ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 37 за 2012 г.

Багдасаров Г.А., Дьяченко С.В., Ольховская О.Г.

Измерение производительности и масштабируемости программного комплекса MARPLE3D

Рекомендуемая форма библиографической ссылки: Багдасаров Г.А., Дьяченко С.В., Ольховская О.Г. Измерение производительности и масштабируемости программного комплекса MARPLE3D // Препринты ИПМ им. М.В.Келдыша. 2012. № 37. 22 с. URL: http://library.keldysh.ru/preprint.asp?id=2012-37

Ордена Ленина
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Российской академии наук

Г.А. Багдасаров, С.В. Дьяченко, О.Г. Ольховская

Измерение производительности и масштабируемости программного комплекса MARPLE3D

Москва — 2012

Г.А.Багдасаров, С.В.Дьяченко, О.Г.Ольховская

Измерение производительности и масштабируемости программного комплекса MARPLE3D

Выполнены измерения производительности пакета прикладных программ MARPLE3D для моделирования импульсной магнитоускоренной плазмы при решении модельных и производственных задач из предметной области пакета на высокопроизводительных ЭВМ. Рассмотрены такие характеристики, как ускорение, эффективность и стоимость вычислений. Выявлен показатель, определяющий практическую производительность, – количество узлов сетки на расчетном узле, и его оптимальные значения для рассматриваемого программного обеспечения – от 10 до 50 тысяч.

Ключевые слова: пакеты прикладных программ, высокопроизводительные вычисления, ускорение, эффективность, масштабируемость.

G.A.Bagdasarov, S.V.Dyachenko, O.G.Olkhovskaya

Program package MARPLE3D performance and scalability measurements

Performance was measured of the program package MARPLE3D for simulation of pulsed magnetically driven plasma. The measurements were carried out while solving both test and practical problems using high performance computing. Speedup, efficiency and cost of computations were examined. Practical performance criteria was defined which is the number of computational mesh cells per computing node. Its optimal value for the studied software was found to be from 10 to 50 thousand.

Key words: application software, high performance computations, speedup, efficiency, scalability

Оглавление

Введение

На основе численных методик с применением неструктурированных расчетных сеток в ИПМ им. М.В.Келдыша РАН создан пакет прикладных программ MARPLE3D, предметной областью которого являются задачи радиационной плазмодинамики [1]. Созданный код не только позволяет решать задачи двухтемпературной радиационной магнитной гидродинамики в трехмерных областях произвольной геометрии, но и включает ряд инноваций, призванных сделать его полнофункциональным исследовательским кодом для задач мультифизики, использующим как актуальные физические и математические модели и численные методы, так и возможности современной высокопроизводительной вычислительной техники.

Код MARPLE3D представляет собой совокупность вычислительной среды (инфраструктуры) для проведения параллельных расчетов начально-краевых задач на неструктурированных вычислительных сетках и реализованных в рамках этой среды физических солверов.

Основные физические модели MARPLE:

- одножидкостная двухтемпературная магнитогидродинамическая модель + эффект Холла + обобщенный закон Ома;
- расчет переноса лучистой энергии: многогрупповое приближение, сеточно-характеристический метод, диффузионное приближение;
- модель длительного плазмообразования;
- электротехническое уравнение полной цепи (генератор, подводящие системы и разрядная камера с плазмой);
- транспортные и кинетические коэффициенты, оптические свойства и уравнения состояния на базе таблиц, широкодиапазонные уравнения состояния.

Основные численные методы MARPLE:

- технология неструктурированных и блочно-структурированных сеток, смешанные тетраэдральные, гексаэдральные и призматические элементы;
- явные TVD/WENO схемы повышенной точности для бездиссипативной МГД, неявные схемы на основе варианта метода

Работа поддержана отделением математических наук РАН (ПФИ 3.6 ОМН РАН) и Российским фондом фундаментальных исследований (проекты 11-07-93939-Джи8_а, 11-02-01027-а). Расчеты выполнены на суперкомпьютерах ЛОМОНОСОВ (НИВЦ МГУ) и МВС-100К (МСЦ РАН).

Галеркина с разрывными базисными функциями для диссипативных процессов;
- ♦ схема суммарной аппроксимации для РМГД системы;
- ♦ схема "предиктор-корректор" 2-го порядка точности по времени.

Основные принципы архитектуры MARPLE:
- ♦ использование высокопроизводительных распределенных вычислений;
- ♦ большое число сервисных функций (ввод-вывод данных, операции с сетками, поддержка параллельных вычислений, динамическая работа с вычислительными объектами);
- ♦ динамическое создание, уничтожение и настройка различных вычислительных объектов (солверы, аппроксимации, граничные условия, свойства вещества), автоматизация управления памятью;
- ♦ использование объектно-ориентированного подхода к проектированию ПО, приемов объектно-ориентированного и обобщенного программирования (язык реализации C++), соответствие современным мировым стандартам коммерческой разработки ПО;
- ♦ кросс-платформенность.

Для реализации описанной модели на каждом временном шаге необходимо вычислять несколько десятков физических величин в каждой ячейке расчетной сетки. Для моделирования существенно трехмерных процессов с учетом реальной геометрии устройств необходимы расчетные сетки, содержащие от нескольких миллионов до десятков и сотен миллионов ячеек. Решение задач такой размерности возможно только с использованием распределенных вычислений на базе современных высокопроизводительных ЭВМ. Поэтому пакет MARPLE3D был сконструирован как программное обеспечение для высокопроизводительных параллельных вычислительных систем. Для выполнения расчетов на массивно-параллельных системах с распределенной памятью применяется известный подход геометрического параллелизма, основанный на декомпозиции расчетной области (разбиении ее на вычислительные домены, каждый из которых обрабатывается отдельным счетным ядром). Поддержка геометрического параллелизма встроена в дискретную модель MARPLE3D на уровне структур сеточных данных. Для декомпозиции смешанных сеток нерегулярной структуры использовался один из самых известных и широко применяемых пакетов такого рода – METIS (ParMETIS). Код может запускаться в качестве параллельного приложения MPI на компьютерах с общей или распределенной памятью. METIS обеспечивает в достаточной мере сбалансированное разбиение сетки в том смысле, что все вычислительные домены содержат приблизительно одинаковое число

расчетных ячеек, и размеры границ доменов также приблизительно одинаковы. Однако при решении реальных задач, в которых имеет место существенная пространственная неоднородность моделируемых физических процессов, баланс загрузки вычислительных ядер может нарушаться, и их использование может оказаться неэффективным.

В нашем случае, когда численная модель включает большое количество различных по ресурсоемкости и организации вычислений солверов, а расчетная сетка имеет нерегулярную структуру, теоретическая оценка эффективности распараллеливания затруднительна. Таким образом, большое значение приобретают практические измерения производительности работы сложных программных комплексов на высокопроизводительных ЭВМ.

