УДК 004.657

Е. В. Цвященко, аспирант, e-mail: eugene.tsviashchenko@gmail.com,
Московский государственный технический университет им. Н. Э. Баумана

Анализ адекватности модели согласования реплик в конечном счете в базах данных NoSQL

Анализируется модель согласования реплик в конечном счете в базах данных NoSQL. Описывается процесс подготовки и проведения натурного эксперимента в облаке для доказательства адекватности модели. Приводятся спецификации программ, с помощью которых осуществляется доступ к системе NoSQL, и программы обработки журналов. Часть полученных экспериментальным путем данных использовалась для адаптации модели, а другая часть — для оценки адекватности. Выполнен анализ адекватности модели.

Ключевые слова: *база данных NoSQL, согласованность в конечном счете, адекватность, адаптация, вероятность рассогласования, реплика*

Введение

Современные распределенные хранилища данных должны быть масштабируемыми, доступными и быстрыми. Эти системы, как правило, распределяют данные между различными машинами (и часто через центры обработки данных) и обеспечивают их тиражирование (репликацию) по двум причинам: во-первых, для обеспечения высокой доступности в случае отказа компонентов и, во-вторых, чтобы обеспечить высокую производительность обработки запросов на нескольких репликах. Такие системы баз данных, построенные на парадигме распределенных хранилищ "ключ/значение", получили название NoSQL (Not-Only-SQL) [1]. Но использование реплик порождают проблему чтения несогласованных данных из разных реплик. Важной характеристикой систем NoSQL является вероятность того, что в процессе распространения обновления данных по нескольким репликам поступит запрос на чтение старых записей. В работе [2] разработана модель оценки такой вероятности. Целью данной статьи является доказательство адекватности этой модели.

Поддержание требуемого уровня согласованности для каждой конкретной предметной области может регулироваться следующими параметрами:

- N — число узлов, на которые в конечном счете будет реплицирована запись (может быть с некоторой задержкой);
- W — число узлов, на которые данные должны быть фактически записаны перед тем, как пользователю (или приложению) будет отправлен ответ об успешном завершении операции (если $W < N$, то система все еще продолжает реплицировать данные на оставшиеся $N—W$ узлов);
- R — число узлов, от которых база данных ожидает ответа для успешного завершения чтения-записи [3].

В работе [4] вводятся понятия *строгой согласованности* записи базы данных (случай $W + R > N$) и *согласованности в конечном счете* ($W + R \leqslant N$).

Строгая согласованность позволяет всегда получать актуальную версию записи, но это приводит к большим задержкам ожидания завершения обновления W реплик базы данных и чтения записей из R реплик. Для широкого класса приложений это имеет решающее значение. Например, для фирмы Amazon дополнительные задержки в 100 мс привели к 1 % падению продаже. В то же время 500 мс задержки в поисковой системе Google привели к снижению трафика на 20 % [4].

Согласованность в конечном счете (КС-согласованность) существенно уменьшает время реакции системы, но в таких системах всегда присутствует определенная вероятность доступа к рассогласованным данным, так как реплики будут согласованы только через некоторое время распространения обновлений. В статье рассмотрен наиболее важный случай такой согласованности (когда $W = R = 1$), обеспечивающий наибольшее быстродействие по записи и чтению данных.

Итак, увеличение задержки выполнения операций записи и чтения данных (для строгой согласованности) связаны с большими экономическими

потерями, но вместе с тем, снижение задержки (для КС-согласованности) приводит к увеличению вероятности рассогласования данных: чем меньше реплик участвуют в запросе чтения, тем меньше гарантий получить самые последние актуальные данные. Приложения часто могут терпеть слабую согласованность, используя "осторожные" шаблоны проектирования, такие как компенсации (например, "воспоминания", "догадки" и "извинения"), ассоциативные и коммутативные операции (например, анализ сроков и журналов, предупреждения) [5]. Тем не менее, при оценке рисков возникает острая необходимость в моделировании согласования реплик и оценке вероятности рассогласования данных при чтении [4].

В статье анализируется адекватность модели КС-согласования реплик для случая $W = R = 1$, разработанной в работе [2]. Для доказательства адекватности был выполнен натурный эксперимент на кластере NoSQL Riak [6] размером до 25 узлов. В работе приводятся спецификации прикладных программ, разработанных для проведения эксперимента и анализа статистики. Часть полученных результатов использовалась для адаптации модели, а другая часть — для оценки ее адекватности.

Эксперименты проводили в облачной среде, когда пользователь не должен заботиться об инфраструктуре и операционной системе [7]. Главными преимуществами в использовании облаков в вычислениях являются: быстрое изменение конфигурации выделенных ресурсов, включая число узлов в компьютерном кластере; плата только за использование ресурсов. Пользователь может запускать на выполнение базы данных NoSQL с большим объемом необходимой оперативной памяти и места для хранения данных тогда, когда это необходимо.

Обзор публикаций по теме статьи

Несмотря на важность проблемы рассогласования реплик в кластерной архитектуре баз данных NoSQL [4, 8—11] научных публикаций по этой тематике немного в связи со сложностью теоретического решения задачи оценки показателей КС-согласования реплик. Это обусловлено необходимостью учета сложных механизмов репликации, а также параметров аппаратных ресурсов, задействованных в процессе тиражирования обновленных данных. Можно выделить пять следующих работ, близких данной тематике.

В работе [4] предлагается подход к количественному измерению показателей КС-согласованности через "вероятностно ограниченное устаревание" записи (пары <ключ, значение>). "Вероятностно ограниченное устаревание" оценивается следующими показателями: k — устаревание, t — видимость и (k, t) — устаревание.

