ИСПОЛЬЗОВАНИЕ ДАННЫХ СИСТЕМНОГО МОНИТОРИНГА ДЛЯ ОПРЕДЕЛЕНИЯ ФАКТОРОВ, УМЕНЬШАЮЩИХ МАСШТАБИРУЕМОСТЬ ПРИЛОЖЕНИЯ

А.С. Антонов, А.М. Теплов

Научно-исследовательский вычислительный центр МГУ имени М.В. Ломоносова

1. Введение

На эффективность и масштабируемость параллельного приложения [1-3] оказывают влияние различные факторы. К ним относятся как факторы, связанные с алгоритмом работы приложения и особенностями его реализации, так и факторы, связанные со свойствами программно-аппаратной среды, в которой приложение выполняется.

Эффективность работы параллельного приложения связана с другими динамическими характеристиками его работы и определяется отношением реальных значений показателей этих характеристик к максимальным теоретическим значениям. Различают *эффективность распараллеливания* – отношение реально полученного ускорения к максимально возможному ускорению, и *эффективность реализации* - отношение полученной производительности к пиковой производительности использованных вычислительных ресурсов. *Масштабируемостью* параллельного приложения называется зависимость изменения динамических характеристик параллельной программы от параметров запуска. Чаще всего рассматривают зависимость изменения динамических характеристик от числа использованных процессоров (*сильная масштабируемость*) и объёма входных данных (*масштабируемость вширь*).

В силу взаимосвязи между понятиями эффективности и масштабируемости параллельного приложения, влияние на масштабируемость будут оказывать, в том числе, те факторы, которые оказывают влияние и на эффективность выполнения. Однако при анализе масштабируемости приложения важнее не то, как сильно влияет каждый фактор в отдельности, а то, каким образом это влияние меняется при изменении параметров работы приложения.

В данной статье рассмотрено, каким образом факторы, влияющие на масштабируемость приложения, отражены в данных системного мониторинга, собранных при работе параллельного приложения. Такие данные могут быть полезны для анализа работы приложения методом «чёрного ящика» при отсутствии доступа к исходному коду, а также при невозможности использования или же отсутствии иных специальных средств анализа. Данные мониторинга считываются с аппаратных датчиков и не требуют модификации кода программы, поэтому их получение является довольно простым, практически не оказывающим влияние на работу самого приложения.

1. Использованный инструментарий для исследования

Приводимые в работе данные получены с помощью систем мониторинга работы приложений, установленных на суперкомпьютерах СКИФ МГУ «Чебышёв» [4] и «Ломоносов» [5]. Система мониторинга использует данные, полученные со множества аппаратных датчиков. Данные мониторинга со всех узлов суперкомпьютера поступают в общее хранилище данных. После этого данные из хранилища ассоциируются с приложениями, выполняемыми на суперкомпьютере.

Система JobDigest [6], разработанная для анализа характеристик приложений, выполняемых на системах Суперкомпьютерного комплекса МГУ, использует данные мониторинга для визуального представления состояния вычислительной системы и сохранения показаний различных аппаратных датчиков во время выполнения параллельной программы. В зависимости от конфигурации отчёт о работе программы может содержать в себе информацию о различных датчиках, и данные с этих датчиков могут быть сгруппированы по-разному.

Инструментарий WEKA [7] является свободно распространяемым программным пакетом с открытым исходным кодом для анализа последовательностей векторов данных. WEKA содержит в себе большое число алгоритмов интеллектуального анализа данных и визуализации. Для использования этого пакета данные системы JobDigest преобразуются в вектор значений различных датчиков таким образом, чтобы в одном векторе значения датчиков соответствовали одному и тому же временному интервалу. Значения от каждого датчика в формируемом векторе занимают соответствующие позиции таким образом, чтобы в каждом полученном векторе в одной и той же позиции находились данные от одного и того же датчика.

2. Факторы, уменьшающие масштабируемость параллельных программ

По результатам проведённых исследований были выделены факторы, в наибольшей степени влияющие на масштабируемость параллельных программ. Описание данных факторов разобьём на несколько групп.

К первой группе отнесём факторы, связанные с использованием коммуникационной сети, как одной из наиболее важных компонент современного суперкомпьютера. Посредством коммуникационной сети происходит взаимодействие процессов, поэтому влияющие на эффективность этого взаимодействия факторы будут влиять и на масштабируемость параллельной программы. При увеличении числа взаимодействующих процессов накладные расходы на организацию сетевого взаимодействия будут увеличиваться и тем самым оказывать влияние на масштабируемость. К таким факторам будем относить латентность коммуникационной сети, пропускную способность, топологию сети и т.п.

Ко второй группе отнесём факторы, связанные с эффективным использованием компонентов вычислительного узла суперкомпьютера. К ним относятся факторы, связанные с объёмом и характеристиками всей иерархии памяти вычислительного узла. При увеличении числа процессов на узле каждому процессу достаётся меньший объём быстрой памяти, что приводит к резким изменениям эффективности работы при выходе за границы этого участка.