Методики оценки производительности и эффективности для приложений, ориентированных на многоядерные системы

В данном разделе по материалам работы [2] описаны основные параметры, характеризующие эффективность параллельных алгоритмов, методы их оценки и анализа.

Показатели эффективности параллельного алгоритма

Ускорение, получаемое при использовании параллельного алгоритма для p процессоров, по сравнению с последовательным вариантом выполнения вычислений, определяется величиной

$S_p(n)=T_1(n)/T_p(n),$

т.е. как отношение времени решения задач на скалярной ЭВМ T_1 к времени выполнения параллельного алгоритма T_p (величина n применяется для параметризации вычислительной сложности решаемой задачи и может пониматься, например, как количество входных данных задачи).

Эффективность использования параллельным алгоритмом процессоров при решении задачи определяется соотношением

$E_p(n)=T_1(n)/(pT_p(n))=S_p(n)/p.$

Величина эффективности определяет среднюю долю времени выполнения алгоритма, в течение которой процессоры реально задействованы для решения задачи.

Из приведенных соотношений следует, что в наилучшем случае $S_p(n)=p$ и $E_p(n)=1$. При практическом применении данных показателей для оценки эффективности параллельных вычислений следует учитывать два важных момента:

1) При определенных обстоятельствах ускорение может оказаться больше числа используемых процессоров $S_p(n)>p$ - в этом случае говорят о

существовании сверхлинейного ускорения. Несмотря на парадоксальность таких ситуаций (ускорение превышает число процессоров), на практике сверхлинейное ускорение может иметь место. Одной из причин такого явления может быть неодинаковость условий выполнения последовательной и параллельной программ. Например, при решении задачи на одном процессоре оказывается недостаточно оперативной памяти для хранения всех обрабатываемых данных, и тогда становится необходимым использование более медленной внешней памяти (в случае же использования нескольких процессоров оперативной памяти может оказаться достаточно за счет разделения данных между процессорами). Еще одной причиной сверхлинейного ускорения может быть нелинейный характер зависимости сложности решения задачи от объема обрабатываемых данных. Так, например, известный алгоритм пузырьковой сортировки характеризуется квадратичной зависимостью количества необходимых операций от числа упорядочиваемых данных. Как результат, при распределении сортируемого массива между процессорами может быть получено ускорение, превышающее число процессоров. Источником сверхлинейного ускорения может быть и различие вычислительных схем последовательного и параллельного методов.

2) При внимательном рассмотрении можно обратить внимание на то, что попытки повышения качества параллельных вычислений по одному из показателей (ускорению или эффективности) могут привести к ухудшению ситуации по другому показателю, ибо показатели качества параллельных вычислений являются часто противоречивыми. Так, например, повышение ускорения обычно может быть обеспечено за счет увеличения числа процессоров, что приводит, как правило, к падению эффективности. И наоборот, повышение эффективности достигается во многих случаях при уменьшении числа процессоров (в предельном случае идеальная эффективность $E_p(n)=1$ легко обеспечивается при использовании одного процессора). Как результат, разработка методов параллельных вычислений часто предполагает выбор некоторого компромиссного варианта с учетом желаемых показателей ускорения и эффективности.

При выборе надлежащего параллельного способа решения задачи может оказаться полезной оценка стоимости вычислений, определяемой как произведение времени параллельного решения задачи и числа используемых процессоров $C_p=pT_p$. В связи с этим можно определить понятие стоимостно-оптимального параллельного алгоритма как метода, стоимость которого является пропорциональной времени выполнения наилучшего последовательного алгоритма.

Оценка максимально достижимого параллелизма

Оценка качества параллельных вычислений предполагает знание наилучших (максимально достижимых) значений показателей ускорения и эффективности, однако получение идеальных величин $S_p=p$ для ускорения и $E_p=1$ для эффективности может быть обеспечено не для всех вычислительно трудоемких задач. Рассмотрим ряд закономерностей, которые могут быть чрезвычайно полезны при построении оценок максимально достижимого параллелизма.

1) Закон Амдаля. Достижению максимального ускорения может препятствовать существование в выполняемых вычислениях последовательных расчетов, которые не могут быть распараллелены. Пусть f есть доля последовательных вычислений в применяемом алгоритме обработки данных, тогда, в соответствии с законом Амдаля (Amdahl), ускорение процесса вычислений при использовании p процессоров ограничивается величиной

$$S_p \le \frac{1}{f+(1-f)/p} \le S^* = \frac{1}{f}.$$

Так, например, при наличии всего 10% последовательных команд в выполняемых вычислениях эффект использования параллелизма не может превышать 10-кратного ускорения обработки данных. Например, при вычислении суммы значений для каскадной схемы доля последовательных расчетов составляет $f=\log_2 n/n$ и, как результат, величина возможного ускорения ограничена оценкой $S^*=n/\log_2 n$.

Закон Амдаля характеризует одну из самых серьезных проблем в области параллельного программирования (алгоритмов без определенной доли последовательных команд практически не существует). Однако часто доля последовательных действий характеризует не возможность параллельного решения задач, а последовательные свойства применяемых алгоритмов. Поэтому доля последовательных вычислений может быть существенно снижена при выборе более подходящих для распараллеливания методов.

Следует отметить также, что рассмотрение закона Амдаля происходит в предположении, что доля последовательных расчетов f является постоянной величиной и не зависит от параметра n, определяющего вычислительную сложность решаемой задачи. Однако для большого ряда задач доля $f=f(n)$ является убывающей функцией от n, и в этом случае ускорение для фиксированного числа процессоров может быть увеличено за счет увеличения вычислительной сложности решаемой задачи. Данное замечание может быть сформулировано как утверждение, что ускорение $S_p=S_p(n)$ является возрастающей функцией от параметра n (данное утверждение часто именуется "эффект Амдаля"). Так, в рассмотренном примере вычисления суммы значений при использовании фиксированного числа процессоров p суммируемый набор данных может быть разделен на блоки размера n/p, для которых сначала параллельно могут быть вычислены частные суммы, а далее эти суммы можно

сложить при помощи каскадной схемы. Длительность последовательной части выполняемых операций (минимально возможное время параллельного исполнения) в этом случае составляет $T_p=(n/p)+\log_2 p$, что приводит к оценке доли последовательных расчетов как величины $f=(1/p)+\log_2 p/n$.

Как следует из полученного выражения, доля последовательных расчетов f убывает с ростом n, и в предельном случае мы получаем идеальную оценку максимально возможного ускорения $S^*=p$.

2) Закон Густавсона – Барсиса. Оценим максимально достижимое ускорение исходя из имеющейся доли последовательных расчетов в выполняемых параллельных вычислениях:

$$g = \frac{\tau(n)}{\tau(n) + \pi(n)/p},$$

где $\tau(n)$ и $\pi(n)$ есть времена последовательной и параллельной частей выполняемых вычислений соответственно, т.е.

