УДК 681.324
ББК 32.973.202-018.2

ОПТИМАЛЬНЫЙ АЛГОРИТМ МИГРАЦИИ ДАННЫХ В МАСШТАБИРУЕМЫХ ОБЛАЧНЫХ ХРАНИЛИЩАХ[1]

Петров Д. Л.[2]
(*Санкт-Петербургский Государственный Электротехнический Университет «ЛЭТИ», Санкт-Петербург*)

Исследуется многокритериальная задача оптимизации плана миграции данных в распределенном облачном хранилище. Разработан алгоритм миграции данных в масштабируемых облачных хранилищах. Доказано, что алгоритм является полиномиальным и дает оптимальный результат по первому критерию.

Ключевые слова: облачные вычисления, распределенные хранилища данных, миграция данных.

Введение

Традиционно оптимизация вычислительных ресурсов в распределенных системах осуществляется при помощи процедуры балансировки нагрузки [18]. Балансировка заключается в назначении возникающих заданий определенным устройствам-обработчикам, объединенным в кластер. Это подразумевает, что любое задание может быть обработано любым из устройств в кластере. К сожалению, это условие не выполняется в распределенных хранилищах данных. Каждое из устройств не способно хранить все данные, которые могут быть востребованы. Поэтому каждый элемент данных (блок данных, файл) хранится лишь на

[1] *Научный руководитель – руководитель Центра Новых Информационных Технологий СПбГЭТУ, к.т.н., доцент Татаринов Ю. С.*

[2] *Петров Дмитрий Леонидович, начальник лаборатории (DLPetrov@mail.eltech.ru)*

ограниченном наборе устройств хранения. Процедура балансировки нагрузки в распределенных хранилищах является относительно простой и заключается лишь в выборе устройства из этого ограниченного набора.

Основная же задача распределенных хранилищ данных – регулярная переконфигурация, которая должна осуществляться всякий раз после изменения потребностей к данным [8, 9, 10, 14, 16]. Переконфигурация заключается в оптимизации распределения данных по устройствам хранения, входящим в состав хранилища. Этот процесс требует перемещения больших объемов данных и занимает много времени.

Одновременное перемещение больших объемов данных приводит к резкому падению производительности. Поэтому обычно принимают, что любое из устройств хранения одновременно может участвовать не более чем в одной операции передачи данных. Также принимают, что перемещаемые элементы данных имеют фиксированный размер и время передачи между любыми устройствами хранения. Задача составления оптимального плана перемещения данных в хранилище называется задачей миграции данных [10]. Полученный в результате ее решения план миграции данных позволяет выполнить процедуру миграции предельно быстро, т. е. оптимизация плана миграции приводит к оптимизации времени миграции. Доказано, что задача миграции данных *NP*-сложная [10], т. е. невозможно получить оптимальное решение за полиномиальное время. Существуют аппроксимационные алгоритмы ее решения [9, 10, 15].

Облачные вычисления сильно меняют взгляды на технологии [5, 17] и, в частности, на технологии хранилищ данных [16, 19]. Хранилища имеют фиксированное количество устройств хранения, соединенных сетью. Облачными хранилищами будем называть хранилища, основанные на «облачной» инфраструктуре, т. е. инфраструктуре, предоставляемой по требованию *IaaS* (*Infrastructure as a Service*). *IaaS*-инфраструктура позволяет, не ограничиваясь фиксированным количеством устройств хранения, временно арендовать дополнительные устройства в слу-

чае необходимости. И наоборот, высвобождать устройства, когда необходимость в них отпала. Изменение состава устройств хранения называется масштабированием. Именно возможность масштабироваться отличает облачные хранилища от традиционных [19]. Устройства, добавляемые к хранилищу или высвобождаемые из хранилища, будем называть масштабируемыми устройствами. Добавление или удаление устройств должно происходить во время переконфигурации хранилища.