Вероятность, что считанный из распределенного хранилища набор версий записей не будет содержать актуальную версию записи, является функцией, зависящей от времени. Однако формула

$$p = 1 - p_{sk} = 1 - \left(\frac{C_{N-W}^{R}}{C_{N}^{R}} \right)^{k}, \qquad (1)$$

определяющая вероятность того, что считанный из распределенного хранилища набор версий записей будет содержать версию из последних k-обновлений, не учитывает распространение обновлений (вероятность не зависит от времени) [4]. Здесь N, W, R — параметры, регулирующие требуемый уровень согласованности (см. введение).

В формуле

$$p = 1 - p_{skt} = 1 - \left(\frac{C_{N-W}^{R}}{C_{N}^{R}} + \sum_{c \in [W, N)} \frac{C_{N-c}^{R}}{C_{N}^{R}} [P_w(c+1, t) - P_w(c, t)] \right)^{k} \qquad (2)$$

авторы попытались учесть распространение обновлений во времени [4]. Но они предлагают получить функцию $P_w(c, t)$ с помощью имитационного моделирования или измерений, что, на наш взгляд, является нереальным (в работе [4] эта функция так и не была получена). В формулах (1), (2) не учитывается интенсивность запросов на чтение.

В работе [8] показатели КС-согласованности измерялись экспериментально. Эксперимент состоял в последовательном считывании записи до того момента, пока устаревшее значение не перестанет возвращаться. Разница между временем последнего считывания устаревшей версии пары <ключ/значение> и временем последнего обновления записи отражала окно рассогласованности.

Для экспериментов была разработана очень простая система хранения с уровнем репликации $N = 3$, $W = 1$, $R = 1$. Для наглядности была добавлена искусственная задержка перед реплицированием (тиражированием) данных, равная 1000 мс. Обновление выполнялось раз в 5 с, чтение — раз в 10 мс. Каждые 10 мин добавлялся читающий процесс, таким образом, их число в ходе эксперимента изменялось от 1 до 12.

Сравнивались размеры окон рассогласованности, полученные из журналов базы данных NoSQL, с результатами проведенных экспериментов. Из результатов сравнения видно, что наблюдаемое значение окна рассогласованности (полученное из экспериментов) никогда не превышает размера окна рассогласованности, ориентированного на данные (т. е. полученного из журналов). Также видно, что после некоторого числа читающих процессов трудно достичь высокой точности наблюдаемого значения.

Приводятся результаты измерений окна рассогласования реплик в облачной среде AMAZON S3 в зависимости от зоны обновления и чтения записей.

Модели не разрабатывались, поэтому не ясно, насколько сделанные выводы являются общими.

В статье [9] предлагается метод *LibRe* (Library For Replication) согласования данных в базах данных NoSQL. Метод основывается на КС-согласованности. Как известно, для КС-согласованного хранилища данных характерна некоторая задержка распространения изменений в базе данных (окно рассогласованности), в течение которой возможно чтение неактуальных записей. Но этот режим согласованности обеспечивает большую производительность системы. Задача состоит в том, чтобы добиться высокой согласованности данных при чтении в сочетании с высокой скоростью доступа к данным.

Подход заключается в следующем. Управляющий модуль LibRe располагается рядом с балансировщиком нагрузки. Балансировщик передает на вход LibRe для каждой операции обновления набор узлов, на которых будет выполнена данная операция. После выполнения операции LibRe хранит некоторое время информацию о том, какие узлы ее выполнили. Для каждой операции чтения LibRe возвращает набор узлов, где операция обновления уже выполнена.

Этот подход имеет существенные недостатки: не каждая база данных NoSQL позволяет встраивать подобный модуль; его работа увеличивает время выполнения операций обновления и чтения данных.

В статье [10] рассматривается проблема согласованности и долговечности в распределенных хранилищах данных через применение "bolt-on" слоя. Данный слой предполагает причинную согласованность — согласованность, основанную на связи (случилось — до). Данная согласованность является сильной (строгой). Операции приложения создают упорядоченные отношения между операциями (случилось — до), и система учитывает этот порядок. Таким образом, все операции записи описывают причинную историю.

Каузальные связи (причинные) чаще всего описываются в двух видах: потенциальная и явная причинная связь. В потенциальной каузальности все записи, которые могут повлиять на другую запись, должны быть видны до того, как станет видна последняя запись. Явная причинная связь создается напрямую в приложении.

В статье предлагается архитектура bolt-on: на каждую клиентскую машину предлагается ввести прослойку (shim), содержащую метаданные и локальное хранилище. Каждый клиент обращается к данной прослойке, которая в свою очередь обращается к КС-согласованному хранилищу данных. Данный слой ограничивает нарушения согласованности, которые может "увидеть" клиент. Хранилище данных ответственно за распределение процессов записи: для того чтобы прочитать новые версии других процессов, каждый shim отправляет соответствующий запрос к хранилищу данных.

Локальное хранилище содержит согласованный набор записей, с которыми могут проводиться операции чтения и добавления в любое время без нарушения ограничений безопасности. Данный набор записей обозначается как "причинная вырезка данных" $x1 \to y1 \to z1$ (causal sat). Пусть другой клиент сохранил запись $y2 \to$ NULL. При поиске последней версии записи $y2$ прослойка shim должна убедиться, что в хранилище была запись $y1$. Иначе причинная связь $x1 \to y2$ будет утеряна. В статье описаны алгоритмы работы с shim, а также процедуры проверки целостности причинных связей.

