Определение числа реплик распределенного хранения больших данных

Т. М. Татарникова, Е. Д. Архипцев
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)
tm-tatarn@yandex.ru

***Аннотация*. Обсуждается проблема распределенного хранения больших данных. Рассматриваются схемы репликации, позволяющие ускорить доступ к данным: одноранговая и «ведущий – ведомый». Показано, что репликация приводит к несогласованности данных, которая на практике является компромиссом, сформулированным в теореме CAP: из трех свойств – согласованности и доступности данных, их устойчивости к разделению – можно обеспечить не больше двух свойств. Предложена модель определения необходимого числа узлов в кластере и числа реплик, основанная на теории очередей.**

Ключевые слова: большие данные, вычислительный кластер, коэффициент репликации, согласованность данных, теорема CAP

I. Введение

Основным свойством больших данных является возможность распределенного хранения данных – горизонтальное масштабирование базы данных [1].

Существуют два способа распределения данных: репликация и фрагментация [2].

Репликация подразумевает копирование одних и тех же данных (реплик) на нескольких узлах вычислительного кластера. Количество реплик называется коэффициентом репликации.

Естественным для репликации является необходимость размещения данных, которые запрашиваются одновременно, на одном и том же узле с целью ускорения доступа к ним [3].

Неизбежным недостатком репликации является несогласованность. Согласованность – это обеспечение внутренней непротиворечивости данных. Таким образом, всегда существует риск того, что некоторые пользователи получат недостоверные данные в силу того, что обновления на репликах выполняются с определенной задержкой, называемой окном несогласованности.

Фрагментация подразумевает, что разные части базы данных размещаются на множество узлов одного кластера. Актуальной задачей фрагментации является группировка данных таким образом, чтобы один пользователь в основном получал данные с одного узла. Поэтому в больших данных агрегат является естественной единицей распределения – объединяет данные, которые, как правило, запрашиваются одновременно.

Репликация и фрагментация являются ортогональными методами: можно использовать любую из двух или обе вместе.

A. Схемы репликаций

Репликация бывает двух видов: первая, реализуется по одноранговой схеме, а вторая – по схеме «ведущий-ведомый» [2].

При одноранговой репликации все узлы кластера имеют одинаковый вес и могут выполнять операции записи. Однако, когда выполняется запись в два разных места, есть риск, что два человека попробуют обновить одну и ту же запись в один и тот же момент времени. Таким образом, возникает конфликт «запись-запись». Несогласованность чтения тоже приводит к проблемам, но они являются преодолимыми, а несогласованность записи имеет необратимый характер.

При распределении данных по схеме «ведущий-ведомый» происходит репликация данных по многим узлам кластера. Один из узлов назначается ведущим (master) – доверенным источником. Остальные узлы являются ведомыми (slaves). Процесс репликации синхронизирует ведомые узлы с ведущим.

Репликация "ведущий-ведомый" – это решение для баз данных с интенсивным выполнением операции чтения и неудачное для баз данных с интенсивным трафиком записи: если часто обновлять данные на ведущем узле, то вероятность того, что пользователи будут получать несогласованные данные, растет с увеличением числа ведомых узлов.

Преимуществом репликации "ведущий-ведомый" является отказоустойчивость чтения: если на ведущем узле произойдет отказ, ведомые узлы смогут по-прежнему обрабатывать запросы на чтение. Сбой ведущего узла сделает невозможной запись данных до тех пор, пока его работа не будет восстановлена. И наличие реплик ускоряет процесс восстановления ведущего узла после его сбоя [3].

B. Теорема CAP

Различают строгую и итоговую согласованность. Строгая – гарантирует, что данные вернутся полностью достоверными и не устареют. Итоговая согласованность не может гарантировать, что данные вернуться полностью достоверными, но со временем данные обновятся на всех репликах.

Согласованность проявляется в разных формах.