$T_1 = \tau(n) + \pi(n)$, $T_p = \tau(n) + \pi(n)/p$.

С учетом введенной величины g можно получить

$\tau(n) = gx(\tau(n) + \pi(n)/p)$, $\pi(n) = (1-g)px(\tau(n) + \pi(n)/p)$,

что позволяет построить оценку для ускорения

$$S_p = \frac{T_1}{T_p} = \frac{\tau(n)+\pi(n)}{\tau(n)+\pi(n)/p} = \frac{(\tau(n)+\pi(n)/p)(g+(1-g)p)}{\tau(n)+\pi(n)/p},$$

которая после упрощения приводится к виду закона Густавсона – Барсиса (Gustafson – Barsis's law):

$S_p = g+(1-g)p = p+(1-p)g$.

Применительно к примеру суммирования значений при использовании p процессоров время параллельного выполнения, как уже отмечалось выше, составляет $Tp = (n/p)+\log_2 p$, что соответствует последовательной доле

$$g = \frac{\log_2 p}{(n/p)+\log_2 p}.$$

За счет увеличения числа суммируемых значений величина g может быть пренебрежимо малой, обеспечивая получение идеального возможного ускорения $S_p=p$.

При рассмотрении закона Густавсона – Барсиса следует учитывать еще один важный момент. С увеличением числа используемых процессоров темп уменьшения времени параллельного решения задач может падать (после превышения определенного порога). Однако за счет уменьшения времени вычислений сложность решаемых задач может быть увеличена (так, например, для задачи суммирования может быть увеличен размер складываемого набора значений). Оценку получаемого при этом ускорения можно определить при помощи сформулированных закономерностей. Такая аналитическая оценка тем более полезна, поскольку решение таких более сложных вариантов задач на одном процессоре может оказаться достаточно трудоемким и даже невозможным, например, в силу нехватки оперативной памяти. С учетом

указанных обстоятельств оценку ускорения, получаемую в соответствии с законом Густавсона – Барсиса, еще называют ускорением масштабирования, поскольку данная характеристика может показать, насколько эффективно могут быть организованы параллельные вычисления при увеличении сложности решаемых задач.

Анализ масштабируемости параллельных вычислений

Целью применения параллельных вычислений во многих случаях является не только уменьшение времени выполнения расчетов, но и обеспечение возможности решения более сложных задач (таких постановок, решение которых не представляется возможным при использовании однопроцессорных вычислительных систем). Способность параллельного алгоритма эффективно использовать процессоры при повышении сложности вычислений является важной характеристикой выполняемых расчетов. Поэтому параллельный алгоритм называют масштабируемым, если при росте числа процессоров он обеспечивает увеличение ускорения при сохранении постоянного уровня эффективности использования процессоров. Возможный способ характеристики свойств масштабируемости состоит в следующем.

Оценим накладные расходы, которые имеют место при выполнении параллельного алгоритма

$T_0=pT_p-T_1$.

Накладные расходы появляются за счет необходимости организации взаимодействия процессоров, выполнения некоторых дополнительных действий, синхронизации параллельных вычислений и т.п. Используя введенное обозначение, можно получить новые выражения для времени параллельного решения задачи и соответствующего ускорения:

$$T_p = \frac{T_1 + T_0}{p}, \quad S_p = \frac{T_1}{T_p} = \frac{pT_1}{T_1 + T_0}.$$

Применяя полученные соотношения, эффективность использования процессоров можно выразить как

$$E_p = \frac{S_p}{p} = \frac{T_1}{T_1 + T_0} = \frac{1}{1 + T_0 / T_1}.$$

Последнее выражение показывает, что если сложность решаемой задачи является фиксированной (T_1=const), то при росте числа процессоров эффективность, как правило, будет убывать за счет роста накладных расходов T_0. При фиксации числа процессоров эффективность их использования можно улучшить путем повышения сложности решаемой задачи T_1 (предполагается, что при росте параметра сложности n накладные расходы T_0 увеличиваются медленнее, чем объем вычислений T_1). Как результат, при увеличении числа процессоров в большинстве случаев можно обеспечить определенный уровень эффективности при помощи соответствующего повышения сложности

решаемых задач. Поэтому важной характеристикой параллельных вычислений становится соотношение необходимых темпов роста сложности расчетов и числа используемых процессоров.

Пусть E=const есть желаемый уровень эффективности выполняемых вычислений. Из выражения для эффективности можно получить

$\frac{T_0}{T_1} = \frac{1-E}{E}$, или $T_1 = KT_0$, $K = E/(1-E)$.

Порождаемую последним соотношением зависимость $n=F(p)$ между сложностью решаемой задачи и числом процессоров обычно называют функцией изоэффективности.

Покажем в качестве иллюстрации вывод функции изоэффективности для модельной задачи суммирования числовых значений. В этом случае

$T_0 = pT_p - T_1 = p((n/p) + \log_2 p) - n = p\log_2 p$

и функция изоэффективности принимает вид

$n = Kp\log_2 p$.

Как результат, например, при числе процессоров p=16 для обеспечения уровня эффективности E=0,5 (т.е. K=1) количество суммируемых значений должно быть не менее n=64. Или же при увеличении числа процессоров с p до q ($q>p$) для обеспечения пропорционального роста ускорения $(S_q/S_p)=(q/p)$ необходимо увеличить число суммируемых значений n в $(q\log_2 q)/(p\log_2 p)$ раз.

Исследование масштабируемости параллельного программного комплекса MARPLE3D

Исследования масштабируемости параллельного программного комплекса MARPLE3D выполнялись на примере двух задач: тестовой задачи о распространении тепловой волны (решается уравнение теплопроводности по неявной схеме, для решения СЛАУ используется параллельный пакет линейной алгебры Aztec [3]) и производственной задачи моделирования сжатия плазменного лайнера магнитным полем (решается полная система уравнений радиационной магнитной газовой динамики с табличными уравнениями состояния и учетом диссипативных процессов). В обоих случаях для организации параллельных вычислений использовалась инфраструктура MARPLE, декомпозиция сеток осуществлялась с помощью пакета METIS. Измерения производились средствами внутренней системы журналирования MARPLE (фиксировалось астрономическое время выполнения операций на каждом вычислительном узле). Измерения выполнялись на суперкомпьютерах ЛОМОНОСОВ (НИВЦ МГУ) и МВС-100К (МСЦ РАН).

Исследование масштабируемости при повышении подробности расчетных сеток и увеличении числа вычислительных ядер

Исследование масштабируемости при повышении подробности расчетных сеток (до 10^6 ячеек) и увеличении числа вычислительных ядер (до 128 ядер) выполнялось в НИВЦ МГУ на суперкомпьютере "Ломоносов" с пиковой производительностью 510 Тфлопс.