Такой подход можно использовать для узкого круга приложения, например, для уточнения целей при поиске данных. КС-согласованность не рассматривалась.

В статье [11] предлагается метод обеспечения причинной согласованности со сходящейся обработкой конфликтов (CAUSAL + CONSISTENCY). Данный вид согласованности является расширением классической причинной согласованности. Классический метод причинной связи не рассматривает конкурирующие операции обновления, т. е. операции работы над одним ключом, например, put(x, a) и put(x, b). Потенциальной причинной связи между этими операциями нет. Следовательно, в КС-согласованной системе возможны конфликты и рассогласования при определенных условиях. Конфликты нежелательны по двум причинам: во-первых, реплики могут логически "разойтись"; во-вторых, полученные расхождения могут логически исключать друг друга, что требует специальной обработки. Предложенное в статье [11] расширение классического метода заключается во введении *сходящейся обработки конфликтов*. Обработкой конфликтов занимается специальная функция h, запускаемая для каждой реплики. Функция должна быть коммутативной и ассоциативной. Сходимость заключается в равенстве результатов выполнения функций $h(a, h(b, c))$ на одной реплике и $h(c, h(b, a))$ — на другой, где $a \to b \to c$ — некоторая последовательность операций обновлений и чтений (put и get).

Преимущества метода: усиление согласованности данных за счет введения коммутативной и ассоциативной функций обработки конфликтов.

Недостатки: обнаружение конфликтов является сложной задачей, решение которой вносит существенные задержки в работу системы. Например, одним из трех компонентов системы обнаружения конфликтов является введение явной причинной связи между операциями put текущей и предыдущей версий записи, что требует выполнения дополнительной операции dep_check (проверка зависимости).

Из приведенного анализа следует, что задача разработки новой модели анализа рассогласования реплик является актуальной.

В публикации [2] разработана модель КС-согласования реплик ($W = R = 1$). Будем считать, что обновлена $(N + 1)$-я реплика и изменения (обновления) распространяются на остальные реплики 1, ..., N. Из процесса обновления реплик, описанного в документации NoSQL системы Riak [6], можно сделать вывод о синхронном характере распространения изменений. Для этого режима была получена вероятность того, что в процессе обновления реплик поступит хотя бы одно требование на чтение из необновленных реплик:

$$P = (1 - Q_1(0)) + \sum_{i=2}^{N} ((1 - Q_i(0)) \prod_{j=1}^{i-1} Q_j(0));$$
$$Q_{i+1}(z) = \psi_{i+1}(\lambda(N-i)(1-z)),\ i = 0...(N-1). \quad (3)$$

Здесь $N + 1$ — общее число реплик для обновляемой записи (изменение записи выполняется на $(N + 1)$-й реплике); λ — интенсивность чтения записи из каждой реплики (1, ..., N), на которую распространяются обновления; $\Psi_{i+1}(s)$ — преобразование Лапласа—Стилтьеса (ПЛС) времени обновления $(i + 1)$-й реплики. ПЛС имеет достаточно сложный вид [2], поэтому модель (3) нуждается в проверке адекватности.

Обозначим $C = \{K, V, \mu_{ns}, \mu_p\}$ — множество измеренных параметров системы, $X = \{N, \lambda\}$ — настраиваемые параметры, т. е. параметры, которые меняются от одного эксперимента к другому; $Y = \{\mu_n, \mu_m, \mu_{do}, \text{PRT}\}$ — адаптируемые параметры модели.

Здесь K — размер ключа записи; V — размер значения записи; μ_{ns} — интенсивность передачи данных по шине сети, соединяющей подсети (байт/с); μ_p — число процессорных циклов в секунду; N — число реплик, на которые распространяются обновления; λ — интенсивность чтения записи из каждой реплики 1, ..., N (1/с); μ_n — интенсивность передачи данных по шине локальной сети (байт/с); μ_m — интенсивность чтения данных из оперативной памяти (байт/с); μ_{do} — интенсивность ввода/вывода данных на диск (байт/с); PRT — число итераций, необходимых для вычисления хеша ключа.

Описание экспериментальной установки

Рассмотрим некоторые особенности развертывания компьютерного кластера в облачной среде применительно к решению задачи оценки адекватности модели КС-согласования реплик. На рынке существует большое число компаний, предоставляющих облачные вычислительные ресурсы. Ресурсы бывают виртуальными и выделенными. Выделенными облачными ресурсами (узлами) являются узлы, которым соответствуют физически отдельные серверы. В отличие от выделенных узлов, виртуальные узлы — это виртуальные машины. Аренда выделенных серверов намного дороже, поэтому в нашем эксперименте использовались виртуальные узлы, предоставленные компанией *DigitalOcean* (DO) [12]. Так как веррятность доступа к рассогласованным данным оценивалась при работе пользователя с одной записью <ключ/значение>, база данных NoSQL не нагружала оперативную память, что позволило арендовать недорогие виртуальные серверы с малым объемом оперативной памяти.

Все виртуальные узлы, предоставляемые поставщиком облачных ресурсов, используют ресурсы многопроцессорных машин с SSD дисками. Это позволяет сделать предположение, что другие клиенты DO, которым предоставлены виртуальные узлы на том же физическом сервере, не будут нагружать именно тот процессор, на котором запущена наша виртуальная машина. Следовательно, производительность нашего виртуального узла существенно не зависит от фоновой загрузки процессора, что позволяет отнести параметр μ_p (число процессорных циклов, выполняемых в секунду) к множеству измеренных параметров системы (т. е. к C). При инициализации узла доступно несколько опций, среди которых можно выделить опцию *private networking*. При включении данной опции все узлы, арендованные пользователем, гарантированно находятся в одном центре обработки данных (ЦОД), что означает отсутствие подкластеров сети. Следовательно, параметр μ_{ns} — интенсивность передачи данных по сети, соединяющей подсети, — можно не учитывать.

