УДК 52-423:004.942 Вестник СПбГУ. Сер. 1. Т. 1 (59). 2014. Вып. 4

ИСПОЛЬЗОВАНИЕ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ ДЛЯ МОДЕЛИРОВАНИЯ САМОГРАВИТИРУЮЩИХ СИСТЕМ

И. А. Сивков-Енин

Санкт-Петербургский государственный университет,
Российская Федерация, 199034, Санкт-Петербург, Университетская наб., 7-9

В работе проводится исследование производительности графических процессоров в задаче моделирования самогравитирующих систем. Был сделан вывод о принципиальной возможности моделирования карликовых галактик в масштабе один к одному при использовании разумных аппаратных мощностей. Библиогр. 10 назв. Ил. 6.

Ключевые слова: динамика галактик, графические процессоры, численное моделирование.

Введение

Бурное развитие вычислительной техники, особенно в последние годы, предоставляет возможность проведения численного моделирования самогравитирующих систем с большим количеством частиц. Применительно к звездным системам оно может использоваться для решения широкого круга задач: изучение формирования, динамических свойств и структуры галактик, которые меняются в ходе эволюции, происходящих в них коллективных релаксационных процессов, определение причин возникновения спиральных неустойчивостей и баров и т. п. Во многих случаях численное моделирование является единственным способом решения данных задач, так как лежащие в их основе системы нелинейны и имеют большое число степеней свободы, что делает затруднительным применение аналитического аппарата.

Несмотря на активное применение астрофизиками данного подхода, в области моделирования звездных систем остается много нерешенных проблем. Одна из основных состоит в том, что в реальных галактиках количество звезд может достигать 10^{12}. Уровень развития современных компьютеров не позволяет моделировать настолько большие системы, поэтому при численном эксперименте количество частиц снижается на несколько порядков. При этом, чтобы моделируемая система являлась бесстолкновительной, как и галактики, приходится использовать так называемое сглаживание гравитационной силы для относительно малых расстояний между частицами. Другой особенностью, возникающей при использовании численных методов, является дискретность состояний системы по времени. Совершенно ясно, что данные приближения могут отрицательно сказаться на качестве получаемых результатов. Чтобы свести к минимуму влияние вышеперечисленных негативных факторов, необходимо максимально возможно увеличить число частиц и уменьшить шаг по времени, это, в свою очередь, требует значительного увеличения производительности расчетов.

На данный момент одно из самых перспективных направлений — использовать графические процессоры, которые последние несколько лет активно применяются для расчетов в приложениях, не связанных с графикой, и зачастую справляются с ними лучше, чем центральный процессор [1]. Такая техника использования видеоадаптера получила название GPGPU (**General-purpose** computing on **graphics processing units**). Данное направление активно развивается и поддерживается основными производителями видеокарт (NVIDIA, AMD). Компания NVIDIA разработала специальную технологию CUDA, облегчающую написание программ для GPGPU, и спроектировала семейство вычислительных систем Tesla на основе архитектуры графических

процессоров, в чью задачу входят исключительно научные и технические вычисления. Одним из направлений, в которых с недавнего времени находят свое применение графические ускорители, является моделирование самогравитирующих систем.

1. Особенности GPGPU и технологии CUDA

1.1. История развития GPGPU. Устройства для превращения персональных компьютеров в маленькие суперкомпьютеры известны довольно давно. В начале 80-х годов прошлого века в Великобритании были разработаны так называемые транспьютеры — процессоры с сокращенным набором команд. Первое время их производительность в соответствующих задачах впечатляла, но затем рост быстродействия универсальных процессоров ускорился, они усилили свои позиции в параллельных вычислениях. В итоге транспьютеры как вычислительные приборы остались в прошлом, однако следы некоторых идей, использованных при их создании, можно обнаружить и в современных системах [2].

С начала XXI века параллельные вычисления начали массово применяться для разработки трехмерных компьютерных игр, появились первые графические процессоры (GPU). Тогда же появились первые технологии неграфических расчетов GPGPU. Результаты первых экспериментов прочили большое будущее GPU-вычислениям. Однако существовал ряд ограничений на использование ресурсов графических процессоров, в частности, необходимость использовать специализированные языки и программные интерфейсы (например OpenGL) для работы с компьютерной графикой при написании вычислительных приложений [3].

