ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ЗАДАЧАХ ФИЗИКИ ВЫСОКИХ ЭНЕРГИЙ ДЛЯ ПРОЕКТА NICA/MPD

А.Е. Басалаев, А.С. Бондаренко, С.А. Немнюгин, А.Н. Тимофеев

Введение. Любой современный эксперимент в области физики элементарных частиц требует тщательного моделирования. На этом этапе происходит апробация применяемых алгоритмов, сравнение используемых моделей, тестирование и оптимизация написанного программного кода. Под оптимизацией здесь понимается как алгоритмическое усовершенствование кода, так и внедрение высокопроизводительных и параллельных технологий.

В рамках научной программы по изучению горячей и плотной барионной материи в ОИЯИ (Объединенный Институт Ядерных Исследований, Дубна) реализуется проект по созданию нового ускорительного комплекса с встречными пучками NICA (Nuclotron-based Ion Collider fAcility) [1, 2]. Новый ускорительный комплекс позволит исследовать взаимодействия тяжелых ионов в широком диапазоне атомных масс: от легких ядер до ядер золота при энергии 3-11 ГэВ/нуклон в системе центра масс и светимости 10^{27}см$^{-2}$с$^{-1}$ при частоте 6 кГц.

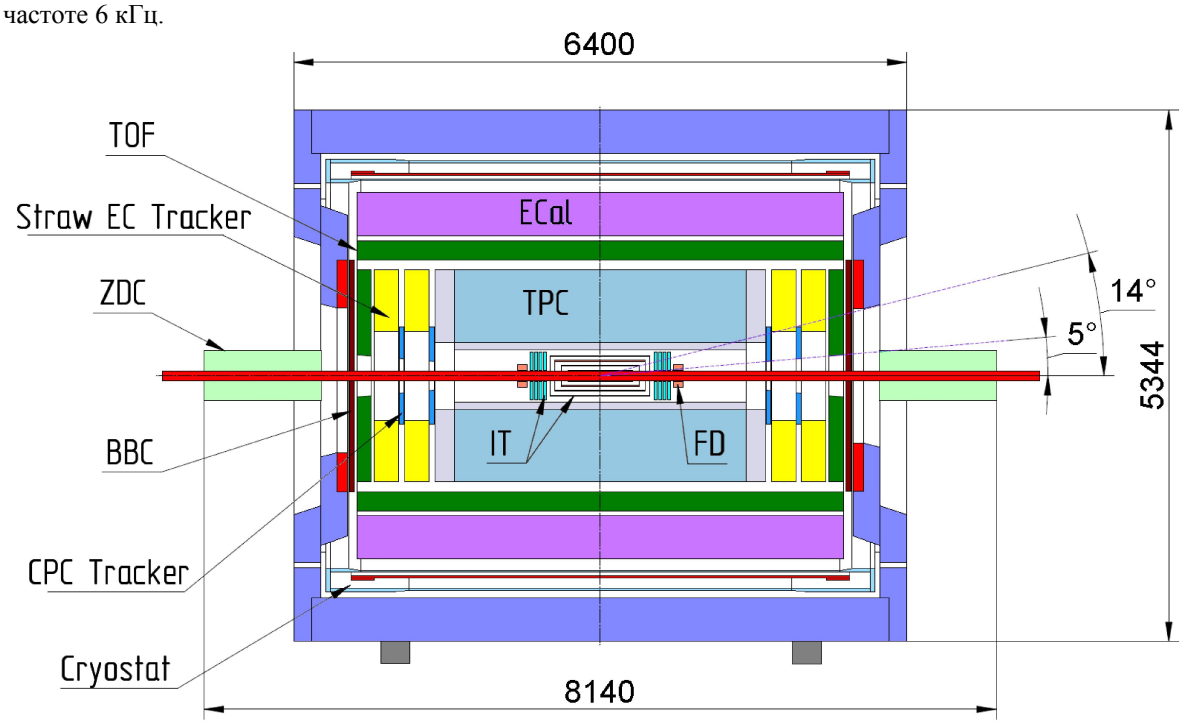


Рис. 1. Схема многоцелевого детектора MPD

На коллайдере предусмотрены две точки пересечения пучков, в одной из которых будет располагаться экспериментальная установка MPD (Multi-Purpose Detector, многоцелевой детектор), предназначенная для исследований столкновений тяжелых ионов [3].