Представим процедуру обновления данных, при которой пользователи User1 и User2 изменяют одни и те же данные в один и тот же момент времени *tw* (рис. 1) – конфликт «запись-запись».

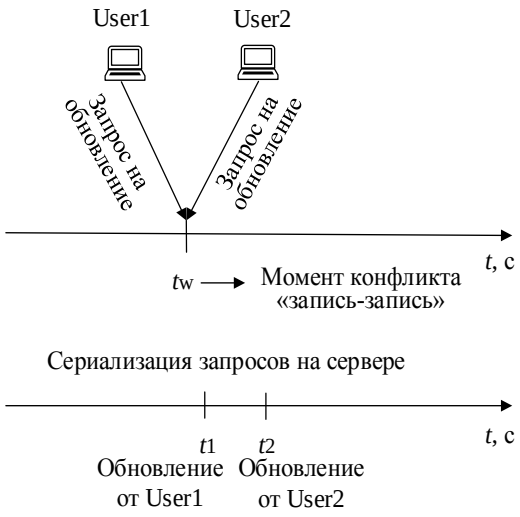

Рис. 1. Сериализация запросов

Когда записи достигают сервера, то он их сериализирует – обрабатывает одну, а потом другую, например, в алфавитном порядке, и реализует сначала обновление первого пользователя User1, а затем обновление User2. Если контроля согласованности нет, то обновление первого пользователя будет выполнено, а затем перекрыто обновлением второго пользователя. В этом случае обновление User1 будет потеряно. Это явление есть пример нарушения согласованности обновления, потому что обновление User2 использовало состояние до обновления User1, но применялось после него.

Представим процедуру чтения данных. На рис. 2 показана ситуация, в которой User2 выполнил чтение в середине процедуры записи, выполняемой User1. Для исключения такой ситуации используется метод логической согласованности. Она гарантирует, что разные элементы данных будут изменяться вместе. Например, для того чтобы решать проблему логической несогласованности при конфликте «чтение-запись», реляционные базы данных используют понятие транзакций. Обе записи первого пользователя упаковываются в одну транзакцию, и тем самым система гарантирует, что User2 будет читать оба элемента данных либо до, либо после обновления. В базах данных NoSQL задача несогласованности решается с помощью агрегатной модели данных – данные, к которым применяются последовательные действия, упаковываются в один агрегат.

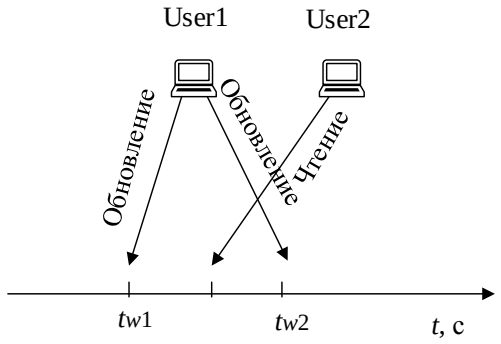

Рис. 2. Конфликт чтения данных

Разумеется, не все данные можно записать в один и тот же агрегат. Поэтому оставляют интервал времени, в течение которого клиенты могут выполнить несогласованное чтение. Продолжительность этого интервала называется окном несогласованности. Например, в документации компании Amazon сказано, что окно несогласованности обычно не превышает секунды – T_w.

Почти всегда можно разработать систему, предотвращающую несогласованность, но практически невозможно сделать это без ущерба для остальных характеристик системы. В результате часто приходится жертвовать согласованностью в пользу чего-то другого.

В больших данных этот компромисс формулируется теоремой CAP. CAP – это акроним от трех свойств:

- Согласованность (**C**onsistency) – непротиворечивость данных.
- Доступность (**A**vailability) – предельное время отклика, которое допустимо.
- Устойчивость к разделению (**P**artition tolerance) означает, что кластер может восстанавливать обмен данными после обрыва связей в кластере.