Задача миграции данных в общем виде является *NP*-сложной [10]. Задача миграции данных в масштабируемом облачном хранилище является частным случаем задачи миграции, она была сформулирована автором этой статьи в [16] как задача многокритериальной оптимизации времени миграции. Первый критерий задачи – оптимизация времени миграции на масштабируемых устройствах хранения, т. е. оптимизация времени масштабирования. Второй критерий – оптимизация времени миграции на остальных устройствах. Эффективное решение поставленной задачи позволит предельно быстро масштабировать хранилища данных, снижая стоимость на аренду устройств хранения.

В этой статье представлен алгоритм решения задачи миграции данных в масштабируемых облачных хранилищах, а также доказана полиномиальность алгоритма и его оптимальность по первому критерию – скорости масштабирования.

1. Существующие алгоритмы

1.1. ЗАДАЧА РАСПРЕДЕЛЕНИЯ ДАННЫХ

Перед переконфигурацией хранилища необходимо вычислить оптимальное расположение данных в хранилище. Выходными данными этой задачи является отображение, в котором каждому элементу данных сопоставляются устройства хранения, на которых он должен быть после переконфигурации. Задача является *NP*-сложной, но были разработаны полиномиальные аппроксимационные алгоритмы ее решения [8, 14]. Вычислив новое расположение данных и имея старое расположение, можно построить направленный мультиграф без циклов G, демонстрирующий пе-

ремещение элементов данных из старой конфигурации в новую. Этот граф называют графом требований [10]:

$$
\begin{aligned}
& G = (V, E, P) \\
& E \subseteq V \times V \\
& P : E \to \mathbb{N} \\
& \forall v, w \in V, P(v, w) = 0 \text{ если } (v, w) \notin E
\end{aligned}
\tag{1}
$$

V – устройства хранения; E – операция перемещения; P – весовая функция мультиграфа.

1.2. ЗАДАЧА МИГРАЦИИ ДАННЫХ

План миграции можно разбить на шаги, поскольку все элементы данных имеют фиксированный размер и одинаковое время передачи. В данной работе мы не учитываем емкость устройств хранения. Задача миграции данных заключается в составлении плана перемещения данных между устройствами хранения согласно графу требований (1) за минимальное число шагов. Очевидно, что задача в таком виде элементарно сводится к задаче раскраски дуг мультиграфа. Минимальное число цветов раскраски дуг мультиграфа называют хроматическим индексом мультиграфа и обозначают χ'. Следует заметить, что направление передачи данных в хранилище не имеет значения с точки зрения задачи миграции [10]. Устройство считается занятым независимо от того, принимает оно данные или передает. Замена направленного мультиграфа на ненаправленный позволит сократить минимальное количество цветов раскраски:

$$
\begin{aligned}
& G = (V, E, P) \\
& E \subseteq V \times V \\
& P : E \to \mathbb{N} \\
& \forall v, w \in V \Rightarrow P(v, w) = P(w, v)
\end{aligned}
\tag{2}
$$

1.3. АЛГОРИТМЫ РАСКРАСКИ РЕБЕР МУЛЬТИГРАФОВ

Задача раскраски дуг мультиграфа в общем виде является *NP*-сложной [12] и, следовательно, алгоритмы ее точного решения требуют полного перебора всех вариантов и имеют экспонен-

циальную вычислительную сложность, т. е. не являются полиномиальными.

Алгоритм 1 (Раскраска ребер мультиграфа)**.** В учебнике [2] описан неполиномиальный алгоритм раскраски ребер мультиграфа, основанный на полном переборе всех вариантов раскраски ребер.

Низкая эффективность неполиномиальных алгоритмов не позволяет использовать их на практике. Для задачи раскраски ребер мультиграфа существует множество аппроксимационных полиномиальных алгоритмов [7, 11].

Алгоритм 2 (Полиномиальная раскраска ребер)**.** В статье [11] описан полиномиальный аппроксимационный алгоритм раскраски ребер мультиграфа с вычислительной сложностью $O(|A|(|V| + \delta))$, где A – множество ребер мультиграфа; V – множество вершин; δ – некоторая константа.