Детектор MPD состоит из цилиндрической и двух торцевых частей (рис. 1). Основным детектором для регистрации треков заряженных частиц и их идентификации в центральной области MPD является время-проекционная камера TPC (Time Projection Chamber).

В данной работе рассмотрены задачи оптимизации процесса вычисления, возникающие при моделировании работы установки NICA/MPD.

Для генерации событий, моделирования работы детектора и реконструкции событий на установке NICA/MPD разрабатывается программный комплекс MpdRoot [4], основанный на пакете для научных расчетов ROOT[5]. Данный комплекс предоставляет множество инструментов как для работы с данными, так и для моделирования в целом, и допускает подключение внешних библиотек для расширения функциональных возможностей.

Использование облачных вычислений. В настоящее время ресурсный вычислительный центр Санкт-Петербургского Государственного Университета располагает несколькими вычислительными системами, одна из которых построена на принципах виртуализации. Система состоит из двух корзин blade-серверов HP BladeSystem c7000, включающих в себя по 16 модулей BL460G7 - 2 процессора Intel® Xeon® X5670 и 96Гб оперативной памяти каждый. Для хранения данных используются две системы HP StorageWorks P4500 G2

емкостью 240 ТБ каждая. Характеристики системы: 32 сервера, 64 процессора, 384 ядра, 3 ТБайт ОЗУ, 480 ТБайт дискового пространства, пиковая производительность 4,5 Тфлопс. Вычислительная система находится под управлением VMWare vSphere [6].

Для работы из общей системы был выделен массив из 10 виртуальных серверов, каждый из которых оснащен четырехъядерным процессором Intel® Xeon® Processor X5670 и 10 Гб оперативной памяти. Используемые средства виртуализации позволяют реализовать динамическое распределение серверных ресурсов, обеспечивая предоставление нужного ресурса нужному приложению в соответствии с приоритетами. Среды исполнения приложений могут увеличиваться и уменьшаться при необходимости: есть возможность «горячего» добавления ЦП и памяти к виртуальным машинам при необходимости и без прерывания, возможность «горячего» расширения виртуальных дисков обеспечивает добавление пространства для хранения данных к работающим виртуальным машинам. Такая возможность динамического выделения ресурсов позволяет оптимизировать нагрузки и энергопотребление, что особенно важно для высоконагруженных научных вычислительных систем.

Разработка интерфейса между генераторами событий и MpdRoot. Модель ядерных столкновений UrQMD (Ultrarelativistic Quantum Molecular Dynamics model, ультрарелятивистская квантово-молекулярная динамическая модель, [7]) и HADGEN (HADron GENerator, адронный генератор, [8]) являются зарекомендовавшими себя во многих экспериментах инструментами для моделирования столкновений тяжелых ядер, но работа с ними не всегда удобна. Данные генераторы работают в одном потоке, а время, необходимое для генерации большого количества событий, весьма значительно; помимо этого, выходные данные сохраняются в виде файлов, что увеличивает затраты на операции чтения/записи, и сами файлы могут занимать десятки гигабайт. Модель ядерных столкновений UrQMD фактически является стандартом в области физики высоки энергий; выбор программного генератора HADGEN обусловлен его высокой точностью описания процессов, происходящих при энергиях NICA (в диапазоне от 3 до 11 ГэВ/нуклон), и тем, что он является составной частью пакета SHIELD [8], планируемого к использованию для расчета радиационной защиты ускорителя и многих его элементов.

В данной работе на примере двух генераторов UrQMD и HADGEN предлагается простой и достаточно универсальный способ ускорить моделирование и обработку данных за счет использования интерфейса подключения этих библиотек к программному комплексу MpdRoot и использования его расширения PROOF (Parallel ROOT facility, [9]).

Оригинальные исходные коды генераторов UrQMD и HADGEN написаны на языке FORTRAN; хранение данных осуществляется в глобальных структурах (common-blocks). Использование глобальных переменных имеет два основных недостатка: 1. снижение надежности программ, их модульности и противоречие принципам инкапсуляции, 2. отсутствие возможности реализовать параллелизм задач такими средствами языка, как POSIX threads или аналогичными инструментами.

