Письма в ЭЧАЯ. 2014. Т. 11, № 4(188). С. 828–846

БЫСТРАЯ РЕКОНСТРУКЦИЯ ТРАЕКТОРИЙ ЗАРЯЖЕННЫХ ЧАСТИЦ В ЭКСПЕРИМЕНТЕ CBM НА ОСНОВЕ ФИЛЬТРА КАЛМАНА С ИСПОЛЬЗОВАНИЕМ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ НА МНОГОЯДЕРНОМ СЕРВЕРЕ ЛИТ ОИЯИ

Т. О. Аблязимов [а], *М. В. Зызак* [б,в], *В. В. Иванов* [а], *П. И. Кисель* [а]

[а] Объединенный институт ядерных исследований, Дубна

[б] Франкфуртский университет им. И. В. Гете, Франкфурт-на-Майне, Германия

[в] Киевский национальный университет им. Т. Шевченко, Киев

Реконструкция траекторий заряженных частиц в режиме реального времени эксперимента CBM (GSI, Дармштадт, Германия) является чрезвычайно сложной задачей. Это обусловлено высокой частотой соударений ионов пучка с мишенью (до 10^7 в секунду), большой множественностью частиц, рождающихся в результате одного соударения (до 1000 частиц), и регистрацией траекторий заряженных частиц координатными детекторами, размещенными в сильно неоднородном магнитном поле. Такая задача может быть решена только с использованием современных высокопроизводительных вычислительных средств. В настоящей работе детально обсуждается алгоритм реконструкции траекторий заряженных частиц на основе фильтра Калмана, реализованный с применением различных методов распараллеливания программного кода. Для решения рассматриваемой задачи использовался гибридный сервер ЛИТ ОИЯИ с двумя центральными процессорами Intel Xeon X5660 и графической картой NVidia GTX 480.

The charged particle trajectory online reconstruction in the CBM experiment (GSI, Germany) is an extremely difficult task. It is conditioned by a high rate of the ion beam — target collisions (up to 10^7/s), high track multiplicity in each nucleus–nucleus collision (up to 1000 particles) and charged particle trajectory registration with the coordinate detectors located in a highly inhomogeneous magnetic field. Such a problem could be solved only by using modern high-performance computers. This work scrutinizes a Kalman filter based track reconstruction algorithm implemented using different parallelization approaches. To perform the analysis, a manycore hybrid server with two Intel Xeon X5660 CPUs and a NVidia GTX 480 GPU (JINR LIT) was used.

PACS: 29.50.+v

ВВЕДЕНИЕ

В настоящее время в GSI (Дармштадт, Германия) на строящемся ускорительном комплексе FAIR (Facility for Antiproton and Ion Research) ведутся работы по созданию экспериментальной установки CBM (Compressed Baryonic Matter). Физическая программа CBM нацелена на всестороннее изучение фазовой диаграммы сильновзаимодействующей

материи и уравнения состояния вещества при экстремально высоких плотностях барионной материи [1]. Для ее реализации необходимы измерения множественности частиц, их распределений в фазовом пространстве, центральности соударений и плоскости реакции. С точки зрения физической программы наибольший интерес связан с очень редкими распадами, регистрация которых потребует проведение эксперимента при экстремально высоких частотах соударения ядер пучка с мишенью и большой множественности заряженных частиц, рождающихся в результате этих соударений.

Схема экспериментальной установки CBM с мюонным детектором представлена на рис. 1.

Непосредственно за мишенью между полюсами сверхпроводящего дипольного магнита располагаются координатные трековые детектирующие системы: MVD (Micro-Vertex Detector), включающая две станции из монолитных микропиксельных детекторов, и STS (Silicon Tracking System), состоящая из восьми двухсторонних кремниевых микростриповых детекторов. Детектирующие системы MVD и STS предназначены для реконструкции траекторий заряженных частиц, восстановления их импульсов с точностью не хуже 1 % и нахождения вторичных вершин в условиях высокой множественности и плотности частиц. Мюонная система MuCh (Muon Chamber), состоящая из чередующихся слоев железных поглотителей и координатных детекторов, предназначена для идентификации вторичных мюонов и реконструкции их треков. Детектор переходного излучения TRD (Transition Radiation Detector) используется для реконструкции треков частиц и идентификации электронов/позитронов в условиях доминирующего фона от пионов. Для идентификации адронов используется детектор времени пролета TOF (Time-Of-Flight). Калориметр PSD (Projectile Spectator Detector) служит для определения плоскости реакции.

Наличие интенсивных потоков регистрируемой экспериментальной информации, которые могут достигать величины 1 Тбайт/с при частоте ядро-ядерных соударений 10 МГц, обуславливает жесткие требования как к архитектуре и быстродействию системы сбора и накопления информации, так и к организации отбора редких событий, представляющих интерес для физиков. Ввиду сложности и неоднозначности критериев отбора таких событий, коллаборацией CBM принято решение не строить общую для всех исследу-

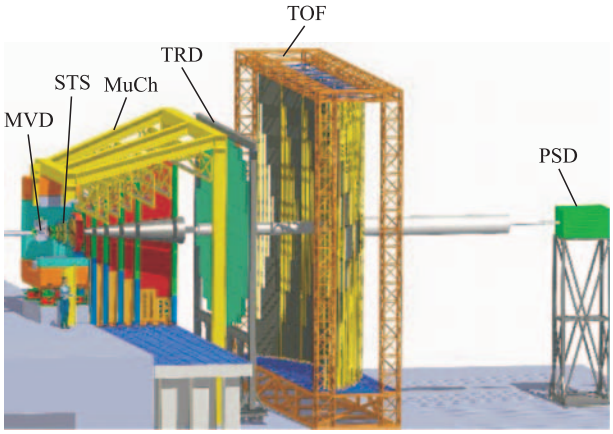

Рис. 1. Схема экспериментальной установки CBM с мюонным детектором MuCh

емых физических процессов триггирующую систему, а проводить обработку всей регистрируемой информации в реальном времени эксперимента. Потоки экспериментальной информации будут управляться посредством высокоскоростной сети, объединяющей высокопроизводительную вычислительную среду с конфигурируемыми вычислительными ресурсами. Эта среда будет использоваться для фильтрации информации и формирования событий. Для включения кандидата на событие в список отобранных событий требуется наличие реконструированных треков, а также восстановленных с помощью детекторов MVD и STS первичной и вторичных вершин. В качестве определяющего фактора для системы DAQ (Data AcQuisition) принята пропускная способность всей вычислительной среды. Такой подход должен позволить с максимальной эффективностью использовать все вычислительные ресурсы установки CBM при наборе статистики, достаточной для регистрации редких процессов, возникающих в ядро-ядерных соударениях при высоких энергиях.

Таким образом, быстрая реконструкция траекторий заряженных частиц с помощью детекторов MVD и STS является ключевой проблемой в задаче отбора полезных событий в эксперименте CBM. Высокая множественность событий (до 1000 треков в каждом ядро-ядерном соударении), интенсивный фон (до 85 % фоновых отсчетов в MVD и STS), неоднородное магнитное поле и необходимость реконструкции всех событий в режиме реального времени (до 10^7 событий в секунду) требуют не только развития новых подходов для решения рассматриваемой задачи, но и максимального использования потенциала современных многоядерных архитектур CPU/GPU.

