–422.
2. *Alessandro B. et al.* A New Measurement of J/ψ Suppression in Pb-Pb Collisions at 158-GeV per Nucleon // European Physical Journal C. — 2005. — Vol. 39, No 3. — Pp. 335–345.
3. *Leitch M.* RHIC Results on J/ψ // Journal of Physics G: Nuclear and Particle Physics. — 2007. — Vol. 34, No 8. — Pp. 453–462.
4. The CBM Collaboration. CBM Compressed Baryonic Matter Experiment. Technical Status Report: Techrep / GSI. — Darmstadt, 2005. — `http://www.gsi.de/onTEAM/dokumente/public/DOC-2005-Feb-447e.html`.
5. The CBM Collaboration // Nuclear Physics A. — 2013. — Vol. 904–905. — Pp. 1059–1062.
6. The CBM Physics Book: Compressed Baryonic Matter in Laboratory Experiments / B. Friman, C. Hohne, J. Knoll et al. // Lecture Notes in Physics. — 2011. — Vol. 814.

7. *Дереновская О. Ю., Васильев Ю. О.* Реконструкция J/ψ в диэлектронном канале распада при энергиях SIS100 в эксперименте CBM // Письма в ЭЧАЯ. — 2013. — Т. 10, № 5 (182). — С. 694–705.
8. *Gorbunov S., Kisel I.* Reconstruction of Decayed Particles Based on the Kalman Filter // CBM-SOFT-note-2007-003. — 2007.
9. IA-32 Intel Architecture Optimization Reference Manual. — 2005.
10. OpenMP. — `http://openmp.org`.
11. OpenCL. — `http://www.khronos.org/opencl`.
12. Threading Building Blocks. — `http://threadingbuildingblocks.org`.
13. Intel Hyper-Threading Technology, Technical User's Guide. — 2003.
14. *Белевенцев А., Кувырков М., Мельник Д.* Использование параллелизма на уровне команд в компиляторе для Intel Itanium // Труды ИСП РАН. — 2006. — Т. 9. — С. 9–22.
15. Silicon Tracking System(STS). Technical Design Report for the CBM: Techrep / GSI. — Darmstadt, 2012. — `http://repository.gsi.de/record/54798`.
16. Анализ эффективности и производительности алгоритма распознавания треков в STS-детекторе эксперимента CBM на многоядерном сервере ЛИТ ОИЯИ / И. С. Кулаков, С. А. Багинян, В. В. Иванов, П. И. Кисель // Письма в ЭЧАЯ. — 2013. — Т. 10, № 2 (179). — С. 253–267.
17. Fast SIMDized Kalman Filter Based Track Fit / S. Gorbunov, U. Kebschull, I. Kisel et al. // Computer Physics Communications. — 2008. — Vol. 178. — Pp. 374–383.
18. Метод фильтра Калмана для реконструкции траекторий заряженных частиц в эксперименте CBM и его параллельная реализация на многоядерном сервере ЛИТ ОИЯИ / Т. О. Аблязимов, М. В. Зызак, В. В. Иванов, П. И. Кисель // Вестник РУДН. Серия «Математика. Информатика. Физика». — 2014. — № 2. — С. 191–196.
19. NVidia GTX 480. — `http://de.geforce.com/hardware/desktop-gpus/geforce-gtx-480/architectur`.
20. Ring Imaging Cherenkov (RICH) Detector. Technical Design Report for the CBM: Techrep / GSI. — Darmstadt, 2013. — `http://repository.gsi.de/record/65526`.
21. *Лебедев С. А.* Математическое обеспечение для реконструкции колец черенковского излучения и идентификации электронов в RICH детекторе эксперимента CBM: Автореф. дис. ... канд. физ.-мат. наук. — Дубна: ОИЯИ, 2011. — 10-2011-14.
22. *Дереновская О. Ю., Иванов В. В.* Реконструкция и отбор распадов $J/\psi \to e^+e^-$, регистрируемых установкой CBM в AuAu-соударениях при энергии пучка 25 ГэВ/нуклон // Письма в ЭЧАЯ. — 2014. — Т. 11, № 4 (188). — С. 862–885.
23. Track Reconstruction Algorithms for the CBM Experiment at FAIR / A. Lebedev, C. Hohne, I. Kisel, G. Ososkov // Journal of Physics: Conference Series. — 2010. — Vol. 219. — Pp. 32–48.
24. *Акишина Т. П.* Особенности применения критерия ω_n^k к задаче идентификации электронов с помощью детектора переходного излучения в эксперименте CBM // Письма в ЭЧАЯ. — 2012. — Т. 9, № 3 (173). — С. 440–462.
25. *Deppner I. et al.* The CBM Time-of-Flight Wall // Nuclear Instruments and Methods. — 2012. — Pp. 121–124.
26. *Дереновская О. Ю., Васильев Ю. О.* Критерии отбора распадов $J/\psi \to e^+e^-$, регистрируемых установкой CBM в AuAu-соударениях при энергии 25 ГэВ/нуклон // Письма в ЭЧАЯ. — 2014. — Т. 11, № 1 (185). — С. 63–73.
27. *Kisel I., Kulakov I., Zyzak M.* Standalone First Level Event Selection Package for the CBM Experiment // IEEE Transactions on Nuclear Science. — 2013. — Vol. 60, No 5. — Pp. 3703–3708.

