Оценка сложности стратегий параллельного построения изображений для систем визуализации на суперкомпьютерах*

О.В. Джосан

Московский Государственный Университет имени М.В. Ломоносова

В работе рассматриваются различные стратегии параллельного построения изображений и видеопоследовательностей на суперкомпьютерах для систем визуализации научных данных. Анализируется их вычислительная сложность. Проводится обобщение существующих стратегий для случая произвольного количества коммуникаций на каждом шаге. Приводятся оценки эффективности и масштабируемости стратегии для различных входных параметров задачи. Практическая апробация предложенных методов проведена на суперкомпьютере BlueGene /P.

1. Введение

Метод объемного построения изображений (volume rendering) – это наиболее распространенный подход к визуализации трехмерных данных большого объема, которые получаются в результате крупномасштабного моделирования на суперкомпьютерах. Вычислительные эксперименты такого рода имеют ряд существенных особенностей. Основная особенность – это размер получаемых данных. В работе [1] рассмотрен пример визуализации данных для задачи моделирования несферических частиц. Объем данных, которые визуализируются в эксперименте, составляет 439 гигабайт. Размер данных моделирования турбулентного горения, которые представлены Институтом Institute for Ultra-Scale Visualization [2] в качестве бенчмарка для методов параллельной визуализации данных, составляет порядка 300 гигабайт. Такой объем требует специализированных методов по организации хранения и ввода-вывода данных. Следующей особенностью является распределенное хранение результатов вычислений. При этом число процессоров, на которых проводится эксперимент, может исчисляться десятками тысяч, и в будущем прогнозируется существенный рост этого показателя. Распределенное хранение данных, которые требуется визуализировать, порождает еще одну особенность параллельной визуализации – существенное время, которое требуется на коммуникации между процессорами в процессе построения изображений. Ввиду этих особенностей реализация параллельного построения изображений в реальном времени для крупномасштабных вычислительных экспериментов сейчас является научной проблемой, которая требует разработки новых методов и подходов к такого рода визуализации.

Методы параллельного построения изображений по распределенным данным активно развиваются в течение последних двадцати лет [3]. Существующие алгоритмы традиционно разделяются на три типа по этапу, на котором происходит обмен распределенными данными: sort-first, sort-middle, sort-last [4]. Примеры различных стратегий параллельного построения изображений можно найти в работах [5-10]. Существует ряд программных продуктов, которые поддерживают параллельное построение изображений, в частности можно выделить системы VisIt[13-14] и ParaView [11-12] с ядром параллельного построения изображений IceT [15]. Однако существующие системы не обеспечивают достаточной скорости работы для построения высококачественных изображений и видео в реальном времени.

В данной работе будут рассмотрены только алгоритмы, относящиеся к классу sort-last, в которых каждый процессор визуализирует свою часть данных, и далее осуществляется только обмен изображениями для построения итогового изображения. Алгоритмы класса sort-last включают в себя два шага: шаг построения изображения (image rendering) и шаг компоновки изображения (image compositing). На первом шаге каждый процессор строит свою часть изображения по части данных, которая была получена на нем при вычислениях. Затем на втором

* Работа выполнена при поддержке ФЦП “Научные и научно-педагогические кадры инновационной России” на 2009 - 2013 годы”, гранта РФФИ 11-07-00614-а.

шаге из полученных изображений формируется итоговая картинка. В данной работе детально проанализированы стратегии, применяемые на шаге компоновки изображения для уменьшения времени работы и объема пересылаемых данных.

Самый тривиальный для реализации подход – пересылка всех данных на один процессор. Такой метод условно называется последовательным [17]. Однако этот метод весьма затратный по вычислениям и времени выполнения, а также может быть легко оптимизирован и упрощен, поэтому в реальности не применяется. В терминах MPI проблема компоновки изображения эквивалентна решению reduce-scatter проблемы. Первой оптимизацией является использование топологии виртуального дерева (virtual tree) для пересылки, что позволяет сократить общее время пересылки данных. Такие подходы подробно описаны в работе[18]. Следующим этапом развития алгоритмов компоновки можно назвать появление метода бинарных обменов (binary swap)[19]. Идея алгоритма заключается в пересылке не целого изображения, а только его части. Недостатком такого алгоритма являлось то, что количество процессоров должно было быть степенью 2. Эта проблема была решена в работе [20], где предложен алгоритм на основе 2-3 обменов. Еще одним принципиальным подходом к организации обменов стал сценарий параллельного конвейера (parallel pipelined)[21]. В работе [22] предложена существенная модификация конвейерного алгоритма, названная циклическим разделением (rotate tiling), которая перенаправляет обмены, сокращая количество пересылок.