Процедура реконструкции траекторий заряженных частиц, регистрируемых с помощью вершинного детектора CBM, включает два последовательных этапа:

1) распознавание треков в условиях высокой множественности и плотности заряженных частиц, интенсивного фона и неоднородного магнитного поля;

2) восстановление параметров трека (место попадания в координатный детектор и направление трека) и импульса заряженной частицы с использованием информации о координатах трека (с детекторов MVD и STS), найденного на шаге 1.

В выполненной ранее работе [2] была представлена реализация быстрого алгоритма распознавания траекторий заряженных частиц в вершинном детекторе эксперимента CBM с использованием параллельных вычислений на многоядерном сервере ЛИТ ОИЯИ с двумя центральными процессорами Intel Xeon 5640 [1].

В основу этого алгоритма положены концепция клеточного автомата и фильтр Калмана. Несмотря на высокую множественность событий, интенсивный фон и неоднородное магнитное поле, алгоритм продемонстрировал высокую эффективность распознавания треков — 96–97 % — при низком уровне некорректно найденных треков — 2–4 %. При этом была достигнута высокая скорость обработки событий на одном логическом ядре, равная в среднем 220 мс на одно центральное событие и 25 мс на одно смешанное событие. Было показано, что с увеличением числа включаемых в обработку ядер практически линейно растет количество обработанных событий.

В настоящей работе приведены результаты исследования производительности алгоритма реконструкции траекторий заряженных частиц, реализованного на основе фильтра

[1] Общее количество физических ядер на двух процессорах равняется 8, а суммарное число логических ядер (с учетом гиперпоточности) составляет 16.

Калмана с применением различных подходов для распараллеливания кода на многоядерном сервере ЛИТ ОИЯИ. В отличие от предыдущей нашей работы [2], этот сервер оснащен двумя более высокопроизводительными процессорами Intel Xeon X5660 и графической картой NVidia GTX 480: суммарное количество физических ядер двух процессоров Intel Xeon X5660 равняется 12, что (с использованием гиперпоточности) соответствует 24 логическим ядрам; общее число вычислительных ядер на карте NVidia GTX 480 составляет 448. Для оценки производительности алгоритма использовались разные подходы для распараллеливания и векторизации программного кода: заголовочные файлы, осуществляющие перегрузку операторов, средства библиотеки Vc (Vector Classes) [3], программные среды OpenMP (Open Multi-Processing) [4] и OpenCL (Open Computing Language) [5].

1. РАСШИРЕННЫЙ ФИЛЬТР КАЛМАНА

Фильтр Калмана — это эффективный вычислительный алгоритм, предназначенный для рекурсивного дооценивания вектора состояния априорно известной дискретизированной по времени линейной динамической системы [6, 7]. Для расчета текущего состояния системы нужно знать текущее измерение и предыдущее состояние самого фильтра. Фильтр Калмана оперирует понятием вектора состояния системы (набором параметров, характеризующих состояние системы в некоторый момент времени) и его статистическим описанием. В общем случае динамика вектора состояния определяется плотностями вероятности распределения его компонент в каждый момент времени. При наличии определенной математической модели производимых наблюдений за системой, а также модели изменения параметров вектора состояния (в данном случае — марковского формирующего процесса) можно записать уравнение для апостериорной плотности вероятности вектора состояния в любой момент времени. Соответствующее уравнение носит название уравнения Стратоновича, которое в общем виде не имеет решения. Аналитическое решение удается получить только в случае ряда ограничений:

- гауссовских априорных и апостериорных плотностей вероятности вектора состояния на любой момент времени (в том числе начальный);
- гауссовских формирующих шумов;
- гауссовских шумов наблюдений;
- белых шумов наблюдений;
- линейности модели наблюдений;
- линейности модели формирующего процесса (который, напомним, должен являться марковским процессом).

Классический фильтр Калмана [8, 9] представляется уравнениями для расчета первого и второго моментов апостериорной плотности вероятности (в смысле вектора математических ожиданий и матрицы дисперсий, в том числе и взаимных) при указанных ограничениях. Ввиду того, что для нормальной плотности вероятности математическое ожидание и дисперсионная матрица полностью задают плотность вероятности, можно сказать, что фильтр Калмана рассчитывает апостериорную плотность вероятности вектора состояния на каждый момент времени. А значит, полностью описывает вектор состояния как случайную векторную величину. Расчетные значения математических ожиданий

при этом являются оптимальными оценками по критерию среднеквадратической ошибки, что и обуславливает его широкое применение.

Существует несколько разновидностей фильтра Калмана, отличающихся приближениями и приемами, которые приходится использовать для сведения фильтра к описанному виду и уменьшения его размерности. В частности, в *расширенном фильтре Калмана* нелинейные модель наблюдений и модель формирующего процесса линеаризуются посредством разложения в ряд Тейлора.

При использовании фильтра Калмана для определения оценок состояния исследуемого процесса по серии зашумленных измерений предполагается, что вектор состояния системы в момент времени k может быть получен из истинного состояния в момент $k-1$ из следующего уравнения:

$$\mathbf{r}_k = \mathbf{F}_k \mathbf{r}_{k-1} + \mathbf{q}_k, \tag{1}$$

где $\mathbf{F}_k$ — матрица эволюции процесса, которая воздействует на вектор $\mathbf{r}_{k-1}$ (вектор состояния системы в момент $k-1$); $\mathbf{q}_k$ — нормальный случайный процесс с нулевым математическим ожиданием и ковариационной матрицей $\mathbf{Q}_k$ описывает случайный характер эволюции системы: $\mathbf{q}_k \sim N(0, \mathbf{Q}_k)$. Ковариационная матрица $\mathbf{Q}_k \equiv \langle \mathbf{q}_k \cdot \mathbf{q}_k^T \rangle$ должна быть известна.

В момент времени k производится измерение $\mathbf{m}_k$ вектора состояния $\mathbf{r}_k$. Векторы $\mathbf{m}_k$ и $\mathbf{r}_k$ связаны между собой уравнением

$$\mathbf{m}_k = \mathbf{H}_k \mathbf{r}_k + \mathbf{v}_k, \tag{2}$$

где $\mathbf{H}_k$ — матрица измерений (наблюдений), связывающая вектор состояния с измерением; $\mathbf{v}_k$ — белый гауссовский шум измерений с нулевым математическим ожиданием и ковариационной матрицей $\mathbf{V}_k$: $\mathbf{v}_k \sim N(0, \mathbf{V}_k)$. Ковариационная матрица $\mathbf{V}_k \equiv \langle \mathbf{v}_k \cdot \mathbf{v}_k^T \rangle$ также должна быть известна.

Начальное состояние системы $\mathbf{r}_0$ и векторы случайных процессов на каждом шаге

$$\{\mathbf{r}_0, \mathbf{q}_1, \ldots, \mathbf{q}_k, \ldots, \mathbf{v}_1, \ldots, \mathbf{v}_k\}$$

полагаются независимыми.

Механизм работы фильтра Калмана носит рекурсивный характер: для вычисления оценки состояния системы на k-м такте нужно знать «состояние фильтра» на предыдущем такте и измерение, выполненное на текущем такте. Состояние фильтра определяется двумя переменными:

- $\mathbf{r}_k$ — оценка вектора состояния системы в момент времени k, полученная по результатам наблюдений вплоть до момента времени k включительно;
- $\mathbf{C}_k$ — ковариационная матрица ошибок, задающая оценку точности вектора состояния и содержащая дисперсии ошибок вычисленного вектора и ковариации, показывающие взаимосвязи между параметрами состояния системы.