В дальнейшем два основных производителя видеокарт, NVIDIA и AMD, разработали и анонсировали соответствующие платформы под названием CUDA (Compute Unified Device Architecture) и AMD Stream Computing соответственно. В отличие от предыдущих концепций программирования для графических процессоров, эти были выполнены с целью прямого доступа к аппаратным возможностям видеокарт и ликвидировали многие важные ограничения предыдущих технологий GPGPU [4].

1.2. Разница между центральным и графическим процессорами. Продаваемые сейчас процессоры содержат лишь до шестнадцати ядер и предназначены для обычных приложений, при этом каждое ядро работает отдельно от остальных, исполняя разные инструкции для разных процессов. Современные видеоадаптеры содержат несколько мультипроцессоров с сотнями вычислительных ядер. И эти ядра исполняют одни и те же инструкции одновременно; такой стиль программирования является обычным для графических алгоритмов и многих научных задач, но требует специфического программирования. Зато это подход позволяет увеличить количество вычислительных блоков за счет их упрощения. Также на видеоадаптерах применяется более быстрая память, что весьма важно для параллельных расчетов, оперирующих с огромными потоками данных [3, 4].

В итоге основой для эффективного использования GPU в научных и иных неграфических расчетах является распараллеливание алгоритмов на сотни ядер, имеющихся в видеокартах. Использование нескольких GPU дает еще больше вычислительных мощностей для решения подобных задач, при этом лучшие результаты достигаются, если число арифметических инструкций значительно превосходит число обращений к памяти [3]. Этим критериям в полной мере удовлетворяет задача N тел в применении к звездным системам: число вычислений силы притяжения растет как N^2, при этом для каждой частицы расчет силы можно проводить независимо от других.

1.3. Технология CUDA. Технология CUDA — это программно-аппаратная вычислительная архитектура компании NVIDIA, основанная на расширении языка C или Fortran, которая дает возможность организации доступа к набору инструкций графического ускорителя и управления его памятью при организации параллельных вычислений.

Конечно же, открытые стандарты, использующие OpenGL, кажутся наиболее универсальными; они позволяют применять один и тот же код для видеокарт разных производителей. Но у таких методов есть масса недостатков, они значительно менее гибкие и не такие удобные в использовании. Кроме того, они не дают использовать специфические возможности определенных видеокарт, такие как присутствующая в современных вычислительных процессорах специальная быстрая память и возможность взаимодействия между параллельно исполняющимися нитями [1]. Важно, что поддержка NVIDIA CUDA есть в видеокартах GeForce серий 8 и старше, которые очень широко распространены. Технологии CUDA доступна на 32-битных и 64-битных операционных системах Linux, Windows и MacOS X.

Модель CUDA отводит GPU роль параллельного сопроцессора. При этом программа, написанная на языке CUDA, использует CPU для выполнения последовательной части кода и подготовительных стадий для вычисления на графическом процессоре. Параллельные участки программы переносятся на GPU, где выполняются большим количеством нитей (*threads*). В случае задачи N тел на GPU производится вычисление силы притяжения между частицами. Для удобства работы с нитями, они разбиваются на одно-, двух- или трехмерные блоки. Блоки в свою очередь компонуются в сетку (*grid*), которая также может иметь до трех измерений. Таким образом, каждая нить и каждый блок имеют трехмерные координаты. В контексте каждой нити значения координат доступны через переменные *threadIdx.x,threadIdx.y,threadIdx.z* и *blockIdx.x, blockIdx.y, blockIdx.z*, а значения размерностей — через *blockDim.x, blockDim.y, blockDim.z u gridDim.x, gridDim.y, gridDim.z*. Взаимодействия между нитями возможно только внутри одного блока через быструю разделяемую память.

```
__global__ void vecAdd(double *a, double *b, double *c, int n)
{

    int id = blockIdx.x*blockDim.x+threadIdx.x;


    if (id < n)
        c[id] = a[id] + b[id];
}
```

Рис. 1. Пример функции, выполняющейся на графическом процессоре. Язык CUDA C/C++.

На рисунке 1 представлен пример функции, выполняющейся на графическом процессоре, которая складывает два вектора. Для каждой нити вычисляется уникальный индекс id в зависимости от ее координат в блоке и координат самого блока, затем элементы векторов с соответствующим индексом складываются и записываются в элемент результирующего вектора.