Алгоритм основан на принципе «разделяй и властвуй». Общее решение для всего графа выстраивается из объединения частных решений для множества раскрашенных подграфов. В начале работы алгоритма максимальный размер подграфов ограничен небольшим числом. Ребра выбираются последовательно в произвольном порядке и формируют новые подграфы или входят в состав уже известных. При определенных условиях новое ребро может объединить два подграфа в один. В процессе работы алгоритма максимальный размер подграфов постепенно увеличивается. После прохода по всем ребрам раскрашенные подграфы объединяются в единый граф с общей раскраской.

В работе существующие алгоритмы не описываются подробно, а лишь приводятся ссылки на них.

Раскраска дуг двудольного мультиграфа является частным случаем задачи раскраски дуг, но эта задача не принадлежит классу *NP*-сложных задач. Вероятно, первое решение этой задачи было предложено в работе [13]. Но существуют алгоритмы с меньшей трудоемкостью:

Алгоритм 3 (Раскраска ребер двудольного мультиграфа)**.** В статье [4] можно найти описание оптимального алгоритма

раскраски ребер с вычислительной сложностью $O(|A| \log D)$, где A – множество ребер; D – максимальная степень вершин мультиграфа.

Алгоритм 3, аналогично алгоритму 2, основан на принципе «разделяй и властвуй». Но в случае простого, двудольного графа можно определить правила добавления ребер к тем или иным подграфам и правила объединения подграфов, которые позволяют формировать заведомо оптимальную раскраску на каждом шаге алгоритма.

Для оценки эффективности алгоритма потребуется следующая теорема, позволяющая приближенно оценить значение хроматического индекса произвольного мультиграфа:

Теорема 1 (Теорема Визинга)**.** $\Delta(G) \leqslant \chi'(G) \leqslant \Delta(G) + 1$, *где G – произвольный граф; $\Delta(G)$ – максимальная степень вершины.*

Доказательство. См. [1] ∎

2. Миграция данных в масштабируемых облачных хранилищах

2.1. МОДЕЛЬ МАСШТАБИРУЕМОГО ОБЛАЧНОГО ХРАНИЛИЩА

Опишем модель масштабируемого облачного хранилища на основе хранилища без ограничений объема устройств хранения (2). Чтобы учесть особенности масштабируемого облачного хранилища, явно выделим подмножество масштабирующих устройств хранения, которое необходимо вывести из строя (или добавить) в результате переконфигурации.

Определение 1. *Масштабирующим (scaling) подмножеством S будем называть подмножество вершин, которые необходимо вывести из строя или добавить к хранилищу в результате переконфигурации. Среди операций передачи данных не может быть операций передачи с одного масштабирующего устройства на другое, т. е.*

$$\forall v, w \in S \subseteq V \Rightarrow P(v, w) = 0 \tag{3}$$

Определение 2. *Масштабируемым облачным хранилищем G будем называть хранилище с явно выделенным масштабирующим подмножеством S.*

$$\begin{aligned} &G = (V, E, P, S) \\ &\forall v, w \in S \subseteq V \Rightarrow P(v, w) = 0 \end{aligned} \tag{4}$$

2.2. РАЗБИЕНИЕ МАСШТАБИРУЕМОГО ОБЛАЧНОГО ХРАНИЛИЩА

Во время масштабирования необходимо выполнить процедуру миграции на всех масштабирующих устройствах S. После этого их можно высвободить и выполнить миграцию данных на оставшихся устройствах хранения.

Разделим процедуру миграции на две части: масштабирование и остаточную миграцию. Для решения подобных задач часто используют методы разделения графов на подграфы [3, 6]. Разделим граф масштабируемого облачного хранилища G на два подграфа: масштабирующий подграф G_S и остаточный G_R.