Процедура вычисления оценки состояния системы с помощью фильтра Калмана состоит из двух последовательных этапов: экстраполяции и коррекции. На этапе экстраполяции вычисляются предварительные оценки вектора состояния $\mathbf{r}_k^-$ и соответствующей ему ковариационной матрицы ошибок $\mathbf{C}_k^-$. В принятых нами обозначениях знак «$-$» означает, что в этих оценках не учитывается измерение k-го такта. На этапе коррекции оценки, полученные при экстраполяции, корректируются путем учета текущего измерения. При этом знак «$-$» заменяется на знак «$+$».

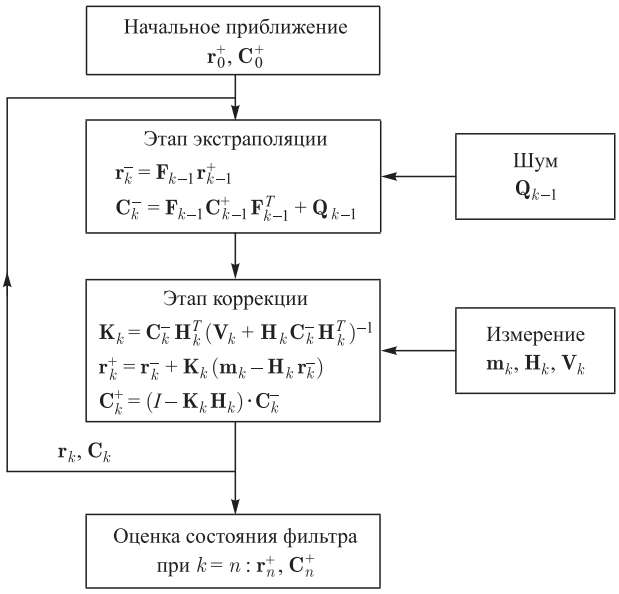

Рис. 2. Блок-схема работы классического оптимального фильтра Калмана

На рис. 2 представлена блок-схема, иллюстрирующая работу классического оптимального фильтра Калмана [10].

Инициализация. Задаются начальные значения вектора состояния $\mathbf{r}_0$ и его ковариационной матрицы $\mathbf{C}_0$. Значение $\mathbf{r}_0$ может быть выбрано произвольным образом, например с использованием некоторой предварительной информации о векторе состояния. В случае отсутствия такой информации, $\mathbf{r}_0$ можно положить равным нулю. В качестве начального приближения для матрицы $\mathbf{C}_0$ берется диагональная матрица с достаточно большими числами на диагонали, чтобы эти числа превосходили дисперсию реальных измерений. В этом случае на первых тактах работы фильтр будет с бóльшим весом использовать результаты измерений, чем априорную информацию. Кроме того, такой выбор позволяет избавиться от зависимости окончательной оценки вектора состояния от начального приближения (см. ниже).

Процедура вычисления оценок состояния системы (экстраполяция и коррекция) продолжается согласно рекурсивному алгоритму, приведенному на рис. 2, вплоть до учета последнего n-го измерения.

Этап экстраполяции. На этом этапе вектор состояния $\mathbf{r}_{k-1}^+$ и его ковариационная матрица $\mathbf{C}_{k-1}^+$ экстраполируются к измерению $\mathbf{m}_k$ следующего такта. При вычислении ковариационной матрицы $\mathbf{C}_k^-$ учитывается случайный шум $\mathbf{Q}_{k-1}$, связанный с эволюцией системы. Экстраполированные оценки вектора состояния $\mathbf{r}_k^-$ и его ковариационная матрица $\mathbf{C}_k^-$ рассчитываются согласно формулам

$$\mathbf{r}_k^- = \mathbf{F}_{k-1}\mathbf{r}_{k-1}^+, \tag{3а}$$

$$\mathbf{C}_k^- = \mathbf{F}_{k-1}\mathbf{C}_{k-1}^+\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1}, \tag{3б}$$

где $\mathbf{r}_{k-1}^{+}$ и $\mathbf{C}_{k-1}^{+}$ — оценки вектора состояния и его ковариационной матрицы после $(k-1)$ шага коррекции. На первой итерации полагается, что $\mathbf{r}_0^{+} = \mathbf{r}_0$ и $\mathbf{C}_0^{+} = \mathbf{C}_0$.

Этап коррекции. На этом этапе вычисляются оптимальная оценка вектора состояния $\mathbf{r}_k^{+}$ и его ковариационная матрица $\mathbf{C}_k^{+}$ с учетом измерения $\mathbf{m}_k$, сделанного на k-м такте.

Вначале определяется разница между измерением $\mathbf{m}_k$ и экстраполированной оценкой измерения:

$$\mathbf{d}_k = \mathbf{m}_k - \mathbf{H}_k \mathbf{r}_k^{-}. \tag{4а}$$

Ковариационная матрица вектора отклонения определяется формулой

$$\mathbf{W}_k = \mathbf{H}_k \mathbf{C}_k^{-} \mathbf{H}_k^{T} + \mathbf{V}_k. \tag{4б}$$

Используя ковариационные матрицы для экстраполированного вектора состояния $\mathbf{C}_k^{-}$ и для вектора отклонения $\mathbf{W}_k$, находим оптимальную по Калману матрицу коэффициентов усиления

$$\mathbf{K}_k = \mathbf{C}_k^{-} \mathbf{H}_k^{T} \mathbf{W}_k^{-1}. \tag{4в}$$

Уточняем полученную на этапе экстраполяции оценку вектора состояния

$$\mathbf{r}_k^{+} = \mathbf{r}_k^{-} + \mathbf{K}_k \mathbf{d}_k. \tag{4г}$$

Вычисляем ковариационную матрицу, отвечающую скорректированному вектору состояния $\mathbf{r}_k^{+}$:

$$\mathbf{C}_k^{+} = (I - \mathbf{K}_k \mathbf{H}_k) \cdot \mathbf{C}_k^{-}. \tag{4д}$$

Результат вычислений. После того как алгоритмом фильтра Калмана учтены все измерения, включая последний такт n, получаем оптимальные оценки вектора состояния $\mathbf{r}_n^{+}$ и его ковариационной матрицы $\mathbf{C}_n^{+}$.

В исследуемой нами задаче реконструкции треков заряженных частиц, регистрируемых системой координатных детекторов MVD и STS, расположенных между полюсами спектрометрического магнита, вектору состояния системы $\mathbf{r}_k$ отвечает набор параметров трека, матрица эволюции системы $\mathbf{F}_k$ реализует экстраполяцию трека в магнитном поле от одного детектора к другому, а матрица шума процесса $\mathbf{Q}_k$ описывает многократное рассеяние частиц в материале детектора. Кроме того, так как в данном случае модель наблюдений и модель формирующего процесса носят нелинейный характер, их линеаризуют посредством разложения в ряд Тейлора.