Перед вызовом функции необходимо выделить память на графическом ускорителе с помощью команды cudaMalloc и перенести в эту память данные с помощью команды cudaMemcpy. Вызов функции имеет следующий вид: vecAdd<<<gridSize,

blockSize>>>(a, b, c, n), где gridSize и blockSize — размерности сетки и блока, которые программист должен задавать сам. Общее количество вызываемых нитей равняется произведению всех размерностей (в общем случае $\prod_{k=x,y,z} blockDim.k \cdot gridDim.k$) и должно соответствовать размерности векторов n.

2. Задание начальных условий и тестирование программного обеспечения

Нами на основе технологии CUDA было разработано программное обеспечение, которое позволяет использовать ресурсы графического процессора для расчета динамической эволюции самогравитирующей системы. Чтобы провести моделирование галактики, мы взяли в качестве начальных условий модель, которую предложил Хернквист [5]. Ее суть состоит в получении самосогласованной системы, состоящей из трех компонентов: балджа, диска и гало. Координаты частиц всех компонентов задаются в соответствии с законами распределения плотности, описанными в [5]. Для этого используется метод моделирования распределения случайной величины. Например, метод отбора-отказа, относящийся к семейству методов Монте-Карло [6]. Далее находятся начальные скорости частиц с помощью уравнения Джинса. При этом для диска используется шварцшильдовское распределение и эпициклическое приближение, а сферические компоненты имеют изотропное максвелловское распределение скоростей (см. [5, 7]). Полученная в итоге самосогласованная модель галактики развивается динамически путем решения уравнений движения с помощью численных методов, например модифицированного алгоритма Эйлера с перешагиванием (leapfrog) (см. [8]).

Для проверки программного обеспечения нами были проведены тестовые расчеты с целью сравнения с аналогичными, которые были проделаны другими авторами с использованием «классических» процессоров.

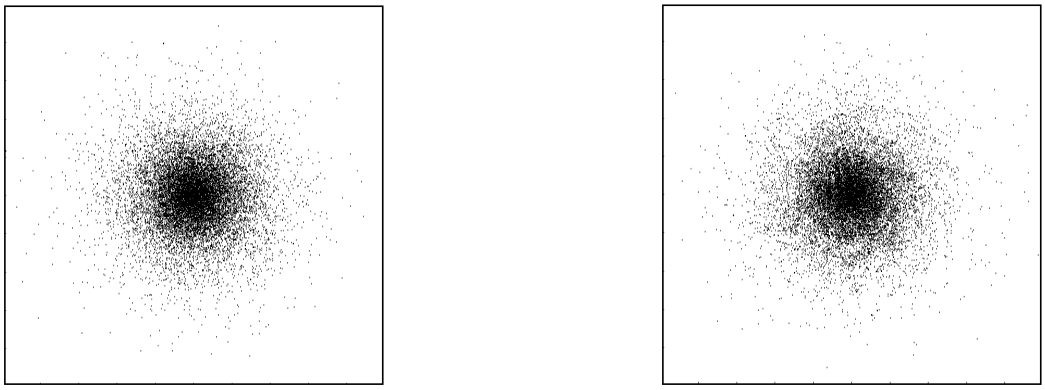
Изображения представляют собой различные проекции галактики в моменты времени t=0, 5 $\times$ 10^8 лет. На рисунке 2 представлена проекция «плашмя», на рисунке 3 — проекция «с ребра». Для сравнения приведены рисунки из статьи Хернквиста [5]. Можно заключить, что результаты, полученные нами, аналогичны результатам, полученным Хернквистом.