В настоящее время алгоритмы компоновки изображений развиваются в двух направлениях: 1) применение гибридных подходов, 2) пересылка только значимых пикселей изображения. Примеры гибридных подходов рассмотрены, например, в работах [23] и [24]. Методы выделения только значимых пикселей предложены в работах [25] и [24].

Приведенный обзор показывает, что выбор методов параллельной компоновки изображений достаточно широк. Однако существующие методы решают далеко не все проблемы, актуальные в настоящее время. Так, например, методы с выделением значимых пикселей сталкиваются с проблемой балансировки загрузки, т.к. в этом случае она должна быть динамической, а это означает применение весьма вычислительно сложных алгоритмов.

Большинство реализаций алгоритмов параллельной композиции для ускорения работы использует сжатие данных. В данной задаче требуется алгоритм с невысокой степенью сжатия, однако сжатие должно быть без визуальных потерь и очень быстрым. Методы, используемые в настоящее время, достаточно примитивны. В работе [26] предложен ряд алгоритмов сжатия, которые могли бы более эффективно использоваться для сжатия при компоновке изображений. Некоторые экспериментальные оценки затрат на коммуникации в сетях различной архитектуры получены в работе [27].

Гибридные модели алгоритмов компоновки в большинстве случаев являются статическими и не могут быть динамически адаптированы под конкретную архитектуру или контекст задачи. Интерес представляет анализ обобщенных гибридных моделей и выделение в них параметров, которые в дальнейшем могут быть использованы для динамической адаптации.

Рассмотренные методы по разному решают вопрос, где и в каком виде хранится итоговое изображение. В большинстве случаев финальное изображение храниться после компоновки распределенно. Для отображения на дисплей нужно это изображение передать с вычислительных узлов на интерфейсную машину или записать информацию на жесткие диски. Если бы мы храним изображение распределенно, то в этом случае может эффективно применяться параллельный ввод-вывод. Однако с появлением мультидисплейных комплексов в процессе компоновки появляется еще одна задача по разделению и передаче итоговой картинки на несколько дисплейных устройств. Ее решение также представляет интерес, т.е. может быть интегрировано в процесс компоновки изображения.

В данной работе рассмотрены теоретические и экспериментальные оценки вычислительной сложности различных методов компоновки изображения при параллельном построении. Оценки проводятся с учетом необходимости отображения на мультидисплейном комплексе, соответственно учитываю возможность совмещения компоновки изображения и распределения изображения между отображающими устройствами. Рассмотрено два похода к организации обменов: 1) на каждом шаге задействуются все процессоры, 2) процессоры разбиты на группы и обработка осуществляется в конвейерном режиме. Рассмотрены некоторые гибридные моде-

ли и их параметры, позволяющие адаптировать процесс компоновки в зависимости от особенностей архитектуры вычислительного комплекса и контекста задачи.

2. Оценка сложности методов параллельного построения изображений

Рассмотрим основные параметры, которые необходимо использовать при оценке сложности методов параллельного построения изображений. Пусть итоговое изображение, которое необходимо построить, имеет размер XxY пикселей. Соответственно площадь этого изображения S пикселей (S=X*Y). N - количество процессоров, используемых для построения изображения. $B_{вх}$ и $B_{исх}$ – количество пикселей, которое может быть получено/отправлено процессоров в секунду (пропускная способность). Пусть $B_{вх} = B_{исх} = B$. Путь мультидисплейный комплекс состоит из решетки pxq дисплеев. Общее количество дисплеев – M=p*q. Пусть выполнено условие M<N.

Рассмотрим оценки для последовательного метода, метода бинарных обменов, метода параллельного конвейера, метода циклического разделения. При этом распределение по процессорам, к которым подключены дисплеи, осуществляется последовательно. Также сделано допущение, что время компоновки пренебрежительно мало по сравнению со временем пересылки изображений между процессорами. После этого опишем гибридный подход для мультидисплейных комплексов с использованием балансировки загруженности.