Определение 3. *Масштабирующим подграфом G_S будем называть подграф масштабируемого облачного хранилища G, образованный множеством вершин S и множеством вершин S_{adj}, смежными с S, а также всеми дугами, соединяющими вершины S с S_{adj}.*

$$\begin{aligned} &G_S = (V_S, E_S, P) \\ &V_S \subseteq V \\ &E_S \subseteq E \\ &v \in S_{adj} \subseteq V \Leftrightarrow \exists w \in S, P(v, w) > 0 \\ &V_S = S \cup S_{adj} \\ &(v,w) \in E_S \Leftrightarrow v \in S, w \in S_{adj}, P(v, w) > 0 \end{aligned} \tag{5}$$

В последней формуле определения (5) выражение $v \in S, w \in S_{adj}, P(v, w) > 0$ эквивалентно выражению $v \in S_{adj}, w \in S, P(v, w) > 0$, поскольку граф G ненаправленный, в соответствии с (2). В левой части рис. 1 показан граф G, подмножество S (черные вершины) и S_{adj} (заштрихованные вершины).

Определение 4. *Остаточным (residual) подграфом G_R будем называть подграф масштабируемого облачного хранилища G, образованный всеми дугами графа G без масштабирующего подмножества S, и все дуги, соединяющие эти вершины.*

$$
\begin{aligned}
&G_R = (V_R, E_R, P)\\
&V_R \subseteq V\\
&E_R \subseteq E\\
&V_R = V \setminus S\\
&(v,w) \in E_R \Leftrightarrow v, w \in V_R, P(v,w) > 0
\end{aligned}
\tag{6}
$$

Очевидно, что множество вершин S_{adj} принадлежит как G_S, так и G_R, в соответствии с их определениями (5) и (6). Дуги, соединяющие вершины S_{adj}, лежат только в G_R. В правой части рис. 1 изображен граф, разделенный на два подграфа G_S и G_R.

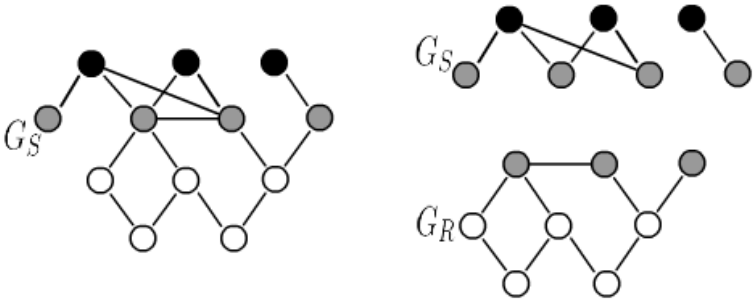

Рис. 1. Разбиение графа масштабируемого облачного хранилища G на два подграфа: G_S и G_R

2.3. ЗАДАЧА МИГРАЦИИ ДАННЫХ В МАСШТАБИРУЕМОМ ОБЛАЧНОМ ХРАНИЛИЩЕ

Используя введенные термины, можно сформулировать задачу миграции данных в масштабируемом облачном хранилище.

Определение 5. *Задача миграции данных в масштабируемом облачном хранилище G – это многокритериальная задача*

оптимизации времени миграции. Основным критерием задачи является время миграции в подграфе G_S, а второй критерий – время миграции в подграфе G_R.

3. Алгоритмы миграции данных в масштабируемом облачном хранилище

Опишем алгоритм миграции данных в масштабируемом облачном хранилище, дающий оптимальный результат по первому критерию – оптимальности времени масштабирования. Для этого докажем, что масштабирующий подграф G_S является двудольным мультиграфом, для которого существуют оптимальные полиномиальные алгоритмы раскраски дуг.

Лемма 1. *Масштабирующий подграф G_S является двудольным.*

Доказательство. Масштабирующий подграф состоит из множества вершин S и вершин S_{adj}, смежных с S, исходя из определения (5). Докажем, что множества вершин S и S_{adj} разбивают граф G_S на две доли, т. е. в графе нет дуг, идущих из S в S или же из S_{adj} в S_{adj}.

Вершины из множества S_{adj} не имеют общих дуг, поскольку это противоречит определению масштабирующего подмножества (3), входящего в состав масштабируемого облачного хранилища (4). Также из (5) следует, что любая его дуга соединяет вершины из S только с вершинами из S_{adj}. Следовательно, вершины из S_{adj} также не могут иметь общих дуг. Таким образом, множества вершин S и S_{adj} составляют две доли двудольного мультиграфа G_S.

