ИССЛЕДОВАНИЕ МЕТОДОВ ТЕСТИРОВАНИЯ КОММУНИКАЦИОННОЙ СРЕДЫ МНОГОПРОЦЕССОРНЫХ СИСТЕМ

А.Н. Сальников, Д.Ю. Андреев

В современном мире одним из приоритетных направлений в области информационных технологий является развитие архитектуры многопроцессорных систем. Но кроме достижения результатов в этой области требует также и развитие средств для создания эффективных алгоритмов для систем с большим числом параллельно работающих вычислительных устройств.

Важным является вопрос о эффективной передаче данных между процессорами, в условиях неоднородности коммуникационной среды многопроцессорной системы. Недостаточно просто разделить передачи данных во времени, разные процессоры могут взаимодействовать с разной скоростью. Поэтому тестирование характеристик коммуникационной среды является актуальной задачей.

Наиболее популярной технологией для многопроцессорных систем на данный момент является Message Passing Interface (MPI). На данный момент существует несколько программных средств тестирования пропускной способности коммуникационной среды на основе MPI, но анализ результатов тестирования не всегда является простой задачей.

В данной работе предложена программная реализация системы для тестирования пропускной способности коммуникационной среды многопроцессорных систем с возможностью графического воспроизведения результатов тестирования.

С технологией программирования MPI основным способом взаимодействия процессоров является передача сообщений. На время передачи сообщения влияют множество факторов: физическое расположение процессоров в системе, длинна сообщения, тип коммуникационной среды, загруженность коммуникационной среды.

На данный момент существует несколько решений вопроса нахождения времени доставки MPI-сообщения. Они делятся на два типа:

Трассировка и профилирование — сбор информации о работе MPI функций в процессе работы существующей параллельной программы.

Синтетические тесты на основе посылки MPI-сообщений различной длинны.

Трассировка и профилирование помогают обнаружить MPI-вызовы, на которые приходится наибольшие временные затраты. Это способствует оптимизации существующего параллельного алгоритма.

Рассмотрим наиболее популярные на данный момент системы тестов MPI, основанные на профилировании:

Visualisation and Analysis of MPI Resources (VAMPIR)[1]
Parallel Visualization and Events Representation (ParaVER)[2]
Tuning and Analysis Utilities (TAU)[3]
IBM High Performance Computing Toolkit[6]

Трассировка существующей программы является хорошим средством для ручной оптимизации программы, но не даёт ответы на многие вопросы о работе самой вычислительной системы.

При увеличении сложности профилируемой программы количество данных результата теста резко возрастает, что может создать дополнительные сложности для анализа и хранения данных.

Синтетические тесты MPI нацелены на получение информации о времени работы отдельных MPI функций в системе.

Вот примеры существующих синтетических MPI тестов:
MPI-bench-suit (в частности mpitest)
Intel MPI Benchmarks[4]
Special Karlsruher MPI Benchmark (SkaMPI)[5]

Данные тесты собирают статистические данные о времени работы различных типов посылки MPI-сообщений.

Вот некоторые типы предлагаемых тестов:
Pingpong — тест посылка обычного сообщения.
Sendrecv — тест двухсторонняя посылка и приём сообщений.
Bcast — тест посылка широковещательного сообщения.
Allgather — тест функции получения сообщения одним процессором со всех других процессоров в группе.
Alltoall — тест посылки сообщений от всех процессоров ко всем процессорам.

Reduce — тест функции передачи вектора данных на различные процессоры

Allreduce — тест функции передачи данных от процессора на все другие процессоры одновременно для всех процессоров .

Barrier — тест функция синхронизации.

Один и тот же тест производится один или несколько раз, после чего в результате получаем максимальное, минимальное и среднее значение. При этом практически нет возможности узнать влияние работы частей вычислительной системы не участвующих в тестах на результаты тестирования.

Большинство из этих систем тестов предоставляют результат для разных длин сообщений. Файл результат содержит текстовую информацию о тесте и числовые данные результатов тестирования.

Анализ подобных данных сильно усложняется при увеличении числа процессоров, матрицы размера 100x100 с числами с плавающей точкой уже практически невозможно анализировать вручную. Но ни одна из данных систем не предоставляет своего удобного средства графического представления полученных данных.