3. Оценка производительности GPU в задаче моделирования самогравитирующих систем

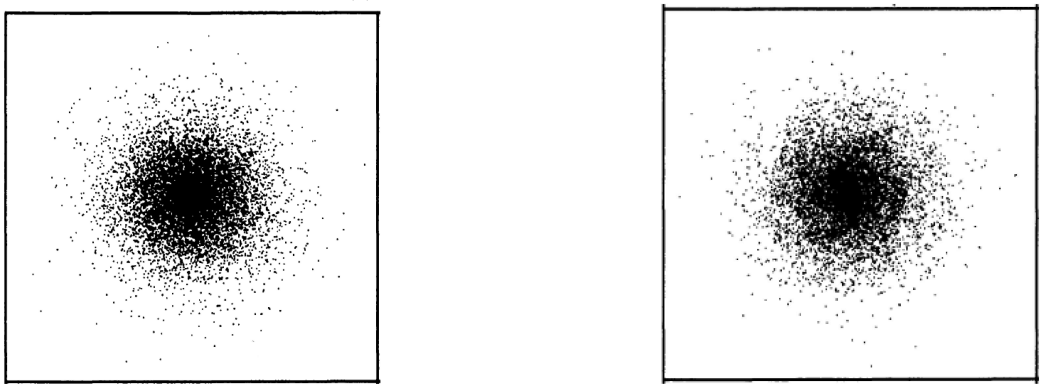
В этом параграфе мы проведем сравнение быстродействия центрального и графического процессоров в задаче N тел. Вначале вычисления выполнялись на персональном компьютере с целью сравнения быстродействия бытовой видеокарты и стандартного центрального процессора. Использовалась машина со следующими характеристиками:

- процессор AMD Athlon II X3 460 (3.4 ГГц);
- графический процессор NVIDIA GeForce GTX 660 Ti (1344 ядра, 0.980 ГГц, 2Гб памяти);
- оперативная память 4 Гб.

На рисунке 4 приведены графики зависимости затраченного на моделирование времени t_m от числа использованных в модели частиц N при проведение расчетов на центральном и графическом процессорах. Обе величины представлены в логарифмическом масштабе.



(a) Результаты, полученные нами



(b) Результаты, полученные Хернквистом

Рис. 2. Галактика в проекции «плашмя» в моменты времени t=0, 500 $\times$ 10^6 лет.

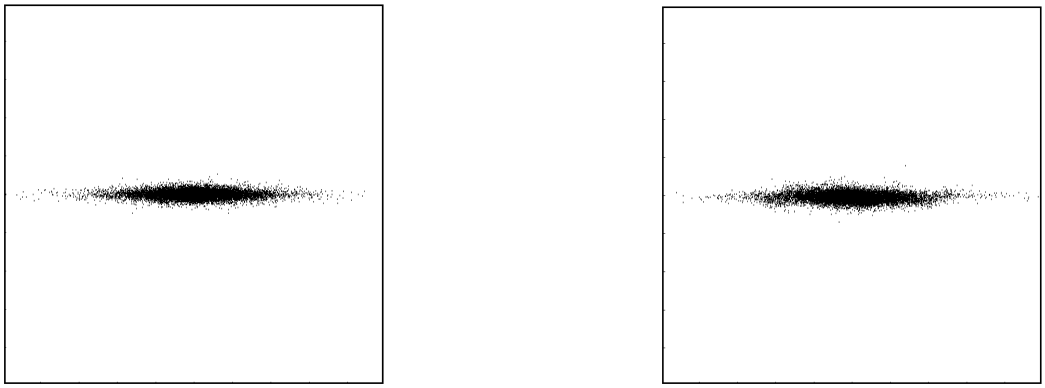
Из рисунка видно, что графический процессор значительно превосходит в производительности центральный и разница с увеличением N растет.

Далее с использованием технологий MPI (см. [9]) и OpenMP нами было разработано программное обеспечение, позволяющее использовать несколько графических процессоров, в том числе расположенных на различных узлах компьютерного кластера. Технология OpenMP применяется для создания нескольких потоков исполнения центрального процессора (процессоров) внутри одного процесса, выполняющегося на одном узле. Каждый поток работает с отдельным графическим процессором, при этом GPU рассчитывает силу притяжения, действующую на конкретную группу частиц. Технология MPI, в свою очередь, позволяет передавать информацию между процессами, выполняющимися на разных узлах кластера.

