ИССЛЕДОВАНИЕ ЗАВИСИМОСТИ МАСШТАБИРУЕМОСТИ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ОТ РАЗМЕРНОСТИ ЗАДАЧИ И ОБЪЕМА ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСОВ

М.Я. Даниленко, А.М. Теплов

Морской гидрофизический
институт НАН Украины
г. Севастополь, ул. Капитанская, 2
E-mail:danilenko-michail@yandex.ru

В статье приведены результаты исследования зависимости масштабируемости параллельных программных комплексов от числа задействованных для решения задачи вычислительных ядер и от размерности задачи, т.е. объема входных данных задачи.

Введение. Часто задачи, решаемые в современных системах контроля окружающей среды, имеют огромные объемы, и решать их иногда необходимо в реальном масштабе времени. Для этих целей все чаще применяются многопроцессорные вычислительные комплексы, функционирующие на основе параллельных программных приложений. Это позволяет добиваться приемлемых результатов во многих современных научных расчетах.

Эффективность выполнения программ по обработке экспериментальных данных на реальных вычислительных системах очень невысока. Проблема напрямую связана со сложностью создания приложения, в котором сочетался бы высокоэффективный алгоритм задачи и его реализация в коде, учитывающая особенности архитектуры конкретной вычислительной системы. Поэтому до начала вычислений часто приходится проводить процесс оптимизации приложений. Это позволяет адаптировать существующий код приложения под особенности, как алгоритмической модели, так и под особенности архитектуры применяемой вычислительной системы. Порой это занимает больше времени, чем создание самого приложения.

Постановка проблемы. Вопрос эффективного использования вычислительных систем, способных справиться с объемными научными и производственными вычислениями, напрямую связан с масштабируемостью – свойством параллельных программ, характеризующим зависимость качества работы параллельной программы от числа использованных процессоров и объема входных данных. Этому вопросу посвящено немало исследований, отраженных в научных статьях и диссертациях. Чаще всего вопрос масштабируемости рассматривается только с точки зрения зависимости характеристик выполнения приложения от числа использованных процессоров. В некоторых источниках вопрос рассматривается с использованием функции изоэффективности. Однако, в таких работах вопрос зависимости динамических характеристик параллельных программ одновременно от числа использованных процессоров и от размерности решаемых задач пока отражен недостаточно. Данная статья касается вопроса анализа масштабируемости программ, как с учетом объема вычислительных ресурсов, так и от размерности решаемых задач, а также вопросов подхода к сравнению масштабируемости параллельных приложений.

Анализ последних достижений. Для поиска причин, снижающих эффективность работы программного комплекса, в настоящее время существует довольно большое количество методов. Для каждого из этих методов разработаны различные инструментарии. Но, несмотря на значительное многообразие существующих методов исследования масштабируемости, эта сторона анализа параллельных программ все еще требует дальнейшего и всестороннего развития. Более того, большое многообразие методов исследования масштабируемости говорит о необходимости дальнейшего поиска более простых и универсальных методов, приемлемых для более широкого круга пользователей параллельных программ [1, 2].

Цель работы. Целью работы, результаты которой представлены в данной статье, является дальнейшая отработка методики исследования масштабируемости параллельных программных

комплексов на примере сравнительного анализа масштабируемости двух относительно простых параллельных приложений.

Методика работы. В предлагаемой работе были проанализированы масштабируемости двух параллельных приложений:

– перемножение двух матриц;

– скалярное произведение двух векторов.

Масштабируемостью параллельной программы будем называть свойство параллельной программы, характеризующее зависимость значений динамических характеристик выполнения параллельной программы от параметров запуска на всех конфигурациях параллельной программы, доступных для рассматриваемой вычислительной системы. Таким образом, если параллельное приложение может быть запущено на различном числе процессоров и для решения задачи разного размера, то масштабируемостью данного приложения будет зависимость времени работы приложения или других динамических характеристик приложения от числа использованных процессоров и размера задачи.