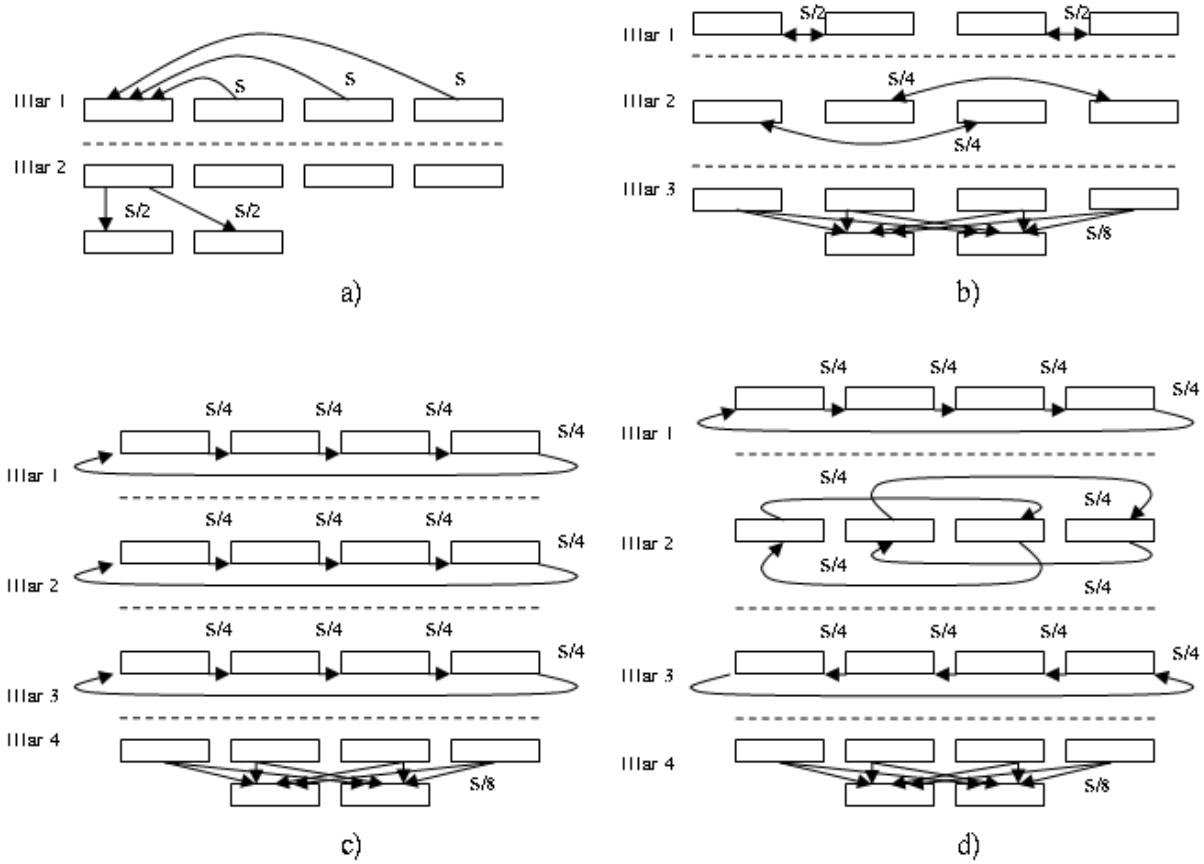

Рис. 1. Схемы обменов при разных стратегиях пересылки для случая N=4, p=1, q=2. a) последовательный метод; b) метод бинарных обменов; c) метод параллельного конвейера; d) метод циклического разделения.

2.1 Оценки сложности для последовательного метода

Суть последовательного метода состоит в том, что все данные передаются на один процессор, далее с него распределяются по процессорам, к которым подключены дисплеи. Сценарий

работы для этого метода для случая N=4, p=1, q=2 проиллюстрирован на Рис.1a. Общий объем передаваемых данных (в пикселях): V_{SS}=S*(N-1)+S =S*N. Поскольку все выполняется последовательно, по время на пересылку вычисляется по формуле: T_{SS}= S*N/B.

