МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра дискретной математики и информационных технологий

МОДЕЛИРОВАНИЕ ПРОЦЕССОВ ПЕРЕНОСА НА СОПРОЦЕССОРАХ INTEL XEON PHI

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 271 группы
направления 09.04.01 — Информатика и вычислительная техника
факультета КНиИТ
Зимина Глеба Алексеевича

Научный руководитель
доцент, к. ф.-м. н. _____ А. Д. Панферов

Заведующий кафедрой
доцент, к. ф.-м. н. _____ Л. Б. Тяпаев

Саратов 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Применение массового параллелизма с использованием специализированных ускорителей является в настоящее время основным направлением развития вычислительных систем большой производительности. Это определяет основные вызовы при программировании приложений для таких систем. Эффективное использование аппаратного параллелизма в его разнообразных воплощениях и реализациях становится обязательным требованием при разработке программных решений, ориентированных на использование на вычислительных кластерах и массово параллельных системах.

Передо мной была поставлена задача адаптации вычислительного блока программного комплекса, обеспечивающего моделирование процессов переноса заряда в графене, для работы на вычислительных ускорителях Intel Xeon Phi. С выполнением тестирования его эффективности и демонстрацией работоспособности на примере моделирования результатов действия на образец графена мощного короткого импульса терагерцового диапазона.

Архитектура и состав программного обеспечения для сопроцессора Intel Xeon Phi ориентированы на выполнение высокопроизводительных приложений, способных максимально использовать возможность одновременного выполнения сотен потоков. Встроенное ПО позволяет использовать его в системах с шиной PCI Express, работающих под управлением операционных систем Linux или Windows. Библиотеки обеспечивают базовую функциональность, такую как определение сопроцессоров в системе, передачу данных между хостом и сопроцессором, загрузку исполняемых файлов на Xeon Phi и их запуск. Инструментальные средства позволяют управлять настройкам сопроцессора, получать информацию о его состоянии, обновлять энергонезависимую память. Поскольку сопроцессор работает под управлением собственной операционной системы возможен даже прямой терминальный доступ при использовании SSH. Однако исполняемый код для запуска на сопроцессоре требует специальной адаптации, в том числе обеспечивающей корректное взаимодействие с базовой системой.

В исходном вычислительном блоке основными ресурсоёмкими процессами являются процедуры численного решения систем обыкновенных дифференциальных уравнений. Они реализованы средствами библиотеки GSL из коллекции свободного ПО проекта GNU. Распараллеливание реализовано

средствами MPI.

На первом этапе я занимался изучением архитектурных особенностей, процедур и приёмов использования сопроцессора. Вариантов организации его взаимодействия с базовой системой, процедур обмена данными. Следующим этапом было знакомство с целевым программным модулем системы моделирования процессов переноса, который мне необходимо было переработать и адаптировать. Базовыми принципами и задачами работы всей программной системы. И выполнение работ по адаптации. Завершающим этапом работы было тестирование разработанного решения. Сначала это было сделано на специально подготовленных задачах различного уровня сложности и, затем, проведён полный цикл моделирования конечного состояния материала в условиях реалистичного по набору параметров внешнего воздействия.

Работа выполнялась с использованием аппаратных и программных ресурсов вычислительного кластера СГУ.

1 Постановка задачи

Целью представляемой работы является ускорение процедуры моделирования процессов переноса в графене за счет исполнения наиболее ресурсоемкой части кода на специализированных для выполнения математических операций сопроцессорах Intel Xeon Phi.

В качестве целевой программной системы, в составе которой предстоит функционировать разрабатываемому модулю, выступает программный комплекс, разработанный для такого моделирования с использованием итерационной процедуры вычисления функции распределения носителей заряда на адаптивной сетке с переменным шагом, строящейся в форме двумерного квадродерева.

В исходной версии программы системы вычисления реализовывались в параллельном режиме но с использованием только CPU. Программа написана на языке программирования Си. Распараллеливание реализовано средствами библиотеки MPI.