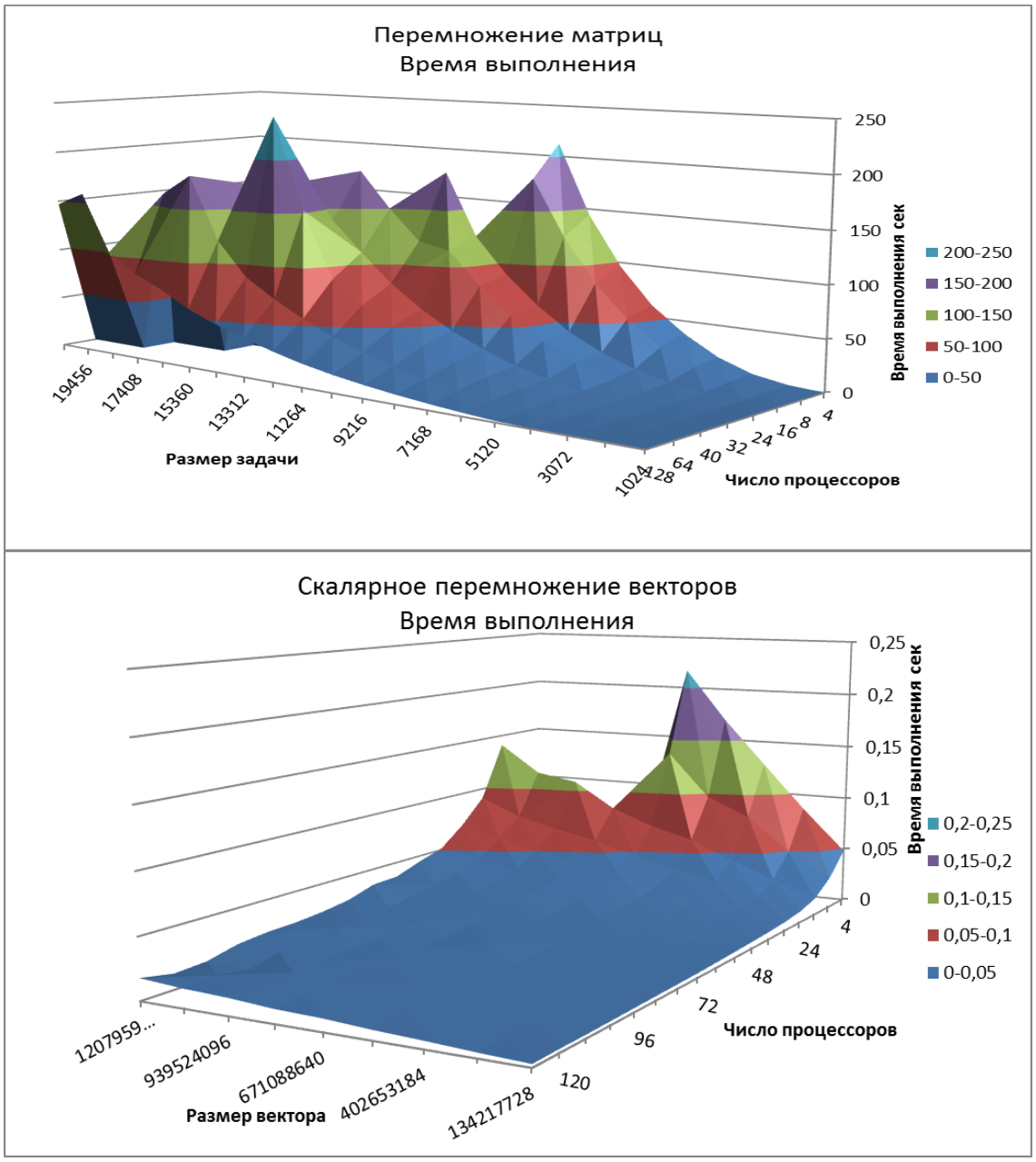
Основной характеристикой, показывающей реальную производительность параллельного приложения, является время его выполнения. Время выполнения – характеристика параллельной программы, которая является абсолютной характеристикой, которую можно измерить. Для изучения масштабируемости время выполнения может быть либо измерено при реальном выполнении параллельной программы, либо использован какой-либо иной принцип для вычисления времени выполнения на каждой интересующей конфигурации. Время выполнения параллельного приложения зависит от большого числа факторов.

Кроме времени выполнения важными характеристиками параллельной программы являются также ускорение времени решения задачи и эффективность использования вычислительных ресурсов.

Конфигурацией запуска параллельной программы будем называть набор параметров, каждый из которых численно характеризует каждый из изменяющихся факторов выполнения параллельной программы. Опорной конфигурацией будем называть такую конфигурацию параметров, относительно которой вычисляются относительные характеристики параллельной программы.

Ускорение параллельной программы на рассматриваемой конфигурации – отношение времени выполнения параллельной программы на самой медленной конфигурации выполнения ко времени выполнения на рассматриваемой конфигурации. Таким образом, ускорение параллельной программы – относительная величина, значение которой зависит от времени выполнения программы, выбранного как время выполнения на самой медленной конфигурации и относительно которого вычисляется ускорение. Величина ускорения может быть больше единицы, если время выполнения программы уменьшилось или принадлежать отрезку [0,1], если конфигурация была выбрана не самой медленной из всех рассматриваемых конфигураций. Если ускорение меньше 1, это означает что время выполнения программы больше, чем на опорной конфигурации.

Масштабируемость параллельной программы – свойство параллельной программы, характеризующее изменение характеристик работы параллельной программы на всех рассматриваемых конфигурациях запуска программы. То есть, если при исследовании масштабируемости параллельного приложения в наборе экспериментов конфигурации запуска изменяются только по размеру решаемой задачи и числу используемых процессоров, картина масштабируемости приложения будет представлять трехмерный набор точек. Каждая точка из этого набора будет представлять усредненный результат эксперимента, проведенного на конфигурации соответствующей координатам по оси размера задачи и по оси числа использованных процессоров. Третья координата точки будет одной из рассматриваемых характеристик работы параллельного приложения (или время выполнения, или ускорение, или эффективность, или какой-то другой).