2. ОСОБЕННОСТИ РЕАЛИЗАЦИИ АЛГОРИТМА РЕКОНСТРУКЦИИ ПАРАМЕТРОВ ТРЕКОВ В ЭКСПЕРИМЕНТЕ CBM

CBM — это эксперимент с фиксированной мишенью (рис. 1), и, как следствие, большинство частиц, образованных в соударении тяжелых ионов при высоких энергиях, летят в направлении падающего пучка. При этом траектории заряженных частиц между полюсами дипольного магнита описываются следующей системой дифференциальных

уравнений:

$$\frac{dt_x}{dz} = c\,t_r\,\frac{q}{p}\,(t_y(B_z + t_x B_x) - (1 + t_x^2)B_y), \tag{5а}$$

$$\frac{dt_y}{dz} = c\,t_r\,\frac{q}{p}\,(-t_x(B_z + t_y B_y) + (1 + t_y^2)B_x), \tag{5б}$$

$$t_r = \sqrt{t_x^2 + t_y^2 + 1}, \tag{5в}$$

где (x, y) — координаты трека при заданной z-координате [1]; $t_x = dx/dz$ и $t_y = dy/dz$ — тангенсы угла наклона трека соответственно в плоскостях XOZ и YOZ; q/p — отношение заряда частицы q на ее импульс p; B_x, B_y, B_z — компоненты магнитного поля в точке (x, y, z); c — скорость света в вакууме.

С учетом вышесказанного вектор состояния заряженной частицы, движущейся в пространстве между полюсами дипольного магнита (см. рис. 1), можно записать в виде

$$\mathbf{r} = \left\{x, y, t_x, t_y, \frac{q}{p}\right\}. \tag{6}$$

Для получения начальной оценки параметров трека использовался метод наименьших квадратов и делались следующие два приближения:

1) принималась во внимание только основная компонента магнитного поля B_y;

2) не учитывалось многократное рассеяние в веществе координатных детекторов.

Эти допущения позволяют получить оценки параметров трека с точностью, достаточной для обеспечения быстрой сходимости алгоритма.

Координатные плоскости STS-детектора состоят из набора стрипов, размещенных на разных сторонах микростripового сенсора под углом друг к другу. При этом измерения, полученные обеими сторонами сенсора, не коррелируют между собой, что позволяет представить их как два одномерных измерения. Регистрирующая плоскость детектора MVD состоит из пиксельных сенсоров, поэтому измерение места ее пересечения заряженной частицей включает в себя обе координаты — x и y. Поскольку эти координаты нескоррелированы между собой, следовательно, по аналогии с детектором STS их можно рассматривать как два независимых измерения. Такое допущение позволяет единообразно обрабатывать измерения с указанных детекторов, упрощать уравнения фильтрации и ускорять вычисления. Так как представленные таким образом измерения зависят линейно от места попадания частицы в плоскость детектора, модель измерения принимает следующий вид:

$$\mathbf{H}_k = \{\cos(\alpha_k),\ \sin(\alpha_k),\ 0,\ 0,\ 0\}, \tag{7}$$

где α_k — угол между стрипом и осью Y.

Матрица эволюции системы $\mathbf{F}_k$ отвечает за движение заряженной частицы в неоднородном магнитном поле $B(x, y, z)$. Траектория частицы в магнитном поле описывается уравнениями (5а)–(5в), численное решение которых требует слишком много времени на

[1] Установка CBM размещается в декартовой системе координат XYZ, начало которой совпадает с мишенью, ось Z направлена по пучку, Y — вертикально вверх, а X находится в горизонтальной плоскости.

компьютере. Поэтому для ускорения процедуры реконструкции траектории движения частицы в магнитном поле использовалась приближенная аналитическая формула из [13]. Скорость и точность аппроксимации траектории частицы в этом подходе определяются числом членов разложения в указанной формуле. Такое приближение позволило получить точность, сопоставимую с точностью, достигаемой при решении системы уравнений (5а)–(5в) методом Рунге–Кутта 4-го порядка. При этом удается более чем вдвое повысить скорость вычислений.

С помощью указанной приближенной формулы с параметрами трека в k-м детекторе частица транспортируется до $(k+1)$-детектора, и, таким образом, находится зависимость между параметрами трека на $(k+1)$-й и k-й станциях: $\mathbf{r}_{k+1} = \mathbf{f}_k(\mathbf{r}_k)$. Эта зависимость линеаризируется посредством разложения в ряд Тейлора:

$$\langle \mathbf{r}_{k+1} \rangle = \mathbf{f}_k(\mathbf{r}_k) \approx \mathbf{f}_k(\mathbf{r}_k^{\mathrm{lin}_p}) + \mathbf{F}_k(\mathbf{r}_k - \mathbf{r}_k^{\mathrm{lin}_p}), \tag{8}$$

$$F_{k_{ij}} = \left.\frac{\partial \mathbf{f}_{k_i}(\mathbf{x})}{\partial x_j}\right|_{\mathbf{x}=\mathbf{r}_k^{\mathrm{lin}_p}}. \tag{9}$$

В качестве точки линеаризации здесь используется вектор состояния

$$\mathbf{r}_k^{\mathrm{lin}} = \left\{ x_k^+,\ y_k^+,\ t_{x\ k}^+,\ t_{y\ k}^+,\ \frac{q}{p_0} \right\},$$

где x_k^+, y_k^+, $t_{x\ k}^+$, $t_{y\ k}^+$ — параметры трека в k-м детекторе, а p_0 — импульс частицы, заданный на этапе инициализации параметров. Таким образом, вычисляется матрица эволюции $\mathbf{F}_k$ исследуемой нами системы.

Для более эффективного использования кэш-памяти процессора и ускорения работы алгоритма реконструкции треков карта магнитного поля аппроксимировалась полиномиальной зависимостью. В работе [10] было показано, что для аппроксимации с высокой точностью поля в области детектирующих станций достаточно использовать полиномы пятого порядка. Кроме того, так как при транспортировке частицы требуются компоненты поля только вдоль одного трека, поведение поля между станциями можно аппроксимировать параболой, используя координаты трека на трех соседних станциях. Такая замена полной карты поля не сказалась на точности реконструкции параметров трека. В то же время скорость вычислений выросла в 50 раз.

В эксперименте CBM координатные детекторы MVD и STS состоят из тонких модулей (толщиной не более 300 мк). В этом случае при фитировании трека шум процесса, описываемый матрицей $\mathbf{Q}_k$, обусловлен в основном многократным рассеянием заряженной частицы в материале детектора. Потерями энергии частицы в материале детектора можно пренебречь, так как их вклад в рассматриваемый нами процесс пренебрежимо мал. При этом каждое отдельное рассеяние частицы на материале детектора происходит на малые углы. В то же время этого материала достаточно для того, чтобы в нем происходило до 20 таких рассеяний.

С учетом вышесказанного и согласно центральной предельной теореме, мы можем использовать для описания суммарного угла отклонения частицы в веществе детектора нормальное распределение. Дисперсия данного распределения может быть вычислена с

помощью следующей формулы [14]:

$$\sigma(\theta) = \frac{13{,}6\ \text{МэВ}}{\beta p} q \sqrt{\frac{S}{X_0}} \left[1 + 0{,}038 \ln\left(\frac{S}{X_0}\right)\right], \tag{10}$$

где p — импульс частицы; q — ее заряд в единицах элементарного заряда; S — толщина материала, сквозь который проходит частица; X_0 — радиационная длина материала. Исходя из приведенных выше допущений, можно, по аналогии с [15], записать $\mathbf{Q}_k$ в виде

$$\mathbf{Q}_k = \sigma_k^2(\theta) \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & (t_x^2+1)t_r^2 & t_x t_y t_r^2 & 0 \\ 0 & 0 & t_x t_y t_r^2 & (t_y^2+1)t_r^2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{11}$$

Обычно приложения, реализованные на основе фильтра Калмана, используют вычисления с двойной точностью. Однако с учетом того, что применение одинарной точности позволяет более эффективно использовать SIMD-модуль, кэш-память центрального процессора и графические ускорители, алгоритм фитирования треков был реализован в одинарной точности.