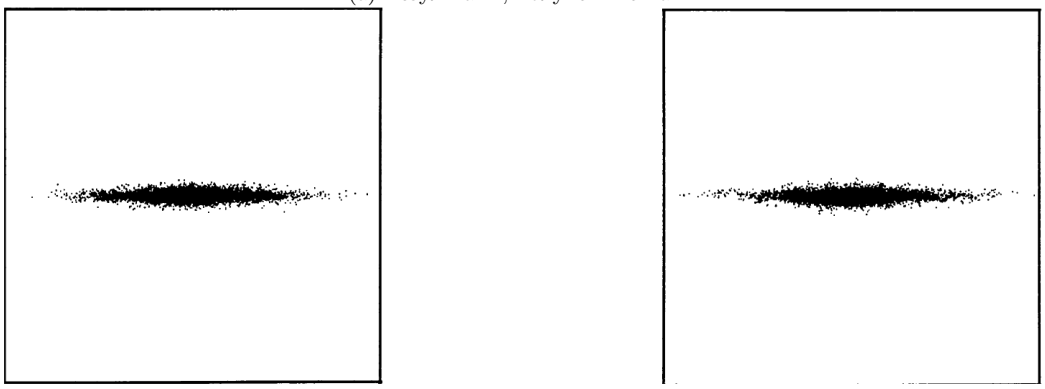
Нами были проведены расчеты с целью выявления быстродействия программного продукта в условиях распределенной вычислительной системы. Вычисления проводились на гибридном кластере Вычислительного центра СПбГУ.

Аппаратные характеристики одного узла:

- центральный процессор 2 $\times$ Intel Xeon X5650 (2 $\times$ 6 ядер);
- графический процессор TESLA M2050 (3 $\times$ 448 графических ядер, 3Гб памяти);
- оперативная память 32Гб;
- Net: Адаптер QDR Infiniband Mellanox, 1Гбит Ethernet.



(a) Результаты, полученные нами



(b) Результаты, полученные Хернквистом

Рис. 3. Галактика в проекции «с ребра» в моменты времени t=$0,500 \times 10^6$ лет.

Результаты приведены на графиках. Тестовые расчеты были выполнены как для прямого вычисления силы гравитационного притяжения, так и для сглаженного, использующего алгоритм TREE-code. В последнем случае использовалось программное обеспечение, разработанное Дж. Бедорфом, Е. Габуровым и С. П. Звартом [10].

На рисунках 6 (a–c) приведены результаты тестов для различного числа узлов, содержащих по 3 графических процессора каждый. В качестве индикатора производительности мы взяли значение обратное времени, затраченному на моделирование. Таким образом в данной работе под производительностью понимается не количество арифметических операций, совершаемых в секунду, а скорость, с которой алгоритм совершает некоторое количество итераций. Стоит отметить, что для прямого алгоритма при большом количестве частиц($\approx 10^6$) с увеличением количества применяемых графических процессоров наблюдается линейный рост производительности, который при использовании нескольких узлов практически сохраняется. Это связано с тем, что соотношение времени, затрачиваемого на вычисления силы тяжести на каждом шаге, ко времени, затрачиваемому на обращение к глобальной памяти, достаточно велико.

Хотя расчеты, выполненные нами, проводились для моделей с $N = 10^6$, учитывая что время расчета для алгоритма TREE-code растет, как $N \log N$, можно провести экстраполяцию на большее количество частиц. Мы сделали вывод о возможности

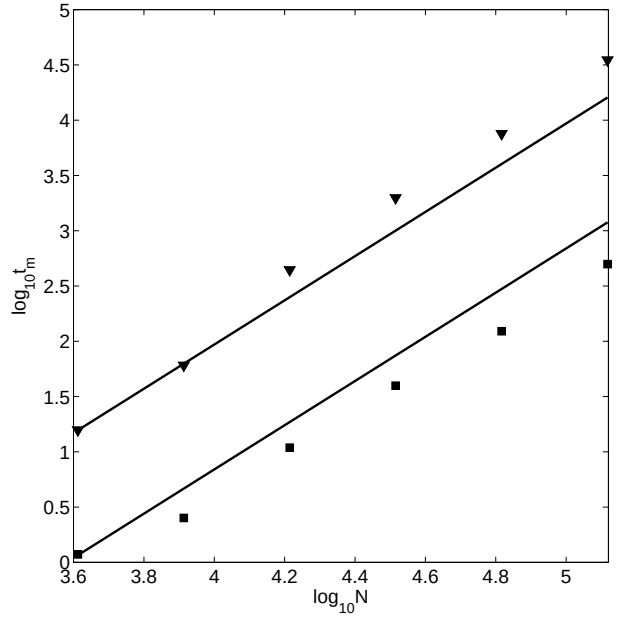


Рис. 4. Время в секундах, затраченное на моделирование на интервале $1.5 \cdot 10^7$ лет, в зависимости от числа частиц: ▲ — центральный процессор; ■ — графический процессор. Сплошные линии соответствуют сложности прямого алгоритма $O(N^2)$. Ось абсцисс — $\log_{10}(N)$. Ось ординат — $\log_{10}(t)$.