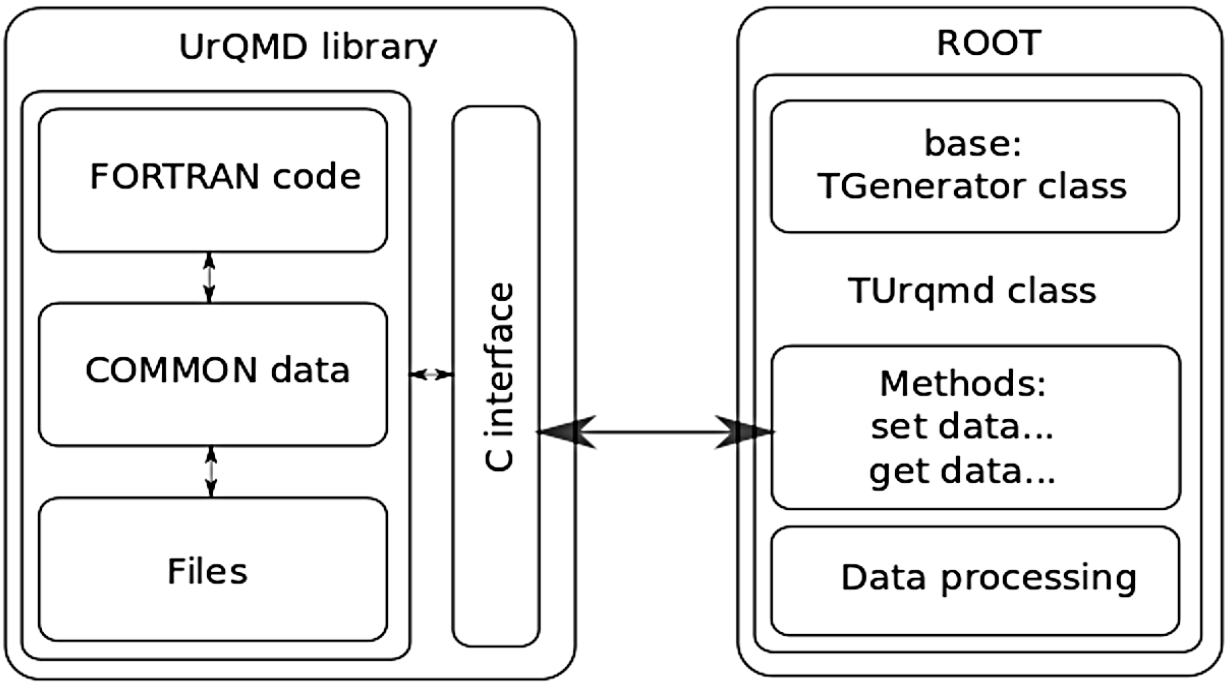


Рис. 2. Архитектура враппера для модели ядерных столкновений UrQMD

В качестве первого шага оптимизации были разработаны интерфейсы на языке C, реализация которых обеспечивает безопасный доступ к данным (рис. 2). Исходные коды генераторов вместе с интерфейсами собираются в динамические библиотеки.

Вторым шагом является создание в рамках MpdRoot классов-оболочек (wrappers) на языке C++, взаимодействующих с интерфейсами библиотек. Данные классы содержат методы, необходимые для работы с генератором, введения начальных параметров моделирования и получения выходных данных в формате, удобном для дальнейшего использования.

Для достижения максимальной производительности был использован инструмент PROOF - расширение пакета ROOT, позволяющее реализовать интерактивную, параллельную обработку больших массивов данных или, в общем случае, добиться параллельности выполнения кода на уровне задач.

PROOF-кластер строится по стандартной схеме master-slave (рис. 3). Благодаря многоуровневой архитектуре, позволяющей создать иерархию из master и submaster узлов, такой подход может быть легко адаптирован к широкому диапазону виртуальных кластеров, географически распределённых между доменами и гетерогенными машинами (GRID). Клиентом системы является пользователь, который хочет использовать ресурсы сайта, чтобы выполнить свою задачу. Master – это точка входа к вычислительному средству: он разбирает запросы клиента, распределяет работу между ресурсами, собирает и соединяет результаты. Координация работы отдельных серверов достигается за счет использования специального протокола передачи данных PROOF.