Переход на одинарную точность позволил резко повысить скорость выполнения алгоритма, однако из-за роста ошибок округления часть треков стала реконструироваться некорректно. Для исправления указанной ситуации были выполнены следующие улучшения алгоритма [10]:

- более точная начальная оценка параметров трека,
- специальная процедура для добавления первого измерения,
- стабилизация вычисления ковариационной матрицы на шаге фильтрации.

При переходе от одного измерения к следующему могут возникать вычислительные проблемы в случае, если экстраполируемые параметры значительно превышают допустимый предел их изменения. Проведенный в работе [8] анализ показал, что если какой-то из параметров более чем в четыре раза превышает ошибку измерения, то, ограничив его указанным предельным значением, можно добиться устойчивой работы алгоритма. Аналогичная проблема имеет место и при добавлении первого измерения, так как в этом случае ошибки параметров задаются в виде бесконечно больших чисел. Для этого специфического случая формулы можно упростить аналитически и избежать вычислений с большими числами.

3. РАСПАРАЛЛЕЛИВАНИЕ ПРОГРАММНОГО КОДА

Для достижения максимальной производительности алгоритма реконструкции параметров треков на современных многоядерных серверах проведено распараллеливание программного кода с использованием векторных регистров SIMD-архитектур и распределением потоков между ядрами центрального и/или графического процессоров.

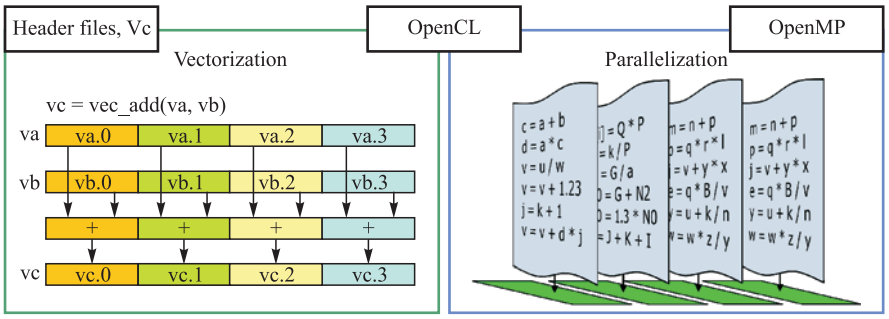

Рис. 3. Подходы, используемые для векторизации и распараллеливания алгоритма реконструкции параметров треков

Векторизация программного кода реализована с помощью заголовочных файлов, библиотеки Vc [3] и/или программной среды OpenCL [11]; для распараллеливания задач между ядрами использовались программные среды OpenMP [12] и OpenCL (рис. 3).

Заголовочные файлы позволяют перегружать арифметические и логические операторы, используя SIMD-инструкции. Это делает код программы компактным и легко читаемым. К примеру, простая функция для вычисления полинома второго порядка с использованием SIMD-инструкций выглядит довольно сложно:

```
__m128 y = _mm_add_ps(_mm_mul_ps(a,x),b);
```

в то время как код функции, использующий заголовочные файлы

```
fvec y = a*x + b;
```

с перегрузкой операторов

```
friend fvec operator+(const fvec &a, const fvec &b) {
  return _mm_add_ps(a,b); }
friend fvec operator*(const fvec &a, const fvec &b) {
  return _mm_mul_ps(a,b); }
```

смотрится просто и читается легко.

Вследствие простоты реализации данный подход гибок по отношению к различным архитектурам центральных процессоров. Он позволяет оставлять код программы неизменным, подключая соответствующий заголовочный файл. Для отладки программного кода существует также заголовочный файл со скалярной реализацией.

Библиотека Vc — это удобный инструмент для применения SSE-, AVX- и LRBni-инструкций [3]. Как и заголовочные файлы, она предоставляет доступ к основным арифметическим и логическим операциям и функциям. Для тестирования разрабатываемого кода Vc содержит и скалярную реализацию. Функция для вычисления полинома второй степени с применением библиотеки Vc будет иметь следующий вид:

```
float_v y = a*x + b;
```

Библиотека Vc автоматически определяет платформу, под которой она запущена, и при сборке кода выбирает соответствующие инструкции. В дополнение к операциям, использующим вектор целиком (вертикальные операции), в Vc реализованы операции внутри одного SIMD-вектора (горизонтальные операции), позволяющие, например, выполнить суммирование или сортировку элементов вектора. Для реализации интерфейсов условных операций все функции могут опционально принимать в качестве аргумента маску. Например, чтобы заменить все отрицательные элементы вектора y из предыдущего примера нулями, нужно выполнить следующие операции:

```
float_m mask = y<0;
y(mask) = 0;
```

OpenCL (Open Computing Language) — открытый стандарт для написания компьютерных программ, связанных с параллельными вычислениями на различных графических и центральных процессорах, а также FPGA [11]. Программная среда OpenCL включает язык программирования, который базируется на стандарте C99, и интерфейс программирования приложений (API). OpenCL обеспечивает параллелизм на уровне инструкций и на уровне данных. При этом создаваемые программы подходят как для CPU, так и GPU, что избавляет разработчиков от необходимости поддерживать две версии одной и той же программы. OpenCL также позволяет использовать обе опции современных процессоров — векторизацию и распараллеливание между ядрами.

OpenMP (Open Multi-Processing) — открытый стандарт для распараллеливания программ на языках Си, Си++ и Фортран [12]. Эта программная среда содержит набор директив компилятора, библиотечных процедур и переменных окружения, предназначенных для программирования многопоточных приложений на многопроцессорных системах с общей памятью (SMP-системах). OpenMP реализует параллельные вычисления с помощью многопоточности, в которой «главный» поток создает набор подчиненных потоков, между которыми распределяется решаемая задача. Участки задачи, выполняемые потоками параллельно, так же как и данные для их выполнения, описываются с помощью специальных директив препроцессора. Количество создаваемых потоков может регулироваться как в самой программе путем вызова библиотечных процедур, так и извне при помощи переменных окружения.

4. ОБСУЖДЕНИЕ РЕЗУЛЬТАТОВ

К алгоритму реконструкции параметров треков предъявляются два основных требования:

1) максимально возможная точность восстановления (x, y)-координат места попадания трека в конкретный координатный детектор, наклонов трека в этой точке и импульса частицы;

2) высокая скорость реконструкции треков в реальном времени эксперимента.

Первое играет ключевую роль, в частности, при реконструкции типа и места распада исследуемых в эксперименте наблюдаемых. В свою очередь, скорость выполнения алгоритма крайне важна для CBM, поскольку планируется проводить реконструкцию событий в реальном времени эксперимента при частоте ядро-ядерных соударений до 10^7 Гц.