Измерения выполнялись для тестовой задачи о распространении тепловой волны [4].

Решалось уравнение теплопроводности в форме:

$$\frac{\partial \varepsilon}{\partial t} = \frac{\partial}{\partial s}(k\frac{\partial T}{\partial s}) + f, \qquad \varepsilon(T) = T \quad (C_V = \left.\frac{\partial \varepsilon}{\partial T}\right|_V = 1), \quad k = k_0 T^{\alpha}.$$

Это уравнение имеет автомодельное решение

$$T(s,t) = \left[\frac{\alpha D}{k_0}\left(Dt - s + S_1\right)\right]^{\frac{1}{\alpha}}, \quad S_1 = const.$$

Это решение описывает тепловую волну, распространяющуюся в направлении s со скоростью D. Точка, в которой температура становится нулевой, имеет координату $S_{fr}(t)=S_1+Dt$. Решение в области $s \geq 0$ существует при следующих начальных и граничных условиях:

$$T(s,0) = \begin{cases} \left[\frac{\alpha D}{k_0}(S_1 - s)\right]^{\frac{1}{\alpha}}, & 0 < s \leq S_1 \\ 0, & s > S_1 \end{cases}, \qquad T(0,t) = \left[\frac{\alpha D}{k_0}(S_1 + Dt)\right]^{\frac{1}{\alpha}}, \quad t > 0.$$

В качестве автомодельной переменной s выбиралась координата x. Тестовые расчеты выполнялись для следующего набора параметров:
α=2; k_0=0,5; S_1=0,5; D=5; $0 \leq t \leq 0{,}4$.

Расчетная область – куб 3x3x3, шаг по времени $\Delta t = 10^{-4}$ (для решения задачи выполняется 400 шагов).

Использовались 4 расчетные сетки, включающие соответственно 125 тысяч (50x50x50), 250 тысяч (100x50x50), 500 тысяч (200x50x50) и 1 миллион (400x50x50) ячеек. На каждой из этих сеток решалось уравнение теплопроводности с использованием 16, 32, 64 и 128 вычислительных ядер. С помощью встроенных средств журналирования MARPLE измерялось время решения задачи в целом и отдельно время выполнения первичной подготовки данных. По результатам измерений вычислены ускорение, эффективность использования параллельным алгоритмом процессоров и стоимость вычислений. Результаты измерений и обработки полученных данных приведены в таблице 1 и далее на рисунках 1 – 4.

Таблица 1. Время расчета при увеличении расчетной сетки.

	p	подго-товка данных	решение уравне-ния	весь расчет	подго-товка данных	решение уравне-ния	весь расчет	подго-товка данных	решение уравне-ния	весь расчет	подго-товка данных	решение уравне-ния	весь расчет
сетка (тыс. ячеек)		125	125	125	250	250	250	500	500	500	1000	1000	1000
Tp (c)	16	16	166	182	34	327	361	97	709	806	270	2107	2377
Sp	16	16	16	16	16	16	16	16	16	16	16	16	16
Ep	16	1	1	1	1	1	1	1	1	1	1	1	1
Cp (c· ядро)	16	256	2656	2912	544	5232	5776	1552	11344	12896	4320	33712	38032
Tp (c)	32	9	99	108	20	201	221	35	363	398	110	1037	1147
Sp	32	28,44	26,82	26,96	27,2	26,02	26,14	44,34	31,25	32,4	39,27	32,51	33,16
Ep	32	0,89	0,84	0,85	0,85	0,81	0,82	1,39	0,98	1,01	1,23	1,02	1,04
Cp (c· ядро)	32	288	3168	3456	640	6432	7072	1120	11616	12736	3520	33184	36704
Tp (c)	64	4	73	77	9	121	130	20	223	243	51	534	585
Sp	64	64	36,38	37,82	60,44	43,24	44,43	77,6	50,87	53,07	84,71	63,13	65,01
Ep	64	1,0	0,57	0,59	0,95	0,68	0,7	1,21	0,80	0,83	1,7	0,99	1,02
Cp (c· ядро)	64	256	4672	4928	576	7744	8320	1280	14272	15552	3264	34176	37440
Tp (c)	128	3	69	72	5	98	103	10	149	159	22	327	305
Sp	128	85,33	38,49	40,44	108,8	53,39	56,08	155,2	76,13	81,11	196,4	103,09	124,7
Ep	128	0,67	0,3	0,32	0,85	0,42	0,44	1,2	0,6	0,63	1,53	0,86	0,9
Cp (c· ядро)	128	384	8832	9216	640	12544	13184	1280	19072	20352	2816	39040	41856

Особенностью исследуемого алгоритма является наличие этапа предварительной подготовки данных, выполняемого перед началом собственно расчета (решения уравнений). Предварительная подготовка данных заключается в чтении описания расчетной сетки из файла, анализе отношений смежности и инцидентности элементов сетки, выделении сеточных элементов, используемых в межпроцессорных обменах, вычислении геометрических соотношений, необходимых для аппроксимации решаемых уравнений, построении портрета матрицы СЛАУ и др. Такая достаточно сложная предварительная работа с сеточными данными характерна для большинства алгоритмов, использующих сетки нерегулярной структуры. Этот этап заметно отличается от последующего по характеру выполняемых на нем операций с сеткой и данными и, как следствие, по показателям эффективности распараллеливания. В рассмотренном примере этот этап занимал от 5 до 10 процентов общего времени счета, в других случаях (больше время моделирования) эта доля может быть меньше. Поэтому далее показатели для двух этапов расчета будут обсуждаться отдельно.

Графики показателей эффективности для этапа предварительной подготовки данных подписаны "data preparing", для этапа решения уравнений – "iterations", для расчета в целом – "total".

Ускорение рассчитывалось по формуле $S_p=(T_{16} / T_p)\cdot 16$, т.к. минимальное число процессов, для которых проводились измерение, было равно 16.

