СРЕДСТВО АВТОМАТИЧЕСКОЙ КЛАСТЕРИЗАЦИИ РЕЗУЛЬТАТОВ ТЕСТИРОВАНИЯ КОММУНИКАЦИЙ В МНОГОПРОЦЕССОРНЫХ СИСТЕМАХ

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 требуется 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 множество программных средств тестирования пропускной способности коммуникационной среды на основе MPI, они предоставляют данные в цифровом или графическом формате, и анализ результатов тестирования не всегда является простой задачей.

Для достаточно большого числа процессоров данные результатов теста обладают большим объёмом, что усложняет не только их анализ, но и хранение.

В данной работе создана программная система, позволяющая автоматизированно выделять группы пар процессоров, со схожей скоростью передачи данных. Система предназначена для кластеризации результатов исполнения программы network_test[2][3] на одной из многопроцессорных вычислительных систем. Программа network_test является средством тестирования производительности коммуникационной среды на основе посылки MPI-сообщений, и входит в состав системы Parus[4].

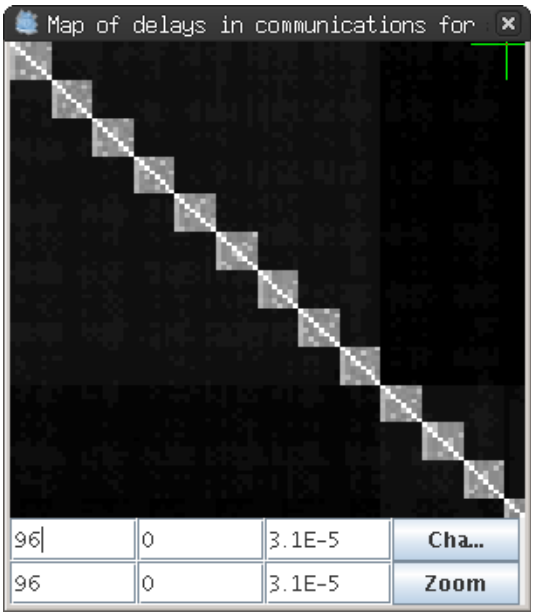

Рис. 1: Графическое представление результатов тестирования

В результате работы network_test создаётся несколько текстовых файлов, содержащих параметры теста и его результаты, как набор матриц вещественных чисел. Каждая матрица соответствует определённой длине MPI-сообщений. Достоинствами текстового представления является его переносимость между разными архитектурами вычислительных систем и возможность читать результаты без каких либо специальных программных средств, но текстовый формат объёмен и не позволяет позиционированного доступа к данным.

В работе [2] было показано, что довольно часто результаты тестирования обладают выраженной структурой(Рис.1). Было сделано предположение о соответствии данной структуры топологии коммуникационной среды.

Решая задачу кластеризации результатов тестирования мы можем автоматически выделить структуру топологии и тем самым значительно сократить объём хранимых данных.

Было принято решение о хранении результатов тестирования в переносимом бинарном формате NetCDF[5], для обеспечения позиционного доступа к данным. Были произведены соответствующие модификации программы network_test.

Текстовый формат однозначно отображается в новом NetCDF-формате. описание данного формата представлено далее (Рис.2).

```
netcdf info {

dimensions:
        x = <количество процессоров> ;
        y = <количество процессоров> ;
        n = UNLIMITED ;
variables:
        int proc_num ;              /*
        int test_type ;              * Информация о типе теста
        int data_type ;              * и его параметрах
        int begin_mes_length ;       */
        int end_mes_length ;
        int step_length ;
        int noise_mes_length ;
        int num_noise_mes ;
        int num_noise_proc ;
        int num_repeates ;
        double data(n, x, y) ;
//data – матрицы, содержащие результаты теста.
}
```

Рис. 2: Заголовок NetCDF формата данных

Данный формат позволяет получать доступ к каждой матрице по её номеру, и по координатам (i,j) к каждому элементу матрицы не прочитывая весь файл целиком до этого места.

Задача кластеризации состоит в выделении множеств пар процессоров, которые на некотором промежутке длин сообщений, для каждой длинны сообщения из промежутка, обладают схожими значениями скорости передачи.

Алгоритм кластеризации можно разбить на несколько стадий:

- Считывается первая матрица. Все элементы матрицы сортируются в порядке возрастания величины задержки при передаче сообщений. Первые n – минимальных значений выбираются и объединяются в первый диапазон значений. Предположительно это будут диагональные элементы матрицы, которые означают передачу данных самому себе. Здесь n — размерность матрицы (число процессоров в системе). Правая граница диапазона в полученном упорядоченном массиве двигается в сторону возрастания значения до тех пор, пока разница между максимальным и минимальным значениями в диапазоне не превысит заданного в параметрах алгоритма порога. Далее диапазон преобразуется во временный диапазон сдвинутый вправо на 1. Производится попытка расширить этот диапазон в правую сторону. Если попытка удаётся, то временный становится постоянным и заменяет собой исходный, в противном случае он выбрасывается из рассмотрения. Расширение производится также как и для начального интервала. Процесс сдвигов с расширением продолжается до тех пор, пока это возможно. Значения до диапазона формируют отдельные, независимые кластеры, а сам диапазон формирует один сплошной кластер.
- Процесс выделения кластеров продолжается после выделения первого диапазона. По аналогии с первым диапазоном следующий ищется точно также от элемента следующего за последним элементом первого диапазона. Процесс продолжается до исчерпания всех значений матрицы.
- Процесс выделения кластеров производится для нескольких следующих матриц. Число матриц задаётся параметром алгоритма. Ищется максимальное пересечение элементов каждого кластера первой матрицы со всеми элементами кластеров остальных матриц. $max\left(\mu\left(L_i\right)\right), L_i = C_{1,i} \cap C_{2,i_2} \cap \ldots \cap C_{m,i_m}$. Здесь $i_2, \ldots, i_m$ меняют своё значение от 1 до числа выделенных кластеров в каждой матрице. Для каждого фиксированного i производится перебор по всем $i_2, \ldots, i_m$.
- Для всех выделенных максимальных пересечений производится их расширение следующим образом. Пусть $C_1, C_2, \ldots, C_m$ набор кластеров в каждой матрице, которые образуют максимальное

пересечение, возьмём объединение H_i элементов составляющих все возможные пересечения из $m-1$ кластера.. Элементы матрицы образующие множество $L_i \cup H_i$ составляют предварительный результирующий кластер и не могут в дальнейшем входить в другие предварительные результирующие кластеры.

- Читается ещё некоторое количество матриц, в них по шагам 1 и 2 алгоритма выделяются кластеры. Полученная картина предварительных результирующих кластеров используется в качестве первой матрицы для шага 3, а сами прочитанные матрицы составляют матрицы для поиска максимального пересечения. При построении максимального расширения допускается отбрасывание не более чем заданный в параметрах процент элементов от размера предварительного результирующего кластера. Таким образом производится операция удлинения кластера. Если данная операция невозможна, то предварительный результирующий кластер становится окончательным кластером, в противном случае удлинение продолжается до тех пор, пока не будет получен окончательный кластер.
- Шаги с первого по пятый продолжаются с чтением очередной порции матриц из файлов. Алгоритм останавливается по исчерпании всего диапазона длин сообщений.

Данный алгоритм не даёт лучшие значения разделения на множества, но не требует больших ресурсов оперативной памяти благодаря тому, что большая часть результатов теста может продолжать храниться в файле.

Для хранения кластеризованных результатов тестирования был разработан ещё один формат хранения данных. Этот формат записи обеспечивает сжатие данных на основе выделения единственного представителя для всего кластера. При этом необходимо хранить информацию о соответствии пар процессоров определённому кластеру. Каждый кластер соответствует какому-то определённому промежутку длин сообщений.

```
netcdf info {
dimensions:
        x = <количество процессоров> ;
        y = <количество процессоров> ;
        n = UNLIMITED ;
variables:
        int proc_num ;              /*
        int test_type ;              * Информация о типе теста
        int data_type ;              * и его параметрах
        int begin_mes_length ;       */
        int end_mes_length ;
        int step_length ;
        int noise_mes_length ;
        int num_noise_mes ;
        int num_noise_proc ;
        int num_repeates ;
        int info(n, x, y) ;
        //info – матрица, содержащая номер позиции начала экземпляра
        //  кластера, к которому принадлежит пара процессоров (x,y).
        int length(n); // соотвествие матрицы n длинне сообщения.
}

netcdf data {
dimensions:
        n = UNLIMITED ;
        double data(n) ; // значения экземпляров, описывающих кластеры.
}
```

Рис. 3: Заголовок NetCDF-формата данных кластеризованных результатов тестирования.

В NetCDF записи результаты теста описываются двумя типами файлов:

Длинна одного кластера соответствует количеству шагов между двумя соседними матрицами в переменной info.

Для доступа к отдельному элементу(x,y) отдельной матрицы(n) требуется сначала определить номер(k) матрицы в info, соответствующей левой границе диапазона длин, к которому принадлежит матрица искомого элемента. В результате позиция искомого элемента вычисляется, как сумма позиции элемента (x,y) в матрице(k) из info и количестве шагов длин сообщений в тесте от левого края диапазона, до длины, соответствующей матрице(n).

Эффективность и точность предложенного алгоритма была исследована на наборе тестовых данных для некоторых многопроцессорных вычислительных систем. Данные для кластеризации были получены для комплекса mvs100k и СКИФ-МГУ «Чебышёв».

Тесты запускались с использованием 128, 256, 512 процессоров.

Были заданы следующие параметры:

--begin 1000 --end 10000 --step 500 --num_iterations 100

--begin 100 --end 2000 --step 100 --num_iterations 100

В качестве рассматриваемых результатов использовались результаты среднего значения и медианы.