Алгоритм 4 (Миграция данных)**.** Для решения задачи будем использовать метод уступок. Выделив два критерия оптимизации, будем оптимизировать алгоритм по первому критерию, для чего потребуется поступиться эффективностью по второму критерию. Первый частный критерий – время масштабирования, т. е. миграция данных на масштабирующем подграфе G_S. Второй критерий – время миграции на остальных устройствах, т. е. миграция дан-

ных на подграфе G_R. Опишем алгоритм в виде последовательности шагов, описанных выше.

1. Выделить подграф G_S из графа G на основе известного подмножества S.

2. Выделить подграф G_R из G на основе подграфа G_S.

3. Использовать алгоритм 3 для расчета плана миграции двудольного мультиграфа G_S.

4. Использовать переборный алгоритм 1 для расчета плана миграции мультиграфа G_R.

5. Получить общий плана миграции масштабируемого облачного хранилища G путем последовательного объединения планов миграции мультиграфа G_S с планом миграции G_R.

Алгоритм 5 (Полиномиальная миграции данных)**.** Шаги аналогичны алгоритму 4. Но на шаге 4 используется полиномиальный алгоритм 2 вместо переборного алгоритма 1.

Теорема 2 (Оптимальность алгоритмов 4 и 5)**.**
Предложенные алгоритмы 4 и 5 оптимальны по первому критерию задачи миграции данных в масштабируемом облачном хранилище, сформулированной в определении 5.

Доказательство. Рассмотрим сначала алгоритм 4. В соответствии с доказанной леммой 1, подграф G_S является двудольным мультиграфом, и для нахождения оптимальной раскраски дуг G_S в алгоритме используется полиномиальный алгоритм 3, который дает оптимальное время миграции подграфа G_S. Исходя из постановки задачи, сформулированной в определении 5, оптимальность алгоритма по первому критерию обеспечивается благодаря оптимальности времени миграции подграфа G_S (шаг 3 алгоритма). То есть алгоритм 4 является оптимальным по первому критерию.

Полиномиальный алгоритм 5 отличается от 4 только шагом 4, а оптимальность, как показано выше, обеспечивается на шаге 3. Следовательно, алгоритм 5 также является оптимальным по первому критерию. ∎

Теорема 3 (Полиномиальность алгоритма 5)**.**
Предложенный алгоритм 5 имеет полиномиальную вычис-

лительную сложность.

Доказательство. Очевидно, что шаги 1 и 2 алгоритма 5 являются полиномиальными, поскольку вычисление вершин S_{adj}, смежных с S, – задача тривиальная, а выделение подграфов G_S и G_R, состоящих из заданных вершин, имеет линейную сложность от количества вершин $O(|V|)$. Шаги 3 и 4 также являются полиномиальными, их вычислительная сложность равна $O(|A|\log\Delta)$ и $O(|A|(|V|+\delta))$, в соответствии с алгоритмом 2 и алгоритмом 3, где A – множество ребер; V – множество вершин; Δ – максимальная степень вершин; δ – некоторая константа.

Результат выполнения шагов 3 и 4 — два плана миграции, состоящие из последовательности операций перемещения данных (дуг подграфов), а шаг 5 состоит из простой операции последовательного объединения двух планов миграции. Очевидно, что шаг 5 имеет линейное время выполнения $O(|A|)$, зависящее от числа ребер графа G .