При первоначальной настройке узла (*Droplet*) необходимо выбрать операционную систему (ОС) или снимок, ранее созданный пользователем. Использовалась ОС Ubuntu Server 14.04 [13], предустановленная поставщиком. В качестве системы NoSQL была использована база данных Riak. Для установки и настройки Riak необходимо выполнить ряд действий на каждом из $N + 1$ узлов. Установка базы данных "с нуля" требует много времени. Поэтому для ускорения подготовки кластера общая часть действий по установке системы, описанных в [6], была выполнена один раз на одном узле, далее был сделан снимок узла, который впоследствии был растиражирован на остальные узлы. Индивидуальная часть действий по настройке Riak, описанная в [6], была сохранена в качестве *bash*-скрипта. Данные скрипты являются *Linux* аналогом *bat*-скриптов для ОС *Windows*. После установки и настройки системы все узлы были соединены в кластер, для чего предусмотрены следующие команды Riak:

1) riak-admin cluster join riak@<ip_first_node>;
2) riak-admin cluster plan;
3) riak-admin cluster commit.

Команда 1 выполнялась на каждом узле, кроме одного, к которому присоединены остальные узлы (*ip_first_node*). После выполнения команды 1 необходимо было проверить конфигурацию кластера на любом из узлов командой 2, после чего сохранить изменения командой 3. После выполнения всех команд кластер был готов к работе, однако потребовалось некоторое время для переноса уже хранящихся данных из первого узла на все остальные. В течение этого времени система может отвечать *not found* на запросы клиентов. Так как исследуется КС-согласованность, настройки согласованности для каждого эксперимента задавались следующим образом: ($N + 1$, $W = 1$, $R = 1$).

Подготовка эксперимента

В модели входящий поток требований на чтение к одной реплике принимался пуассоновским с параметром λ. Для того чтобы эксперимент соответствовал модели, требования на чтение должны поступать независимо к каждой из 1, ..., N реплик с интенсивностью λ. К $(N + 1)$-му узлу поступают требования на изменение записи с интенсивностью 1,25 (1/с). Для выполнения последовательного чтения и записи были разработаны соответствующие прикладные программы. База данных Riak предоставляет библиотеки для доступа к системе на языках Java, Erlang, Pyton, Ruby, из которых была выбрана библиотека Java. Java-приложения транслируются в промежуточный байт-код, который исполняется на любой виртуальной машине, что облегчило отладку приложения. Все процессы чтения и записи выполнялись непосредственно на узлах. Ниже приведен алгоритм процесса записи, выполняемого на узле $N + 1$.

Вход: N_ITER_W — число итераций, KEY — ключ обновляемой записи, REC_VAL — счетчик версии записи (значение).

Алгоритм:

```
REC_VAL = 1
ЦИКЛ по N_ITER_W
    TIME_STAMP = CURR_TIME
    ЗАПИСАТЬ в БД пару <KEY, REC_VAL>
    ЗАПИСАТЬ в журнал CURR_TIME
    ЗАПИСАТЬ в журнал REC_VAL
    REC_VAL + = 1
    DELAY = EXPONENTIAL(1.25)
    DELAY -= (CURR_TIME — TIME_STAMP)
    ЗАДЕРЖАТЬ выполнение на время DELAY
КОНЕЦ ЦИКЛА
```

Функция CURR_TIME возвращает текущее время системы в миллисекундах. Уменьшение DELAY компенсирует время выполнения алгоритма. При проведении подобных экспериментов в кластере возникает вопрос синхронизации часов между виртуальными узлами. Эту проблему решает поставщик, устанавливая единое время при инициализации узлов.

Ниже приведен алгоритм процессов чтения, выполняемых на узлах 1, ..., N.

Вход: λ — интенсивность поступления требований на чтение, KEY — ключ обновляемой записи, N_ITER_W — число итераций процесса записи.

Алгоритм:

```
N_ITER_R = N_ITER_W / 1.25 · λ
ЦИКЛ по N_ITER_R
    TIME_STAMP = CURR_TIME
    СЧИТАТЬ из БД пару <KEY, REC_VAL>
    ЗАПИСАТЬ в журнал CURR_TIME
    ЗАПИСАТЬ в журнал REC_VAL
    DELAY = EXPONENTIAL(λ)
    DELAY -= (CURR_TIME — TIME_STAMP)
    ЗАДЕРЖАТЬ выполнение на время DELAY
КОНЕЦ ЦИКЛА
```

После завершения процессов чтения и записи на каждом узле журналы должны быть скопированы на локальную машину для расчета экспериментальной вероятности. Ниже приведен алгоритм программы оценки вероятности.

Вход: LOG_W — файл журнала процесса записи, LOGS_R — файлы журналов процессов чтения, N_ITER_W — число итераций записи.

Алгоритм:

```
COUNT = 0
ДЛЯ каждой записи <TIME_W, REC_VAL_W>
    из журнала LOG_W
    ЕСЛИ ∃ < TIME_R, REC_VAL_R > ∈
    LOGS_R:TIME_R >= TIME_W □
    REC_VAL_R < REC_VAL_W,
    ТО COUNT + = 1
КОНЕЦ ДЛЯ
ВЕРНУТЬ COUNT / N_ITER_W
```

Увеличение переменной COUNT на единицу объясняется следующим: после того как завершена операция записи в одну реплику, поступило хотя бы одно требование на чтение из еще необновленных реплик.