Наконец, к третьей группе отнесены факторы, связанные с характеристиками используемого алгоритма или исследуемой параллельной программы. Это предел декомпозиции данных для вычислителя, дисбаланс нагрузки на вычислители и другие особенности, связанные с алгоритмическим устройством программы, приводящие к уменьшению масштабируемости.

3. Описание проведённых экспериментов

Для выявления факторов, влияющих на масштабируемость приложений [8], и исследования данных системного мониторинга [9-11] использованы как реальные приложения пользователей суперкомпьютерного комплекса, работающие в общей очереди задач, так и специально написанные для этого тесты.

Для моделирования факторов, связанных с работой коммуникационной сети, использована система специально сконструированных тестов. Данные тесты не производят никаких вычислений, а занимаются только отправкой и приёмом сообщений. Это необходимо, чтобы максимально увеличить нагрузку на коммуникационную сеть и минимизировать влияние вычислений на общую картину, получаемую на основе данных мониторинга.

Система тестов коммуникационной сети состоит из трёх типов приложений, которые пересылают сообщения между участвующими процессами по разной схеме.

Тест «master» работает следующим образом: один главный процесс рассылает сообщения всем остальным процессам приложения. Параметрами запуска регулируются длина сообщения и число циклов рассылки. В данном тесте основная нагрузка приходится на рассылающий процесс, и поэтому важны свойства коммуникационной сети. При высокой загрузке сообщениями малой длины ограничивающим масштабируемость фактором станет латентность сети.

Тест «snake» работает следующим образом: на первом шаге все процессы с чётным номером пересылают сообщение своим соседям с номером на единицу бóльшим. На втором шаге процессы с нечётным номером пересылают сообщение соседям с номером, также бóльшим на единицу. Последний процесс пересылает сообщения нулевому. В этом тесте бóльшая часть процессов обменивается с соседними по номеру процессами. При запуске по несколько процессов на вычислительный узел значительная часть пересылок будет приходиться на пересылки внутри узла, поэтому коммуникационная сеть будет задействована в меньшей степени. Физическое размещение процессов по вычислительным узлам может зависеть от настроек системы очередей конкретного суперкомпьютера, но, как правило, при плотном размещении процессов по узлам (на каждое ядро узла по одному процессу) на один узел попадают соседние процессы.

Тест «butterfly» работает следующим образом: каждый процесс с номером меньше половины общего числа процессов отсылает сообщение процессу с симметричным номером с конца (нулевой процесс посылает последнему, второй — предпоследнему и т.д.) Таким образом, при обычном размещении процессов по вычислительным узлам бóльшая часть сообщений направлена не на соседние процессы, а на процессы, находящиеся физически на разных вычислительных узлах. Поэтому большая часть сообщений проходит не внутри узла, а по коммуникационной сети.

Этот набор тестов позволяет оценить пригодность данных мониторинга для определения основных свойств коммуникационной сети, влияющих на масштабируемость параллельного приложения. Для выявления факторов, связанных с эффективным использованием компонентов вычислительного узла суперкомпьютера, анализировались также приложения, по-разному использующие иерархию памяти, в том числе такие, как реализация блочного перемножения матриц и тест Linpack [12].

Для получения данных системного мониторинга использовались отчёты о работе приложения JobDigest. Данные отчёта анализировались на предмет нахождения корреляций между различными графиками системных датчиков. Анализировались как численные показатели датчиков, так и динамика изменения их значений.

Для анализа зависимостей между различными показателями данные JobDigest преобразовывались в представление данных для анализатора WEKA. Для этого был разработан скрипт, преобразующий несколько временных рядов по различным датчикам в один временной ряд, объединяющий все показания всех датчиков мониторинга в вектор.

В WEKA данные атрибутов анализировались на предмет характерных зависимостей между показаниями различных датчиков в некоторый момент времени. Для этого рассматривались диаграммы зависимостей между показаниями различных датчиков.

Также для собранных данных состояния всех датчиков применялся алгоритм кластеризации. С его помощью определялось, какую часть времени работы программы она провела в некотором состоянии.

4. Результаты и анализ проведённых экспериментов

Рассмотрим результаты некоторых проведённых экспериментов с целью выявления факторов, влияющих на масштабируемость приложений.

Пропускная способность и топология коммуникационной сети

В системах с распределённой памятью пропускная способность коммуникационной сети может стать ограничивающим фактором масштабируемости приложения в случае, если процессы обмениваются между собой очень большими объёмами данных. Тогда пропускная способность будет определять, какое время будет расходоваться на пересылку каждого сообщения. Таким образом, чем больше процессов будут пересылать такие сообщения, тем выше будет нагрузка на коммутаторы, соединяющие вычислительные узлы.

Рассмотрим запуск теста «butterfly» с параметрами запуска соответствующими сообщениям различной длины (1, 1 000 000, 100 000 000).