Благодаря переходу к бинарному формату, для программы тестирования уменьшились задержки при выводе результатов, что значительно ускорило процесс тестирования.

Объём занимаемого файлового пространства в NetCDF представлении данных результатов теста уменьшился примерно на 20% по сравнении с текстовым представлением.

Ожидаемые результаты:

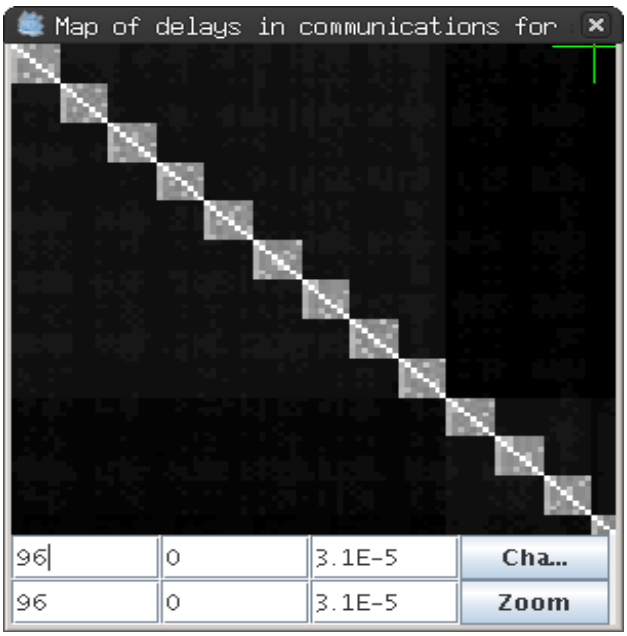

Рис. 4. Результаты тестов

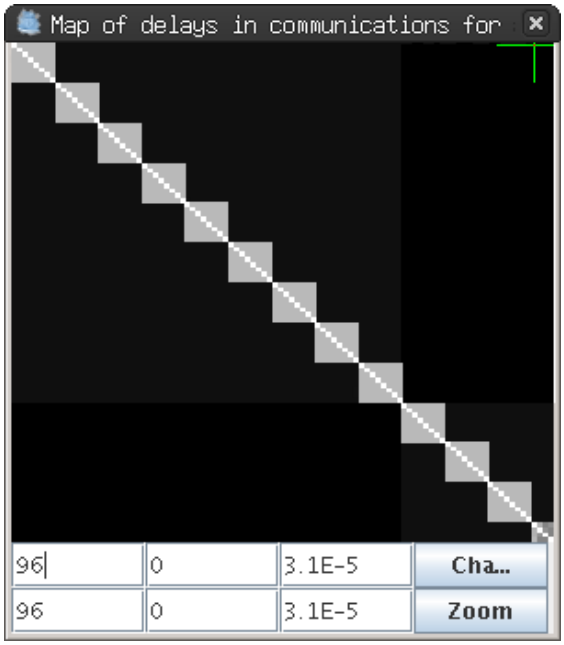

Рис. 5. Ожидаемый результат кластеризации

Тестирование алгоритма на полученных данных показало, что выделяется намного больше кластеров, чем ожидалось, но границы полученных кластеров в большинстве случаев близки к ожидаемым. На данный момент с помощью разработанного алгоритма нельзя получить разбиение, из которого можно автоматизированно выделить информацию о топологии коммуникационной среды, но полученные разбиения близки к желаемым.

В среднем для полученное количество кластеров для одного промежутка:

128 процессоров — 20.

256 процессоров — 27.

512 процессоров — 32.

Это позволило уменьшить объём файлов с данными теста более чем в 10 раз.

Работа проводится при поддержке грантов: МК-1606.2008.9 и РФФИ 08-07-00445, а также при поддержке студенческой исследовательской лаборатории Intel в МГУ.

ЛИТЕРАТУРА:

1. 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
2. А.Н. Сальников, Д.Ю. Андреев «Исследование методов тестирования коммуникационной среды многопроцессорных систем» // Научный сервис в сети Интернет: решение больших задач: Труды Всероссийской научной конференции, издательство МГУ имени М.В. Ломоносова, 2008г. Стр. 237-240
3. Alexey N. Salnikov and Dmitry Y. Andreev «An MPI-Based System for Testing Multiprocessor and Cluster Communications» Lecture Notes in Computer Science (LNCS5205) Recent Advantages in Parallel Virtual Machine and Message Passing Interface, Volume 5205, pp. 332-333, 2008, ISBN: 978-3-540-87474-4.

4. Alexey N. Salnikov «PARUS: A Parallel Programming Framework for Heterogeneous Multiprocessor Systems» Lecture Notes in Computer Science (LNCS 4192) Recent Advantages in Parallel Virtual Machine and Message Passing Interface, Volume 4192, pp. 408-409, 2006, ISBN-10: 3-540-39110-X ISBN-13: 978-3-540-39110-4.
5. NetCDF Tutorial http://www.unidata.ucar.edu/software/netcdf/docs/netcdf-tutorial.html