Проведение эксперимента

Перед проведением эксперимента необходимо определить значение переменной N_ITER_W для каждого из экспериментов. Для этого воспользуемся теоремой Ляпунова [14]. Известно, что функция распределения случайной величины $\frac{\sum_{i=1}^{k} \xi_i}{k} - M(\xi)$

стремится к нормальному закону при увеличении объема выборки k. Поэтому

$$P\left(\left|\frac{\sum_{i=1}^{k}\xi_i}{k} - M(\xi)\right| < \delta\right) \approx \gamma = 2\Phi(x) = 2\frac{1}{\sqrt{2\pi}}\int_0^x \mathbf{e}^{-\frac{z^2}{2}}\,dz,$$

$$x = \frac{\delta\sqrt{k}}{\sigma},\ \delta = \frac{x\sigma}{\sqrt{k}}, \qquad (4)$$

где σ — оценка среднеквадратического отклонения случайной величины ξ_i; величина x зависит от надежности γ: $x|_{\gamma=0,95} = 1{,}96$, $x|_{\gamma=0,99} = 2{,}58$, $x|_{\gamma=0,999} = 3{,}29$. Случайная величина ξ_i определяется на i-й итерации обновления записи так: если после обновления этой записи хотя бы один клиент прочитает старую версию записи, то $\xi_i = 1$, иначе $\xi_i = 0$. Эти величины распределены по одинаковому закону, поэтому математическое ожидание $M(\xi_i) = M(\xi)$ — это и есть оцениваемая вероятность P (см. (3)). Так как случайная величина ξ_i принимает значение 1 с вероятностью P и 0 с вероятностью $1 - P$, то

$$\sigma = \sqrt{P(1-P)}. \qquad (5)$$

Пусть необходимо получить точность оценки вероятности P с надежностью $\gamma = 0{,}95$. Тогда из (4) с учетом (5) получим

$$k \geqslant (x\sigma/\delta)^2 = \frac{1{,}96^2 P(1-P)}{\delta^2} = \frac{3{,}84P(1-P)}{\delta^2}. \qquad (6)$$

Рассчитаем объем выборки k по выражению (6):

1) пусть $P \sim 10^{-3}$, $\delta = 0{,}001$, тогда $k \geqslant 3840$;

2) пусть $P = 2{,}5 \cdot 10^{-2}$, $\delta = 0{,}005$, тогда $k \geqslant 3744$.

Основываясь на приведенных выше расчетах, объем выборки для всех экспериментов можно принять равным 4000.

Были проведены две серии экспериментов. В табл. 1 приведены значения настраиваемых параметров X (N и λ) и результаты первой серии экспериментов.

В табл. 2 приведены значения настраиваемых параметров X и результаты второй серии экспериментов.

Анализ результатов эксперимента

1. Адаптация модели КС-согласованности реплик

Значения общих параметров множества C для всех серий экспериментов представлены ниже:

- $K = 20$ байт — размер ключа изменяемой записи;
- $V = 1024$ байт — размер значения изменяемой записи;
- $\mu_{ns} = 0$ (т. е. не учитывался) — интенсивность передачи данных по сети, соединяющей подсети;
- $\mu_p = 2400 \cdot 10^6$ — число операций, выполняемых в секунду процессором Intel Xeon CPU E5-2630L v2.

Задача адаптации модели решалась методом наименьших квадратов по схеме

$$\sum_{i=1}^{L}(P(C, X_i, Y) - Z_i)^2 \xrightarrow[Y]{} \min,$$

где функцию $P(C, X_i, Y)$ определяет формула (3); Z_i — вероятность, полученная при проведении i-го эксперимента; L — число экспериментов, по которым проводилась адаптация модели (они отмечены в табл. 1 и 2 серым цветом. Остальные эксперименты были использованы при оценке адекватности модели (3)). Оптимальные значения адаптируемых параметров Y были получены методом наискорейшего спуска [15].

Эксперименты проводились в два подхода, в разное время, следовательно, при разной фоновой загрузке ресурсов. Фоновая загрузка узлов зависит от работы других клиентов облачных ресурсов и может меняться время от времени. Поэтому адаптация модели проводилась отдельно для первой и второй

Таблица 1

Результаты первой серии экспериментов

N	λ	Оценка вероятности	N	λ	Оценка вероятности
3	10	0,025	16	3	0,192
	15	0,039		5	0,325
	20	0,065		10	0,539
	30	0,082		15	0,692
8	10	0,174	24	3	0,228
	15	0,223		5	0,375
	20	0,324		7	0,471
	30	0,415		10	0,611

Таблица 2

Результаты второй серии экспериментов

N	λ	Оценка вероятности	N	λ	Оценка вероятности
4	16	0,118	6	5	0,092
	18	0,126		7	0,130
	20	0,155		10	0,163
	22	0,176		12	0,203
	24	0,185		15	0,223
	26	0,201		17	0,274
	28	0,221		20	0,330
	30	0,280		22	0,358

Таблица 3

Адаптируемые параметры

Первая серия экспериментов		Вторая серия экспериментов	
Параметр	Значение	Параметр	Значение
μ_n	35,1 Мбит/с	μ_n	18,6 Мбит/с
μ_m	8330 Мбайт/с	μ_m	6440 Мбайт/с
μ_{do}	215 Мбайт/с	μ_{do}	215 Мбайт/с
PRT	16	PRT	16

серии экспериментов. После решения вариационной задачи были получены адаптируемые параметры, представленные в табл. 3.