Авторами разработано программное средство для тестирования коммуникационной network_test. Тесты основаны на использовании MPI стандарта версии 1.2. Данная утилита входит в состав системы автоматизированного распараллеливания программ PARUS. Тесты предоставляют возможность имитировать использование коммуникационной среды процессорами не задействованными в подсчёте результатов тестирования. Так же система PARUS имеет средство графической визуализации результатов тестирования network_viewer.

Приложение network_test написано на языке C++ с использованием библиотеки MPI, оно поддерживает шесть режимов тестирования коммуникаций:

one_to_one — в этом режиме одновременно сообщениями обменивается только пара MPI-процессов. Для обмена используются функции MPI_Send и MPI_Recv. В матрицу записывается время выполнения функции MPI_Recv.

all_to_all — в этом режиме все MPI-процессы вовлечены в обмен сообщениями. Сообщения передаются с помощью функций MPI_Isend и MPI_Irecv, а в матрицу записывается промежуток времени между инициализацией MPI_Irecv и окончанием обмена с MPI_Waitany.

send_recv_and_recv_send — режим в котором пара MPI-процессов обменивается сообщениями в обе стороны.

async_one_to_one — тоже самое, что режим one_to_one, но используются неблокирующие вызовы функций посылки сообщений.

noise_blocking — MPI-процессы разделяются на три группы: пара «целевых» процессов, множество «шумящих» процессов и множество «молчащих» процессов. Сперва выбирается пара целевых процессов, остальные помечаются как молчащие, затем из молчащих процессов случайным образом выбирается определённое число шумящих процессов. Число шумящих процессов определяется пользователем в параметрах теста. Целевые процессы производят передачу данных также, как и для теста one_to_one. При этом, время их передачи заносится в матрицу. Шумовые процессы обмениваются сообщениями так же как и в all_to_all режиме, но задержки для них никуда не сохраняются. Размер и количество шумовых сообщений определяются в параметрах теста.

noise — аналогично noise_blocking, но используется неблокирующие вызовы функций посылки сообщений.

Начальный вариант реализаций перечисленных видов тестов был представлен в работе [7].

Пользователем указывается несколько параметров: интервал длин сообщений, шаг приращения длины сообщения и число повторов на каждом шаге по длине сообщения. Тест начинает свою работу с самой маленькой длины сообщения в интервале, длина сообщения увеличивается на значение шага на каждой итерации до тех пор, пока не будет достигнута верхняя граница интервала. По достижении верхней границы тестирование прекращается. Для каждой длины сообщения тест производит некоторое количество передач через коммуникационную среду. Тип и объём передач зависит от выбранного пользователем режима, который указывается в параметрах командной строки вместе со всеми остальными параметрами. Параметр «число повторов» задаёт число независящих друг от друга итераций для фиксированной длины сообщения, в результате чего получается некоторая выборка. Выборка в дальнейшем используется для поиска минимального значения, медианы, подсчёта среднего значения и стандартного отклонения. Найденные значения формируют множество матриц, где каждая матрица отвечает своей длине сообщения. В позиции i,j матрицы содержится задержка при передаче сообщения от MPI-процесса с номером i к MPI-процессу с номером j. Задержка оценивается при помощи функции MPI_Wtime. Данные для минимального значения, медианы, среднего значения и квадратичного отклонения записываются в отдельные текстовые файлы.

Приложение network_viewer написано на языке Java. Утилита обладает графическим интерфейсом. Данных результатов теста можно увидеть в виде двухмерного графического изображения любой из матриц. Графическое изображение строится в виде области с клетками цвета градаций серого для каждого значения матрицы (Рис 1). При этом цвет выбираются в зависимости от типа нормализации. При глобальной нормализации чёрный