2.2 Оценки сложности для метода бинарных обменов и конвейерного метода

Метод бинарных обменов для случая N=4, p=1, q=2 проиллюстрирован на Рис.1b. На первом шаге алгоритма процессоры обмениваются S/2 пикселями изображения, на следующем шаге S/4 и т.д. Количество итераций в этом случае $\log_2 N$. Объем данных, который последовательно передается на i-й итерации, составляет: $V_i=S/2^i$. Время на i-ой итерации составляет $T_i= S/(B*2^i)$. Таким образом суммарное время на сборку и объем передаваемых последовательно данных можно вычислить следующим образом:

$$T_{BS}'= \sum_{i=1}^{\log_2 N} \frac{S}{B*2^i} = \frac{S}{B}\left(1-\frac{1}{N}\right); \quad V_{BS}'= \sum_{i=1}^{\log_2 N} \frac{S}{2^i} = S\left(1-\frac{1}{N}\right);$$

После выполнения алгоритма на каждом процессоре получится S/N пикселей итогового изображения. Поскольку сборка на M процессоров для отображения на дисплеи осуществляется последовательным образом, то количество данных, которое необходимо последовательно передать, можно посчитать как количество данных, которое придет на каждый из M процессоров. Таким образом, V_2=S/M; T_2=S/(B*M). В итоге получаем, что время, которое тратится на построения изображения в формате для мультидисплеев с помощью метода бинарных обменов:

$$T_{BS} = \frac{S}{B}\left(1-\frac{1}{N}\right)+\frac{S}{BM} = \frac{S}{B}\left(1-\frac{1}{N}+\frac{1}{M}\right);$$

Объем передаваемых данных:

$$V_{BS} = S\left(1-\frac{1}{N}+\frac{1}{M}\right);$$

Идея конвейерного метода состоит в том, что на каждой итерации процессор передает и получает S/N пикселей изображения. При этом выполняется N-1 итераций. После этих итераций осуществляется последовательная сборка на M Процессоров. С учетом оценок полученных в работе [22] получаем, что оценка времени работы и объема передаваемых данных с учетом мультидисплейности совпадает с показателями для метода бинарных обменов. Работа метода проиллюстрирована на Рис.1c.

2.3 Оценки сложности для метода циклического разделения

Верхние оценки времени работы алгоритма и объема передаваемых данных получены авторами метода в работе [22]. Используя эти оценки для случая мультидисплейности получаем:

$$V_{RT} = S\left(\frac{1}{N}\sum_{i=1}^{\log_2 N}\left(1-\frac{1}{N}\right)^{i-1} * \frac{1}{2^{i-1}} - \frac{1}{N}+\frac{1}{M}\right);$$

$$T_{RT} = \frac{S}{B}\left(\frac{1}{N}\sum_{i=1}^{\log_2 N}\left(1-\frac{1}{N}\right)^{i-1} * \frac{1}{2^{i-1}} - \frac{1}{N}+\frac{1}{M}\right);$$

Таким образом из проанализированных классических методов сборки изображения метод циклического разделения требует наименьший размер передаваемых последовательно на каждой итерации данных и время его работы меньше остальных. Однако в каждом из перечисленных методов в оценках присутствует слагаемое, получаемое из применения последовательного метода перераспределения изображения для мультидисплейного комплекса. Очевидно, что этот шаг можно оптимизировать, применив гибридную стратегию, в рамках которой компоновка изображения совмещена с распределением его по мультидисплейному комплексу.

3. Метод гибридной сборки с учетом мультидисплейности

3.1 Описание гибридной стратегии сборки для мультидисплейного комплекса

В данной статье автором предлагается стратегия гибридной сборки для параллельной компоновки изображения. Стратегия гибридной сборки представляет собой объединение шагов компоновки изображения и распределения итогового изображения по процессорам для отображения на мультидисплейном комплексе. Возможно применение различных алгоритмов компоновки изображения при таком подходе. Однако различные стратегии будут давать различный результат при исполнении программы на различных параллельных архитектурах. Это связано с тем, что аппаратно не обеспечивается одинаковая скорость при коммуникации между различными процессорами. Для эффективной реализации параллельной компоновки изображения и его распределения возможно использование коллективных операций MPI, в частности операции reduce.

Для использования этой операции требуется обеспечить дополнительное условие, что операция компоновки изображения будет ассоциативной по отношению к порядку, котором происходит сборка изображения. Однако в случае компоновки изображения это условие может быть ослаблено. Достаточно, чтобы ошибка, которая вносится при разном порядке компоновки, была визуально не заметна.

Наибольшего ускорения возможно было бы добиться при использовании неблокурующих вызовов reduce, но такая функциональность будет поддерживаться только в стандарте MPI-3. В текущем стандарте MPI используются блокирующие операции reduce, поэтому компоновка итогового изображения для M дисплеев эквивалентна последовательному выполнению M вызовов reduce для каждого из процессоров, к которому подключен дисплей. Каждый шаг операции reduce проводится для части изображения, которая соответствует текущему дисплею. В общем виде схему выполнения операции можно представить следующим образом (Рис.2).