Пользователь, работая в сессии программы ROOT, может запускать процессы, которые связываются с PROOF-кластером и подают запросы на обработку заданий. Получив запрос на обработку задания, на мастере и на рабочих узлах для каждой сессии пользователя стартует специальное ROOT-приложение – proofserv. Процесс, исполняющийся на мастере, координирует работу между рабочими узлами и объединяет результаты в единое целое. На рабочих узлах процесс proofserv делает непосредственно вычислительную работу, обрабатывая отдельные задания.

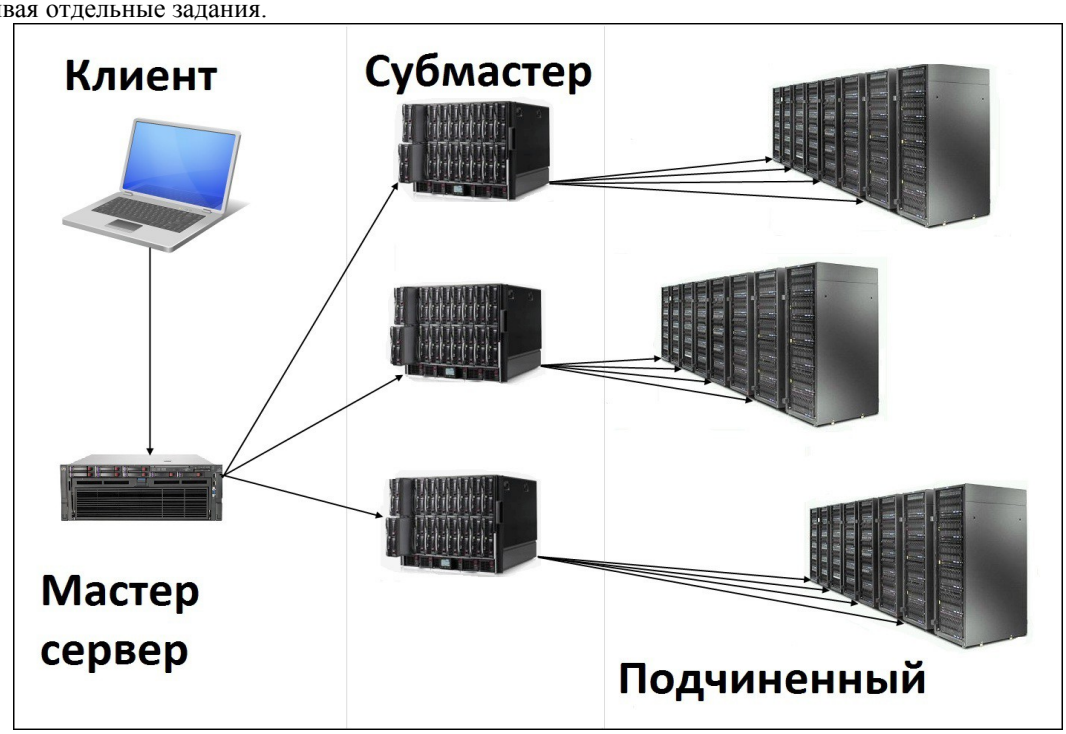


Рис. 3. Схема кластера PROOF

Задания для обработки на PROOF-кластере пишутся особым образом; для этого объявляется класс C++, наследуемый от встроенного в ROOT класса TSelector. В данном классе обязательно должен быть переопределен набор методов, реализующий функционал вычислительного узла (рис. 4).

Рис. 4. Иерархия вызовов обязательных методов при работе с PROOF

Для работы с PROOF-кластером создается экземпляр класса TProof:

```
TProof * p = TProof::Open("proof.cluster.address");

p->SetParallel(nThreads);

p->Process("ClassImplementation.C", nIterations);
```

Рис. 5. Пример работы с PROOF-кластером

Функция void Process(..) служит для запуска задания в параллельном режиме с заданным количеством итераций nIterations.