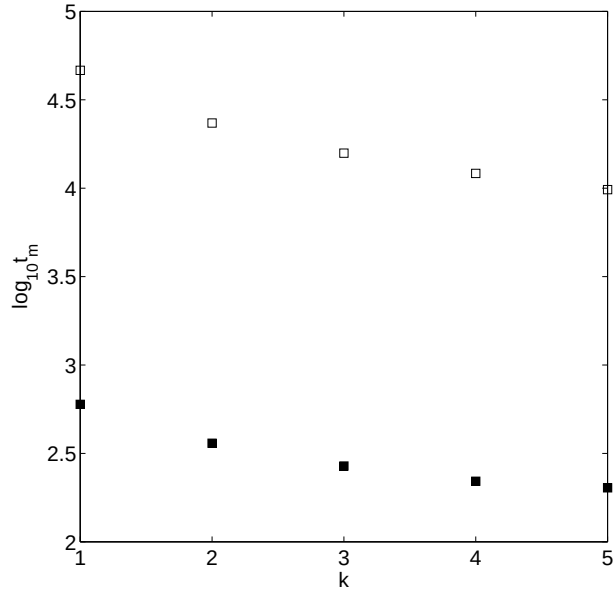
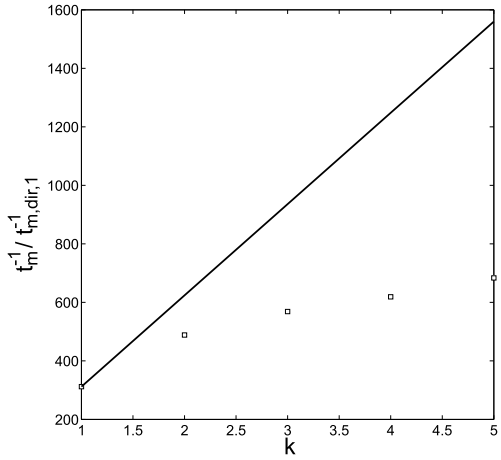
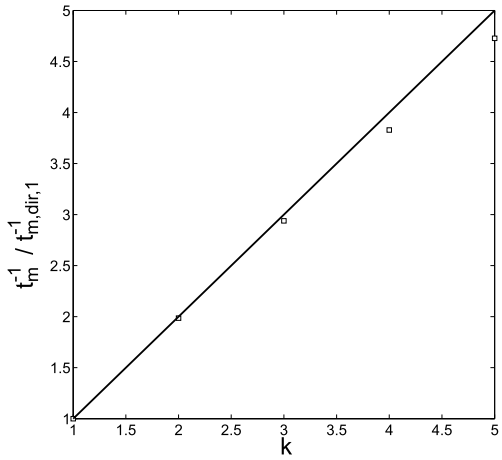


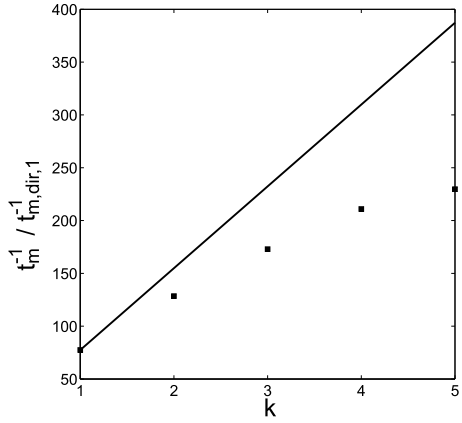
Рис. 5. Время t_m в секундах, затраченное на моделирование $N = 10^6$ частиц на интервале 10^9 лет, в зависимости от количества используемых узлов кластера k: □ — прямой алгоритм; ■ — алгоритм TREE-code. На оси абсцисс отложено k. На оси ординат — $\log_{10}(t_m)$.