Поскольку алгоритм 5 является последовательным объединением шагов 1–5, его вычислительная сложность равна сложности наиболее затратного шага: $\max\{O(|V|), O(|A|\log\Delta), O(|A|(|V|+\delta)), O(|A|)\} = \max\{O(|A|\log\Delta), O(|A|(|V|+\delta))\}$. Из этого следует, что алгоритм имеет полиномиальную сложность. ∎

4. Экспериментальная оценка эффективности алгоритма

4.1. ЗАДАЧА ОЦЕНКИ ЭФФЕКТИВНОСТИ

Используя традиционные алгоритмы миграции данных, масштабирование облачного хранилища можно произвести только после полного выполнения процедуры миграции данных. Это связанно с тем, что масштабирующиеся устройства хранения могут быть задействованы в процедуре миграции данных до самых последних шагов. Разработанный алгоритм выделяет в графе хранилища G масштабирующий подграф G_S, содержащий все масштабирующиеся устройства. Это позволяет скорее выполнить процедуру миграции на этих устройствах и высвободить их с целью экономии затрат на аренду устройств. Обратной стороной алго-

ритма является необходимость выполнения остаточной миграции в подграфе G_R после выполнения миграции в G_S и последовательное объединение шагов миграции, которое увеличивает общее время миграции.

С практической точки зрения интересны ответы на следующие вопросы:

- на сколько уменьшается время (количество шагов) масштабирования?
- на сколько увеличивается время полной процедуры миграции?

Такие оценки удобно производить в процентах относительно общего времени миграции G с использованием традиционных алгоритмов.

Количество шагов в плане миграции равно количеству цветов раскраски дуг соответствующего мультиграфа хранилища, т. е. его хроматическому индексу χ'. Значения хроматических индексов $\chi'(G)$, $\chi'(G_S)$ и $\chi'(G_R)$ соответствуют количеству шагов в плане миграции для соответствующих графов хранилищ. В качестве приближенной оценки χ', в соответствии с теоремой 1, можно использовать максимальную степень вершин мультиграфа Δ плюс 1. Будем обозначать эту оценку как Δ'. Использование приближенных оценок Δ' вместо χ' позволяет быстро и достаточно точно произвести оценку, не прибегая к трудоемкой процедуре вычисления раскраски ребер мультиграфа.

Эффективность переборного алгоритма 4 можно оценить формально, поскольку он дает оптимальный результат. Эффективность алгоритма 5 оценивать не будем, поскольку он является приближением алгоритма 4 и отличается от алгоритма 4 лишь планом миграции в подграфе G_R, что не существенно с точки зрения первого критерия оптимизации (определение 5).

4.2. МЕТОДИКА ОЦЕНКИ

Исходя из задачи оценки эффективности алгоритмов миграции данных в облачном хранилище будем использовать следующие параметры оценки:

- $P_{step} = \chi'(G) - \chi'(G_S) \approx \Delta'(G) - \Delta'(G_S)$ – время (кол-во шагов) масштабирования, которое было сэкономлено в результате применения алгоритма;
- $P_{rel} = \dfrac{P_{step}}{\chi'(G)} \approx \dfrac{\Delta'(G) - \Delta'(G_S)}{\Delta'(G)}$ – относительный выигрыш от масштабирования, соответствующий сэкономленному времени по сравнению с традиционным алгоритмом;
- $L_{step} = \chi'(G_S) + \chi'(G_R) \approx \Delta'(G_S) + \Delta'(G_R)$ – общее время миграции данных с новым алгоритмом;
- $L_{rel} = \dfrac{L_{step}}{\chi'(G)} - 1 \approx \dfrac{\Delta'(G_S) + \Delta'(G_R)}{\Delta'(G)} - 1$ – относительный проигрыш времени от разделения графа, соответствующий дополнительно затраченному времени на миграцию от общего времени миграции данных.

Можно вычислить абсолютный выигрыш от масштабирования в машино-часах (точнее в машино-шагах), которые будут сэкономлены в результате применения алгоритма: $P = P_{step} * |S|$.

Проигрыш времени процедуры миграции (абсолютный проигрыш L) не приводит к потере ресурсов, но он приводит к увеличению времени работы хранилища в режиме переконфигурации и, соответственно, к увеличению времени между последовательными процедурами миграции. Для некоторых видов хранилищ это может быть важной характеристикой.