Рис. 1. Скорость передачи данных по сети Infiniband (длина 1), тест «butterfly»

Рис. 2. Скорость передачи данных по сети Infiniband (длина 1 000 000), тест «butterfly»

Рис. 3. Скорость передачи данных по сети Infiniband (длина 1 000 000), тест «snake»

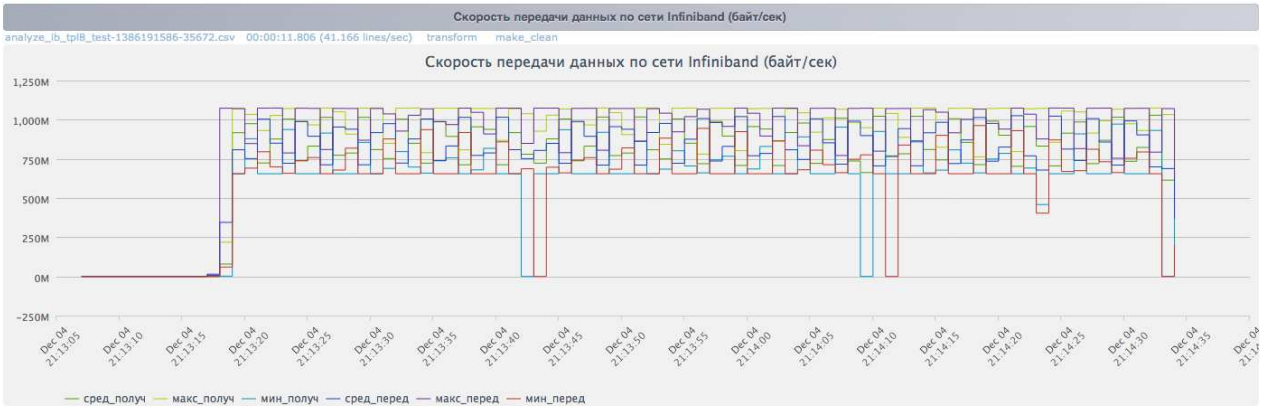

Рис. 4. Скорость передачи данных по сети Infiniband (длина 100 000 000), тест «butterfly»

На графиках скорости передачи данных по сети Infiniband (рис. 1) видна низкая интенсивность использования сети со скоростью около 100 МБ/сек при длине сообщения 1. При увеличении длины сообщения до 1 000 000 (рис. 2) виден рост интенсивности использования сети до 900 МБ/сек. Разница между максимальным и минимальным значением скорости становится более существенной. Средние значения скорости передачи и получения данных попеременно находятся возле максимального и минимального значения. Такое поведение может говорить о том, что проявляется предел пропускной способности отдельных узлов, что вызывает задержку во времени передачи и приёма большого числа сообщений большой длины. И это подтверждается экспериментами с дальнейшим увеличением длины сообщения.

В третьем эксперименте с тестом «butterfly» длина сообщения увеличена до 100 000 000 (рис. 4). Максимальное значение скорости передачи и приёма данных находится на уровне 1100 МБ/сек, что превышает значение в предыдущем эксперименте (900 МБ/сек). Однако минимальное значение скорости передачи и приёма данных при очень длинных сообщениях находится на уровне 650 МБ/сек. Это значительно меньше значения в предыдущем эксперименте (850 МБ/сек), которое ненамного меньше максимального значения. Схема пересылок осталась прежней, но разрыв между максимальным и минимальным значением скорости передачи и приёма данных по сети сильно увеличился. Минимальное значение скорости передачи данных стало меньше, чем в эксперименте с меньшим размером сообщения. Сильно возросли отклонения значений средней величины скорости передачи и приёма данных, которые имеют ступенчатую структуру.

Структура графика скорости передачи данных по сети Infiniband указывает на то, что предел пропускной способности сети передачи данных определяет время выполнения операции обмена сообщениями.

Если преобразовать данные JobDigest в представление анализатора WEKA и рассмотреть зависимости атрибутов минимальной, максимальной и средней скорости передачи сообщений от соответствующих значений скорости приёма для сообщений длины 1 000 000 и сообщений длины 100 000 000, то станет понятно, что предел скорости передачи действительно достигнут.