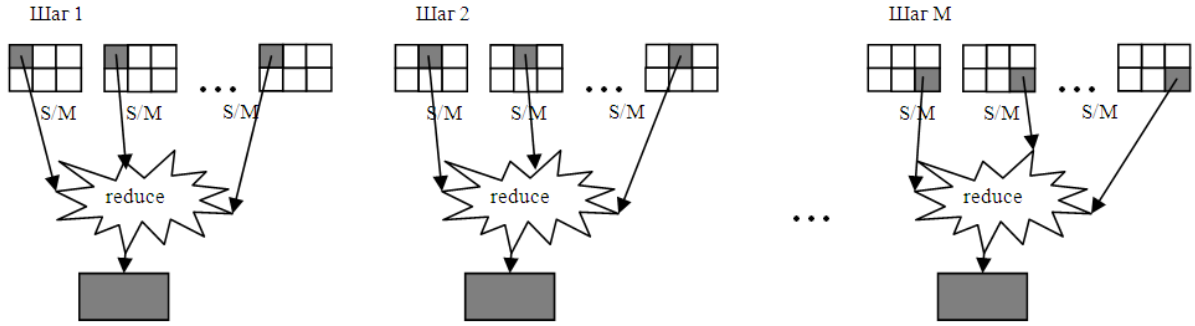

Рис.2. Схема компоновки изображения для мультидисплейного комплекса с использованием MPI-операции reduce.

Теоретически выполнение операции reduce на каждой конкретной параллельной архитектуре будет выполняться быстрее, чем произвольный алгоритм компоновки изображения, т.к. разработчики аппаратного и системного программного обеспечения реализуют базовые функции MPI максимально эффективно для своей архитектуры. В частности, на суперкомпьютере Blue Gene/P используется специальная коммуникационная сеть для выполнения коллективных операций, что позволяет получить существенных выигрыш по времени выполнения по сравнению с самостоятельно реализованными алгоритмами обменов, использующих коммуникации точка-точка. Поэтому получение теоретической оценки ожидаемого времени выполнения затруднительно, т.к. требует детального знания архитектуры конкретной параллельной системы и особенностей реализации операций MPI. Однако эти оценки могут быть получены экспериментально. Соответствующий эксперимент был выполнен на суперкомпьютере Blue Gene/P и его результаты приведены в разделе 4 данной работы.

3.2 Использование балансировки по данным для оптимизации времени выполнения

При визуализации научных данных в получаемом изображении далеко не все пиксели являются значащими. Больша*я* часть пикселей относится к фоновым и имеет постоянное значение. Для эффективной передачи данных можно использовать различные алгоритмы сжатия перед компоновкой. Однако использование сжатия приводит дисбалансу загруженности процессоров, т.к. у некоторых процессоров окажется часть изображения, где большинство пикселей значащие, а некоторые процессоры получат только фоновые пиксели, которые будут эффективно сжаты.

В качестве одного из подходов к балансировке загруженности процессоров при использовании сжатия может быть применено неравномерное разбиение по площади для частей, которыми обмениваются процессоры. Но такой подход не применим, если в итоге должно быть получено не одно изображение, а несколько изображений для мультидисплейного комплекса.

Требуется предложить способ разбиения данных между процессорами, при котором возможно будет использовать сжатие, при этом будет сохранена равномерная загруженность процессоров и в итоге получится изображение для мультидисплейного комплекса. Для выполнения этой задачи предполагается следующий подход. Предлагается выполнять операцию reduce не для части данных, которые относятся к одному дисплею, а по полосе данных, которая включает в себя данные для q дисплеев в ряду. При таком способе получается более равномерное распределение значащих пикселей. Далее полученное сжатое изображение пересылается на q процессоров с помощью MPI операции scatter.

4. Экспериментальная оценка эффективности и масштабируемости на Blue Gene /P

4.1 Описание эксперимента

Эксперимент проводился на суперкомпьютере Blue Gene/P МГУ. Получения экспериментальных оценок эффективности выполнения была использована реализация 5 методов:

1) последовательный (SS)
2) бинарных обменов (BS)
3) конвейерный метод (RS)
4) метод на основе reduce-вызовов (RED)
5) метод на основе reduce-вызовов с использованием оптимизации по данным (REDopt)

Методы 1) - 3) были реализованы на основе стратегий библиотеки IceT[15]. Эксперимент проводился для визуализации данных молекулярной динамики для молекул с порядком атомов 10^5. Однако в эксперименте оценивалось только время выполнения и эффективность компоновки изображения, поэтому входными данными алгоритма можно считать изображения в S пикселей, полученные на каждом процессоре, по данным, которые у него были.