2. Анализ адекватности модели КС-согласованности реплик

В табл. 4 представлены результаты натурных экспериментов, которые были использованы при анализе адекватности модели (см. в табл. 1 и 2 неотмеченные строки), а также результаты соответствующих модельных экспериментов.

Средняя относительная погрешность (за исключением максимального и минимального значений) по двум сериям экспериментов составила 9,4 %, а для некоторых экспериментов получена погрешность меньше 1 %. На рис. 1 и 2 показаны зависимости вероятности рассогласования P от λ при различных значениях N, построенные по результатам натурных и модельных экспериментов. На рисунках "мод" и "экс" обозначают графики, построенные соответственно по модели и экспериментальным данным. Из рис. 1 видно, что погрешность увеличивается с ростом N. Это можно объяснить увеличением влияния фоновой нагрузки. В реальной ситуации значение N редко устанавливают выше 10, поэтому большая погрешность при $N = 24$ не является критичной.

Полученное при адаптации модели значение интенсивности передачи байта данных внутри сегмента сети μ_n (см. табл. 3) означает, что сеть сильно перегружена (например, другими клиентами сети). На рис. 3 показано, какой была бы зависимость вероятности P от интенсивности λ, если сеть не была бы пе-

Таблица 4

Анализ адекватности модели

N	λ	Вероятность		Относительная погрешность, %
		Эксперимент	Модель	
3	15	0,039	0,045	15,4
	20	0,065	0,059	9,23
8	15	0,223	0,241	7,72
	20	0,324	0,308	4,94
16	5	0,325	0,294	9,54
	10	0,539	0,501	7,05
24	5	0,375	0,535	42,0
	7	0,471	0,658	39,7
4	16	0,118	0,135	14,4
	20	0,155	0,165	6,45
	24	0,185	0,195	5,41
	28	0,221	0,224	1,36
6	7	0,130	0,124	4,62
	12	0,203	0,204	0,50
	17	0,274	0,276	0,73
	22	0,358	0,341	4,75

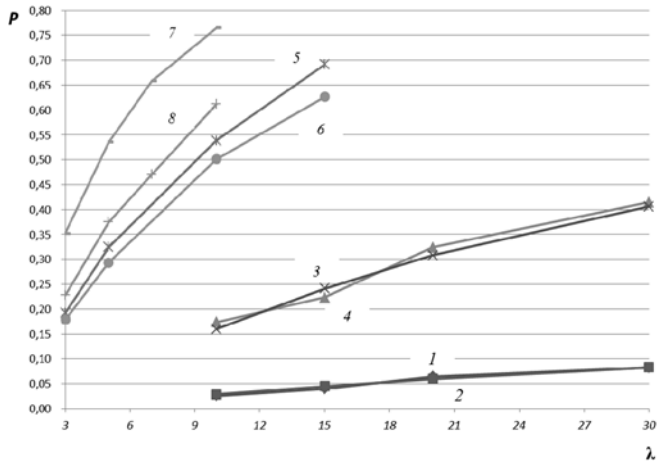

Рис. 1. Зависимости вероятности поступления хотя бы одного требования на чтение в процессе обновления N реплик от λ для первой серии экспериментов:
1 — N = 3_мод, *2* — N = 3_экс, *3* — N = 8_мод, *4* — N = 8_экс, *5* — N = 16_экс, *6* — N = 16_мод, *7* — N = 24_мод, *8* — N = 24_экс

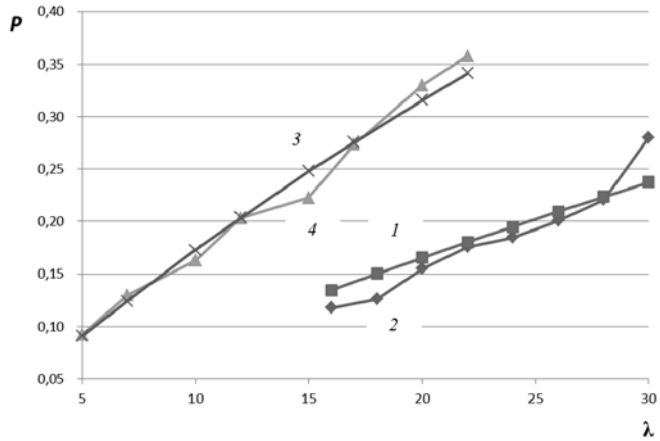

Рис. 2. Зависимости вероятности поступления хотя бы одного требования на чтение в процессе обновления N реплик от λ для второй серии экспериментов:
1 — N = 4_мод, *2* — N = 4_экс, *3* — N = 6_мод, 4 — N = 6_экс

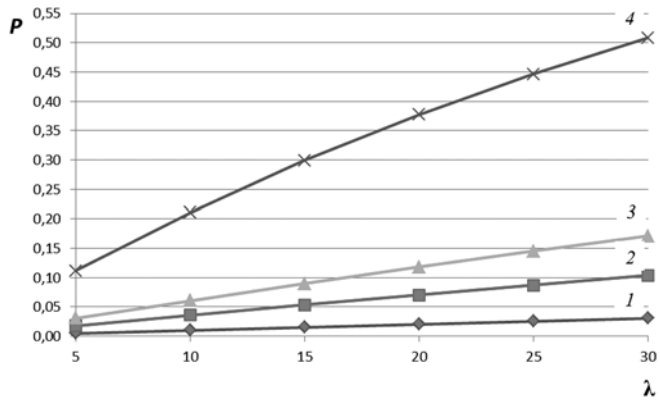

Рис. 3. Зависимости поступления хотя бы одного требования на чтение в процессе обновления N реплик от λ для μ_n = 100 Мбит/с:
1 — N = 3_мод, *2* — N = 6_мод, *3* — N = 8_мод, *4* — N = 16_мод

регружена — например, при μ_n = 100 Мбит/с (расчеты выполнены на модели). Вероятность рассогласования снижается более чем в 2 раза.