В ходе решения задач проекта NICA появилась потребность в моделировании Ridge-эффекта [9] для подробного рассмотрения структуры события столкновения тяжелых ядер. Природа данного эффекта на данный момент не объяснена, существует несколько теорий. Эффект состоит в наличии корреляций между частицами, получающимися при столкновении тяжелых ядер. Имеется корреляция по азимутальному углу даже “далеких” друг от друга частиц, показывающему отклонение частиц в сторону. Для наблюдения Ridge-эффекта предъявляются некоторые требования по отбору частиц: величина их поперечного импульса должна лежать в определенных пределах. Задача моделирования данного эффекта во многом обусловлена необходимостью объяснить, почему только частицы с определенным спектром импульсов имеют подобные корреляции. Для этого требуется моделирование с большим объемом статистики (порядка миллионов событий), поэтому был использован кластер виртуальных машин PROOF. Благодаря тому, что написание класса оболочки намного проще, чем попытки внедрить параллелизм в код самого генератора, и наличию легко масштабируемой вычислительной системы, удалось добиться высокой производительности с малыми затратами.

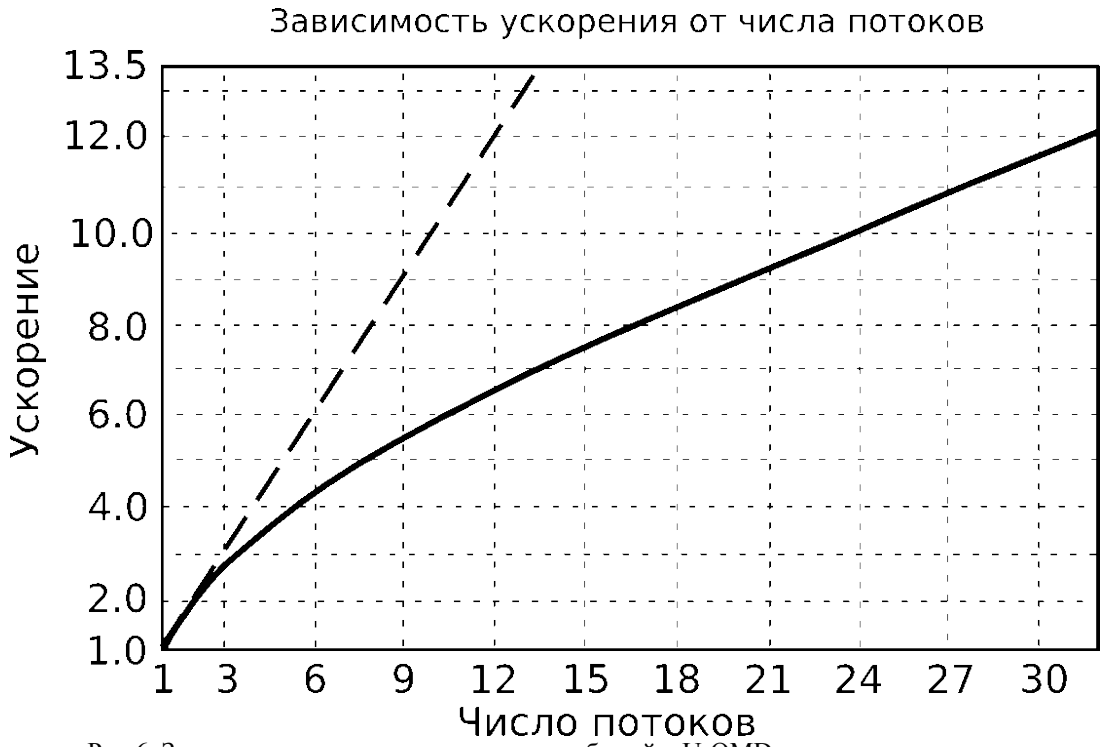


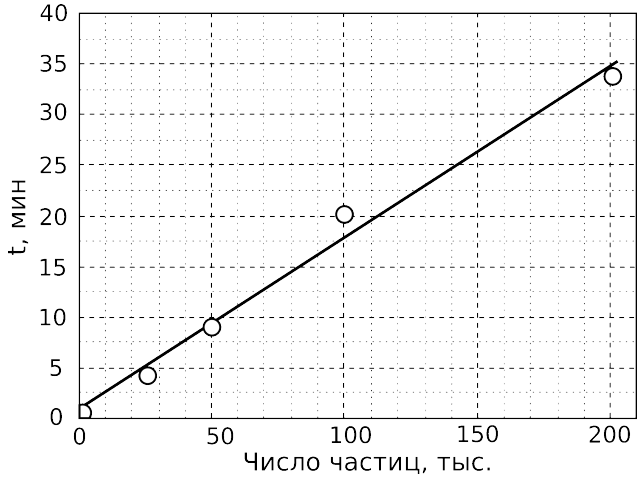
Рис.6. Зависимость ускорения генерации событий в UrQMD от числа потоков при использовании многоядерной архитектуры PROOF

