Метод уменьшения сложности графа задержек коммуникационной среды вычислительного кластера

Н.О. Жуков, А.Н. Сальников

МГУ имени М.В. Ломоносова

В данный момент уществует проблема отображения информации о задержках в коммуникационной среде. Отображение такой информации в виде ориентированного взвешенного графа даёт возможности для визуального анализа характеристик коммуникационной среды. Однако при таком подходе мы получаем граф огромного расмера, с очень большим числом рёбер.

Важным с практической точки зрения является то, что чем быстрее будет построен граф топологии суперкомпьютера, чем более наглядна будет визуализация, тем быстрее будет найдена ошибка или неполадка в работе вычислительного кластера. Так же будет проще использовать кластер с максимальной производительностью, так как будет проще подогнать программу под архитектуру конкретного вычислительного кластера.

В предыдущей работе [1] был предложен метод, когда граф задержек в коммуникационной среде отображается в трёхмерном пространстве следующим образом: ждой вершине ставится в соответствие точка с трёхмерными координатами, а ребро представляется в виде отрезка, соединяющего две соответствующие вершины (точки). Длины всех отрезков пропорциональны величене задержки при передаче. Отображение строится при помощи метода многомерного шкалирования. Предложенный метод интегрирован с программу *network_viewer2* [2].

Предполагаемый в данной работе метод осуществляет предварительное сжатие графа, Сжатие выполняется в несколько этапов. Вначале граф разбивается, используя специальные алгоритмы разбиения графов, затем подграф все вершины которого принадлежат одной группе сжимаются в вершину. Алгоритм параметризован параметром «степень сжатия», который задаёт допустимую разницу между расстояними внутри группы вершин и между группами. В интерфейсе программы *network_viewer2* для более детального просмотра графа реализуется возможность раскрытия вершины, соответствующей группе в которую был сжат подграф. К методу предъявляются следующие требования:

1. сохранение относительной длины дуг графа;
2. разбивается граф таким образом, что сумма весов ребер внутри подграфа меньше, чем между подграфом и другими ребрами графа;
3. вес ребра между двумя вершинами в сжатом графе есть среднее арифметическое всех весов ребер, соединяющих все вершины в данных подграфах.

Так же для выявления иерархической структуры графа реализуется метод сжатия, основанный на иерархических алгоритмах кластеризации.

Алгоритм работает на построении иерархических множеств. На каждом шаге производится разбиение графа на множество подграфов. Здесь строится массив, где элемент — это номер кластера в следующей иерархии. Процесс продолжается до тех пор, пока число вершин не достигнет указанного пользователем в параметрах алгоритма. В дальнейшем, в рамках процедуры «раскрытия вершины» в граф добавляются вершины для кластера из предыдущей иерархии.

Реализация алгоритмов выполнена на языках C, C++, графический пользовательский интерфейс написан с использованием кроссплатформенной библиотеки Qt. Визуализация графов происходит с помощью библиотеки OpenGL, функционал которой предоставлен Qt-модулем QtOpenGL.

Работа программы опробована на данных задержек полученых с суперкомпьютера «Ломоносов» для 1000 узлов, что соответствует четвёртой части всех узлов общедоступной очереди «regular4».

Программа временные характеристики работы представлены в таблицах 1 и 2.

Таблица 1: время работы программы в зависимости от числа рёбер

Число разбиений	число рёбер	время (мс.)
0(полный граф)	123064	1968254
100	4465	> 400
50	1225	> 100
10	45	> 10

Таблица 2: время работы алгоритма кластеризации в зависимости от соотношения разности расстояний внутри кластера и снаружи

разность %	число рёбер	время (мс.)
40	2211	29640
20	2080	27520
7	105	3230
3	3	10

Литература

1. Bannikov P.S, Salnikov A.N. Retrieving topology of interconnections in computational cluster based on results of MPI benchmarks // Moscow University Computational Mathematics and Cybernetics. – 2014. – Vol. 38, no. 2. – P. 73–82. DOI: 10.3103/S0278641914020022

2. Сайт проекта по тестированию вычислительных кластеров «ClustBench»: URL: `https://github.com/clustbench/network-tests2` (Дата обращения 15.06.2015)