Точность восстановления параметров трека методом фильтра Калмана. Для оценки точности реконструкции параметров трека используем такое понятие, как остаток ρ. При этом остаток ρ_x, в частности для x-координаты трека в одном из детекторов STS-системы, определяется как разность между величиной x_{mc}, полученной в результате Монте-Карло моделирования прохождения трека через детектор, и значением x_{reco},

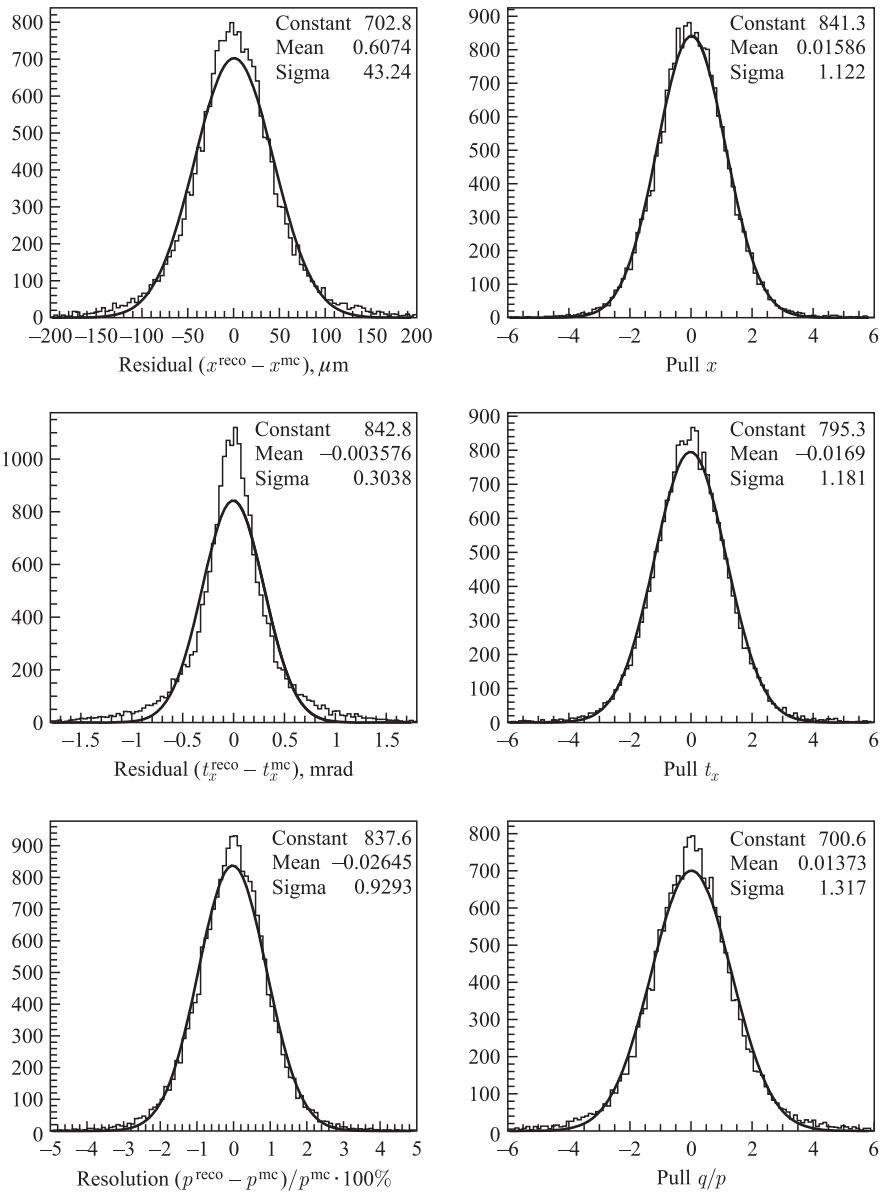

Рис. 4. Распределение остатков и пулов для параметров треков, реконструированных с помощью фильтра Калмана

реконструированным с помощью рассматриваемого алгоритма:

$$\rho_x = x_{\mathrm{reco}} - x_{\mathrm{mc}}. \tag{12}$$

В качестве характеристики надежности реконструкции параметров трека используются нормированные остатки (пулы):

$$P(x) = \frac{\rho_x}{\sqrt{C_{xx}}}, \tag{13}$$

где C_{xx} — диагональный элемент соответствующей ковариационной матрицы, полученной в результате реконструкции трека. В идеальном случае пулы должны быть распределены по закону Гаусса с единичной дисперсией.

Для оценки качества реконструкции треков с помощью рассматриваемого алгоритма нами использовались треки, найденные в STS-системе эксперимента CBM методом клеточного автомата [2, 16]. Для этого было отобрано 20 000 длинных первичных треков, координаты которых были зарегистрированы во всех STS-станциях. Качество реконструкции треков оказалось одинаковым для всех реализаций упомянутых выше средств векторизации и распараллеливания кода.

Результаты тестирования представлены на рис. 4. Поскольку распределения для x- и y-координат и соответствующих им наклонов абсолютно идентичны, даны результаты только для переменной x.

Из приведенных распределений видно, что алгоритм, реализованный на основе фильтра Калмана, позволяет реконструировать параметры треков с высокой точностью. В частности, точность восстановления импульса заряженных частиц составила примерно 1 % на всем импульсном интервале. Распределения пулов хорошо аппроксимируются нормальным законом с дисперсией, близкой к единице. Этот результат указывает на корректность процедуры фитирования. Пул, отвечающий обратному импульсу, наиболее чувствителен к влиянию многократного рассеяния частицы в материале детектора, и поэтому его дисперсия превышает единицу.

Производительность алгоритма на многоядерных системах. Как отмечалось выше, скорость выполнения алгоритмов играет ключевую роль в современной физике высоких энергий, поэтому очень важно с максимальной эффективностью использовать ресурсы современных вычислительных машин.

В настоящее время в развитии высокопроизводительных серверов наблюдается тенденция увеличения как числа самих процессоров, так и количества ядер на них. При этом важно, чтобы пропорционально росла и общая скорость выполнения алгоритмов.

Масштабируемость алгоритма на центральных процессорах. Для оценки производительности алгоритма реконструкции параметров треков на основе фильтра Калмана использовался многоядерный сервер cuda.jinr.ru ЛИТ ОИЯИ (см. рис. 5), оснащенный двумя процессорами Intel Xeon X5660, каждый из которых содержит шесть физических ядер с частотой 2,8 ГГц и 12 Мбайт кэш-памяти третьего уровня. Используя технологию гиперпоточности, на каждом из физических ядер можно запускать одновременно два потока (процесса выполнения программ), за работу каждого из которых отвечает соответствующее логическое ядро. Таким образом, на двух центральных процессорах сервера cuda.jinr.ru может быть запущено одновременно 24 потока.

При использовании программной среды OpenCL и вышеперечисленных средств распараллеливания (заголовочные файлы, библиотеки Vc) время обработки одного трека на одном логическом ядре сервера составило 0,5 мкс.