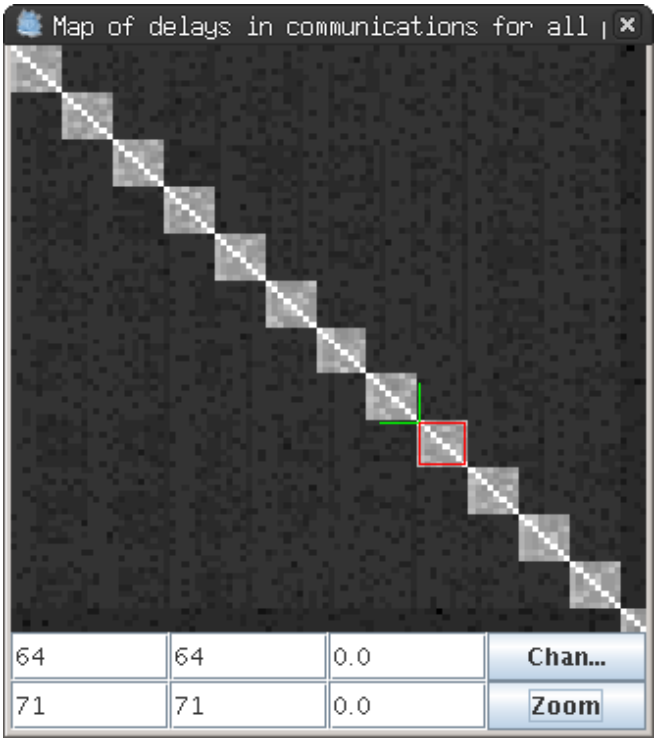

Рисунок 1: Изображение матрицы результатов теста one_to_one для 100 процессоров

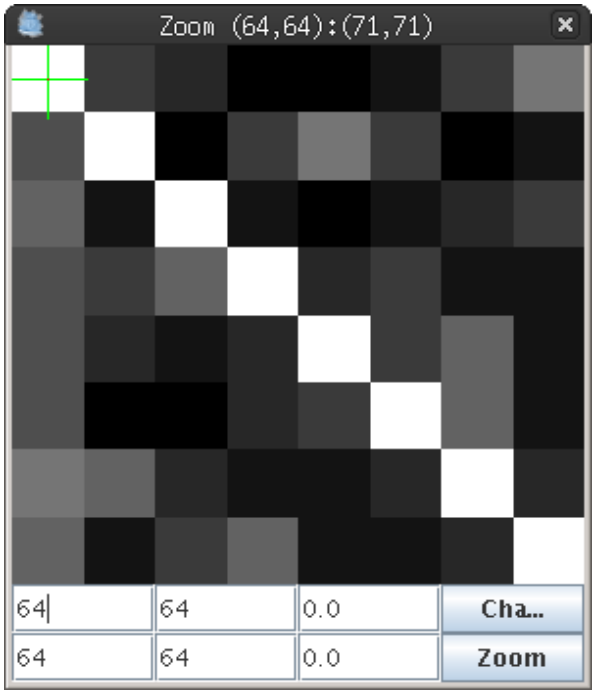

Рисунок 2: Изображение части матрицы

цвет берётся для максимального значения всех матриц, а белый — для минимального, при локальной нормализации – чёрный и белый цвета берутся для максимальных и минимальных значений одной матрицы соответственно.

Даётся возможность выбора любого процессора для просмотра числового значения его результата. И возможность выбора части матрицы для просмотра значений с локальной нормализацией в пределах этой части (Рис 2).

С помощью описанных приложений были получены данные для машин: mvs100k (кластер с 470x4 процессорами Intel Xeon 5160 связанных Infiniband сетью) и IBM pSeries 690 (SMP система с 16 процессорами).Исходный код доступен со страницы проекта PARUS (http://parus.sf.net).