Теорема гласит, что из трех свойств – согласованности и доступности данных, их устойчивости к разделению – можно обеспечить не больше двух свойств. На практике CAP реализует следующее утверждение: в системе, которая подвержена разделению, следует искать компромисс между согласованностью и доступностью. Полученная в результате система не будет ни хорошо согласованной, ни идеально доступной, но она будет представлять собой разумное сочетание этих свойств.

II. Определение количества реплик

Введем следующие обозначения:

R – коэффициент репликации;

W – количество узлов, участвующих в записи;

N – количество узлов, участвующих в чтении.

Согласно теореме CAP, не нужно, чтобы все узлы подтверждали запись для обеспечения строгой согласованности; нужно, чтобы это сделали большинство. Это явление называется кворумом записи и выражается неравенством

$$W>N/2. \quad (1)$$

Аналогично существует кворум чтения. Кворум чтения зависит от того, сколько узлов должны подтвердить запись. Пусть R=3. Если все записи должны подтвердить два узла (W=2), то необходимо установить контакт по крайней мере с двумя узлами, чтобы гарантировать получение последних данных. Если же записи подтверждаются только одним узлом (W=1), надо связаться со всеми тремя узлами, чтобы гарантировать получение последних обновлений. В последнем случае нет кворума записи, поэтому возникает конфликт обновлений, но, контактируя с достаточно большим количеством читателей, обнаружение конфликта гарантированно. Таким образом, можно получить строго согласованные результаты чтения, даже если нет строгой

согласованности записей. Неравенство получения строгой согласованности:

$$R+W>N. \quad (2)$$

Неравенства (1) и (2) выведены для одноранговой модели распределения.

В случае распределения «ведущий-ведомый», чтобы избежать конфликтов «запись-запись», достаточно записать данные на ведущий узел. Чтобы избежать конфликтов «чтение-запись», достаточно выполнять чтение только с ведущего узла.

Уточним, что количество узлов в кластере и коэффициент репликации – это разные параметры вычислительного центра (ВЦ). Например, кластер может иметь 100 узлов при коэффициенте репликации, равном 3. При распределенном хранении базы данных необходимо найти баланс (золотую середину) между этими параметрами. С одной стороны, время отклика на запрос пользователя не должно быть выше допустимого времени $T_{\text{доп}}$, которое определяет гарантированное время терпеливости клиента и, соответственно, качество функционирования ВЦ [4]. С другой стороны, окно несогласованности должно гарантировать, что конфликт чтения данных устраним за ограниченное время [5].

A. Определение оптимальной структуры кластера

Кластер представим замкнутой системой массового обслуживания, в которой N − количество узлов кластера. Узлы задаются временем обслуживания T_i^{ser}, $i=\overline{1,N}$. Переход запроса клиента распределенной базы данных задается матрицей вероятностей p_{ij}. Необходимо найти оптимальное количество узлов кластера, такое чтобы запрос клиента выполнялся за допустимое время $T_{\text{доп}}$.

Для решения поставленной задачи предлагается следующая рекуррентная процедура:

$$\bar{T}_i(J)=T_i^{ser}\left(1+\left(\frac{\bar{L}_i(J-1)}{N}\right)\right),\ i=\overline{1,N}; \quad (3)$$

$$\bar{T}(J)=\sum_{i=1}^{N} e_i\bar{T}_i(J); \quad (4)$$

$$\bar{\Lambda}(J)=\frac{J}{\bar{T}(J)}; \quad (5)$$

$$\bar{L}_i(J)=\bar{\Lambda}(J)e_i\bar{T}_i(J), \quad (6)$$

где $\bar{T}_i(J)$ – среднее время пребывания заявки в i-м узле при наличии в сети J заявок; $\bar{T}(J)$ – среднее время пребывания заявки в кластере при наличии в сети J заявок; $\bar{\Lambda}(J)$ – пропускная способность кластера при наличии в нем J заявок; вектор $e=\left[e_i\right]_{i=\overline{1,N}}$ является решением системы линейных уравнений

$$e_i=\sum_{j=1}^{N} e_{ij}p_{ij}, \quad (7)$$

которая определяет стационарное распределение цепи Маркова, управляющей переходами заявок с матрицей вероятностей переходов p_{ij}, $i,j=\overline{1,N}$.