UDC 004.93'1:519.237:51-72 PACS 02.50.Sk; 02.70.Rr; 29.85.-c; 07.05.-t

Vectorization and Parallelization of Algorithms for Selection and Reconstruction of $J/\psi \to e^+e^-$ Decays in Real Time of the CBM Experiment

O. Yu. Derenovskaya, V. V. Ivanov

Laboratory of Information Technologies
Joint Institute for Nuclear Research
6, Joliot-Curie str., Dubna, Moscow region, Russian Federation, 141980

The measurements of J/ψ decays is one of the key goals of the CBM experiment. The technique of J/ψ registration in its dielectron channel has been developed, which includes a chain of methods of trajectories and momentum reconstruction of charged particles with STS, electron/positron identification with RICH, TRD and TOF, as well as construction of the J/ψ-candidates and their characteristics using the KFParticle package. Taking into account that selection and reconstruction decays of $J/\psi \to e^+e^-$ are planned to be carried out in real time of the CBM experiment, the used methods and algorithms should be not only effective but also fast.

In this paper the time-consuming estimation of the existing algorithms based on their acceleration via code vectorization by means of SIMD instructions and parallelization between the processor cores that are implemented using OpenMP, OpenCL software environments and TBB library has been carried out. This analysis allowed to establish weak points in this chain, which are under investigation, as well as offer a fast and efficient parallel algorithm for the identification of the charged particles with TRD based on the ω_n^k criterion.

Key words and phrases: multivariate analysis, CBM experiment, selection of $J/\psi \to e^+e^-$ decays, parallel computation, SIMD instructions.

References

1. T. Matsui, H. Satz, J/ψ Suppression by Quark-Gluon Plasma Formation, Physics Letters B 178 (14) (1986) 416–422.
2. B. Alessandro, et al., A New Measurement of J/ψ Suppression in Pb-Pb Collisions at 158-GeV per Nucleon, European Physical Journal C 39 (3) (2005) 335–345.
3. M. Leitch, RHIC Results on J/ψ, Journal of Physics G: Nuclear and Particle Physics 34 (8) (2007) 453–462.
4. The CBM Collaboration. CBM Compressed Baryonic Matter Experiment. Technical Status Report, Tech. rep., GSI, Darmstadt (2005).
   URL `http://www.gsi.de/onTEAM/dokumente/public/DOC-2005-Feb-447e.html`
5. The CBM Collaboration, Nuclear Physics A 904–905 (2013) 1059–1062.
6. B. Friman, C. Hohne, J. Knoll, S. Leupold, J. Randrup, et al., The CBM Physics Book: Compressed Baryonic Matter in Laboratory Experiments, Lecture Notes in Physics 814 (2011) 1–980.
7. O. Y. Derenovskaya, Y. Vassiliev, J/ψ Reconstruction in the Dielectron Decay Channel at SIS100 Energies in the CBM Experiment, Physics of Particles and Nuclei Letters 10 (5 (182)) (2013) 424–430, in Russian.
8. S. Gorbunov, I. Kisel, Reconstruction of Decayed Particles Based on the Kalman Filter, CBM-SOFT-note-2007-003.
9. IA-32 Intel Architecture Optimization Reference Manual (2005).
10. OpenMP.
    URL `http://openmp.org`
11. OpenCL.
    URL `http://www.khronos.org/opencl`
12. Threading Building Blocks.
    URL `http://threadingbuildingblocks.org`
13. Intel Hyper-Threading Technology, Technical User's Guide (2003).