4.3. ЭКСПЕРИМЕНТ

Для проведения эксперимента было выбрано несколько случайных графов, в которых число масштабирующихся устройств хранения $|S|$ невелико по сравнению с общим числом устройств хранения $|V|$, и количество дуг между устройствами подграфа G_S также невелико по сравнению с дугами между обычными устройствами из G_R. Эти условия типичны для прикладных задач, в которых масштабирующиеся устройства нельзя отключать (или подключать) от хранилища в большом количестве, а также

при отключении (включении) устройств хранения не все данные должны передаваться из (на) этого хранилища сразу.

Проведено 4 эксперимента с графами различного размера. Граф хранилища из эксперимент №1 изображен на рис. 1.

Таблица 1. Результаты экспериментов

№	$\|V\|$	$\|S\|$	$\Delta'(G)$	$\Delta'(G_S)$	$\Delta'(G_R)$	P_{rel}	P	L_{rel}
1	12	3	5	3	3	40%	6	20%
2	20	5	13	6	13	54%	35	46%
3	28	12	17	8	16	53%	108	41%
4	37	16	22	11	22	50%	176	50%

В результате эксперимента было выяснено, что для хранилищ с относительно небольшим количеством масштабирующихся устройств хранения предложенный алгоритм способен обеспечивать экономию времени масштабирования на 40–60% по сравнению с существующими алгоритмами. При этом общее время миграции данных может увеличиться на 20–50%.

4.4. ВОЗМОЖНЫЕ УЛУЧШЕНИЯ АЛГОРИТМА

Увеличение общего времени миграции на 20–50% для некоторых видов хранилищ может являться существенной проблемой, несмотря на оптимальное время масштабирования. Предложенный алгоритм можно улучшить. Основной способ улучшения алгоритма – замена последовательного объединения планов миграции в подграфах G_S и G_R на «параллельное объединение» с целью сокращения общего количества цветов раскраски. Проще говоря, во время миграции данных в подграфе G_S можно параллельно выполнять миграцию в подграфе G_R таким образом, чтобы она не влияла на миграцию в G_S, сохраняя тем самым оптимальность алгоритма по первому критерию.

Допустимы как минимум два подхода сокращения общего количества цветов раскраски:

- раздельная раскраска G_S и G_R (как это делается в алгоритмах 4 и 5), но при составлении общего плана миграции G

для каждого цвета в раскраске G_S производить поиск такого цвета в G_R, чтобы эти цвета можно было объединить в один цвет в общей раскраске G

- расширить двудольный граф G_S вершинами и ребрами из G_R так, чтобы количество цветов раскраски расширенного G_S не увеличилось, а количество цветов раскраски G_R уменьшилось

Указанные улучшения требуют дополнительной проработки и подробно не рассматриваются в данной работе.

5. Выводы

В статье описана многокритериальная оптимизационная задача миграции данных в масштабируемых облачных хранилищах и предложены алгоритмы ее решения. Задача разделяет общую задачу миграции на две подзадачи: масштабирования и остаточной миграции. Основной критерий оптимизации – минимизация времени (шагов) масштабирования. Показано, что задача является *NP*-сложной.

Предложено два алгоритма решения задачи: переборный алгоритм и аппроксимационный алгоритм полиномиальной вычислительной сложности $\max\{O(|A|\log\Delta), O(|A|(|V|+\delta))\}$, где A – множество ребер; V – множество вершин; Δ – максимальная степень вершин; δ – некоторая константа. Доказано, что алгоритмы дают оптимальный результат по первому критерию — минимизации времени масштабирования.

Экспериментально показано, что для хранилищ с относительно небольшим количеством масштабирующихся устройств хранения предложенный алгоритм способен обеспечивать экономию времени масштабирования на 40–60% по сравнению с существующими алгоритмами. При этом общее время миграции данных может увеличиться на 20–50%.

Оптимальная скорость масштабирования позволит облачным хранилищам предельно быстро арендовать и высвобождать устройства хранения данных и, тем самым, минимизировать срок

и стоимость аренды устройств, предоставляемых как инфраструктура по требованию *IaaS*.