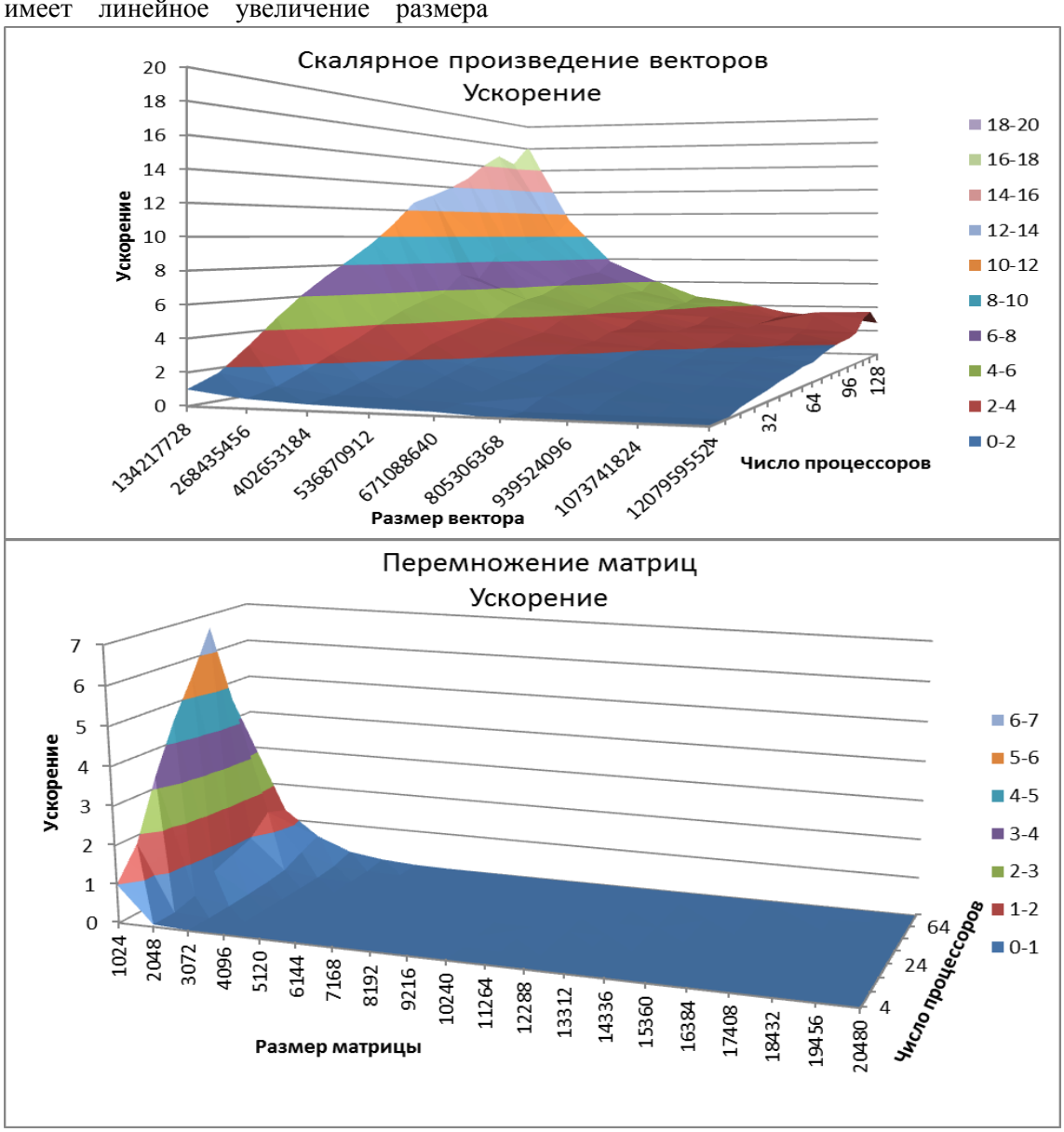


Р и с. 1. Графики времени выполнения

Эксперименты над обеими программами проводились, начиная с минимальной конфигурации вычислительной системы и некоторого минимального размера задачи, на которой время выполнения программы являлось существенным для измерений. Далее увеличивался размер решаемой задачи до тех пор, пока данные для вычислений все еще помещались в оперативную память вычислительных узлов. После достижения максимального размера задачи для данного числа вычислителей увеличивалось число использованных процессоров, и на новом числе процессоров повторяли увеличение размера решаемой задачи. Размер задачи увеличивался согласно ограничениям, накладываемым структурой выполняемой программы. Таким образом, размер решаемой задачи увеличивался с увеличением числа использованных узлов, потому что рос общий объем оперативной памяти.

Результаты экспериментов собирались на основе нескольких прогонов на каждой конфигурации и усреднялись. В результате были собраны усредненные результаты времени выполнения анализуемых параллельных программ.

Перемножение матриц и перемножение векторов имеет различную логическую структуру, и потому имеют различный шаг конфигураций, как по числу использованных процессоров, так и по размеру задачи. Перемножение векторов имеет линейное увеличение размера задачи от 134217728 до 671088640. Перемножение матриц имеет квадратичное увеличение размера задачи от 10242 до 204802. Оба приложения запускались на числе процессоров от 4 до 128.