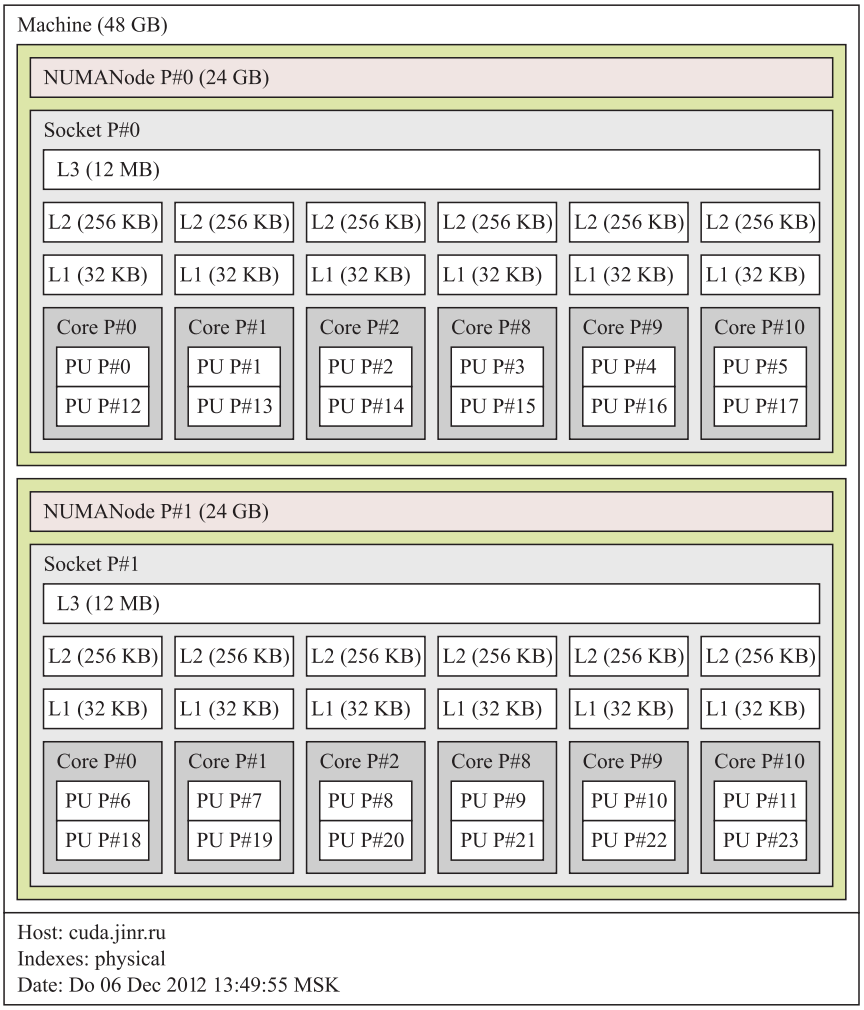

Рис. 5. Структура ядер, оперативной и кэш-памяти на сервере cuda.jinr.ru. Каждому физическому ядру отвечают два логических ядра, 32 Кбайт кэш-памяти первого уровня и 256 Кбайт второго. На каждый из двух процессоров приходится 24 Гбайт оперативной памяти и 12 Мбайт кэш-памяти третьего уровня, которая делится между всеми ядрами

Для распараллеливания алгоритма на ядрах двух центральных процессоров сервера применялись две программные среды — OpenMP и OpenCL.

В среде OpenMP процедура загрузки ядер задачами заключалась в следущем. Вначале запускалась первая задача на первом логическом ядре первого физического ядра первого процессора, затем запускалась вторая задача на втором логическом ядре этого же физического ядра. Далее по такой же схеме запускались задачи на остальных логических ядрах первого процессора. После полной загрузки первого процессора проводилась загрузка второго.

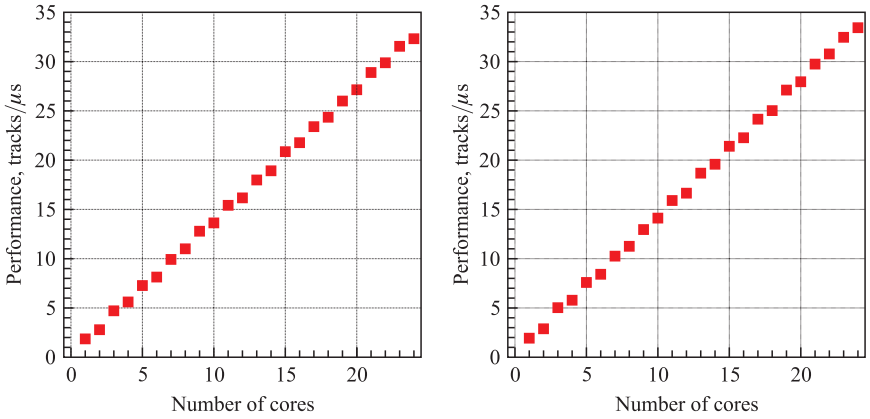

Рис. 6. Масштабируемость алгоритма реконструкции треков заряженных частиц в зависимости от числа запущенных в среде OpenMP логических ядер при использовании для векторизации кода заголовочных файлов (слева) и библиотеки Vc (справа)

На рис. 6 приведены графики масштабируемости алгоритма реконструкции треков заряженных частиц в зависимости от числа запущенных в среде OpenMP логических ядер при использовании для векторизации кода заголовочных файлов (слева) и библиотеки Vc (справа). Видно, что оба метода обеспечивают линейную зависимость числа обработанных треков от количества используемых логических ядер. Из приведенных зависимостей также следует, что применение гиперпоточности позволило дополнительно увеличить скорость вычислений примерно на 20 %.

Оба метода показывают одинаковую производительность на сервере cuda.jinr.ru и позволяют достичь скорости обработки, равной 34 трека за 1 мкс. Отметим, что использование заголовочных файлов обеспечивает гибкую реализацию алгоритма, что, соответственно, значительно упрощает перенос программы и ее тестирование на различных платформах. Вместе с тем библиотека Vc предоставляет доступ к горизонтальным операциям и возможность работать с масками, что расширяет возможности при написании программного кода.

Так как в среде OpenCL разработчику программ не предоставляется возможности самому контролировать загрузку ядер процессора, логические ядра загружаются согласно системной нумерации. На сервере cuda.jinr.ru нумерация логических ядер организована таким образом, что вначале на каждом из физических ядер запускается по одному потоку (точки от 0 до 12 на рис. 7), а затем с помощью технологии гиперпоточности к вычислениям подключаются вторые логические ядра. Алгоритм, реализованный в среде OpenCL, также демонстрирует линейную масштабируемость (рис. 7).

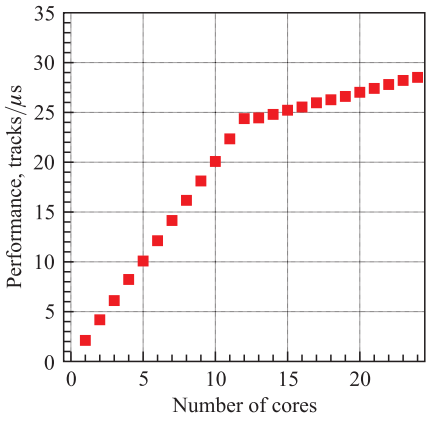

Рис. 7. Масштабируемость алгоритма реконструкции параметров треков заряженных частиц по отношению к числу логических ядер центрального процессора в среде OpenCL

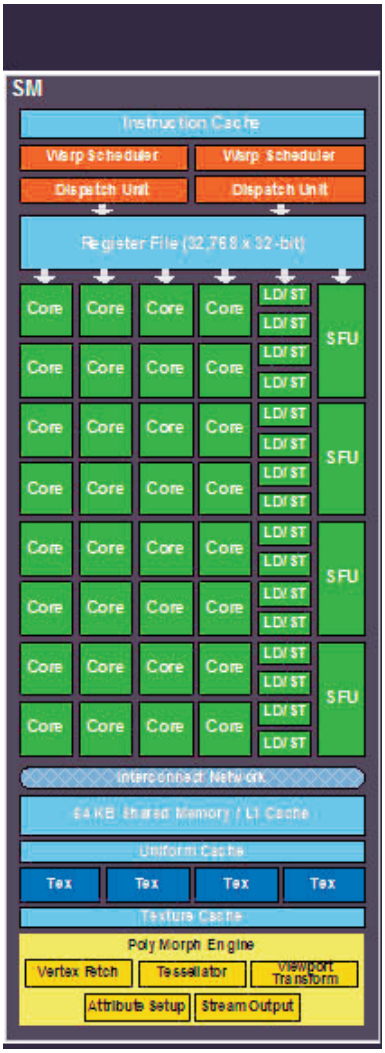

Рис. 8. Структура потокового мультипроцессора графической карты NVidia GTX 480

Поскольку при использовании среды OpenMP можно более эффективно управлять памятью и оптимизировать программный код, удалось достичь большей производительности, чем в среде OpenCL: 34 трека против 27 за 1 мкс.

