1.6. Примеры применения современных ИКТ в науках о Земле, науках о Живом и в образовании.

Скважинный каротаж.
Поиск мотивов в геноме.
Молекулярная динамика.
Виртуальная студия.

Моделирование реальных данных

Разложение сейсмических кубов по волновым пакетам
Математическая постановка
Сложности реализации
Результаты
Задача поиск мотивов
Современные аппаратные средства
Адаптация программы под GPU
Адаптация программы под FPGA
Результаты

Моделирование плазмы
Актуальность данной работы связана с необходимостью точного расчета характеристик неустойчивостей, возбуждаемых релятивистским электронным пучком в субтермоядерной плазме на установке ГОЛ-3 (ИЯФ СО РАН), а также выяснения причин возникновения аномальной электронной теплопроводности [1]. В силу того, что плазма в данном случае является существенно неравновесной, использование упрощенных (например, магнитогидродинамических) методов невозможно, расчет необходимо производить в рамках кинетического подхода, то есть с помощью метода частиц в ячейках, что требует больших вычислительных ресурсов. Конкретной физической задачей в данном случае является расчет спектра плазменной турбулентности. Для ее решения необходимо проведение трехмерных расчетов с высоким разрешением по всем трем координатам в силу того, что плазменные колебания в данном случае являются существенно анизотропными. Для описания динамики плазмы решается уравнение Власова в бесстолковительной форме и уравнения Максвелла. Используемые методы решения этих уравнений описаны в статьях [2, 3, 4].

Рассмотрим следующую постановку задачи. В начальный момент в трехмерной области решения (размер области L), которая имеет форму прямоугольного параллелепипеда:
$0 <= x <= L_X$, $0 <= y <= L_Y$, $0 <= z <= L_Z$
находится плазма, состоящая из электронов и ионов. Модельные частицы распределены по области равномерно. Задается плотность плазмы и температура электронов, температура ионов считается нулевой. Дополнительно в области присутствуют электроны пучка, которые также распределены по области равномерно (предполагается, что пучок уже вошел в расчетную область). Электроны пучка отличаются от электронов плазмы тем, что они имеют кинетическую энергию направленного движения e = 1 МэВ, а их температура равна нулю. Модельные частицы, соответствующие электронам пучка, имеют меньшую массу, нежели модельные частицы, соответствующие электронам плазмы (отношение их масс равно отношению плотности плазмы и плотности пучка). Итак, исходными параметрами задачи являются: плотность и

температура электронов плазмы, отношение плотности электронов плазмы к плотности электронов

пучка, энергия электронов пучка:

• плотность электронов плазмы n0 = 1014 см3.
• температура электронов плазмы T0 = 1 КэВ
• отношение плотности электронов пучка к плотности электронов плазмы _ = 10−3
• энергия электронов пучка " = 1 МэВ

Распараллеливание выполнено методом декомпозиции расчетной области по направлению, перпендикулярному направлению движения электронного пучка. Используется смешанная эйлерово–лагранжева декомпозиция. Сетка, на которой решаются уравнения Максвелла, разделена на одинаковые подобласти по одной из координат. С каждой подобластью связана группа процессоров (в том случае, когда вычисления производятся на многоядерных процессорах, процессором для единообразия будет именоваться отдельное ядро). Далее, модельные частицы каждой из подобластей разделяются между процессорами связанной с этой подобластью группы равномерно, вне зависимости от координаты.

Каждый из процессоров группы решает уравнения Максвелла во всей подобласти. Далее решаются уравнения процессоров группы производит обмен граничными значениями тока и полей с соседними подобластями, и затем рассылает полученные граничные значения всем процессорам своей группы.