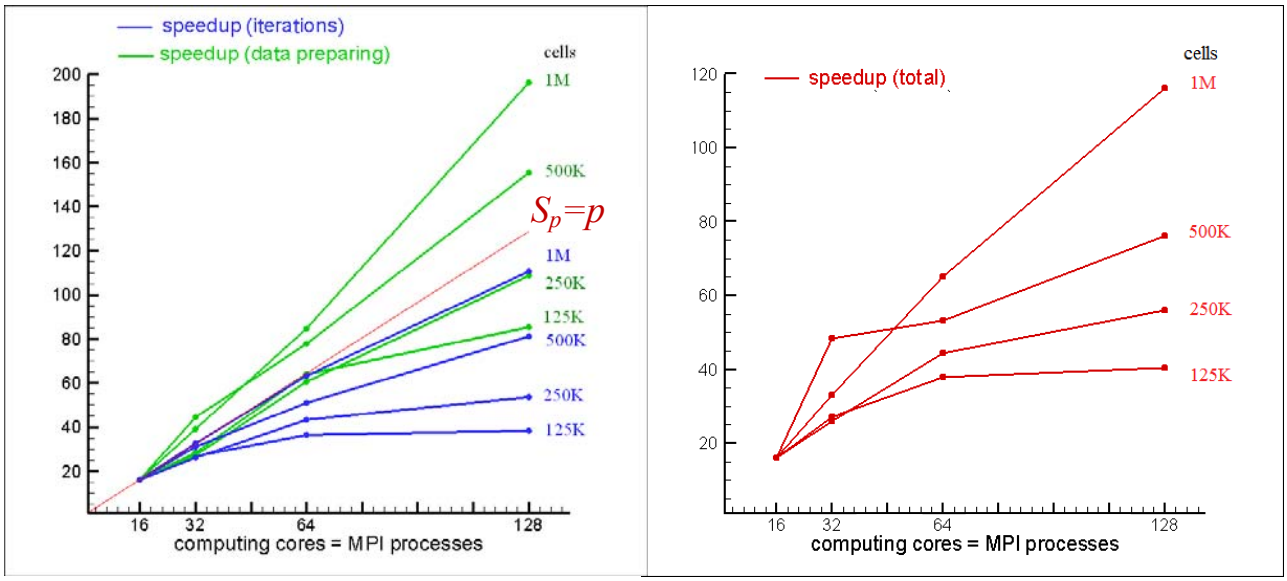

Рисунок 1. Ускорение.

Синие графики – ускорение при расчете шага по времени, зеленые – ускорение первичной подготовки данных, красные – расчета в целом. Справа от графиков указан размер расчетной сетки (количество ячеек), для которой проводились измерения.

Для первичной подготовки данных на больших сетках (500 тысяч и 1 миллион ячеек) имеет место сверхлинейное ускорение (ускорение в 9,7 и 12,3 раза при увеличении числа процессов в 8 раз), это связано с особенностями алгоритмов работы со структурами данных.

Из рисунка 1 видно, что масштабируемость алгоритма улучшается при увеличении размера сетки. На сетке в 1 миллион ячеек для расчета шага по времени практически достигается ускорение, близкое к максимально возможному (красная линия $S_p = p$ на графике) – ускорение в 6,5 раз при увеличении числа процессов в 8 раз. Для меньших сеток ускорение значительно меньше. Таким образом, наблюдается эффект Амдаля – ускорение является возрастающей функцией от вычислительной сложности задачи (размера сетки). Эффект усиливается при увеличении числа вычислительных ядер.

Эффективность использования параллельным алгоритмом процессоров при решении задачи определялась соотношением $E_p=(T_{16}\cdot 16) / (pT_p)=S_p/p$, т.е. за единицу принималась эффективность использования 16 процессоров (минимальное число, для которого проводились измерения). С этим связано незначительное превышение единичного уровня на графиках эффективности на рисунке 2.

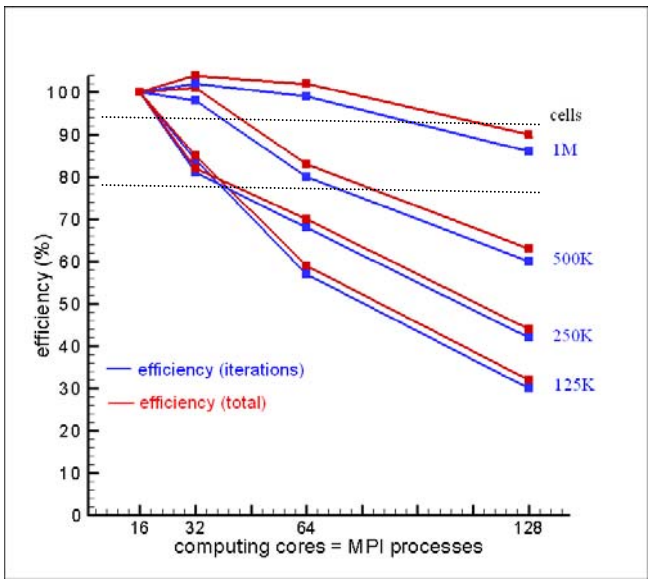

Рисунок 2. Эффективность.

Синие графики – эффективность при расчете шага по времени, красные – эффективность всего расчета в целом. Справа от графиков указан размер расчетной сетки (количество ячеек), для которой проводились измерения. Видно, что эффективность, близкая к максимально возможной, достигается при расчете на сетке, содержащей 1 миллион ячеек. Кроме того, в этом случае эффективность мало снижается с увеличением числа процессов (от 100 до 90% при увеличении числа процессов в 8 раз), тогда как при использовании сеток меньшего размера эффективность падает в 2-3 раза.

Данных измерений недостаточно для полного анализа функции изоэффективности, однако можно утверждать, что для высоких значений эффективности $80\%<E<90\%$ (пунктирные линии на рисунке 2) эта функция практически линейная $n = Kp$. Здесь мерой сложности вычислений n служит число ячеек расчетной сетки, p – количество вычислительных ядер. Это означает, что при увеличении сетки в q раз для сохранения эффективности распараллеливания следует использовать в q раз больше вычислительных ядер. Размер сетки, обрабатываемой одним ядром, остается при этом постоянным.

Таким образом, можно сделать вывод, что распараллеливание решения задачи более эффективно при использовании больших (миллион и более ячеек) расчетных сеток. В этом случае параллельный алгоритм можно считать

масштабируемым, т.к. при росте числа процессоров он обеспечивает увеличение ускорения при сохранении постоянного уровня эффективности использования процессоров.

Стоимость вычислений определялась как произведение времени параллельного решения задачи и числа используемых процессоров $C_p=pT_p$.