Ускорение, как видно из рис. 6, ниже теоретического предела Амдала. В данном примере, каждый поток моделировал 100 столкновений тяжелых ядер золота. При увеличении нагрузки на поток наблюдается увеличение производительности, так как потери на пересылку данных и накладные расходы на создание и мониторинг потоков, становятся малыми по сравнению с временем вычислений.

Вычисление радиационных длин для детектора MPD. Одной из актуальных задач на стадии проектирования эксперимента NICA/MPD является исследование возможности помещения дополнительных трековых детекторов за торцевыми стенками TPC (см. рис. 1), на которых размещена электроника для съема информации. Важной является оценка влияния аппаратуры на торцевых стенках TPC на пролетающие сквозь них частицы. Это влияние носит случайный характер и скорректировать его невозможно. Характеристикой материала, показывающей, насколько велико это влияние, является радиационная длина - средняя толщина вещества, на которой энергия электрона уменьшается в e раз, умноженная на плотность вещества. Её необходимо минимизировать.

Для выполнения этой задачи необходимо получить распределение радиационных длин в детекторе и вычислить интегральную радиационную длину для TPC. Точность расчетов зависит от плотности покрытия точками области сканирования. Поэтому одной из задач в данной работе было не просто вычислить радиационные длины для MPD, но сделать это с высоким разрешением.

Данная задача была решена на виртуальном кластере в среде MpdRoot. Для ускорения расчетов применялось распараллеливание с помощью PROOF.



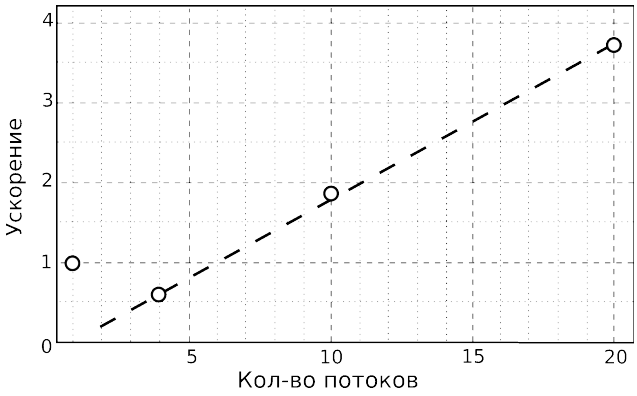


Рис. 7. Анализ производительности кода для расчета радиационных длин

Использование PROOF при проведении расчета в одном потоке сильно снижает производительность, как видно на рис. 7. Однако при использовании большого количества потоков наблюдается выигрыш в скорости, при этом рост производительности близок к линейному. Безусловно, конкретная задача еще нуждается в дополнительной оптимизации, однако преимущество PROOF очевидно.