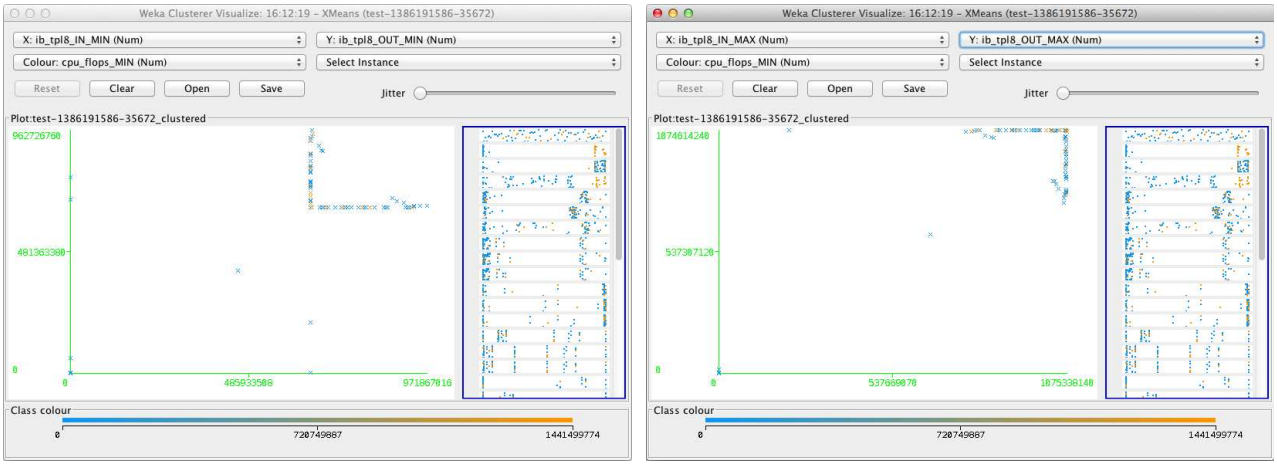

Рис 5. Зависимость атрибутов минимальной (слева) и максимальной (справа) скорости входящей передачи данных от скорости получения данных по сети Infiniband для теста «butterfly» с сообщениями длиной 100 000 000

По графикам на рисунке 5 можно сделать вывод, что максимальная и минимальная скорости передачи и приёма данных при использовании очень длинных сообщений ограничены соответственно сверху и снизу константным значением, а изменения максимального и минимального значения как приёма, так и передачи сообщений не влияют друг на друга. Однако среднее значение демонстрирует обратную зависимость между скоростью приёма и скоростью передачи данных, которая в сумме остается константной (рис. 6). Это говорит о том, что при использовании очень длинных сообщений предел скорости передачи данных достигнут, что влияет на масштабируемость такого приложения.

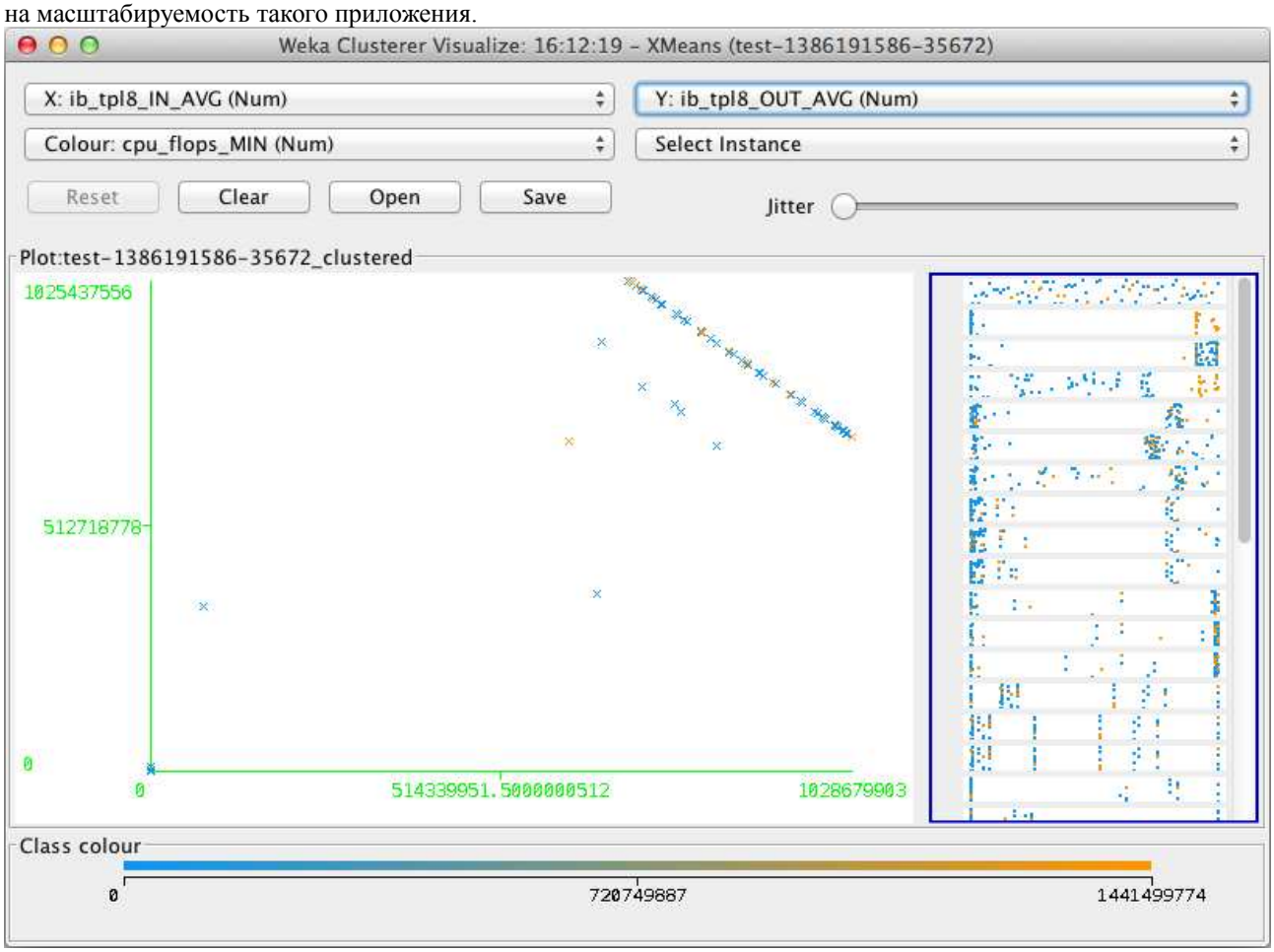

Рис 6. Зависимость атрибутов средней скорости входящей передачи данных от скорости получения данных по сети Infiniband для теста «butterfly» с сообщениями длиной 100 000 000