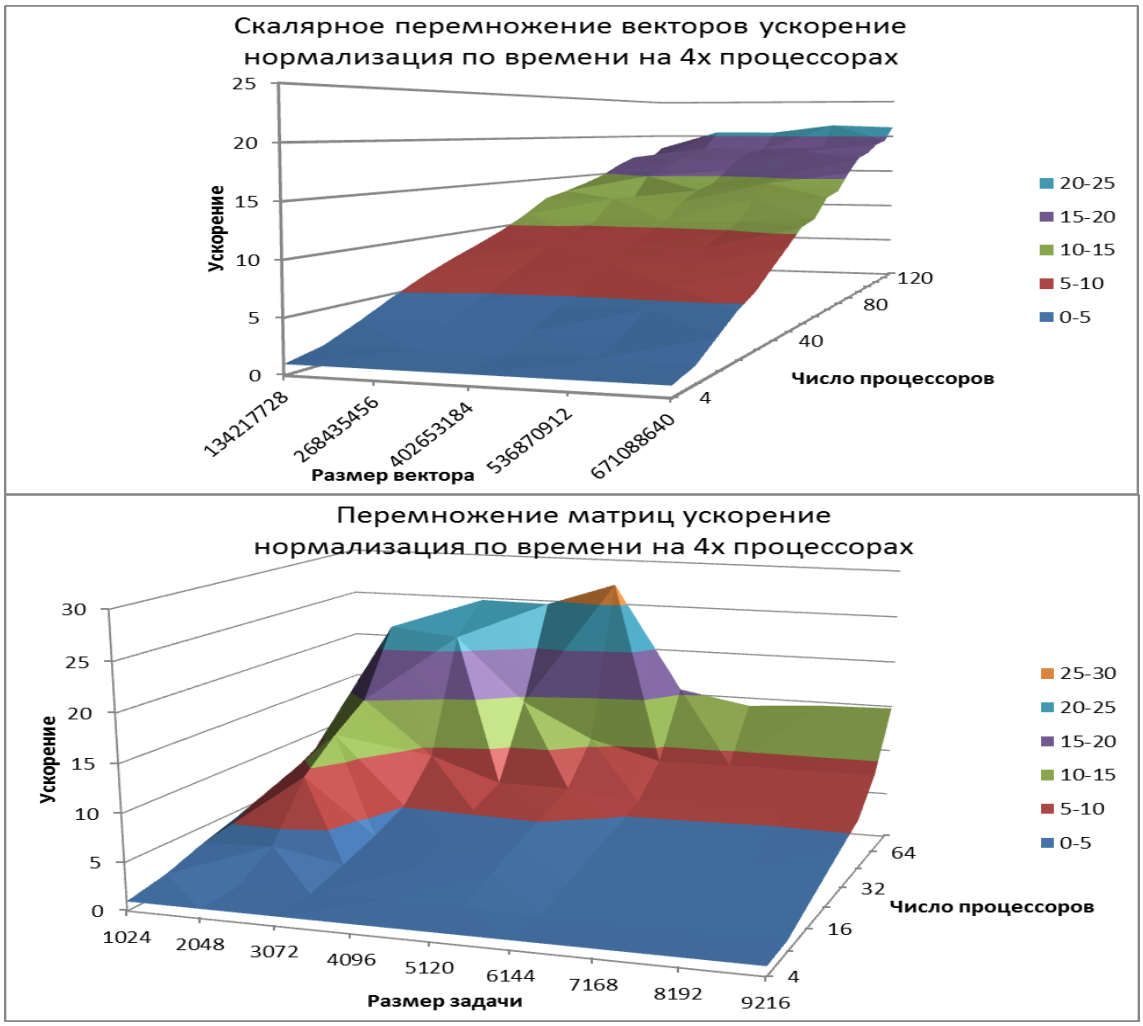


Р и с. 2. Графики ускорения при большом числе процессоров

Полученные результаты. Как видно из графиков, приведенных на рис.1, время выполнения программ отличается очень существенно: время выполнения перемножения матриц превышает 200 секунд, в то время как на скалярное перемножение векторов уходит менее 0,25 секунды. Но характер зависимости у обоих приложений имеет схожую структуру, хотя значения времени выполнения отличаются на порядки. Рассмотрим на рис.2 графики ускорения для каждого приложения.

Расчет ускорения проводился по формуле $S_i=T_i/T_0$, где S_i – ускорение, T_i – время выполнения на i-й конфигурации запуска, T_0 – время выполнения на опорной конфигурации запуска.

Часто в качестве T_0 выбирается время выполнения задачи на минимальном числе процессоров с минимальным размером задачи. Такой подход оправдан в случае фиксированного размера задачи. В случае изменения размера задачи, на графике ускорения будет появляться значительная область значений в промежутке [0,1].