В работе [4] приведено выражение (1), согласно которому вероятность, что считанная из распределенного хранилища версия записи не будет актуальной (т. е. последней обновленной, $k = 1$), равна (для случая $W = R = 1$):

$$P = 1 - \frac{1}{N+1}, \qquad (7)$$

где $N + 1$ — число реплик записи.

Значения этой вероятности представлены в табл. 5 для разных N.

Таблица 5

Значение вероятности P по формуле (7)

N	3	4	6	8	16	24
P	0,750	0,800	0,857	0,889	0,941	0,960

Эти значения никак не соответствуют экспериментальным данным (см. рис. 1, 2). Объясняется это тем, что формула (7) является упрощенной и не учитывает ни механизма распространения обновлений по репликам, ни интенсивности чтения записей из этих реплик. Разработанная модель (3) не имеет указанных недостатков, а также учитывает параметры аппаратных ресурсов (оперативной памяти, диска, сети и процессора), задействованных в процессе тиражирования обновленных данных [2].

Заключение

По результатам натурного эксперимента, выполненного с базой данных NoSQL Riak в облачной среде компании *Digital Ocean* на кластере с числом узлов до 25, доказана адекватность разработанной модели (3) (см. табл. 4). Средняя относительная погрешность составила 9,4 %. Сильное расхождение модели с экспериментом наблюдается только при $N = 24$, но подобное значение N редко встречается в реальных системах.

Задача адаптации модели (3) решалась методом наименьших квадратов. При этом была использована только часть полученных экспериментальных данных, другая часть использовалась для доказательства адекватности модели.

Приведены алгоритмы работы прикладных программ для проведения натурных экспериментов для режима КС-согласования реплик и оценки вероятности того, что за время обновления N реплик поступит хотя бы одно требование на чтение из необновленных реплик (т. е. вероятности чтения неактуальных данных). На основе теоремы Ляпунова обоснован объем выборки, необходимый для достижения надежности оценки, равной 0,95.

Список литературы

1. **NoSQL.** [Электронный ресурс]. URL: http://ru.wikipedia.org/wiki/NoSQL (дата обращения 09.03.2015).

2. **Григорьев Ю. А., Цвященко Е. В.** Анализ характеристик согласования реплик в конечном счете в базах данных NoSQL // Информатика и системы управления. 2014. № 3. С. 3—11.

3. **Редмон Э., Уилсон Д. Р.** Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL. М.: ДМК Пресс, 2013. 384 с.

4. **Bailis P., Venkataraman Sh., Franklin M. J., Hellerstein J. M., Stoica I.** Probabilistically Bounded Staleness for Practical Partial Quorums, 2012: [Электронный ресурс]. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-4.pdf (дата обращения 09.03.2015).

5. **Alvaro P., Conway N., Hellerstein J. M., and Marczak W. R.** Consistency analysis in Bloom: a CALM and collected approach // CIDR, 2011. P. 249—260.

6. **Riak** documentation. [Электронный ресурс]. URL: http://docs.basho.com/index.html (дата обращения 09.03.2015).

7. **Романова А. О.** Научные вычисления в облаках // Молодежный научно-технический вестник. Электронный журнал. № 9, 2012. URL: http://sntbul.bmstu.ru/doc/479373.html.

8. **Bermbach D., Tai S.** Eventual Consistency: How soon is eventual? // ACM MW4SOC '11, December 12, 2011, Lisboa, Portugal. [Электронный ресурс]. URL: http://dl.acm.org/citation.cfm?id=2093186. Проверено 22.03.2014.

9. **Kumar S. P., Chiky R., Lefebvre S., Soudan E. G.** LibRe: A Consistency Protocol for Modern Storage Systems // ACM COMPUTE'13 Aug 22—24, 2013, Vellore, Tamil Nadu, India.

10. **Bailis P., Chodsi A., Hellerstein J. M., Stoica I.** Bolt-on Causal Consistency // ACM SIGMOD'13, June 22—27, 2013, New York, 2013. P. 761—772.

11. **Lloyd W., Freedman M. J., Kaminsky M., Andersen D. G.** Don't Settle for Eventual: Scalable Causal Consistency for Wide-Area Storage with COPS // ACM SOSP' 11, October 23—26, 2011, Cascais, Portugal. — pp. 401—416.

12. **Digital** Ocean. [Электронный ресурс]. URL: https://www.digitalocean.com (дата обращения 09.03.2015).

13. **Ubuntu** OS 14.04. [Электронный ресурс]. URL: http://releases.ubuntu.com/14.04 (дата обращения 09.03.2015).

14. **Теорема** Ляпунова. [Электронный ресурс]. URL: режим доступа: https://ru.wikipedia.org/wiki/Теорема_Ляпунова (дата обращения 09.03.2015).

15. **Метод** наименьших квадратов. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Метод_наименьших_квадратов (дата обращения 09.03.2015).