Проведены следующие эксперименты:

1) анализ времени выполнения на различном количестве процессоров при фиксированном размере изображения и фиксированной сетке мультидисплейного комплекса;
2) анализ времени выполнения при изменяющемся размере изображения;
3) анализ времени выполнения при изменяющейся сетке мультидисплейного комплекса.

В экспериментах использовалось 128, 256, 512, 1024 процессора. Размер изображения менялся от 2000x2000, 1000x1000, 500x500 для каждого дисплея из мультидисплейного комплекса. Сетки мультидисплейного комплекса брались размером 2x2, 2x4 и 4x4.

4.2 Результаты

Графики полученных в вычислительном эксперименте результатов приведены на Рис.3.

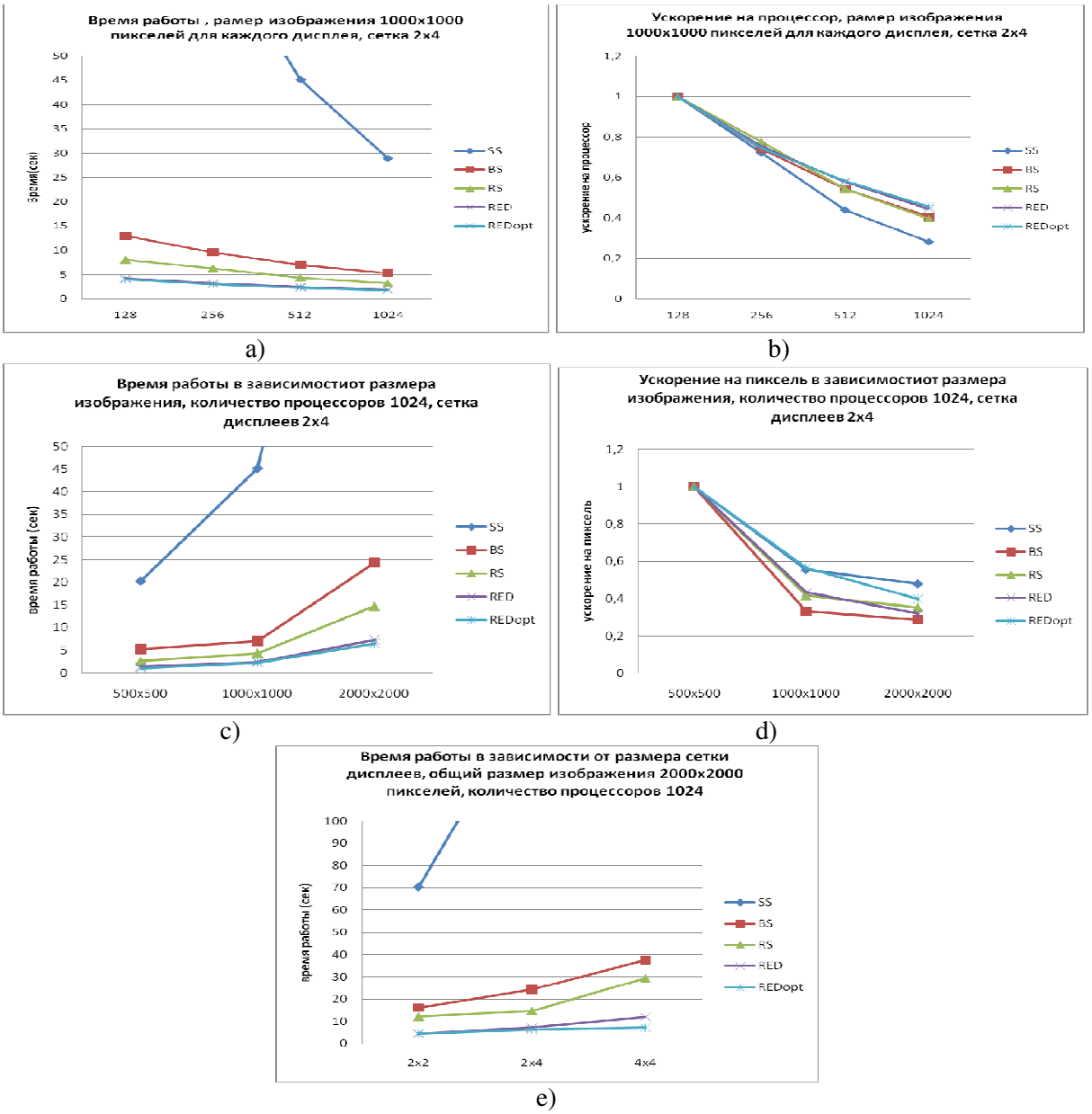

Рис.3. Результаты вычислительных экспериментов на Blue Gene/P.