Данный вывод подтверждает также то, что при рассмотрении той же зависимости для атрибутов того же теста с использованием сообщений длиной 1 000 000 характер зависимости сильно отличается. Средняя

скорость передачи данных почти не отличается от максимальной и минимальной, и меняется в малых пределах, но пропорционально: с увеличением скорости приёма увеличивается и скорость передачи данных (рис. 7).

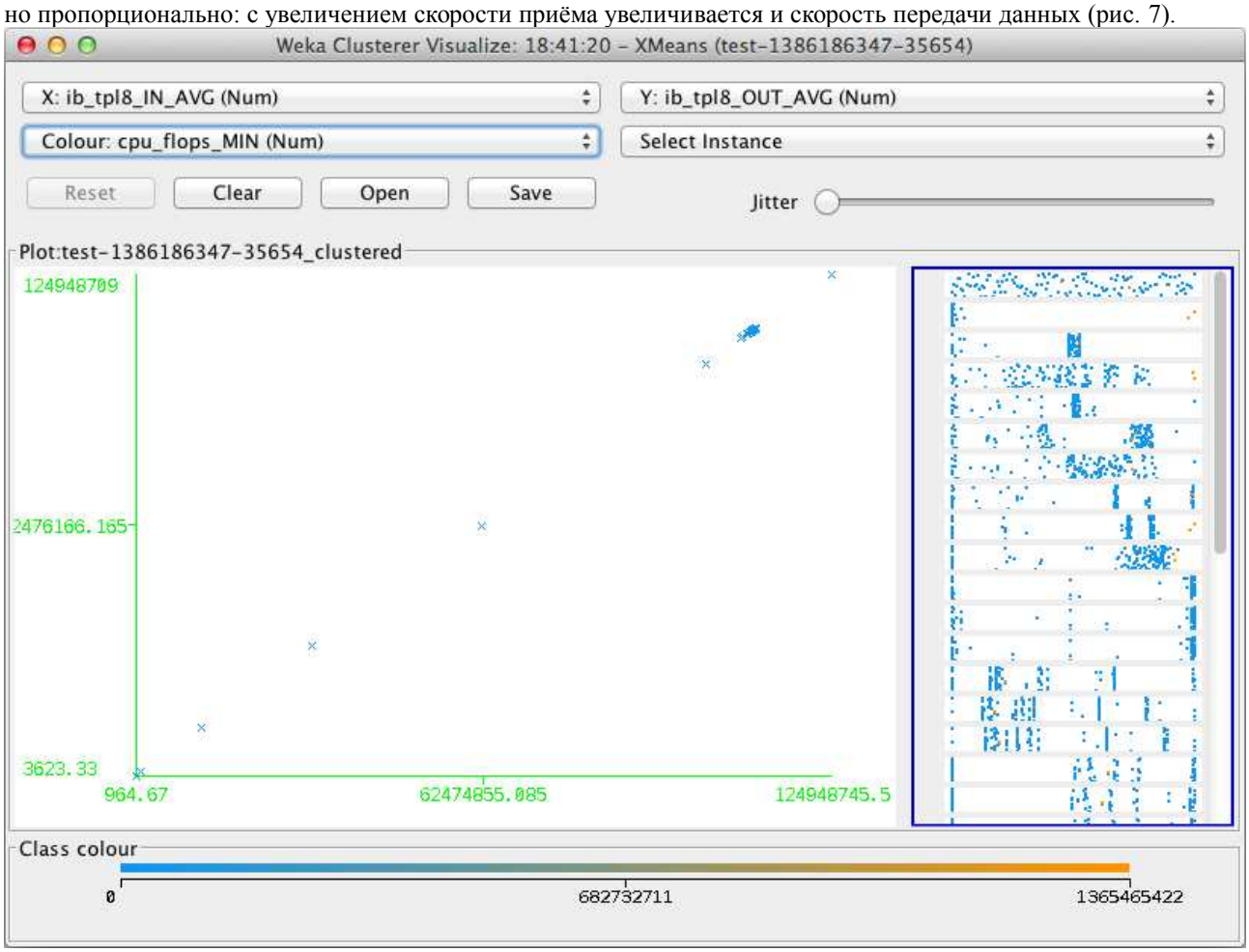

Рис. 7 Зависимость атрибутов средней скорости входящей передачи данных от скорости получения данных по сети Infiniband для теста «butterfly» с сообщениями длиной 1 000 000

Если же рассмотреть различие между запуском тестов «butterfly» и «snake» (рис. 2, 3), то проявляется влияние топологии расположения процессов. В этих двух экспериментах общее число одновременно пересылаемых сообщений и их длина одинаковы, но интенсивность использования сети отличается почти в 4 раза (около 950 МБ/сек для теста «butterfly» и около 230 МБ/сек для теста «snake» при длине сообщения 1 000 000).