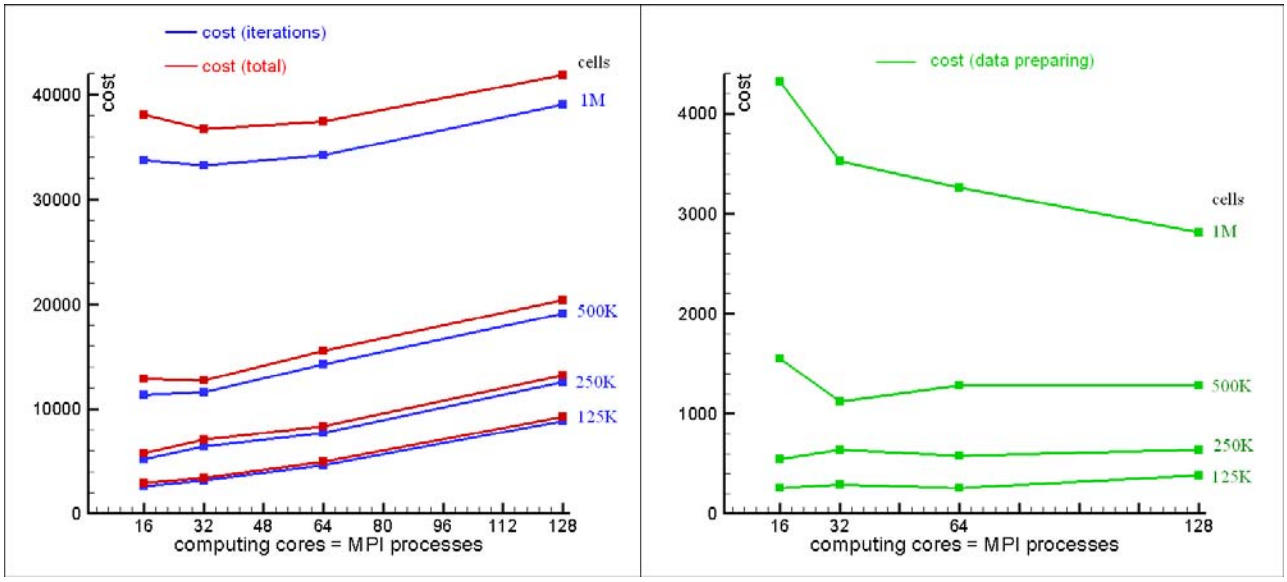

Рисунок 3. Стоимость вычислений (сек·ядро).

На рисунке 3 слева представлена полная стоимость вычислений (красные графики) и стоимость вычислений при расчете шагов по времени (синие графики). Справа от графиков указан размер расчетной сетки (количество ячеек). Справа представлена стоимость первичной подготовки данных (зеленые графики). Поведение этих графиков заметно различается. Полная стоимость вычислений остается практически постоянной при расчете на сетке в 1 миллион ячеек и существенно возрастает с увеличением числа процессов (в 2 и даже 3 раза) для сеток меньших размеров. Стоимость подготовки данных на меньших сетках практически постоянна, а на сетке в 1 миллион ячеек уменьшается более чем в 1,5 раза при увеличении числа процессов в 8 раз.

На рисунке 4 представлена зависимость стоимости вычислений (сек·ядро) от размера расчетной сетки (тысяч ячеек): слева – полная стоимость вычислений (красные графики) и стоимость вычислений при расчете шагов по времени (синие графики), справа – стоимость первичной подготовки данных (зеленые графики). Числа рядом с графиками показывают число вычислительных процессов, для которых проводились измерения.

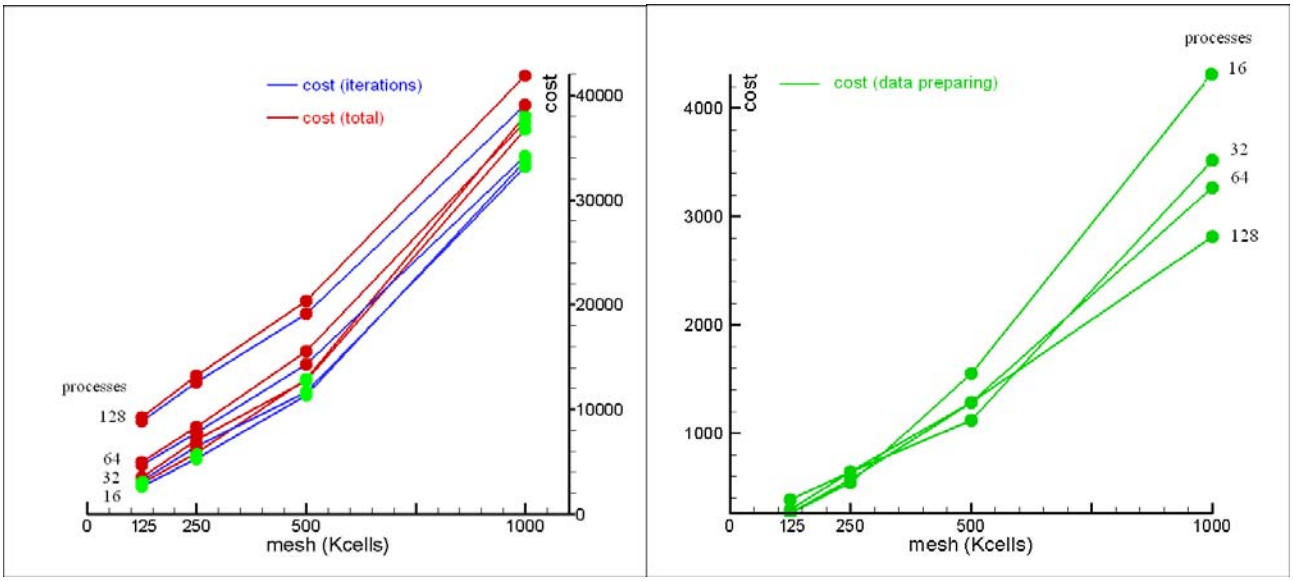

Рисунок 4. Стоимость вычислений в зависимости от размера сетки.

На левом графике видно, что стоимость вычислений при 128 процессах заметно выше для всех проверенных сеток. При малом числе параллельных процессов (16, 32) рост стоимости быстрее линейного – при увеличении сетки в 8 раз стоимость возрастает в 13 и 11 раз соответственно. Практически линейный рост стоимости наблюдается в случае 64 процессов (рост стоимости в 7,6 раза при увеличении сетки в 8 раз).

В таблице 2 приведены приблизительные размеры сетки (тысяч ячеек), приходящейся на каждый вычислительный процесс в рассмотренных случаях.

Таблица 2. Размер сетки в вычислительном домене.

Размер исходной сетки (тыс. ячеек) / количество вычислительных процессов	125	250	500	1000
16	7,8	15,6	31,25	62,5
32	3,9	7,8	15,6	31,25
64	1,95	3,9	7,8	15,6
128	0,97	1,95	3,9	7,8

Слева на рисунке 4 отмечены зеленым цветом точки, соответствующие верхнему правому треугольнику таблицы (ячейки с серым фоном). Точки,

соответствующие нижнему левому треугольнику таблицы (ячейки с белым фоном), отмечены красным. Зеленым точкам соответствуют стоимостно-оптимальные вычисления. Таким образом, можно утверждать, что существует оптимальный размер сетки в вычислительном домене, обрабатываемом одним процессом. Для данного алгоритма он составляет приблизительно от 10 до 50 тысяч ячеек. При меньшем числе ячеек резко возрастает стоимость вычислений из-за роста доли накладных расходов на междоменные обмены. При увеличении размера сетки в домене снижается эффективность и уменьшается ускорение.