Производительность алгоритма на графических ускорителях и распределение задач по рабочим группам. На современных высокопроизводительных серверах для вычислений используются не только центральные процессоры, но и графические карты, что позволет существенно повысить суммарную производительность серверов. Поэтому важно, чтобы рассматриваемый алгоритм можно было запускать и на графических процессорах. Реализация алгоритма реконструкции параметров треков в среде OpenCL позволила решить эту задачу.

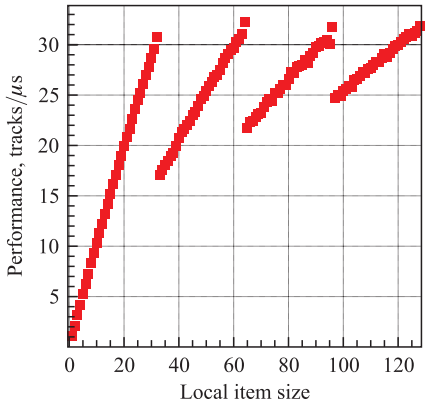

Рис. 9. Результаты производительности графического процессора NVidia GTX 480 в зависимости от числа треков в рабочей группе

Для тестирования алгоритма использовалась графическая карта NVidia GTX 480 сервера cuda.jinr.ru. Процессор этой карты содержит 480 CUDA-ядер, которые оснащены арифметико-логическими устройствами для проведения вычислений как с целыми числами, так и с «плавающей точкой». Во время выполнения программы ядра распределяются между потоковыми мультипроцессорами (SMP) — по 32 ядра на каждый. На рис. 8 приведена структура потокового мультипроцессора карты NVidia GTX 480 [17].

При запуске программы на графическом процессоре весь набор треков, подлежащих обработке, распределяется между рабочими группами. Каждая из таких групп запускается своим потоковым мультипроцессором. Поскольку на используемой нами карте рабочие группы разбиваются на 32 потока, выполняющихся параллельно, максимальная производительность может быть достигнута только тогда, когда группа содержит число треков, кратное 32. В противном случае ресурсы карты, выделяемые на одну рабочую группу, будут использоваться неоптимально.

На рис. 9 приведены результаты производительности графического процессора NVidia GTX 480 в зависимости от числа треков в рабочей группе. Видно, что при числе треков в группе, кратном 32, достигается максимальная производительность, равная 33 трека за 1 мкс.

ЗАКЛЮЧЕНИЕ

В настоящей работе изучена и продемонстрирована возможность проведения быстрой реконструкции параметров траекторий заряженных частиц, регистрируемых системой координатных детекторов MVD и STS эксперимента CBM, на основе фильтра Калмана с использованием параллельных вычислений на многоядерном сервере ЛИТ ОИЯИ. К разрабатываемому алгоритму предъявлялись два основных требования: максимально возможная точность реконструкции нужных параметров трека (координата и направление) и частицы (импульс); высокая производительность алгоритма.

Решение по первому требованию было найдено в результате правильного выбора в качестве метода для реализации указанной задачи рекурсивного фильтра Калмана, а также за счет применения ряда приближений, позволивших без потери точности вычислений обеспечить высокую надежность и скорость обработки.

Второе требование было удовлетворено за счет разработки параллельного алгоритма с использованием различных современных средств по распараллеливанию и векторизации кода, таких как заголовочные файлы, осуществляющие перегрузку операторов, библиотека Vc, программные среды OpenMP и OpenCL. Указанные технологии позволяют запускать созданный программный код на высокопроизводительных многоядерных и гибридных системах, оснащенных векторными (SIMD) модулями и графическими ускорителями.

Разработанный алгоритм был протестирован на сервере cuda.jinr.ru ЛИТ ОИЯИ с двумя центральными процессорами Intel Xeon X5660 и графической картой NVidia GTX 480. Все программные реализации показали линейную масштабируемость и высокую производительность (34 трека за 1 мкс на центральном процессоре и 33 трека за 1 мкс на графическом). Таким образом, уже в существующей комплектации сервер позволяет обрабатывать до 70 треков за 1 мкс.

СПИСОК ЛИТЕРАТУРЫ

1. The CBM Physics Book / Eds.: Friman B. et al. // Lecture Notes in Physics. 2011. V. 814. P. 960.
2. *Kulakov I. S. et al.* Performance Analysis of Cellular Automaton Algorithm to Solve the Track-Reconstruction Problem on a Multicore Server at the Laboratory of Information Technologies, Joint Institute of Nuclear Research // Phys. Part. Nucl. Lett. 2013. V. 10, No. 2. P. 162–170.
3. Vector Classes. http://gitorious.org/vc.
4. OpenMP. http://openmp.org.
5. OpenCL. http://www.khronos.org/opencl.
6. *Kalman R. E.* A New Approach to Linear Filtering and Prediction Problems // Trans. ASME. Ser. D. J. Basic Eng. 1960. V. 82. P. 35–45.
7. http://en.wikipedia.org/wiki/Kalman_filter
8. *Simon D.* Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches. John Wiley & Sons, 2006.
9. *Frühwirth R. et al.* Data Analysis Techniques for High-Energy Physics. Second Ed. Cambridge Univ. Press, 2000.
10. *Gorbunov S. et al.* Fast SIMDized Kalman Filter Based Track Fit // Comp. Phys. Commun. 2008. V. 178. P. 374–383.
11. http://ru.wikipedia.org.org/wiki/OpenCL
12. http://ru.wikipedia.org.org/wiki/OpenMP
13. *Gorbunov S., Kisel I.* Analytic Formula for Track Extrapolation in Non-Homogeneous Magnetic Field // Nucl. Instr. Meth. A. 2006. V. 559. P. 148–152.
14. *Lynch G. R., Dahl O. I.* Approximations to Multiple Coulomb Scattering // Nucl. Instr. Meth. B. 1991. V. 58. P. 6–10.
15. *Wolin E. J., Ho L. L.* Covariance Matrices for Track Fitting with the Kalman Filter // Nucl. Instr. Meth. A. 1993. V. 329. P. 493–500.
16. *Kisel I.* Event Reconstruction in the CBM Experiment // Nucl. Instr. Meth. A. 2006. V. 566. P. 85–88.
17. NVidia GTX 480. http://de.geforce.com/hardware/desktop-gpus/geforce-gtx-480/architecture.

Получено 3 декабря 2013 г.