Литература

1. ВИЗИНГ В. Г. *Хроматический класс мультиграфа* // Кибернетика. – 1965. – №3. – С. 29–39.
2. ЕВСТИГНЕЕВ В., КАСЬЯНОВ В. *Графы в программировании: обработка, визуализация и применение*. – С.-Пб.: БХВ-Петербург, 2003. – 1104 с.
3. BERGER M., BOKHARI S. *Partitioning strategy for nonuniform problems on multiprocessors* // IEEE Transactions on Computers. – 1987. –C-36(5). – P. 570–580
4. COLE R., OST K., SCHIRRA S. *Edge-coloring bipartite multigraphs in O(E log D) time* // Combinatorica. – 2001. – №21. – P. 5–12.
5. DELIC K. A., WALKER M. A. *Emergence of the academic computing clouds* // ACM Ubiquity. – 2008. – Vol. 9, Iss. 31. – P. .
6. GEORGE A., LIU J. *Computer Solution of Large Sparse Positive Definite Systems*. – Prentice-Hall, Englewood Cliffs NJ, 1981.
7. GOLDBERG M. K. *Edge-coloring of multigraphs: Recoloring technique* // J. Graph Theory. – 1984. – №8. – P. 121–137.
8. GOLUBCHIK L., KHANNA S., KHULLER S., THURIMELLA R. AND ZHU A. *Approximation Algorithms for Data Placement on Parallel Disks* // Proc. of ACM-SIAM SODA, 2000.
9. GOLUBCHIK L., KHULLER S., KIM Y. A., SHARGORODSKAYA S., WAN Y. *Data Migration on Parallel Disks: Algorithms and Evaluation* // Algorithmica. – 2006. – №45(1). – P. 137–158.
10. HALL J., HARTLINE J., KARLIN A., SAIA J., WILKES J. *On Algorithms for Efficient Data Migration* // ACM Symposium on Discrete Algorithms. – 2001. – P. 620–629.

11. HOCHBAUM D. S., NISHIZEKI T., SHMOYS D. B. *A better than "Best Possible" algorithm to edge color multigraphs* // J. off Algorithms. – 1986. – №7. – P. 79–104.
12. HOLYER I. *The NP-completeness of Edge-Coloring* // SIAM J. Comp. – 1982. №11. – P. 117–129.
13. HOPCROFT J., KARP R. *An n. 5/2 algorithm for maximum matchings in bipartite graphs* // SIAM J. Comput. – 1973. – №2. – P. 225–231.
14. KASHYAP S., KHULLER S. *Algorithms for Non-Uniform Size Data Placement on Parallel Disks* // Conference on Foundations of Software technology and Theoretical Computer Science, LNCS 2914. – 2003. – P. 265–276.
15. KHULLER S., KIM Y. A. *Algorithms for Data Migration with Cloning* // SIAM Journal on Computing. – 2004. – Vol. 33, №2. – P. 448–461.
16. PETROV D. L., TATARINOV Y. *Data migration in the scalable storage cloud* // IEEE, International Conference on Ultra Modern Telecommunications, ICUMT, St. Petersburg, 2009.
17. ROBISON S. *A Bright Future in the Cloud* // Financial Times, March 4, 2008.
18. SHIRAZI B. A., KAVI K. M., HURSON A. R. *Scheduling and Load Balancing in Parallel and Distributed Systems* // IEEE Computer Society Press. – 1995. – 448 p.
19. ZUCKERMAN B. *Scalable storage in the cloud* // Cloud Slam Conference, 2009. – URL: http://cloudslam09.com.

OPTIMAL ALGORITHM FOR DATA MIGRATION IN SCALABLE STORAGE CLOUDS

Dmitry L. Petrov, Saint Petersburg Electrotechnical University «LETI», Saint Petersburg, Head of Laboratory (DLPetrov@mail.eltech.ru).

Abstract: The problem of multi-criteria optimization of data migration schedule in distributed cloud storage is investigated. The algorithm for data migration in scalable cloud storage is developed. We prove that the algorithm has polynomial computational complexity and results in the optimal value of the first criterion.

Keywords: computing clouds, distributed storage system, data migration.

Статья представлена к публикации членом редакционной коллегии М. В. Губко