ЭФФЕКТИВНОСТЬ РАСПАРАЛЛЕЛИВАНИЯ

Основная цель создания параллельной программы для моделирования аномальной теплопроводности в плазме _ возможность счета на больших сетках и с большим количеством модельных частиц. Поэтому эффективность распараллеливания вычислялась следующим образом

$$k = T_2/T_1 \times N_1/N_2 \times S_2/S_1 \times 100\% \quad (1)$$

здесь T_1 - время счета с использованием N_1 процессоров, T_2 – время счета с использованием N_2 процессоров, S1, S2 – характерные размеры задач, в данном случае число узлов сетки по координате X. При этом размер задачи увеличивается пропорционально числу процессоров, т.е. нагрузка на отдельный процессор не возрастает. Цель такого определения эффективности – понять, насколько увеличивается время счета при увеличении числа процессоров и неизменной нагрузке на один процессор. В идеале время должно остаться тем же самым (k = 100% в идеале).

За основу для сравнения при этом берется модель, требующая большого времени вычислений (сетка 256×128×128 узлов, 150 частиц в ячейке). В расчетах по определению эффективности увеличивается только размер сетки по X, все остальные параметры остаются неизменными. Вопрос о том, насколько быстрее можно посчитать на суперкомпьютере задачу, которую можно посчитать и на настольной машине, в данной работе не рассматривается: такие задачи не представляют интереса с физической точки зрения.

Из этого рисунка можно сделать вывод, что созданная программа способна эффективно использовать более тысячи процессорных ядер. С другой стороны, падение ффективности довольно заметное, и очевидно, что используемая двумерная декомпозиция не сможет обеспечить эффективность для десятков тысяч и более ядер.

С помощью программы Intel Trace Analyzer & Collector была проведена профилировка программы, результаты представлены в таблице (расчет на 512 ядрах, 120 временных шагов) 3. Каждый из процессов считает движение своей части модельных частиц. Затем происходит суммирование значений тока, полученных по каждой группе частиц процедура MPI_Allreduce). Таким образом получается, что группа процессоров, отвечающая за определенную подобласть, работает как SMP-компьютер. Учитывая тот факт, что движение модельных частиц вычисляется полностью независимо, а также то, что данные,

необходимые для расчета движения одной модельной частицы, имеют очень небольшой объем, отсюда можно сделать вывод, что замена группы процессоров при расчете отдельной подобласти на GPU, могла бы значительно повысить скорость счета. Основной вывод, который следует из таблицы 3: процедуры MPI, все взятые вместе, занимают немного времени по сравнению с расчетными процедурами, и поэтому оптимизацию программы по скорости нужно проводить, ускоряя именно расчетные процедуры. Среди расчетных процедур самой затртной является процедура расчета движения модельных частиц (от 60 % до 90 % времени в зависимости от компьютера).

СОКРАЩЕНИЕ ВРЕМЕНИ СЧЕТА: РАСЧЕТ НА ГРАФИЧЕСКИХ УСКОРИТЕЛЯХ

Расчет динамики частиц, выполненный на графической карте GeForce 9400, на 16 ядрах, сам по себе показал значительное повышение производительности по сравнению с расчетом на процессорах Xeon или Nehalem. Необходимо отметить, что время счета включает в себя также время обращения к памяти, и таким образом сильно различается на разных системах (аналогично, преимущество графических ускорителей достигается за счет быстрой локальной памяти).

Таким образом, расчет с помощью одного ядра графической карты происходит на два порядка быстрее. Приведенные цифры относятся только к реализации метода частиц и не могут быть использованы для оценки и сравнения производительности указанных систем в общем случае.

Относительно перспектив реализации метода частиц на перспективных гибридных суперЭВМ необходимо сделать два замечания. Во-первых, для этого необходимо упорядочивание частиц по ячейкам, так чтобы один поток графической карты работал бы только с частицами внутри одной ячейки, или компактной группы ячеек. Во-вторых, все пересылки между подобластями должны выполняться только отдельно от вычислений: один, специально выделенный процессор взаимодействует с графическим ускорителем, другой одновременно с этим выполняет пересылки. Важно отметить, что вычислительные потоки CUDA ни в коем случае не выполняют пересылок (не возникает такой необходимости), и никак не взаимодействуют с потоками, связанными с другой подобластью.

Таким образом, если использовать графический ускоритель как замену группы процессоров (раздел 1.2), используемых для расчета движения модельных частиц в рамках одной подобласти, то можно во-первых, значительно ускорить расчет, во вторых, тем самым полностью исключаются коммуникации между потоками, связанными с одной подобластью (внутри одного графического ускорителя), что же касается пересылок между подобластями (связанными с разными графическими ускорителями), то они не могут быть более медленными, чем сейчас, когда данные пересылаются между группами процессоров, из-за меньшего количества физических связей в суперкомпьютере.