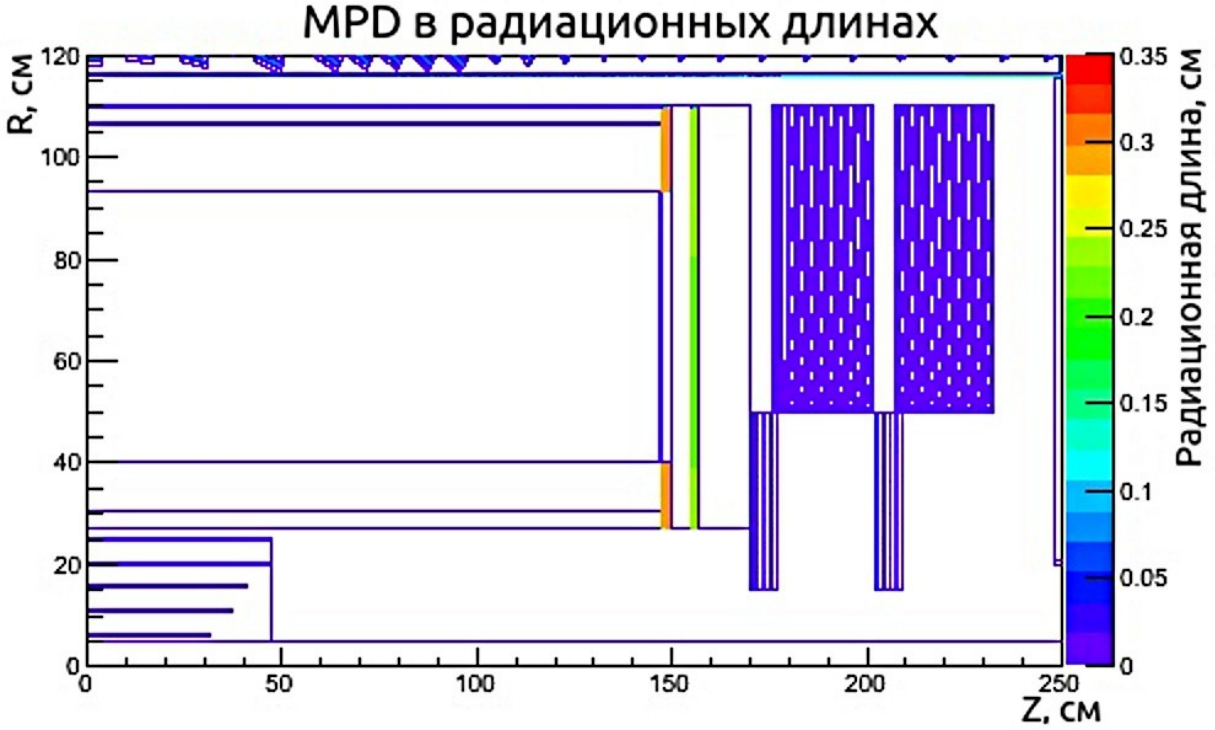


Рис. 8. Представление MPD в радиационных длинах

На рис. 8 представлен результат моделирования - картина детектора MPD в радиационных длинах.

Заключение. Использование высокопроизводительных технологий всегда являлось неотъемлемой частью передовых научных исследований. Но наличие мощной вычислительной системы не всегда позволяет добиться хороших результатов, так как написание параллельных программ, их отладка и тестирование сами по себе являются очень трудоемкими задачами. В данной работе описан способ эффективного и мало затратного создания программ для фактически любого вычислительного комплекса. Особенно хорошо данный метод подходит для решения задач физики высоких энергий, так как все вычисления производятся в программной среде ROOT, которая изначально разрабатывалась в Европейском Центре Ядерных Исследований для решения соответствующих задач.

ЛИТЕРАТУРА:

1. Ускорительно-накопительный комплекс NICA. Технический проект / Под общ. ред. Мешкова И. Н. и Сидорина А. О. Дубна: ОИЯИ. 2009. 72 с.
2. Ускорительно-накопительный комплекс NICA - база фундаментальных исследований и инновационных разработок / Под общ. ред. Кекелидзе В. Д.; Сост.: Коваленко А. Д. и др. Дубна: ОИЯИ. 2011. 32 с.
3. Abraamyan Kh. U. et al. The MPD detector at the NICA heavy-ion collider at JINR // Nuclear Instruments and Methods in Physics Research. 2011. A628. P. 99-102.
4. Simulation and Analysis Framework for NICA/MPD Detectors. http://mpd.jinr.ru/.
5. ROOT - A Data Analysis Framework. http://root.cern.ch/.
6. Ресурсный вычислительный центр СПбГУ http://ptc.spbu.ru/hpc.
7. Bleicher M. et al. Relativistic Hadron-Hadron Collisions in the Ultra-Relativistic Quantum Molecular Dynamics Model // J. Phys. G: Nucl. Part. Phys. 1999. V25. P.1859-1896.
8. Dementyev A. V. and Sobolevsky N. M., SHIELD, a Monte Carlo Hadron Transport Code, Institute of Nuclear Research of Russian Academy of Science, Moscow, Russia.
9. Brun R. et al. Parallel interactive data analysis with PROOF // Nuclear Instruments and Methods in Physics Research. 2006. A559. P. 13-16.
10. STAR Collaboration. Correlation Structures from Soft and Semi-hard Components in pp Collisions at s= 200 GeV. // Acta Phys. Polon. 2005. B36. P. 353.