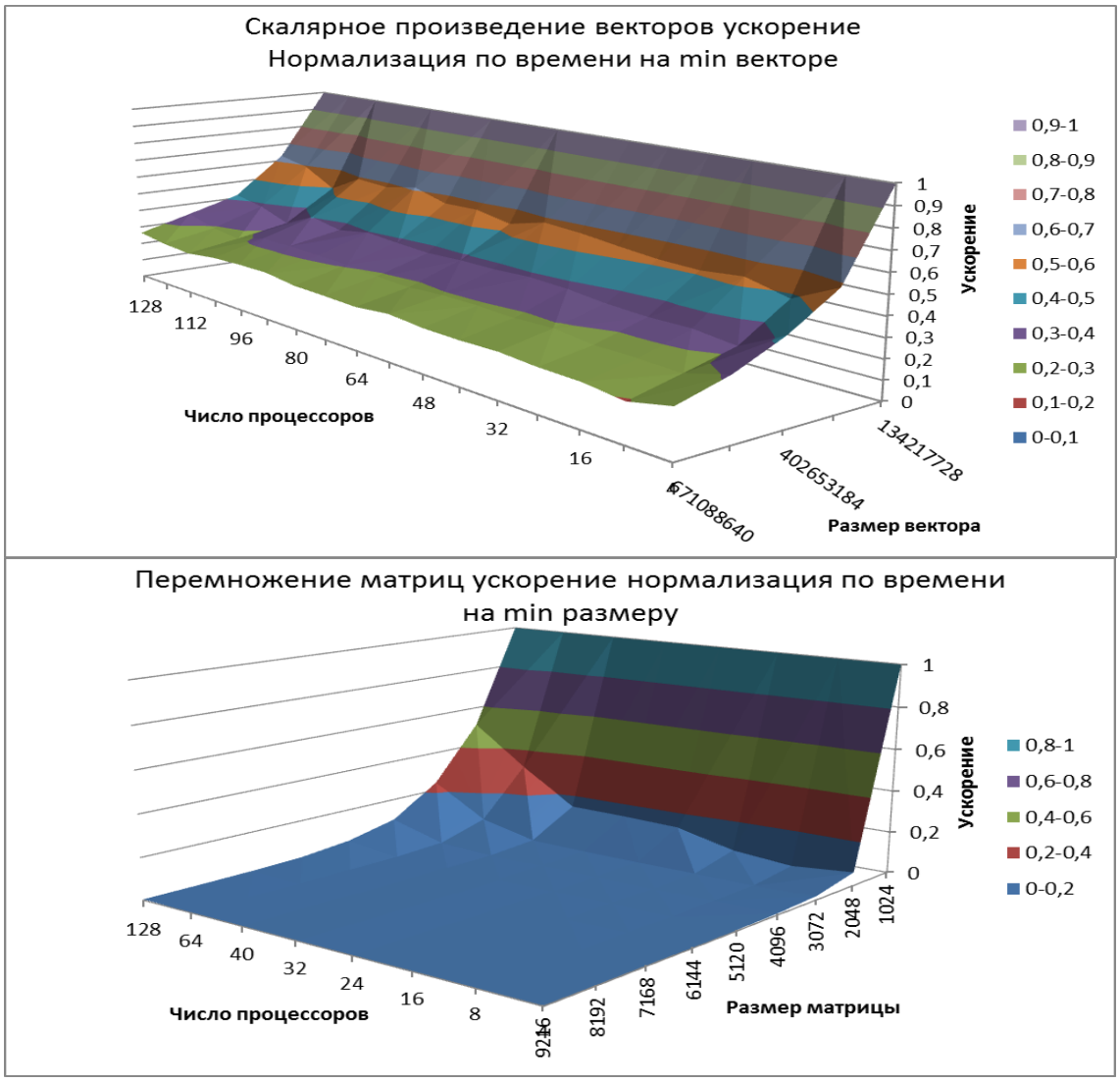
Р и с. 3. Графики ускорения при малом числе процессоров (нормализация по времени на 4-х процессорах)

Как видно из графиков на рис.2 ускорения приложений с ростом размера приложений замедляются. Однако перемножение матриц замедляется значительно быстрее.

Ввиду недостаточной информативности графиков в сравнительно большой области, где ускорение лежит в промежутке [0,1], необходимо рассмотреть, как ведет себя ускорение на этом промежутке более детально. Это можно сделать несколькими способами. Например, можно выяснить, как ведет себя ускорение приложения, фиксируя размер задачи или число процессоров и «послойно» рассматривая каждую получившуюся зависимость. Таким образом, мы как бы нормализуем ускорение по времени работы по одной из координат.

Вначале рассмотрим ускорение приложения, фиксируя размер задачи и «послойно» будем рассматривать каждую получившуюся зависимость. Таким образом, мы нормализуем ускорение по времени работы и (или) по времени выполнения на минимальном числе процессоров.

.



Р и с. 4. Графики ускорений при нормализации по времени на min размере

Согласно графикам на рис. 3, при таком представлении ускорения, вне зависимости от размера задачи, ускорение на 4-х процессорах принимается, равном единице. На графиках ускорения обоих приложений видна тенденция к увеличению величины ускорения с ростом размерности задачи. Это может объясняться довольно быстрым насыщением ускорения при небольших размерах задачи, тогда как приближается предел декомпозиции данных. Однако на графике ускорения перемножения матриц видна область, на которой достигается сверхлинейное ускорение. Эта область на графике занимает ту часть графика, на которой отношение размерности задачи к числу использованных процессоров является приблизительно равной величиной. Это может указывать на тот факт, что на данных конфигурациях запуска данные для расчетов хорошо уложились в КЭШ-память. Таким образом, ускорение увеличивается быстрее, чем растет число использованных процессоров за счет лучшего использования памяти. На графике ускорения для скалярного произведения векторов такого эффекта не наблюдается.

Такое представление данных является достаточно удобным, однако не отражающим, как реально ведет себя ускорение работы программы с увеличением размера задачи. Реальная картина ускорения сильно отличается от прямой линии, равной 1.

Рассмотрим ускорение приложения, фиксируя число процессоров и «послойно» будем рассматривать каждую получившуюся зависимость. Таким образом, мы нормализуем ускорение по времени работы на минимальном размере задачи.