Исследование масштабируемости решения задачи плазменной мультифизики

Исследование масштабируемости программного комплекса MARPLE3D при увеличении числа вычислительных ядер (до 1024 ядер) выполнялось в МСЦ РАН с использованием суперкомпьютера МВС-100К с пиковой производительностью 140,16 ТФлопс.

Измерения проводились для типичной задачи плазменной мультифизики из предметной области MARPLE. Постановка задачи и результаты расчетов подробно описаны в [1]. Моделировалось сжатие проволочной сборки магнитным полем мощного токового импульса. Под действием токового импульса вещество проволочек нагревается, испаряется и переходит в плазменное состояние. Возникающее азимутальное магнитное поле сжимает плазму к оси сборки, где в момент максимального сжатия происходит быстрый переход кинетической энергии в энергию излучения. Рассматривалась одиночная проволочная сборка цилиндрической формы, состоящая из 240 алюминиевых проволок, имеющих диаметр 10,4 μм и высоту 50 мм. Диаметр сборки 140 мм, диаметр разрядной камеры 150 мм. В качестве граничного условия задавался ток генератора (амплитуда 6 МА, время нарастания тока 700 нс), и вычислялась напряженность магнитного поля на внешней стенке камеры. Начальная конфигурация проволочной сборки задавалась в форме плазменной оболочки с плотностью $1{,}5\cdot10^{-4}$г/см^3 и температурой 2 эВ. Плотность фоновой плазмы 10^{-7}г/см^3. Уравнения состояния, транспортные коэффициенты и оптические свойства использованы в форме таблиц. Постановка задачи соответствует эксперименту № 637 на установке SPHINX Исследовательского Центра CEA Gramat (Франция).

Использовалась расчетная сетка блочной структуры со смешанными элементами (гексаэдры и треугольные призмы), состоящая из 4 миллионов ячеек. Такой размер сетки обусловлен требованиями физической точности – на менее подробной сетке невозможно с необходимой точностью вычислить параметры центрального пинча, формирующегося на оси сборки. На этой сетке решалась полная система уравнений РМГД, включающая диссипативные

процессы, с использованием 16, 32, 64, 128, 256, 512 и 1024 вычислительных ядер. Результаты измерений приведены в таблице 3.

Таблица 3. Время расчета при увеличении числа вычислительных ядер.

количество вычислительных ядер	тыс. ячеек/ ядро	подготовка данных (мин:сек)	шаг по времени без расчета переноса излучения (мин:сек)	шаг по времени с расчетом переноса излучения (мин:сек)
16	250	498:56	2:00	4:00
32	125	119:41	1:09	2:24
64	60	30:43	0:41	1:17
128	30	8:48	0:20	0:55
256	15	2:18	0:12	0:26
512	7	1:15	0:07	0:17
1024	3.6	0:31	0:04	0:13

По результатам проведенных измерений в алгоритме выявлено три части, которые масштабируются принципиально по-разному: начальная подготовка данных, расчет временного шага без расчета переноса излучения и расчет шага с расчетом переноса излучения. Для расчета переноса излучения используется многогрупповое по спектру диффузионное приближение, что связано с многократным (по числу спектральных групп) решением уравнения диффузии по неявной схеме, а также с интенсивной работой с таблицами оптических свойств. Поэтому расчет переноса излучения не только занимает от 50% до 70% времени расчета временного шага, но и показатели эффективности его распараллеливания отличаются от показателей других частей алгоритма. Первичная подготовка данных занимает в производственном расчете значительно больше времени, чем в тестовой задаче. Это происходит из-за того, что вычисление геометрических соотношений, необходимых для аппроксимации решаемых уравнений, выполняется в нескольких различных солверах. Тем самым становятся практически неприемлемыми расчеты на малом количестве вычислительных ядер. Например, расчет "первого шага" на 16 ядрах занял более 8,5 часов.

Вычисленные по результатам измерений показатели эффективности распараллеливания представлены в таблице 4 и на рисунках 5 и 6.

Таблица 4. Показатели эффективности распараллеливания при увеличении числа вычислительных ядер.

p	n/p (тыс.)	подготовка данных			шаг по времени без расчета переноса излучения			шаг по времени с расчетом переноса излучения		
		S_p	E_p	C_p	S_p	E_p	C_p	S_p	E_p	C_p
16	250	16	1	478976	16	1	1920	16	1	3840
32	125	66,7	2,1	229792	27,8	0,87	2208	26,7	0,83	4608
64	60	259,8	4,1	117952	46,8	0,73	2624	49,9	0,78	4928
128	30	907,2	7,1	67584	96	0,75	2560	69,8	0,55	7040
256	15	3470,4	13,6	35328	160	0,63	3072	147,7	0,58	6656
512	7	6386,2	12,5	38400	274,3	0,54	3584	225,9	0,44	8704
1024	3.6	15451,2	15,1	31744	480	0,47	4096	295,4	0,29	13312

Ускорение рассчитывалось по формуле $S_p=(T_{16}/T_p)\cdot 16$, т.к. минимальное число процессов, для которых проводились измерение, было равно 16.

На рисунке 5 слева – ускорение при расчете шага по времени, справа – ускорение первичной подготовки данных.

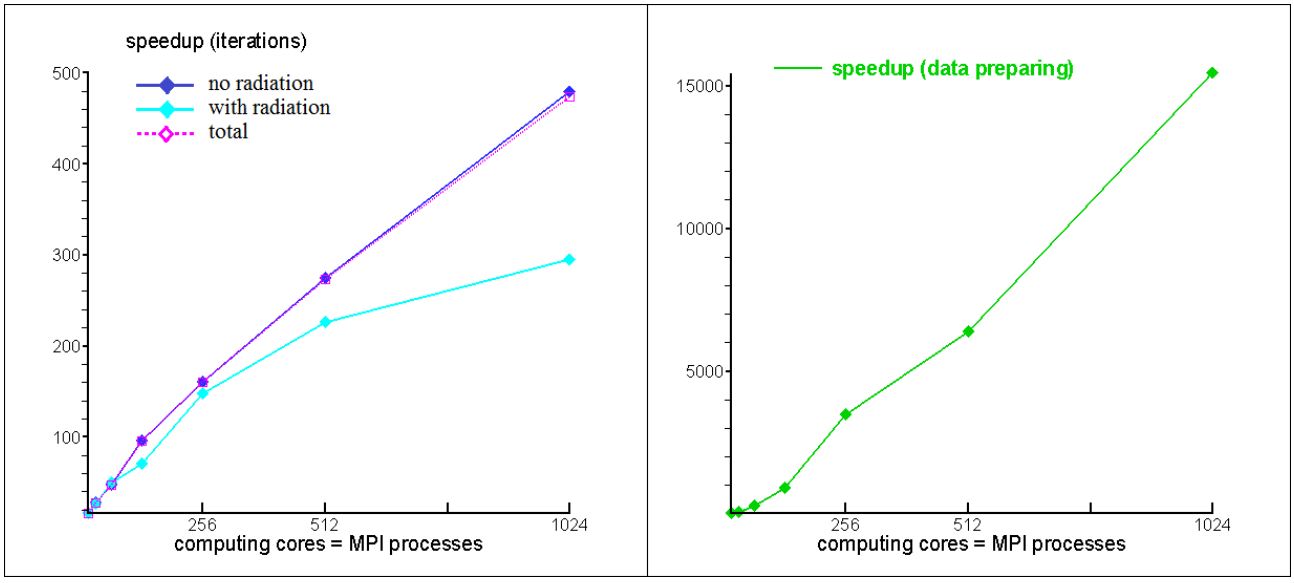

Рисунок 5. Ускорение.