(a) Прямой алгоритм, $N = 10^4$



(b) Прямой алгоритм, $N = 10^6$



(c) Алгоритм TREE-code, $N = 10^6$

Рис. 6. Производительность(t_m^{-1}) в зависимости от количества используемых узлов кластера k: сплошная линия соответствует линейному росту производительности; ■ — полученная нами производительность. Ось абсцисс — количество узлов кластера, ось ординат — производительность, нормированная на производительность одного узла для прямого алгоритма $t_{m,dir,1}^{-1}$ при $N = 10^6$.

моделирования карликовых галактик в масштабах приближенных к значению 1:1 с помощью вполне доступных мощностей (например, гибридного кластера Вычислительного центра СПбГУ), если вычисления проводятся с применением распределенных систем, использующих графические процессоры.

Заключение

На основе технологии CUDA, позволяющей использовать для счета ресурсы графического процессора, мы разработали программное обеспечение для моделирования динамической эволюции галактики. С использованием данного программного обеспечения было проведено исследование производительности графического процессора в задаче моделирования самогравитирующих систем в сравнении с центральным процессором. На его основе можно сделать вывод о значительном превосходстве GPU в области динамического моделирования галактик. С помощью технологий MPI и OpenMP нами была изучена производительность систем, имеющих нескольких графических процессоров. При этом для алгоритма прямого счета силы гравитационного

притяжения мы выявили практически линейный рост производительности с увеличением числа видеоадаптеров даже при использовании нескольких вычислительных узлов кластера. Был сделан вывод о принципиальной возможности расчета динамической эволюции карликовых галактик в реальном масштабе при подключении вполне разумных мощностей, основанных на графических процессорах. Написанное нами программное обеспечение было проверено путем проведения тестового моделирования и сравнения результатов с полученными ранее другими авторами с использованием только центрального процессора. Программное обеспечение может быть использовано как в учебных, так и в научных целях.

Хотим выразить благодарность Ресурсному центру «Вычислительный центр СПбГУ» за аппаратное обеспечение расчетов и Дж. Бедорфу, Е. Габурову и С. П. Зварту за программное обеспечение, позволяющее использовать алгоритм TREE-code.

Литература

1. *Боресков А. В., Харламов А. А.* Основы работы с технологией CUDA. М.: ДМК Пресс, 2011.

2. *Черняк Л.* Транспьютеры — британский феномен восьмидесятых // Суперкомпьютеры. 2011. Т. 5.

3. *Боресков А. В., Харламов А. А., Марковский Н. Д. и др.* Параллельные вычисления на GPU. Архитектурная и программная модель CUDA / М.: Издательство МГУ, 2012.

4. *Берилло А. А.* Nvidia cuda — неграфические вычисления на графических процессорах // iXBT.com. 2008. URL: http://www.ixbt.com/video3/cuda-1.shtml (дата обращения: 01.12.2013).

5. *Hernquist L.* N-body realizations of compound galaxies // The Astrophysical Journal Supplement Series. 1993. Vol. 86. P. 389–400.

6. *Боровков А. А.* Математическая статистика. М.: Наука, 1984.

7. *Binney J., Tremaine S.* Galactic dynamics. Princeton Series in Astrophysics. Second edition edition. New Jersey: Princeton University Press, 2008.

8. *Бэдсел Ч., Ленгдон А.* Физика плазмы и численное моделирование, перевод с английского. М.: Энергоатомиздат, 1989.

9. *Антонов А. С.* Параллельное программирование с использованием технологии MPI. М.: Издательство МГУ, 204.

10. *Bédorf J., Gaburov E., Zwart S. P.* A sparse octree gravitational n-body code that runs entirely on the gpu processor // Journal of Computational Physics. 2012. 5. Vol. 231. P. 2825–2839.

Статья поступила в редакцию 26 июня 2014 г.

Сведения об авторе

Сивков-Енин Иван Алексеевич — аспирант; i.a.sivkov@gmail.com

USING GPU FOR MODELLING OF SELF-GRAVITATING SYSTEMS

Ivan A. Sivkov-Enin

St.Petersburg State University, Universitetskaya nab., 7-9,
St.Petersburg, 199034, Russian Federation; i.a.sivkov@gmail.com

We analyse the performance of graphic processor units in N-body simulations of self-gravitating systems. We find that it is possible to model dwarf galaxies on a one-to-one scale using available hardware. Refs 10. Figs 6.

Keywords: dynamics of galaxies, GPU, N-body simulation.