На графиках рис. 4 видно, что все значения графиков лежат в промежутке [0,1] и значения ускорения для перемножения матриц убывают значительно быстрее, чем значения ускорения у скалярного перемножения векторов.

Однако в силу того, что значения данного представления данных расположены на небольшом отрезке значений, а также того, что зависимость является по сути обратной и график является, по сути, не графиком ускорения, а графиком замедления, то такое представление является не очень наглядным и сравнивать их сложно.

Одним из серьезных недостатков такого подхода к построению графиков ускорения, (т.е. нормализовав данные по значениям на минимальном числе процессоров или же по минимальному размеру задачи), является то, что при этом не получится корректно сравнить данные, которые получены на конфигурациях запуска, в которых размер задачи не уместился на минимальной конфигурации вычислительной системы. Это существенно сужает рассматриваемую область значений времени выполнения.

Наибольшей наглядности ускорения работы программы на всех конфигурациях запуска можно добиться, выбрав опорной такую конфигурацию запуска, на которой время выполнения было максимальным. Для этого рассмотрим графики ускорения приложения, построенные по такому принципу (Рис.5).

Для обеих программ опорной конфигурацией стала (точнее примем) конфигурация с максимальным размером задачи на минимальном числе процессоров. Из графиков видно, что при таком отображении более наглядно видна зависимость ускорения задачи от уменьшения ее размеров и увеличения числа процессоров. Для графика ускорения перемножения матриц видно очень сильное ускорение (до 10000 раз) по сравнению с самой медленной конфигурацией запуска. Этот факт может объясняться сверхлинейным ускорением (т. е. более оптимальным использованием памяти), которое было отмечено и на других представлениях ускорения данной программ мы. Ускорение скалярного произведения векторов превышает 80 раз, что отражает тот факт, что с уменьшением размера задачи время выполнения также сильно уменьшается.

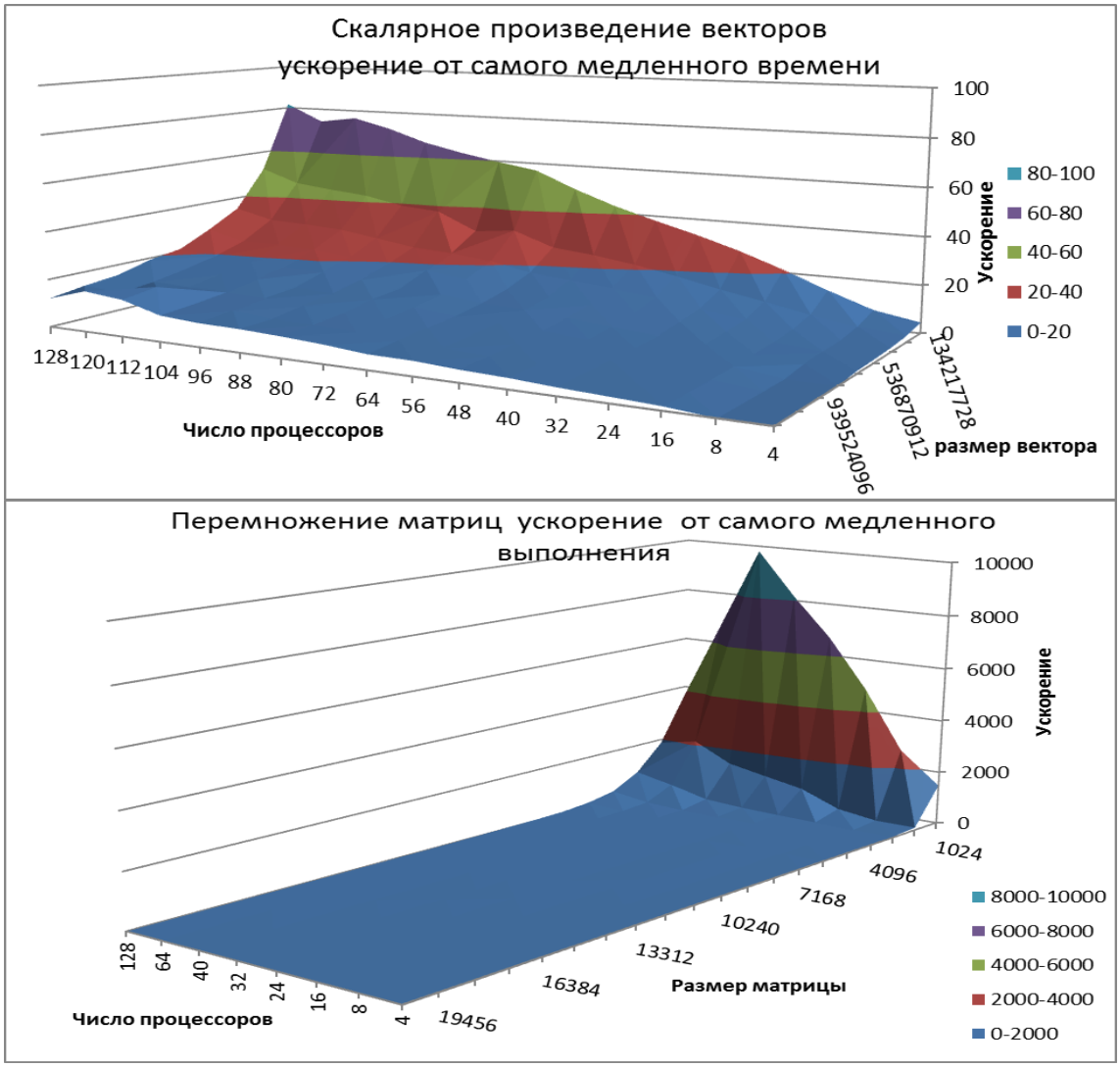
Чтобы сравнить масштабируемость двух приложений, удобнее было бы использовать график эффективности (Рис.6). Для случая, когда фиксирован размер задачи, для расчета эффективности можно применить формулу $E=S_i*p_0/p_i$, где S_i – ускорение, полученное на i-й конфигурации, на которой использовалось p_i процессоров, p_0 – число процессоров на опорной конфигурации запуска. Ограничение на применимость этой формулы сводится как раз к тому, что нельзя корректно сравнивать данные между различными срезами, потому как в формуле не учитывается тот факт, что при росте размера задачи должна расти и эффективность.