На Рис.3a показан график времени работы методов при изменении количества процессоров. При этом фиксирован размер изображения для каждого дисплея 1000x1000 пикселей и сетка мультидисплейного комплекса: размер 2x4. На Рис.3b показано ускорение на процессор в этом же эксперименте. Как видно из графика, полученное решение довольно плохо масштабируемо и дает низкую эффективность при большом количестве процессоров. Однако предложенные методы RED и REDopt более эффективны, чем известные ранее методы.

На Рис.3c показано время работы методов при использовании различного размера изображения для каждого из дисплеев. При этом фиксировано количество процессоров – 1024 и сетка мультидисплейного комплекса 2x4. На Рис.3d показано ускорение на пиксель в зависимости от размера изображения. Использование в методе RED оптимизации по данным и сжатия передаваемых данных позволило существенно улучшить показатель по этому параметру.

На Рис.3e показана зависимость времени работы метода от изменения сетки дисплеев. При этом фиксировано значение размера входного изображения 2000x2000 пикселей и количество используемых процессоров: 1024x1024. Как видно из графика оптимизация по данным позволила уменьшить время работы алгоритма RED на сетке из 16 дисплеев в два раза. Необходимо

провести дополнительное исследование на сетке большей размерности и при большем общем размере изображения.

5. Заключение

В работе проведен анализ методов параллельной компоновки изображения при визуализации научных данных большого размера на суперкомпьютерах методом объемного построения изображений. Приводится теоретическая оценка сложности методов для случая, когда на выходе получается не одно изображение, а несколько для отображения на мультидисплейном комплексе. Предложен гибридный метод компоновки изображения, объединяющий стадии компоновки изображения и распределения изображения между несколькими экранами. Предложена оптимизация это метода, позволяющая сократить размеры передаваемых данных и соответственно общее время работы метода. Проведено экспериментальное исследование эффективности предложенного метода и его оптимизации на суперкомпьютере Blue Gene/P. В дальнейших исследованиях предполагается совершенствование метода сжатия данных для ускорения работы программы, исследование масштрабируемости предложенных методов на суперкомпьютере «Ломоносов» и оптимизация предложенных методов для этого суперкомпьютера.

Литература

1. Götz J., Iglberger K., Stürmer M., Rüde U. Direct Numerical Simulation of Particulate Flows on 294912 Processor Cores // Proc. Conf. High Performance Computing, Networking, Storage and Analysis. Washington, DC, USA, pp.1 -11, 2010.
2. The SciDAC Ultra-Scale Visualization Institute. URL:http://vis.cs.ucdavis.edu/Ultravis/datasets/ (дата обращения: 10.01.2011).
3. Survey of Parallel Volume Rendering Algorithms URL:www.hpl.hp.com/research/mmsl/presentations/3d/pdpta98.pdf (дата обращения: 10.01.2011).
4. Mueller C. The sort-first rendering architecture for high-performance graphics // In ACM SIGGRAPH ASIA 2008 courses (SIGGRAPH Asia '08). New York, USA, Article 36 , 11 pages, 2008.
5. Crockett T.W., Orloff T. A MIMD rendering algorithm for distributed memory architectures // Proceedings of the symposium on Parallel rendering -1993, p.35-42, San Jose, California, USA, 1993.
6. Pajarola R. Cluster parallel rendering // In ACM SIGGRAPH ASIA 2008 courses (SIGGRAPH Asia '08). New York, USA, Article 34, 12 pages, 2008.
7. Martin K.M., Geveci B., Ahrens J., Law C. Large Scale Data Visualization Using Parallel Data Streaming // IEEE Comput. Graph. Appl., Vol. 21, pp. 34-41, 2001.
8. Berkant B.C., Cevdet A. Hypergraph-Partitioning-Based Remapping Models for Image-Space-Parallel Direct Volume Rendering of Unstructured Grids // IEEE Trans. Parallel Distrib. Syst., Vol.18, pp. 3-16., 2007.
9. Moloney B., Ament M., Weiskopf D., Möller T. Sort First Parallel Volume Rendering // IEEE Transactions on Visualization and Computer Graphics, Vol.9, 2010.
10. Finkbeiner B., Entezari A., Van De Ville D., Möller T. Efficient volume rendering on the body centered cubic lattice using box splines // Computers & Graphics, Vol. 34(4). pp. 409 - 423, 2010.
11. Cedilnik A., Geveci B., Moreland K., Ahrens J., Favre J. Remote Large Data Visualization in the ParaView Framework // A. Heirich, B. Raffin, L. P. Santos (eds.) editors, Eurographics Parallel Graphics and Visualization 2006, pp. 162--170, 2006.