16. **Cap** Theorem. [Электронный ресурс]. URL: режим доступа: http://en.wikipedia.org/wiki/CAP_theorem (дата обращения 09.03.2015).

17. **Григорьев Ю. А.** Анализ свойств баз данных NoSQL // Информатика и системы управления. 2013. № 2. С. 3—13.

18. **Григорьев Ю. А., Цвященко Е. В.** Сильная и слабая согласованность в базах данных NoSQL // Информатика и системы управления. 2014. № 4. С. 14—23.

E. V. Tsviashchenko, Postgraduate Student, e-mail: eugene.tsviashchenko@gmail.com
Bauman Moscow State Technical University

Adequacy Analysis of the Model of Replicas Agreement Eventually in NoSQL Databases

In this paper the model of replicas agreement eventually in NoSQL databases is analyzed. With the model, we can estimate the probability, that in process of updating N replicas, at least one read request from stale replicas will received. In order to proof model adequacy the processes of preparation and conduct of the experiment in the cloud are described. For experiment were used virtual cluster with up to twenty five nodes, provided by Digital Ocean company. The specifications of the programs that provides for access to the NoSQL database and log processing are given. After completion of the experiments the one part of obtained results was used for model adaptation and the second part — for adequacy evaluating. Adaptive parameter "intensity data vithin the network segment" value was calculated as near 35 Mb/s for the first experriments series and ner 18 Mb/s for the second means that the cloud network is busy. The analysis showed, that average relative error accounted for 9 percent. Strong disagreement with the experimental model is only observed at N = 24, but like the value of N is rare in real systems. Finally, we consider how he probabilities would change if the network was not busy.

Keywords: *NoSQL database, eventually consistency, adequacy, adaptation, inconsistency probability, replica*

References

1. **NoSQL.** [Electronic resource]. URL: http://ru.wikipedia.org/viki/NoSQL (accessed 09.03.2015).

2. **Grigoriev Y. A., Tsviashchenko E. V.** Analis haractenstik soglasovannosti replic v konechnom schete v basah dannix NoSQL, *Informatika i sistemi ypravlenia*, 2014, no. 3, pp. 3—11.

3. **Redmon E., Yilson D. R.** *Sem bas dannix za sem nedel. Vvedenit v sovremennie basi dannix i ideologiju NoSQL*, Moscow: DMK Press, 2013. 384 p.

4. **Bailis P., Venkataraman Sh., Dranklin M. J., Hellerstein J. M., Stoica I.** Probabilistically Bounded Staleness for Practical Quorums, 2012. [Electronic resource]. URL: http://www.eecs.berkeley.edu/Pubs/TechI pts/2012/EECS-2012-4.pdf (accessed 09.03.2015).

5. **Alvaro P., Conway N., Hellerstein J. M., and Marczak W. R.** Consistency analysis in Bloom: a CALM and collected approach, *CIDR*, 2011, pp. 249—260.

6. **Riak** documentation. [Electronic resource]. URL: http://docs.basho.com/index.html (accessed 09.03.2015).

7. **Romanova A. O.** Nauchnie vichislenia v oblakax, *Molodezhniy nauchno-tehnicheskiy vestnik*, Electronic Journal, 2012, no. 9, URL: http://sntbul.bmstu.ru/doc/479373.html

8. **Bermbach D., Tai S.** Eventual Consistency: How soon is eventual? *ACM MW4SOC '11, December 12, 2011, Lisboa, Portugal.* [Electronic resource], URL: http://dl.acm.org/citation.cfm?id=2093186 (accessed 22.03.2014).

9. **Kumar S. P., Chiky R., Letebvre S., Soudan E. G.** LibRe: A Consistency Protocol for Modern Storage Systems. *ACM COMPUTE '13,* Aug 22—24, 2013, Vellore, Tamil Nadu, India.

10. **Bailis P., Ghodsi A., Hellerstein J. M., Stoica I.** Bolt-on Causal Consistency. *ACM SIGMOD '13*, June 22—27. New York. 2013, pp. 761—772.

11. **Lloyd W., Freedman M. J., Kaminsky M., Andersen D. G.** Don't Settle for Eventual: Scalable Causal Consistency for Wide-Area Storage with COPS, *ACM SOSP '11*, October 23—26, 2011, Cascais, Portugal, 2011, pp. 401—416.

12. **Digital** *Ocean*. [Electronic resource], URL: https://www.digitalocean.com (accessed 09.03.2015).

13. **Ubuntu** *OS* 14.04. [Electronic resource], URL: http://releases.ubuntu.com/14.04 (accessed 09.03.2015).

14. **Lapunov** *theorem*. [Electronic resource], URL: https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0_%D0%9B%D1%8F%D0%BF%D1%83%D0%BD%D0%BE%D0%B2%D0%B0 (accessed 09.03.2015).

15. **Least** *squares*. [Electronic resource]. URL: https://en.wikipedia.org/wiki/Least_squares (accessed 09.03.2015).

16. **Cap** *Theorem*. [Electronic resource], URL: http://en.wikipedia.org/wiki/CAP_theorem (accessed 09.03.2015).

17. **Grigoriev Y. A.** Analiz svojstv baz dannix NoSQL, *Informatika i sistemi ypravlenija*, 2013, no. 2, pp. 3—13.

18. **Grigoriev Y. A., Tsviashchenko E. V.** Silnaya i slabaya soglasovannost v basah dannih NoSQL, *Informatika i sistemi ypravlenija*, 2014, no. 4, pp. 14—23.