На графиках эффективности приложений значения величин, полученных для двух различных приложений, области близких значений пересекаются. Таким образом, можно сравнить масштабируемость двух рассматриваемых приложений с использованием графиков эффективности. На графике перемножения матриц также есть участки с эффективностью более 100%. Эти участки также были отражены и на других представлениях данных о масштабируемости приложения, где, скорее всего, тоже наблюдается сверхлинейное ускорение.

Данные графики наглядно отражают факт того, что эффективность работы этого приложения зависит только от числа использованных процессоров. На числе процессоров, равном 4, оба графика показывают эффективность в 100%

. Это не соответствует действительности, потому как с увеличением объема решаемой задачи и с сохранением числа процессоров эффективность работы должна возрастать.

Такой результат связано в первую очередь с тем, что при сохранении числа процессов увеличивается объем вычислений на каждом ядре процессора. При этом накладные расходы на организацию



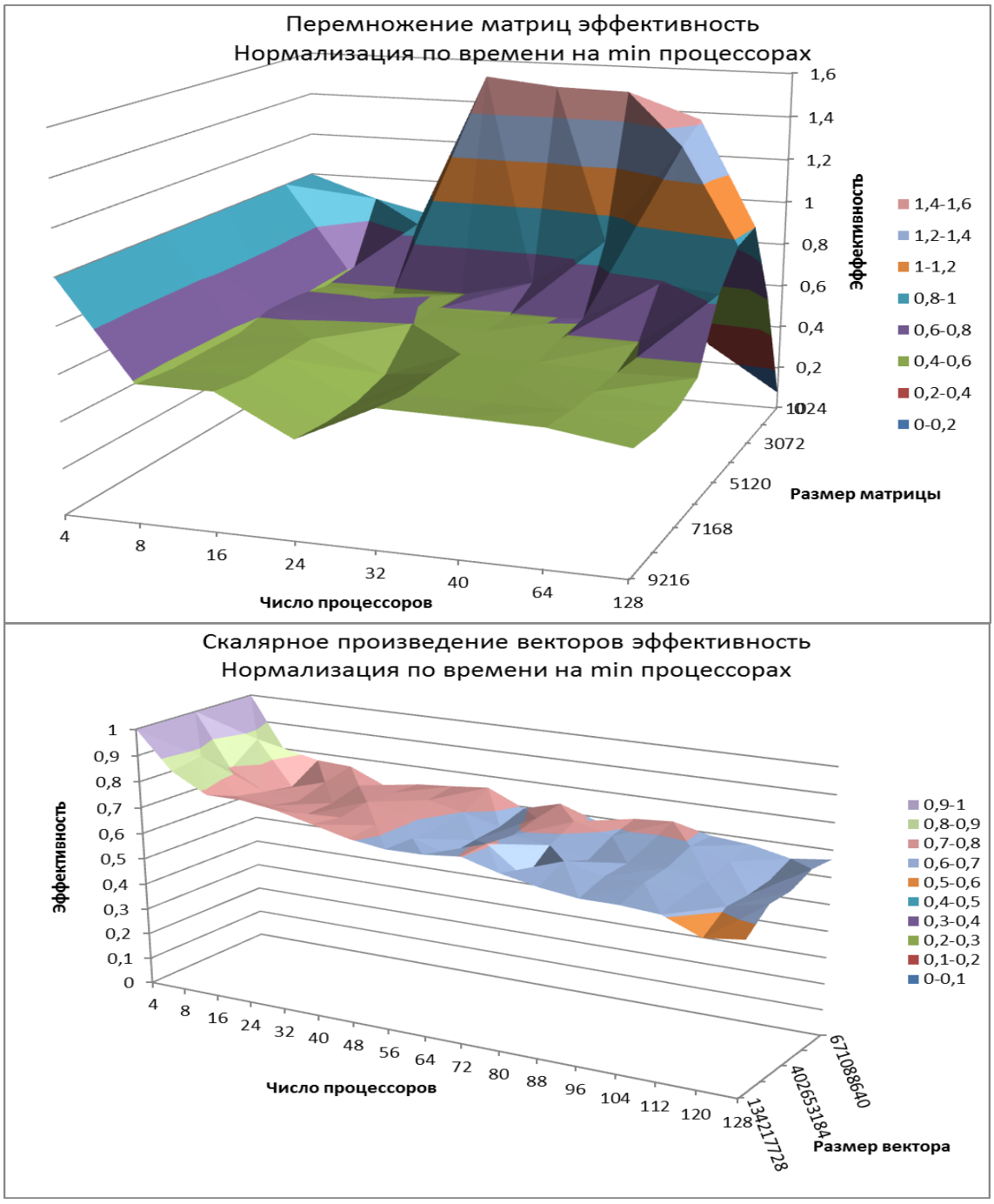
Р и с. 5. Графики ускорения приложений от самого медленного выполнения

параллельных вычислений остаются на прежнем уровне.