1.1 Адаптивная сетка

При решении задач оптимизации процедуры численного интегрирования часто применяется бисекция шага. Например, при использовании правила Рунге для оценки погрешности квадратурных формул и в адаптивных алгоритмах. Эта процедура обобщаема на двумерный и трехмерный случаи путем использования квадро - и окто - деревьев. Для рассматриваемого двумерного пространства (p_1,p_2) квадродерево позволяет определить его полное покрытие квадратами, отношение сторон которых принадлежит ряду значений 2^N, где $N = 1, 2, 3 \ldots$ ряд натуральных чисел. Значение функции в пределах каждого такого квадрата принимается равным её значению в центральной точке. Для обеспечения универсальности при работе с произвольными параметрами моделируемого процесса областью построения адаптивной сетки должна быть вся область определения функции распределения $-\pi \leqslant p_1 \leqslant \pi$, $-\pi \leqslant p_2 \leqslant \pi$. Она выступает в качестве корня квадродерева. В реализованной версии процедуры построения квадродерева центр корневого квадрата всегда размещается в точке $p_1 = 0$, $p_2 = 0$. Значение функции распределения в этой точке принимается равным нулю.

1.2 Архитектура программного комплекса

В коде программы используется специальная адаптированная система единицы, естественная для рассматриваемой задачи.

Физические параметры задачи задаются в отдельном файле `task_q.txt`. Все параметры задаются в естественной «графеновой» системе единиц. Кроме этих параметров, определяемых и задаваемых до этапа построения решения, существует несколько глобальных параметров, изменяющихся в процессе решения задачи. Они выделены в отдельный файл `task_globe.txt`. После вычисления функции распределения для списка узлов все полученные данные записываются во временный файл `calc_resalt_temp.txt`. И последней структурой является массив для хранения финальных результатов вычислений (файл `q_tree.txt`).

2 Архитектура Intel MIC

Xeon Phi – семейство x86 процессоров корпорации Intel с большим количеством процессорных ядер.

Данные процессоры предназначены для использования в суперкомпьютерах, серверах и высокопроизводительных рабочих станциях. Архитектура процессоров позволяет использовать стандартные языки программирования и технологии MPI и OpenMP.

В основе архитектуры Intel MIC лежит классическая архитектура x86, на ускорителе исполняется ОС Linux. Для программирования MIC предполагается использовать OpenMP, OpenCL, Intel Cilk Plus, специализированные компиляторы Intel Fortran, Intel C++. Также предоставляются математические библиотеки.

2.1 Архитектура Intel Xeon Phi

Сопроцессор Intel Xeon Phi включает до 61 процессорных ядер, соединенных высокопроизводительной встроенной кольцевой шиной. 8 контроллеров памяти обслуживают 16 каналов GDDR5, обеспечивая суммарную производительность 5,5 GT/s (миллиардов пересылок в секунду, при ширине шины 64 байта это дает пропускную способность 352 GB/s). Отдельный компонент реализует клиентскую логику PCI Express. Каждое ядро является полнофункциональным и поддерживает выборку и декодирование инструкций из 4 потоков команд.

2.2 Конвейер ядра Intel Xeon Phi

Ядра Intel Xeon Phi обеспечивают выполнение 32- и 64-битного кода, совместимого с архитектурой Intel64 без поддержки расширений MMX, AVX и SSE (всех версий). Блок векторных вычислений, содержащийся в каждом ядре, дополнительно реализует набор операций над 512-битными векторами.

