ИСПОЛЬЗОВАНИЕ ВЫЧИСЛИТЕЛЬНОЙ МОДЕЛИ MAP-REDUCE ДЛЯ ОРГАНИЗАЦИИ РАБОТЫ ГРАФОВЫХ СТРУКТУР ДАННЫХ В РАСПРЕДЕЛЕННОЙ СРЕДЕ

А. Х. Перез Чернов

Многие прикладные проблемы, возникающие в области информационных технологий, могут быть сформулированы в терминах графов и соответствующих теоретико-графовых задач. Несмотря на возросшие вычислительные мощности персональных компьютеров, для решения некоторых задач по-прежнему требуются значительные вычислительные ресурсы. Настоящая работа предлагает подход к организации специальной структуры данных, которую можно использовать для хранения графа, а также применения к нему базовых операций (удаление вершины, взятие окружения вершины, проверка смежности двух вершин), используя при этом вычислительные ресурсы нескольких персональных компьютеров.

Будем рассматривать *простые* графы, т.е. конечные неориентированные графы без петель и кратных ребер. Под *большими* графами будем понимать графы, для хранения или анализа которых целесообразно использовать несколько компьютеров.

Для больших графов могут быть актуальны: задача увеличения скорости работы алгоритма путем декомпозиции исходного графа и решения индивидуальных подзадач независимо и параллельно на различных машинах; использование общих, объединенных ресурсов (например, памяти) для решения задачи, которая не может быть решена при ограниченных ресурсах.

СУЩЕСТВУЮЩИЕ ПОДХОДЫ К ОРГАНИЗАЦИИ СОВМЕСТНОГО ВЫЧИСЛИТЕЛЬНОГО ПРОСТРАНСТВА

Под *кластеризацией* понимают всякую организацию нескольких машин для совместного решения какой-либо задачи, а соответствующий набор машин называют *кластером*. Отдельно выделяют *статические* и *динамические кластеры*. В статических кластерах мы заранее определяем набор машин, которые будут совместно работать над решением задачи, причем до окончания работы состав машин не может быть изменен. Динамический кластер позволяет в процессе работы изменять состав машин, входящих в его состав.

В настоящее время для решения вычислительных задач с использованием совместной работы нескольких машин могут быть использованы следующие подходы.

1. *Кластеризация на уровне обмена сообщениями* предполагает, что для обмена данными между программами используется механизм сообщений, так что вместо индивидуальных вызовов различных функций, программы «подписываются» на получение и обработку сообщений соответствующего типа.

2. *Кластеризация на уровне мобильного кода* соответствует подходу «перенос вычислений ближе к данным», когда алгоритмы автоматически переносятся и исполняются на различных машинах.

3. *Кластеризация на уровне операционных систем* позволяет осуществлять «миграцию процессов», а именно, когда несколько машин, обычно под управлением одинаковых операционных систем, настраиваются таким образом, что процесс созданный на одной из машин может быть перенесен (мигрирован) для исполнения на другую машину.

4. *Кластеризация на уровне совместно используемых данных* часто соответствует подходу «перенос данных ближе к вычислениям» и предполагает наличие какого-либо сервиса, который распространяет или обеспечивает доступность данных на всех машинах, входящих в кластер.

5. *Кластеризация на уровне модели Map-Reduce* сконцентрирована на выполнение распределенных вычислений с помощью декомпозиции. Для того чтобы решить задачу с помощью этого подхода, исследователь должен предоставить a) быстрый алгоритм разделения общей задачи на набор подзадач, каждую из которых можно выполнять независимо и параллельно; b) алгоритм, решающий каждую из индивидуальных подзадач; c) алгоритм, составляющий общий ответ для исходной задачи из частных индивидуальных ответов. Сам процесс декомпозиции и решение индивидуальных подзадач называют *Map* этапом, так как исходной задаче сопоставляется набор подзадач и далее – набор ответов на индивидуальных подзадачах. Составление общего ответа из частных называется *Reduce* этапом, а вся описанная модель организации вычислений – *Map-Reduce* подходом [1].

Известны две практические реализации указанной модели – первая представлена компанией Google, вторая разработана при содействии специалистов из Yahoo. Ценность указанных реализаций заключается в том, что после того, как задача и алгоритмы представлены в соответствующей форме, достаточно легко осуществить практические вычисления на большом наборе персональных компьютеров. А именно, создаются многочисленные процессы (задания), ответственные за Map и Reduce фазы, и каждый из процессов (заданий) пересылается на доступную в текущий момент времени машину. Дополнительно контролируются возможные отказы в работе индивидуальных процессов, и соответствующие

решаемые подзадачи пересылаются для повторного вычисления другим узлам. Указанная кластеризация близка к кластеризации на уровне совместно используемых данных и кластеризации на уровне мобильного кода, и, тем не менее, накладывает жесткие ограничения на конкретный формат, состав и механизм координации индивидуальных алгоритмов, работающих в таком вычислительном пространстве.