При выборе формулы для эффективности, отражающей уменьшение эффективности работы приложения в зависимости от размера данных, если исходить из общих соображений, картина должна быть похожей, с той лишь разницей, что эффективность должна уменьшаться с уменьшением размера задачи и с увеличением числа процессоров.

Поэтому при сравнении масштабируемости приложений лучшей масштабируемостью можно считать масштабируемость того приложения, которое на больших конфигурациях запуска (с большим числом процессоров) продолжает эффективно использовать ресурсы вычислительной системы. Можем также сравнить масштабируемости, например, сравнивая изменение максимальной эффективности работы приложения по всем размерам решаемой задачи с ростом числа использованных процессоров. Для сравнения масштабируемости двух приложений по принципу сравнения их эффективности понадобится величина порога эффективности.

Пороговым значением эффективности будем называть величину эффективности, на которой будем сравнивать максимальное число процессоров, сохраняющих такую эффективность вычислений.



Р и с. 6. Графики эффективности решения задач при нормализации по времени на min процессоров

Тогда, лучше масштабируемость будет у той программы, у которой для данного порогового значения эффективности максимальное число процессоров, на которых эффективность вычислений больше или равна пороговому значению.

Пороговое значение на графике эффективности будет плоскостью, пересекающей график эффективности рассматриваемого приложения, параллельной двум другим осям.

Таким образом, сравнение масштабируемостей двух приложений будет сведено к сравнению максимального числа процессоров и проекции фигуры, получившейся после пересечения графика эффективности плоскостью порогового значения, на ось числа процессоров.

Недостатком такой оценки является тот факт, что два приложения могут иметь различную оценку масштабируемости при выборе разных пороговых значений. Масштабируемость одного приложения может быть хуже другого приложения при пороговом значении в 70% и лучше другого при пороговом значении в 75%.

Это вполне может быть в случае, если одно приложение имеет очень высокую эффективность на небольшом числе процессоров, и эффективность очень быстро падает с ростом числа процессоров, например, из-за проблем с организацией параллельного взаимодействия. Второе приложение имеет не настолько высокую эффективность, однако, с ростом числа процессоров, сохраняет ее почти без изменений. Тогда до этого высокого порога эффективности масштабироваться будет лучше первое приложение, а на более низком пороге – второе.

Можно также сравнивать масштабируемость приложений, рассматривая среднюю эффективность работы приложения. Средней эффективностью работы приложения на рассматриваемом промежутке конфигураций назовем величину, равную среднему значению эффективности работы приложения на всех рассматриваемых конфигурациях запуска приложения. Тогда из двух параллельных приложений лучшей масштабируемостью будет обладать то, у которого больше максимальное значение числа процессоров на пороговой эффективности, равной средней эффективности приложения.

Изложенный подход к сравнению масштабируемостей рассматривает основой для сравнения величину средней эффективности выполнения приложений. Это означает, что средняя эффективность будет значительно больше при наличии на графике явления сверхлинейного ускорения. Для средней эффективности будет важным и то, как быстро уменьшается эффективность с ростом числа процессоров и с уменьшением размера задачи. При таком подходе будет уменьшено влияние эффективности при выборе разных формул для эффективности приложения. Если даже эффективность работы приложения будет довольно низкой, но этот низкий уровень будет сохраняться на широком числе конфигураций запуска, то даже при низкой средней эффективности выполнения будет отражен тот факт, что приложение масштабируется хорошо. Поэтому, сведение сравнения масштабируемости к сравнению числа процессоров будет не оправдано, так как оно не учитывает реальные значения ни размеров задач, ни показатели реальной эффективности для приложений. Эти величины далеко не всегда можно сравнивать между собой напрямую.

Выводы. Результаты исследований, представленные в данной статье, показывают, что при анализе масштабируемости параллельных программ обязательно необходимо учитывать влияние на динамические характеристики вычислений не только объема использованных вычислительных ресурсов, но и объема данных решаемых задач. Описанный в работе метод позволяет сравнивать масштабируемость программ, имеющих различную структуру алгоритма и различные структуры данных для обработки. Можно также сделать вывод о том, что для достижения высоких степеней масштабируемости суперкомпьютерных приложений требуется соответствующая поддержка как со стороны программного, так и аппаратного обеспечения.

СПИСОК ЛИТЕРАТУРЫ

1. *Антонов А.С., Теплов А.М.* Исследование масштабируемости программ с использованием инструментов анализа параллельных приложений на примере модели атмосферы NH3D. // Вестник Южно-Уральского государственного ун-та/серия: Вычислительная математика и информатика. – Челябинск: Изд-во ЮУрГУ, 2013. № 1, том 2. – С. 5–16.
2. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. – СПб.: БХВ-Петербург. 2002. – 608 с.