12. Moreland K., Avila L., Lee Ann Fisk. Parallel Unstructured Volume Rendering in ParaView. // In Visualization and Data Analysis 2007, Proceedings of SPIE-IS&T Electronic Imaging, vol. 6495, pp. 64950F-1–12, 2007.

13. Foulks A., Bergeron R.D. Uncertainty visualization in the VisIt visualization environment // Proceedings Visualization and Data Analysis 2009, Vol. 7243, 2009.

14. VisIt Visualization Tool. URL: https://wci.llnl.gov/codes/visit/ (дата обращения: 10.01.2011).

15. Moreland K. IceT Users' Guide and Reference. Tech Report SAND 2009-3170, June 2009.

16. Peterka T., Goodell D., Ross R., Han-Wei Shen, Rajeev Thakur. A configurable algorithm for parallel image-compositing applications // Proc.Conf. High Performance Computing Networking, Storage and Analysis. New York, USA, Article 4 , 10 pages, 2009.

17. Porter T., Duff T. Compositing digital images // SIGGRAPH Comput. Graph., Vol. 18, pp. 253-259, 1984.

18. Moreland K., Wylie B., Pavlakos C. Sort-last parallel rendering for viewing extremely large data sets on tile displays // In Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics. IEEE Press, Piscataway, USA, pp. 85-92. 2001.

19. Ma K.-L., Painter J. S., Hansen C. D., Krogh M. F. Parallel volume rendering using binary-swap compositing // IEEE Computer Graphics and Applications, vol. 14, №4, pp. 59–68, 1994

20. Yu H., Wang C., Ma K.-L., Massively parallel volume rendering using 2-3 swap image compositing // Proc. Conf. Supercomputing. Piscataway, USA: IEEE Press, pp. 1–11, 2008.

21. Lee T.-Y., Raghavendra C. S., Nicholas J. B. Image composition schemes for sort-last polygon rendering on 2d mesh multicomputers // IEEE Transactions on Visualization and Computer Graphics, vol. 2, no. 3, pp. 202–217, 1996.

22. Lin C.F., Liao S.K., Chung Y.C. A rotate-tiling image compositing method for sort-last parallel volume rendering systems on distributed memory multicomputers // Journal of Information Science and Engineering, pp.643-664, 2004.

23. Nonaka J., Ono K., Miyachi H. Theoretical and Practical Performance and Scalability Analyses of Binary-Swap image Composition Method on IBM Blue Gene/L // The 1st International Workshop on Super Visualization (IWSV), June 7, 2008, Kos Is., Greece, 2008.

24. Takeuchi A., Ino F., Hagihara K. An improved binary-swap compositing for sort-last parallel rendering on distributed memory multiprocessors // Parallel Comput., vol. 29, no. 11-12, pp. 1745–1762, 2003.

25. Peterka T., Goodell D., Ross R., Shen H.-W., Thakur R. A Configurable Algorithm for Parallel Image-Compositing Applications // Proc. Conf. High Performance Computing, Networking, Storage, and Analysis, Portland, Oregon, USA, 2009. Preprint ANL/MCS-P1624-0509, May 2009.

26. Джосан О.В., Мурынин А.Б., Попова Н.Н. Метод визуализации многомерных динамических данных на многопроцессорных комплексах // Вестник компьютерных и информационных технологий. 2009. № 8. сс. 8-12.

27. Корж А.А., Макагон Д.В., Оценка минимальных требований к аппаратуре и топологии при построении высокоскоростных коммуникационных сетей для суперкомпьютеров с общей памятью // Вычислительные методы и программирование: новые вычислительные технологии. 2008. Т. 9. № 2. сс. 26-31.