Конвейер ядра Intel Xeon Phi содержит 7 этапов, блок векторных вычислений также имеет конвейерную структуру и состоит из 6 этапов, приведенные на рисунке 1. Все этапы основного конвейера кроме последнего (WB), поддерживают спекулятивное выполнение. Каждое ядро может выполнять инструкции 4 потоков, что позволяет уменьшить потери из-за латентности доступа к памяти, выполнения векторных инструкций и т.д. [3]

2.3 Иерархия памяти

Каждое ядро сопроцессора Intel Xeon Phi имеет собственные кэши L1 и L2, все ядра совместно используют оперативную память сопроцессора. Кэши L1 и L2 являются инклюзивными, то есть все данные, хранящиеся в кэше L1, хранятся также в кэше L2. В обоих кэшах при замещении используется псевдо-LRU алгоритм.

2.4 Intel Manycore Platform Software Stack (MPSS)

Архитектура и состав программного обеспечения для сопроцессора Intel Xeon Phi ориентированы на выполнение высокопроизводительных приложений, способных максимально использовать возможность одновременного выполнения сотен потоков. Встроенное ПО позволяет использовать его в системах с шиной PCI Express, работающих под управлением операционных систем Linux или Windows.

Библиотеки обеспечивают базовую функциональность, такую как определение сопроцессоров в системе, передачу данных между хостом и сопроцессором, загрузку исполняемых файлов на Xeon Phi и их запуск. Инструментальные средства позволяют управлять настройкам сопроцессора, получать информацию о его состоянии, обновлять его флеш-память и т.д.; с помощью ssh можно получить терминальный доступ к сопроцессору для запуска на нем программ.

2.5 Модели использования сопроцессора Intel Xeon Phi

Архитектура Intel Xeon Phi поддерживает несколько режимов использования сопроцессора, которые можно комбинировать для достижения максимальной производительности в зависимости от характеристик решаемой задачи. Процесс может быть запущен как в операционной системе базовой системы, так и в ОС сопроцессора; в зависимости от режима использования могут использоваться вычислительные мощности только процессоров базовой системы, либо только сопроцессора, либо процессоров базовой системы и сопроцессора совместно. [5]

Поддерживается два режима выполнения приложений: режим Offload и режим MPI.

2.6 Создание приложений для архитектуры MIC

Разработка приложений для архитектуры Intel Xeon Phi требует наличия тех же знаний и навыков, что и разработка параллельных приложений для распределенных многоядерных систем. Можно использовать следующие программные инструменты:

- средства разработки "Intel Parallel Studio XE 2013 "Intel Cluster Studio XE 2013 "Intel(R) SDK for OpenCL Applications XE 2013 Beta gcc (в настоящий момент не поддерживает векторные инструкции) и др.;
- библиотеки Intel Math Kernel Library (Intel MKL), Intel Threading Building Blocks (Intel TBB), Intel Integrated Performance Primitive (Intel IPP), входящие в состав средств разработки Intel, а также Intel MPI for Linux, MPICH2, Boost и др.
- отладчики (Intel Debugger, gdb, totalview), профилировщики (входят в состав средств разработки Intel), средства виртуализации (xen) и т.д.

2.7 Векторизация

Для того чтобы приложение эффективно использовало вычислительные возможности сопроцессора Intel Xeon Phi, необходимо выполнение двух важных условий: приложение должно обладать высокой степенью параллельности, а так же иметь возможности для векторизации своего кода. Сделать свой код векторным, используя компилятор компании Intel, можно следующими способами:

- В некоторых простых случаях компилятор может сам векторизовать ваш код, дополнительно ему можно давать рекомендации;
- Можно использовать возможности параллельного расширения Intel Cilk Plus (SIMD директивы, элементарные функции и специальную технологию Array Notation для массивов) для самостоятельной векторизации кода;
- Можно воспользоваться библиотеками с уже векторизованным кодом, например, Intel MKL. Следует, понимать, что использование подобных библиотек не всегда приводит к ускорению вашего кода.
- Можно использовать язык ассемблера с векторными инструкциями для оптимизации критичных участков кода, либо оболочки этих инструкций в виде функций языка Си (intrinsics). Существуют также библиотеки

классов SIMD, которые являются надстройкой более высокого уровня над векторными командами процессора.