Для первичной подготовки данных имеет место сверхлинейное ускорение (ускорение в 966 раз при увеличении числа процессов в 64 раза). Зависимость ускорения от числа ядер имеет почти квадратичный характер $S_p \sim p^2$. Этому соответствует квадратичная зависимость времени обработки сеточных данных от размера сетки на вычислительном узле $T_p \sim (n/p)^2$.

При расчете временных шагов без переноса излучения (синий график слева на рисунке 5) ускорение на всем исследованном диапазоне остается линейным, но приблизительно в 2 раза меньше максимально возможного.

Ускорение расчета переноса излучения (голубой график слева на рисунке 5) еще меньше, и его рост заметно замедляется с увеличением числа MPI процессов. Для 1024 процессов ускорение почти в 4 раза меньше максимально возможного. В рассматриваемой задаче оказалось возможным использовать схему дробных шагов, в которой временной шаг расчета диффузии излучения приблизительно на 2 порядка больше временного шага для остальных процессов. Такая схема позволила существенно увеличить скорость и масштабируемость расчета в целом. Ускорение данного алгоритма показано слева на рисунке 5 сиреневой пунктирной линией, практически совпадающей с синим графиком "без излучения".

Эффективность использования параллельным алгоритмом процессоров при решении задачи **и стоимость вычислений** приведены на рисунке 6. Эффективность определялась соотношением $E_p=(T_{16}\cdot 16)\,/\,(pT_p)=S_p/p$, т.е. за единицу принималась эффективность использования 16 процессоров (минимальное число, для которого проводились измерения). Относительная стоимость вычислений определялась как произведение времени параллельного решения задачи и числа используемых процессоров, отнесенная к стоимости вычислений на 16 ядрах $C_p=(pT_p)/C_{16}$. В расчете стоимости и эффективности учитывалось только время решения уравнений (без начальной подготовки данных). При использовании 1024 вычислительных ядер стоимость вычислений возросла более чем вдвое, соответственно, эффективность упала до 50%.

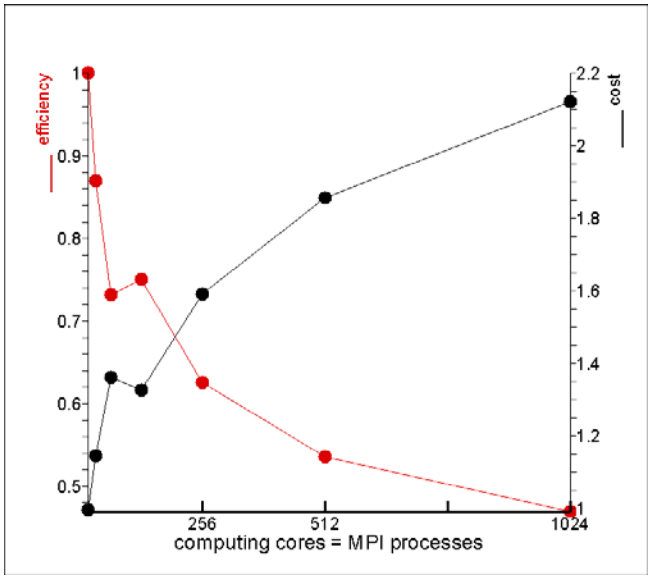

Рисунок 6. Эффективность и стоимость вычислений.

Как и в тестовой задаче, практически приемлемая эффективность вычислений оказалась зависящей от выявленного выше параметра – размера сетки на вычислительном ядре. В данном случае значение этого параметра также лежит в пределах от 50 до 10 тысяч ячеек. Т.е. эффективным было

решение задачи на 64 – 256 ядрах. При меньшем числе ядер неприемлемо долгое время потребовалось для предварительной подготовки данных, при большем – заметно возросла стоимость временных шагов.

Заключение

Для программного комплекса РМГД MARPLE3D, предназначенного для моделирования импульсной магнитоускоренной плазмы на высокопроизводительных ЭВМ, были проведены серии измерений производительности на суперкомпьютерах Ломоносов (НИВЦ МГУ) и МВС-100К (МСЦ РАН). Изучение таких показателей, как ускорение, эффективность использования процессоров и стоимость вычислений, позволило проанализировать масштабируемость параллельных алгоритмов MARPLE3D и выявить следующие их особенности:

- Имеется ресурсоемкий, но хорошо масштабируемый этап предварительной подготовки данных, характерный для большинства алгоритмов, использующих сетки нерегулярной структуры. Наличие этого этапа в начале расчета делает практически неприемлемым решение сложных задач на больших сетках при малом числе вычислительных ядер.
- Имеются различия в масштабируемости разных физических солверов, что необходимо учитывать при построении хорошо масштабируемых алгоритмов.
- Определен показатель практически приемлемой эффективности – размер сетки в вычислительном домене, обрабатываемом одним MPI процессом. Значение этого показателя составило от 50 до 10 тысяч ячеек на вычислительное ядро.

Эти результаты справедливы как для тестовой задачи теплопроводности, так и для типичной производственной задачи из предметной области MARPLE3D, что позволяет предположить применимость сделанных выводов и для других задач, решаемых средствами программного комплекса MARPLE3D. Таким образом, становится возможным априорный анализ эффективности их решения на высокопроизводительных многоядерных системах.

Литература

1. В.А. Гасилов, А.С. Болдарев, С.В. Дьяченко, и др. Пакет прикладных программ MARPLE3D для моделирования на высокопроизводительных ЭВМ импульсной магнитоускоренной плазмы. - Математическое моделирование, 2012, том 24, номер 1, С. 55–87.
2. В.П. Гергель. Теория и практика параллельных вычислений. Интернет-университет информационных технологий - ИНТУИТ.ру, БИНОМ. Лаборатория знаний, 2007 г., 424 с.
3. R.S. Tuminaro, M. Heroux et al. Official Aztec User's Guide Version 2.1, Sandia National Laboratories, Albuquerque, 1999.
4. Я.Б. Зельдович, Ю.П. Райзер. Физика ударных волн и высокотемпературных гидродинамических явлений. М., ФИЗМАТЛИТ, 2008.