Система (7) решается при дополнительном ограничении

$$\sum_{i=1}^{N} e_i=1.$$

Решение процедуры (3)−(7) начинается с $\bar{L}_i(0)=0$, для $i=\overline{1,N}$.

Выходом из рекурсии является условие:

$$\frac{\Lambda(j-1)}{\Lambda(j)}\geq\varepsilon,\ 0,9<\varepsilon<1. \quad (8)$$

Если (8) выполняется, то система вошла в состояние насыщения – найдено максимальное число заявок, при которых система находится в стационарном режиме.

Далее необходимо проверить соответствие времени пребывания заявки в кластере с допустимым временем:

$$T\leq T_{\text{доп.}} \quad (9)$$

Если условие (9) выполняется, то оптимальная структура системы – число узлов кластера найдено, иначе увеличить на единицу число каналов (узлов) и повторить процедуру (3)–(8) для новой структуры. Повторять структурную оптимизацию до выполнения условия (9).

Наращивание вычислительной мощности кластера через увеличение узлов не является единственным способом выбора оптимальной структуры кластера. Воспользовавшись той же процедурой (3)–(8). Можно подобрать характеристики узлов, на которых строится кластер. На практике это означает замену узлов на новые, более производительные.

Таким образом, предлагается следующая методика выбора оптимальной структуры:

1. Установка J=1 – запуск в кластер одной заявки на обновление данных.
2. Вычисление $\bar{\Lambda}(J)=\frac{J}{\bar{T}(J)}$.
3. Проверка условия стационарного режима функционирования системы: $\bar{\Lambda}(J)\geq\bar{\lambda}$, где $\bar{\lambda}$ – средняя интенсивность входного потока заявок на обновление данных. Если условие не выполняется, то переход на п. 5.
4. Проверка условия $\bar{T}\leq T_{\text{доп}}$. Если условие не выполняется, то переход на п. 9, иначе – переход на п. 8.
5. Фиксировать N, $\bar{T}$, $\bar{\Lambda}$ и J переход на п. 10.
6. Проверка условия $\frac{\Lambda(J-1)}{\Lambda(J)}\geq\varepsilon$. Если условие выполняется, то переход на п. 5.

7. Вычисление $\bar{L}_i(J)$. Переход на п. 1.
8. Запуск еще одной заявки: $J=J+1$, переход на п. 2.
9. Увеличиваем число узлов в кластере: $N=N+1$, переход на п. 2.
10. Конец.

Пусть кластер состоит из одинаковых узлов с характеристикой $T_i^{ser}=0,3\,c,\ i=\overline{1,N}$. Установим $T_{\text{доп}}$=0,35 с.

Применим процедуру (3)–(7) к такой структуре кластера. На рис. 3 приведен график, показывающий процесс насыщения системы – с каждой новой заявкой значение ε приближается к условию 0,9< ε<1.

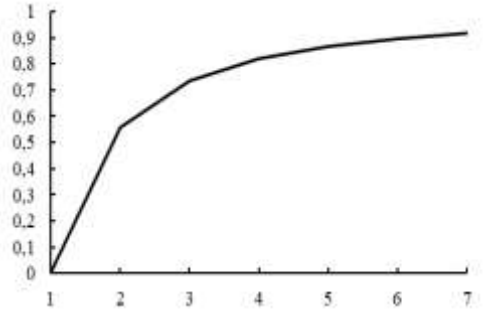

Рис. 3. Процесс насыщения системы

В табл. I приведены значения времени обработки запроса клиента, пропускная способность вычислительного кластера, число заявок, одновременно обрабатываемых в кластере, и средняя длина очереди к узлам кластера.