2.8 Ускорение приложений

Одним из существенных моментов, на которые следует обратить внимание при улучшении приложений для Intel Xeon Phi, является балансировка нагрузки.

Сопроцессор Intel Xeon Phi позволяет запускать одновременно 4 логических потока на ядро. Однако часто бывает эффективнее запускать меньшее их количество, т.к.:

— это позволяет минимизировать нагрузку на кэши разных уровней (L1, L2, TLB), т.к. если потоков на ядре много, они начинают соперничать за доступ в кэш;

— меньше соревнований между потоками за единственный векторный модуль ядра;

— уменьшаются запросы к основной памяти.

3 Практическая часть

В исходной версии программной системы имелась собственный модуль программы, выполняющая вычислений функций распределения f_1 и вспомогательных функций f_2 и f_3 на обычных ЦП с использованием технологии MPI. Время выполнения программы для обсчета 1300 узлов на четырех ядерном процессоре Intel(R) Xeon(R) CPU E5-2603 v2 с использованием 4 потоков составляет около 6 минут.

Сопроцессор Intel Xeon Phi, который необходимо было использовать, имеется на кластере СГУ, поэтому отлаживание и запуск всех программ производился на кластере. Здесь имеется несколько сопроцессоров Xeon Phi модели 5110P/5120D. Они имеют 60 ядер и 240 потоков, 8 гигабайт GDDR5 памяти, при этом могут достигать максимальной производительности в 1010 GFLOPS.

3.1 Запуск кода на сопроцессоре

После тестирования различных вариантов запуска, было предложено запускать программу полностью на сопроцессоре, то есть использовать режим Co-processor-only, в этом случае сопроцессор будет рассматриваться как отдельный многоядерный компьютер. Использование 60 ядер и 240 потоков должно довольно сильно ускорить работу программу.

Intel для работы с Xeon Phi рекомендует использовать созданный ими компилятор **icc**. Чтобы запустить код на сопроцессоре нам необходимо использовать специальный флаг компиляции **-mmic**, кроме этого необходимо указать используемые библиотеки. Компиляция кода будет выглядеть следующим образом:

icc -L/home/gzimin/gsl-2.4/usr/local/lib/ -I/home/gzimin/gsl-2.4/ usr/local/include/ -fopenmp -lgsl -lgslcblas -mmic diploma_calc.c -o calc_mic_native

3.2 Доработка функционала вычислительного модуля

Основные вычисления в заданной программе выполняются в функции `ode_calc()`, которая используется данные из файлов `q_tree` и `task_q`. В главной функции `main` из файла `q_tree` в цикле `while` построчно считываются все данные. Затем они подаются на вход функции `ode_calc()`, которая уже работает с этими данными и записывает их в выходной файл

`calc_result_temp`.

Необходимо переписать код так, чтобы данные построчно считывались и передавались главной счетной функции в цикле `for`, что и было реализованно.

Для цикла `for` необходимо знать количество строк в исходном файле. Поэтому перед основным циклом нужно еще раз прочитать файл, и в отдельную переменную записать количество строк. Теперь можно заменить цикл `while` на `for` и с использованием директив OpenMP распараллелить основной цикл.

Для визуализации результатов программы было решено строить несколько видов графиков.

3.3 Время работы программы

Для сравнения времени работы были использованы входные данные с различным количеством узлов: от 20 до 20 000. Тестирование проводилось при различном количестве потоков. Были использованы инструменты библиотеки OpenMP для вычисления времени выполнения работы программы.

3.4 Тестирование функционирования реализованного модуля в составе полного програмного комплекса

Теперь, имея готовую обновленную программу, считающую функцию распределения носителя заряда в графене, можно реализовать полный цикл моделирования (рассчета функции распределения) для реалистичной физической задачи.