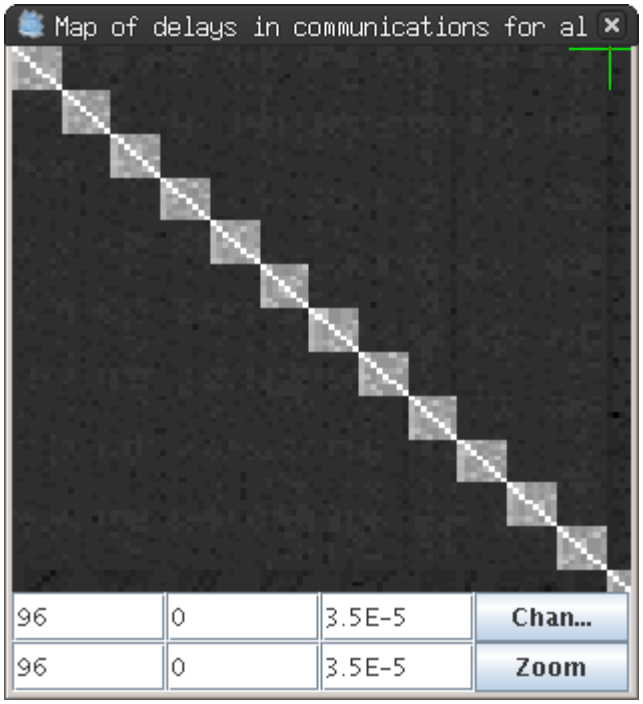

Рисунок 3: Среднее значение

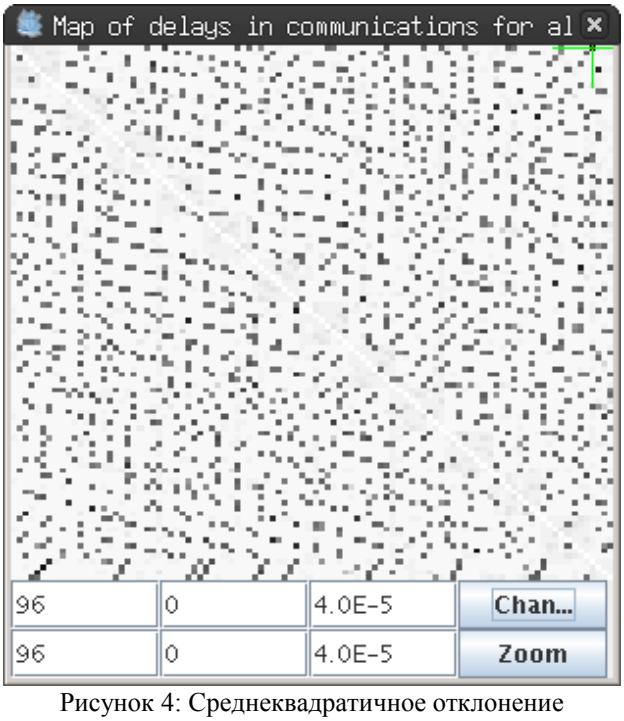

Рисунок 4: Среднеквадратичное отклонение

На 100 процессорах Intel Xeon 5160 комплекса mvs100k при длине сообщения в 15000 байт по изображению результатов тестов (Рис 3) можно делать выводы о архитектурных особенностях системы. При этом значения среднеквадратичного отклонения не были подчинены какой-либо закономерности (Рис 4).

Результаты для IBM pSeries 690 показали менее стабильные результаты. При длине сообщения 10000 байт, средние значения времени доставки сообщения различались иногда более чем в 10 раз.

Работа проводится при поддержке грантов: МК-1606.2008.9 и РФФИ 08-07-00445.

ЛИТЕРАТУРА:

1. W.E. Nagel, A. Arnold, M. Weber, H.-C. Hoppe, K. Solchenbach VAMPIR: "Visualization and analysis of MPI resources" Supercomputer,12(1):69-80, January 1996.
2. V. Pillet "PARAVER: A Tool to visualize and Analyze Parallel Code" Universitat Polit`ecnica de Catalunya. 1995.
3. S. Shende and A. D. Malony "TAU: The TAU Parallel Performance System" International Journal of High Performance Computing Applications, Volume 20 Number 2 Summer 2006. Pages 287-331.
4. Intel MPI Benchmarks : Users Guide and Methodology Description.
5. R.H. Reussner "SKaMPI: The Special Karlsruher MPI-Benchmark User Manual" Technical Report 02/99, Department of Informatics, University of Karlsruhe, Am Fasanengarten 5, D-76128 Karlsruhe, Germany, 1999.
6. T.J. Watson "MPI Performance Analysis Tools on Blue Gene" IBM Research Center. 2007
7. Н.М. Булочникова, В.Ю. Горицкая, А.Н. Сальников"Методы тестирования производительности сети с точки зрения организации вычислений" труды Всероссийской научной конференции "Научный сервис в сети Интернет 2004" Издательство Московского университета 2004г. ISBN 5-211-05007-X стр. 221-223.