Большинство сообщений в тесте snake было доставлено через внутренние каналы связи, так как соседние процессы расположены физически рядом. Поэтому время на передачу данных в тесте snake оказалось значительно меньшим, чем при передаче тех же данных, но по схеме butterfly. Действительно, при такой схеме передачи данных через внешние каналы связи по коммуникационной сети Infiniband будут обмениваться только два процесса на вычислительном узле, а именно, те процессы, которые физически расположены на разных узлах с отправителем и получателем. В случае же теста «butterfly» (рис. 2) все процессы на узле будут использовать сеть Infiniband, так как все их получатели и отправители будут являться удалёнными процессами. Исключением может быть один вычислительный узел, на котором расположен процесс с номером, равным половине количества всех процессов. Важно, что в первом и втором случае объём переданных данных оставался одинаковым, потому что в обоих тестах было передано одинаковое число сообщений одинаковой длины, отличалась только схема их пересылок, которая и оказала сильное влияние на время выполнения тестов.

Таким образом, на графиках JobDigest можно увидеть ограничения, накладываемые на масштабируемость приложения, связанные с характеристиками и топологией коммуникационной сети и схемы пересылок данных в приложении.

Использование при работе жёсткого диска

Поскольку время доступа к жёсткому диску сравнительно велико, его интенсивное использование может стать фактором, сдерживающим масштабируемость приложения. Задержка времени обращения к диску может быть как задержкой времени только на обращение к ячейке диска, если приложение выполняется на вычислительных узлах с локальными дисками, или же может состоять из суммы задержек сети, по которой подключены удаленные дисковые ресурсы, и задержек непосредственно при использовании диска. Наиболее

сильно это будет проявляться, если процессы используют частые обмены с дисками очень небольшими порциями данных. При этом время на передачу данных может значительно превышать время их записи на диск.

Рассмотрим примеры графиков системы JobDigest для работы теста Linpack. В тесте был определён такой размер данных, что они не должны помещаться в оперативную память вычислительных узлов. Тест был запущен в разделе, содержащем локальные диски, поэтому задача должна активно использовать дисковую память и своп.

Рис.8. Количество операций с плавающей точкой в секунду, тест Linpack

Рис.9. Скорость обмена данными с диском, тест Linpack

Рис 10. Интенсивность работы со своп-памятью, тест Linpack

На графике количества операций с плавающей точкой (рис. 8) видна общая низкая производительность приложения. Только на нескольких участках производительность становится относительно высокой. Это подтверждает предположение о том, что общая эффективность работы теста должна быть низкой.

На графике скорости обмена данными с диском (рис. 9) видна активность на протяжении всей работы теста. Такая дисковая активность может свидетельствовать либо о том, что используется запись в файл, либо о том, что данные подкачиваются из своп-памяти.

График интенсивности работы со своп-памятью (рис. 10) подтверждает предположение о том, что на протяжении всей работы приложения используется своп-память.

Если преобразовать данные JobDigest в представление анализатора WEKA и рассмотреть зависимости, то становится понятно, что производительность работы приложения тесно связана с активностью использования своп-памяти.