ИСПОЛЬЗОВАНИЕ МОДЕЛИ MAP-REDUCE ДЛЯ ОБЕСПЕЧЕНИЯ ЭФФЕКТИВНОЙ РАБОТЫ СПЕЦИАЛЬНОЙ ГРАФОВОЙ СТРУКТУРЫ ДАННЫХ

В работе [2] была представлена специальная графовая структура данных *S*, обладающая преимуществами как списков смежности, так и матрицы смежности. Рассмотрим, какие именно операции и этапы работы с указанной структурой данных можно выполнить, используя распределенную вычислительную среду.

Известна следующая проблема организации систем распределенных вычислений: увеличение компьютеров в кластере может ухудшать как теоретические, так и практические показатели работы системы. Лишь некоторые системы допускают неограниченное *масштабирование,* т.е. увеличение входящих в кластер машин, сохраняя или увеличивая при этом вычислительную эффективность. Для многих систем существуют теоретические и практические оценки количества входящих машин в кластер, при которых последующее добавление новых машин нецелесообразно. Ограничениями могут выступать как, например, число созданных процессов, взаимоблокировки, или недостаточно эффективные механизмы распараллеливания программы и координации алгоритмов. Если $f(G)$ – время работы некоторой операции на одной машине, а k – количество машин в кластере, то во многих случаях оптимистической оценкой эффективности указанной операции в распределенной среде будет $f(G)/k$. Структура данных *S* допускает указанное увеличение эффективности в распределенной среде как базовых операций, так и самого процесса своего создания.

Пусть с помощью списков смежности *A* задан граф *G*. При создании структуры данных *S* мы должны вначале получить списки смежности *B*, задающие отношения смежности исходного графа, но в каждом списке которых вершины упорядочены по возрастанию номеров. Это осуществляется следующим образом: находясь в исходном списке $N_A(i)$ смежности вершины i, последовательно для каждой вершины j из $N_A(i)$ добавим в конец создаваемого списка смежности $N_B(j)$ вершину i. Легко заметить, что можно распараллелить обработку каждого списка (окружения вершины) из списков смежности графа. А именно, достаточно разделить

весь список $N(i)$ на множестве индивидуальных непересекающихся списков $N_1(i)$, ..., $N_p(i)$, $p < \deg(i)$, и далее для каждого из списков $N_k(i)$, $k=1,...,p$, выполнять описанные действия. Таким образом, верно

Утверждение 1. *При равномерной нагрузке и отсутствии отказов узлов, входящих в кластер, структуру данных S можно создать по спискам смежности графа $G=(V,E)$ за время $(n+m)/k$, где $m = |E|$, $n=|V|$, k - количество машин в динамическом кластере.*

Операция удаления вершины в структуре данных S реализована с помощью прохождения всех связей соответствующего «столбца» структуры данных и удаления всех связей этой вершины в каждом из соответствующих «строк» – списков смежностей. Создав дополнительные массивы и предоставив специальные Map- и Reduce- алгоритмы можно показать, что справедливо следующее

Утверждение 2. *При равномерной нагрузке и отсутствии отказов узлов, входящих в кластер, удаление вершины v из графа G можно реализовать за время d/k, где $d = \deg(v)$, k – количество машин в динамическом кластере.*

Если граф имеет значительные размеры, то невыгодно заново пересылать необходимые данные каждому из Map-алгоритмов и лучше с помощью некоторого сервиса автоматически предоставлять алгоритмам доступ ко всему графу или какой-либо его части. Можно показать, что хранение структуры данных *S* допускает такую организацию, что возможны как репликация графа на множество машин кластера, так и эффективный доступ каждого из алгоритмов к необходимым данным.

Литература

1. *Dean J., Chemawat S.*, MapReduce: Simplified Data Processing on Large Clusters // OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004, p. 137–150.
2. *Суздаль С. В., Перез Чернов А. Х,* Специальные структуры данных для задач на графах, связанных с понятиям клики или с модульными декомпозициями // Вестник БГУ, Сер. 1, 2007. С. 103–108.

ПРИБЛИЖЕННО КЕЛЕРОВЫ *F*-СТРУКТУРЫ НА ОДНОРОДНЫХ Ф-ПРОСТРАНСТВАХ ПОРЯДКА 6 ПСЕВДООРТОГОНАЛЬНЫХ ГРУПП $O(2, K)$

А. С. Самсонов

Геометрия однородных пространств в значительной степени характеризуется инвариантными аффинорными структурами, образующими алгебру. Известно [1], что в случае однородных регулярных Ф-пространств