Как видно из результатов оптимизации структуры кластера – числа узлов, время обработки запросов при N=3 составило 0,5 с, что не соответствует $T_{\text{доп}}$. Увеличение числа узлов кластера до 10-ти позволяет сократить время отклика на запрос пользователя и удовлетворить условию (9).

ТАБЛИЦА I. РЕЗУЛЬТАТЫ СТРУКТУРНОЙ ОПТИМИЗАЦИИ КЛАСТЕРА

N	T	Λ	J	L
3	0,5	14,5	7	2,3
4	0,431	18,5	8	2
5	0,384	20,8	8	1,6
6	0,367	24,5	9	1,5
7	0,355	28,2	9	1,43
8	0,349	29,2	10	1.32
9	0,341	30,1	10	1,21
10	0,327	30,6	10	1

B. Определение числа реплик

Для случайно поступающих запросов на запись вероятность пересечения двух или более запросов равна

$$P_w = 1 - e^{-\lambda T_w}, \tag{10}$$

где λ – реальная интенсивность поступления запросов на обновление данных в интервале времени T_w.

Определим максимальное количество узлов, участвующих в записи R – число реплик, при котором при заданной интенсивности поступления запросов на обновление можно избежать критического уровня конфликта «запись-запись».

$$R\lambda T_w = \frac{1}{e}, \tag{11}$$

где слева в (11) максимальная производительность ВЦ.

Тогда

$$R = \left(e\lambda T_w\right)^{-1}, \tag{12}$$

В табл. II приведены результаты по выбору числа реплик при разной интенсивности запросов на запись (обновление) данных и размера окна несогласованности.

ТАБЛИЦА II. РЕЗУЛЬТАТЫ ПО ВЫБОРУ ЧИСЛА РЕПЛИК

R	λ	T_w
8	0,05	1
5	0,08	1
4	0,1	1
2	0,3	1
1	0,5	1
5	0,05	1,5
4	0,08	1,5
3	0,1	1,5
1	0,3	1,5
1	0,5	1,5

Таким образом, для снижения риска сериализации запросов можно настраивать частоту обновление данных или при необходимости соблюдения интенсивности обновления данных – оценить число реплик, позволяющих не выходить за пределы окна несогласованности.

III. ЗАКЛЮЧЕНИЕ

Рассмотрены известные способы распределенного хранения больших данных на кластерах: фрагментация и репликация.

В части решения задачи фрагментации предложена рекуррентная процедура и методика выбора оптимальной структуры кластера.

В части решения задачи репликации предложена модель выбора числа реплик при возникновении конфликта «запись-запись» в схеме «ведущий-ведомый».

СПИСОК ЛИТЕРАТУРЫ

[1] Daniel T. Larose and Chantal D. Larose. Discovering knowledge in data: an introduction to data mining. John Wiley & Sons Limited, 2014. 336 p.

[2] Цвященко Е.В. Анализ согласованности базы данных NoSQL на этапе проектирования информационной системы // Информационные системы и технологии. 2015. №1. С. 74-83.

[3] Sovetov B.Ya., Tatarnikova T.M., Poymanova E.D. Storage scaling management model // Information and Control Systems. 2020. № 5(108). С. 43-49. DOI: 10.31799/1684-8853-2020-5-43-49.

[4] Татарникова Т.М., Вольский А.В. Оценка вероятностно-временных характеристик сетевых узлов с дифференциацией трафика // Информационно-управляющие системы. 2018. № 3 (94). С. 54-60. DOI: 10.15217/issn1684-8853.2018.3.54.

[5] Татарникова Т.М., Елизаров М.А. Процедура разрешения коллизий в RFID-системе // Известия высших учебных заведений. Приборостроение. 2017. Т. 60. № 2. С. 150-157. DOI: 10.17586/0021-3454-2017-60-2-150-157.