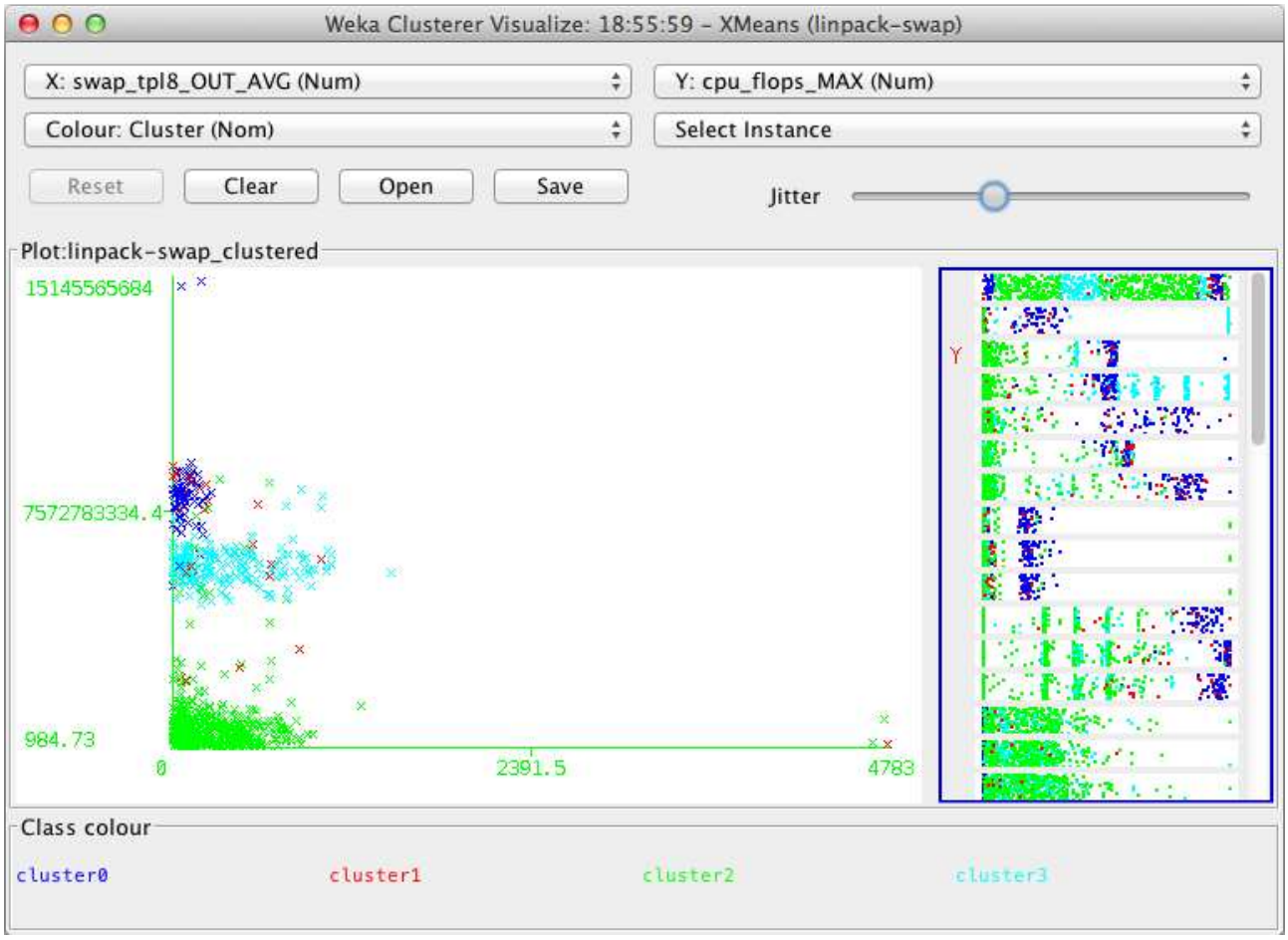

Рис 11. Зависимость активности своп-памяти в приложении (ось X) от производительности (ось Y). Цветами обозначены выделенные классы состояний работы приложения

На графике зависимости использования своп-памяти от производительности (рис. 11) показан результат работы алгоритма кластеризации состояний вычислительной системы. На нём видно, что большую часть времени (67% - cluster2) приложение имело низкую производительность и ненулевую активность своп-памяти, значительно меньшую часть времени (21% - cluster3), приложение имело среднюю производительность при активном обмене с диском, и самую высокую производительность имело только в 9% (cluster0) состояний при отсутствии обменов с диском.

Таким образом, на графиках JobDigest можно увидеть ограничения, накладываемые на масштабируемость приложения пределом объёма оперативной памяти и последующим использованием дисковой памяти для вычислений.

Предел декомпозиции данных

Рассмотрим графики количества операций с плавающей точкой двух запусков одного и того же приложения на одном и том же числе процессоров, но с разным размером задачи.

Рассмотрим пример двух запусков перемножения матриц размера 1024x1024 и 512x512.

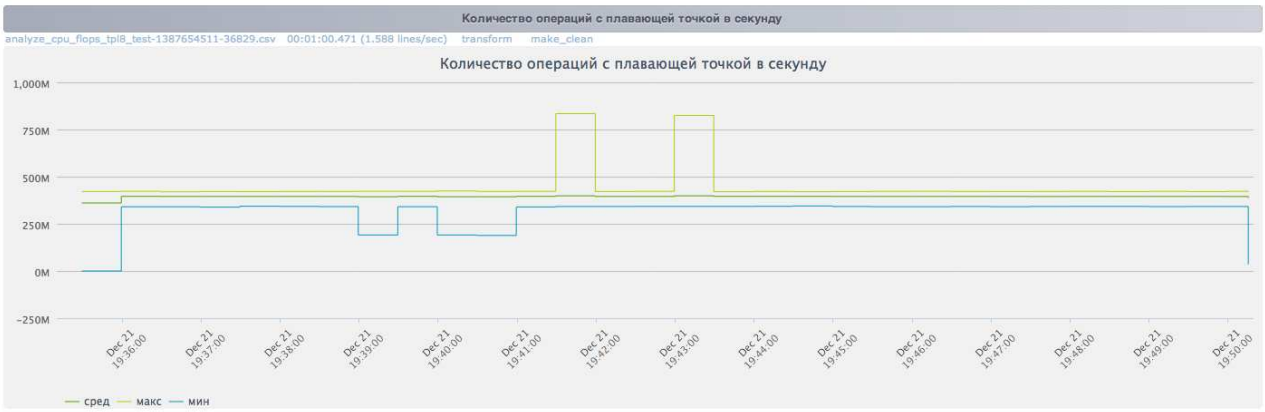

Рис. 12. Количество операций с плавающей точкой в секунду, перемножение матриц (размер 1024x1024)

Рис. 13. Количество операций с плавающей точкой в секунду, перемножение матриц (размер 512x512)

При запуске перемножения матриц размера 1024x1024 среднее количество операций с плавающей точкой в секунду (рис. 12) принимает значения выше 420 Мфлопс.