Оценочное тестирование в составе программного комплекса выполнялось на примере определения вида функции распределения носителей заряда, формируемой в результате действия на образец графена короткого импульса линейно поляризованного электромагнитного излучения с частотой 2 ТГц и амплитудой напряженности электрического поля 300 В/см.

3.5 Демонстрация работы на реальной задаче

Основываясь на полученных графиках можно сделать вывод, что максимально значения функции распределения было достигнуто после второй итерации, то есть в девятом поколении. После этого новые значения функции только уменьшались.

Это можно интерпретировать как дополнительную детализацию поведения функции распределения в области её максимальных значений. Но анализ итерационной работы всего программного комплекса выходит за рамки моей работы. На этом этапе можно только констатировать, что представленные результаты работы с реалистичными параметрами модели демонстрируют адекватную, соответствующую ожидаемой физической картине процесса, работу реализованной версии вычислительного модуля. Представленные выше скоростные характеристики позволяют говорить об очень существенном, на один – два порядка, сокращении временных затрат на вычисление значений функции распределения f_1 и вспомогательных функций f_2 и f_3 в назначаемых узлах покрывающей адаптивной сетки. Эти оценки в силу закона Амдала нельзя рассматривать в качестве ожидаемого ускорения работы всего программного комплекса. Тем не менее общее ускорение работы оказывается очень существенным и, по предварительным оценкам, может достигать 4 – 12 раз. Такой результат, в том числе, был обусловлен и расширением функциональных возможностей программного модуля путем включения в него процедур прямой записи результатов счета в файл `q_tree.txt` вместо вывода во временный файл с последующей обработкой специальной программной процедурой. Представленные модификации отражены в прилагаемом коде.

ЗАКЛЮЧЕНИЕ

В ходе подготовки и написанию ВКР мне пришлось ознакомиться с большим объёмом материала по предметной области, для моделирования процессов в которой было необходимо разработать программное решение, и по составу, структуре и внутренним интерфейсам программного комплекса, в который предстояло интегрировать разрабатываемую мной программу. Эта работа кратко отражена в тексте, представляемом на защиту.

Основные проблемы, которые предстояло решить, были связаны с освоением программных технологий, необходимых для переноса вычислительно сложных элементов кода на специализированные математические сопроцессоры Intel Xeon Phi. Такой перенос был главной целью работы. Он обеспечил возможность эффективно использовать узлы кластера СГУ, оснащенные такими сопроцессорами. При этом производительность таких гетерогенных узлов на моделировании процессов переноса, реализуемом с использованием разработанного программного модуля, оказывается почти на порядок больше производительности использовавшихся ранее CPU узлов. Это обеспечивает высокую скорость и энергоэффективность работы программного комплекса, предоставляет возможность просчитывать более детальные модели процессов.

Поставленные задачи были решены в полном объёме.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 S.A. Smolyansky, A.D. Panferov, D.B. Blaschke, N.T. Gevorgyan. “Nonperturbative kinetic description of electron-hole excitations in graphene in atime dependent electric field of arbitrary polarization”,Particles,2:2 (2019),pp. 208–230. (дата обращения: 4.2020)

2 А. Д. Панферов, А. В. Маханьков, А. А. Трунов. «Использование адаптивной сетки на основе квадродерева для моделирования конечного состояния квантово-полевой системы при импульсном внешнем воздействии». Программные системы: теория и приложения, 2020,11:1(44), с. 79–92. (дата обращения: 4.2020)

3 Архитектура Intel Many Integrated Core [Электронный ресурс]. URL: https://www.intel.ru/content/www/ru/ru/architecture-and-technology/many-integrated-core/intel-many-integrated-core-architecture.html (дата обращения 4.2020) Загл. с экрана. Яз. рус.

4 Anand’s Hardware Tech Page.[Электронный ресурс]. URL: https://www.anandtech.com/show/14305/intel-xeon-phi-knights-mill-now-eol Загл. с экрана. Яз. англ.