Рассмотрим пересылку частиц между двумя соседними подобластями. Если сейчас с одной подобластью связано, например, 8 процессов (каждый занимает одно процессорное ядро), а пересылки частиц из одной подобласти в другую выполняются только между двумя выделенными процессами (условно .головными. процессами), то необходимо выстраивать маршрутизацию для сообщений в системе, физически состоящей из 16 ядер (или, с точки зрения MPI, из 16 потоков). В том случае, когда расчет выполняется на графическом ускорителе (в 8, 100, 200 потоков CUDA...), а пересылки будут выполнятся только между двумя MPI-потоками. Вероятно, такая пересылка займет не больше времени, чем в первом случае.

ОПТИМИЗАЦИЯ СЧЕТА НА GPU

Указанная в таблице 2 производительность была получена после проведения нескольких этапов оптимизации программы. Были учтены как особенность архитектуры графических

процессоров, так и используемых алгоритмов расчета движения частиц. Напомним, что процесс расчета состоит из двух стадий: пересчет поля и движения частиц.
Предварительный анализ программы показал, что основное время тратится на движение частиц и именно этот этап расчетов нуждается в оптимизации.
После того, как исходный код программы был адаптирован для выполнения на графических процессорах и произведена его отладка, была получена производительность в 30 GFlops, что всего лишь в 27 раз быстрее, чем программа для CPU (Xeon). Анализ программы с помошью средств профилирования показал, что реализуется большое количество промахов по кэшу на чтение и запись, есть ветвления, которые приводят к существенному замедлению и ряд других проблем.
Большое количеств промахов объясняется непоследовательностью обращения к памяти. Было произведено переупорядочивание частиц по ячейкам (т.е. привязка частиц к ячейкам, или определение набора частиц, принадлежащих каждой конкретной ячейке). Следует отметить, что не используется сортировка в обычном понимании этого слова, в силу того, что сложность алгоритмов сортировки не менее чем O(N logN), что неприемлемо для метода частиц в ячейкам (при $N = 10^9$). Сложность алгоритма привязки частиц к ячейкам - O(N). Использование привязки частиц к ячейкам само по себе дает ускорение расчета движения частиц в 2 раза даже на Xeon. Но это не позволило полностью избавиться от промахов по кэшу. Переход к текстурам позволил еще больше улучшить эту ситуацию, обеспечив дополнительное ускорение в 5-6 раз 3. Следует отметить, что несмотря на то, что текстуры не позволяют хранить данные типа double, собственно интегрирование производится с двойной точностью. В результате расчет движения частиц был ускорен в 600 раз.
В процессе расчетов используется большое количество значений, которые являются постоянными во время запуска ядра. Таких значений довольно много и была сделана попытка вынесения их в константную память.
Была произведена оптимизация и в плане использования более быстрых мат.операций. Так, ряд делений было заменено на умножение, где это возможно, что дало общее повышение производительности на 15 %.
Использование CUDA Occupancy Calculator показало, что рост производительности упирается в количество регистров, которые нужны ядру во время выполнения. Переписыванием алгоритма было уменьшено количество регистров. Также явным образом было ограничено число доступных ядру регистров во время компиляции, что дало ускорение на 200%.

ЗАКЛЮЧЕНИЕ

Ускорение расчета движения модельных частиц было достигнуто в результате выполнения следующих действий:

- Адаптация программы для выполнения на графических процессорах, ускорение в 27 раз по сравнению с одним ядром процессора Xeon
- Переупорядочивание частиц по ячейкам, ускорение в 2 раза
- Переход к доступу к атрибутам частиц через текстуры, ускорение в 6 раз
- Ограничение число доступных ядру регистров во время компиляции, ускорение в 2 раза.

Итоговое ускорение для процедуры расчета движения частиц составило ~ 600 на графическом ускорителе Tesla по сравнению с одним ядром процессора Xeon.