При запуске того же приложения но с размером матрицы 512x512 (рис. 13) средняя и максимальная производительности не превышают значения 270 Мфлопс.

Если размер задачи не позволяет обеспечить большой объём вычислений на каждом процессе, то наступит момент, когда на организацию вычислений будет уходить время, сравнимое со временем, затраченным на сами вычисления. Таким образом, затраты на организацию работы параллельного приложения станут сравнимы с затратами на выполняемую полезную работу. Это является условием, ограничивающим масштабируемость приложения.

При уменьшении размера задачи ожидаемо было бы увеличение производительности, потому что данных меньше, и доступ к ним должен осуществляться быстрее, если они укладываются в кэш-память. Но если данных уже слишком мало, то производительность будет падать с уменьшением объёма вычислений.

В данном случае размер матрицы слишком мал, поэтому производительность значительно уменьшилась.

Таким образом, на графиках JobDigest можно увидеть достижение предела декомпозиции данных, ограничивающего масштабируемость приложения.

5. Заключение и выводы

Рассмотренные результаты экспериментов подтверждают, что в данных системного мониторинга могут проявляться различные факторы, влияющие на масштабируемость параллельных программ. К ним относятся факторы, относящиеся к влиянию программно-аппаратной среды, свойств алгоритма и другие. Представленные результаты экспериментов демонстрируют некоторые яркие проявления факторов, ограничивающих масштабируемость приложений. Данные в представлении Job Digest, дополненные данными зависимости значений одних атрибутов от других, полученными в WEKA, позволяют сделать выводы о степени влияния различных факторов на масштабируемость как синтетических тестов, так и реальных приложений.

Работа выполняется при поддержке гранта РФФИ 13-07-00790 а.

ЛИТЕРАТУРА:

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб: БХВ-Петербург, 2002.-608 с.
2. Ananth Grama, Anshul Gupta, George Karypis, Vipin Kumar. Introduction to Parallel Computing. (2nd Edition), Pearson, 2003.
3. В.П. Гергель. Высокопроизводительные вычисления для многоядерных многопроцессорных систем. Учебное пособие – Нижний Новгород; Изд-во ННГУ им. Н.И.Лобачевского, 2010.
4. Вл.В. Воеводин, С.А. Жуматий, С.И. Соболев, А.С. Антонов, П.И. Брызгалов, Д.А. Никитенко, К.С. Стефанов, В.В. Воеводин. Практика суперкомпьютера "Ломоносов"// Открытые системы, N 7, 2012. С. 36-39.
5. А.С. Антонов. СКИФ МГУ – основа Суперкомпьютерного комплекса Московского университета // Вторая Международная научная конференция "Суперкомпьютерные системы и их применение" (SSA'2008): доклады конференции (27-29 октября 2008 года, Минск).- Минск: ОИПИ НАН Беларуси, 2008. С. 7-10.
6. А.В. Адинец, П.И. Брызгалов, В.В. Воеводин и др. Jobdigest – подход к исследованию динамических свойств задач на суперкомпьютерных системах // Вестник Уфимского государственного авиационного технического университета. — 2013. — Т. 17, № 2 (55). — С. 131–137.
7. Weka 3: Data Mining Software in Java: http://www.cs.waikato.ac.nz/ml/weka/
8. А.С. Антонов, А.М. Теплов. Исследование масштабируемости программ с использованием инструментов анализа параллельных приложений на примере модели атмосферы NH3D // Вестник Южно-Уральского государственного университета. Серия "Вычислительная математика и информатика", Т.2, N 1, 2013. С. 5-16.
9. А.С. Антонов, С.А. Жуматий, Д.А. Никитенко, К.С. Стефанов, А.М. Теплов, П.А. Швец. Исследование динамических характеристик потока задач суперкомпьютерной системы // Вычислительные методы и программирование: Новые вычислительные технологии (Электронный научный журнал). Том 14, раздел 2, 2013. С. 104-108.
10. Д.А. Никитенко, К.С. Стефанов. Исследование эффективности параллельных программ по данным мониторинга // Вычислительные методы и программирование: Новые вычислительные технологии (Электронный научный журнал), том 13, № 1. С. 97-102.
11. Д.А. Никитенко. Комплексный анализ производительности суперкомпьютерных систем, основанный на данных системного мониторинга // Вычислительные методы и программирование: Новые вычислительные технологии (Электронный научный журнал), том 15, с. 85-97.
12. Dongarra J. J., Bunch J. R., Moler G. B., Stewart G. W. LINPACK Users' Guide. — Society for Industrial and Applied Mathematics, 1979—1993.