14. A. Beleventsev, M. Kuvirkov, D. Melnik, Using the Instruction-Level Parallelism in the Compiler for Intel Itanium, Proceedings of ISP RAS 9 (2006) 9–22, in Russian.
15. Silicon Tracking System(STS). Technical Design Report for the CBM, Tech. rep., GSI, Darmstadt (2012).
URL http://repository.gsi.de/record/54798
16. I. S. Kulakov, S. A. Baginyan, V. V. Ivanov, P. I. Kisel, Performance Analysis of Cellular Automaton Algorithm to Solve the Track-Reconstruction Problem on a Multicore Server at the Laboratory of Information Technologies, Joint Institute for Nuclear Research, Physics of Particles and Nuclei Letters 10 (2 (179)) (2013) 253–267, in Russian.
17. S. Gorbunov, U. Kebschull, I. Kisel, V. Lindenstruth, W. F. J. Müller, Fast SIMDized Kalman Filter Based Track Fit, Computer Physics Communications 178 (2008) 374–383.
18. T. O. Ablyazimov, M. V. Zyzak, V. V. Ivanov, P. I. Kisel, Filter Method for the Charged Particles Trajectories Reconstruction in the CBM Experiment and Its Parallel Implementation at the JINR LIT Manycore Server, Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics" (2) (2014) 191–196, in Russian.
19. NVidia GTX 480.
URL http://de.geforce.com/hardware/desktop-gpus/geforce-gtx-480/architectur
20. Ring Imaging Cherenkov (RICH) Detector. Technical Design Report for the CBM, Tech. rep., GSI, Darmstadt (2013).
URL http://repository.gsi.de/record/65526
21. S. A. Lebedev, Algorithms and Software for Cherenkov Ring Reconstruction and Electron Identification in the CBM Experiment: Abstract of a PhD Thesis, JINR, Dubna, 10-2011-14.
22. O. Y. Derenovskaya, V. V. Ivanov, Reconstruction and Selection of $J/\psi \to e^+e^-$ Decays Registered by CBM Setup in 25 AGeV AuAu Collisions, Physics of Particles and Nuclei Letters 11 (4 (188)) (2014) 560–572, in Russian.
23. A. Lebedev, C. Hohne, I. Kisel, G. Ososkov, Track Reconstruction Algorithms for the CBM Experiment at FAIR, Journal of Physics: Conference Series 219 (2010) 32–48.
24. T. P. Akishina, Peculiarities of Applying the ω_n^k Criterion for the Electron Identification Problem Based on the Transition Radiation Detector in the Compressed Baryonic Matter Experiment, Physics of Particles and Nuclei Letters 9 (3 (173)) (2012) 268–282, in Russian.
25. I. Deppner, et al., The CBM Time-of-Flight Wall, Nuclear Instruments and Methods (2012) 121–124.
26. O. Y. Derenovskaya, Y. O. Vassiliev, Criteria for Selection of $J/\psi \to e^+e^-$ Decays Detected by the CBM Setup in Au-Au Collisions at an Energy of 25 AGeV, Physics of Particles and Nuclei Letters 10 (7)) (2014) 710–716, in Russian.
27. I. Kisel, I. Kulakov, M. Zyzak, Standalone First Level Event Selection Package for the CBM Experiment, IEEE Transactions on Nuclear Science 60 (5) (2013) 3703–3708.