5 Новости технологий, обзоры гаджетов, смартфонов, бытовой техники и автомобилей. [Электронный ресурс]. URL: https://www.ixbt.com/news/hard/index.shtml?15/92/47l (дата обращения 4.2020) Загл. с экрана. Яз. рус.

6 Resource Guide for Intel® Xeon Phi™ Coprocessor Developers.[Электронный ресурс]. URL: https://software.intel.com/content/www/us/en/develop/articles/quick-start-guide-for-the-intel-xeon-phi-coprocessor-developer.html (дата обращения: 4.2020)

7 J. Reinders, J. Jeffers. «Intel Xeon Phi Coprocessor High Performance Programming». 2013, pp.250-255. (дата обращения 4.2020)

8 R. Rahman. «Intel® Xeon Phi™ Coprocessor Architecture and Tools. The Guide for Application Developers». 2013, pp.48-52. (дата обращения 2.2020)

9 GNU Scientific Library. Ordinary Differential Equations. [Электронный ресурс]. URL: https://www.gnu.org/software/gsl/doc/html/ode-initval.html (дата обращения: 3.2020)

10 GNU Scientific Library. Design document. [Электронный ресурс]. URL: https://www.gnu.org/software/gsl/design/gsl-design.html#SEC1 (дата обращения: 3.2020)

11 Сервер компьютерных классов факультета информационных технологий НГУ. Параллельное программирование на OpenMP. [Электронный ресурс]. URL: http://ccfit.nsu.ru/arom/data/openmp.pdf (дата обращения: 11.2018)

12 The OpenMP API specification for parallel programming. [Электронный ресурс]. URL: https://www.openmp.org/wp-content/uploads/openmp-4.5.pdf (дата обращения: 4.2020)

13 Национальный университет ИНТУИТ. [Электронный ресурс]. URL: https://www.intuit.ru/studies/courses/10611/1095/lecture/22908 (дата обращения 4.2020) Загл. с экрана. Яз. англ.

14 Intel Product Specifications.[Электронный ресурс]. URL: https://ark.intel.com/content/www/ru/ru/ark/products/94035/intel-xeon-phi-processor-7250-16gb-1-40-ghz-68-core.html (дата обращения 5.2020) Загл. с экрана. Яз. рус.

15 The C++ Resource Network. [Электронный ресурс]. URL: https://www.cplusplus.com/reference/cstdlib/qsort/ (дата обращения 4.2020) Загл. с экрана Яз. англ.

16 Gnuplot documentation. [Электронный ресурс]. URL: http://gnuplot.sourceforge.net/docs_4.2/node62.html (дата обращения 4.2020) Загл. с экрана Яз. англ.

17 Образовательный комплекс «Введение в принципы функционирования ипримененния современных мультиядерных архитектур (на примере Intel Xeon Phi)». Лекция №3. Выполнение программ на Intel Xeon Phi.[Электронный ресурс]. URL: http://hpc-education.unn.ru/files/courses/XeonPhi/Lection03.pdf (дата обращения 4.2020) Загл. с экрана Яз. англ.

18 Intel®Xeon Phi™ Processor Software User's Guide. [Электронный ресурс]. URL: https://registrationcenter-download.intel.com/akdlm/irc_nas/11673/xppsl_user_guide.pdf (дата обращения 3.2020) Загл. с экрана Яз. англ.

19 ResearchGate. Discover scientific knowledge and stay connected to the world of science. A Survey on Evaluating and Optimizing Performance of Intel Xeon Phi. [Электронный ресурс]. URL: https://www.researchgate.net/publication333384596_A_Survey_on_Evaluating_and_Optimizing_Performance_of_Intel_Xeon_Phi (дата обращения 4.2020) Загл. с экрана Яз. англ.

20 PCWorld. Intel Tests Chip Design With 80-Core Processor. [Электронный ресурс]. URL: https://www.pcworld.com/article/128924/article.html (дата обращения 4.2020) Загл. с экрана Яз. англ.