**КОМПЬЮТЕРНЫЕ ИССЛЕДОВАНИЯ
И МОДЕЛИРОВАНИЕ 2016 Т. 8 № 1 С. 101−112**

Ки&М

МОДЕЛИ В ФИЗИКЕ И ТЕХНОЛОГИИ

УДК: 004.657

Анализ адекватности модели строгого согласования реплик в базах данных NoSQL

Е. В. Цвященко

Московский государственный технический университет им. Н. Э. Баумана,
Россия, 105005, г. Москва, ул. 2-ая Бауманская, д. 5

E-mail: eugene.tsviashchenko@gmail.com

*Получено 2 июля 2015 г.,
после доработки 8 декабря 2015 г.*

В статье анализируется модель сильного согласования реплик в базах данных NoSQL. Описывается процесс подготовки и проведения натурного эксперимента в облаке для доказательства адекватности модели. Приводятся спецификации программ, с помощью которых производился доступ к NoSQL-системе? и программы обработки журналов. Часть из полученных экспериментальным путем данных использовалась для адаптации модели, другая часть — для оценки адекватности. Приводится анализ адекватности модели.

Ключевые слова: база данных NoSQL, сильная согласованность, адекватность, адаптация, время ожидания, преобразование Лапласа–Стилтьеса

Adequacy analysis the model of strong replicas agreement in NoSQL databases

E.V. Tsviashchenko

Bauman moscow state technical university, 5 2-nd Bauman st., Moscow, 105005, Russia

Abstract. — In this article the model of strong replicas agreement was analyzed. The process of preparing and conducting the nature experiment in the cloud in order to proof the model adequacy was described. Specifications of the program for implementation of database access to the NoSQL system and the program for handling journals were presented. One part of obtained experiments results was used for model adaptation, another part — for adequacy evaluating. The adequacy analysis is presented.

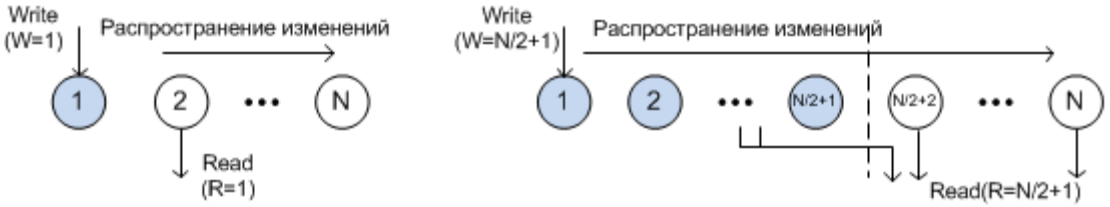
Keywords: NOSQL database, strong consistency, adequacy, adaptation, waiting time, Laplas–Stilties transform

Citation: *Computer Research and Modeling*, 2016, vol. 8, no. 1, pp. 101–112 (Russian).

Введение

Для повышения производительности и отказоустойчивости автоматизированных информационных систем (АИС) в настоящее время все чаще используются системы баз данных, построенные на парадигме распределенных хранилищ «ключ/значение», получившие название NoSQL (Not-Only-SQL) [NoSQL]. Эти системы обладают рядом преимуществ по сравнению с параллельными системами баз данных SQL [Григорьев, Цвященко, 2014а]. К ним можно отнести высокую масштабируемость и отказоустойчивость, возможность обработки неструктурированных данных, большое число реплик и др.

В базах данных NoSQL не поддерживается режим ведения транзакций, поэтому возникает проблема согласования данных. Поддержание требуемого уровня согласованности для каждой конкретной предметной области может регулироваться параметрами (N, W, R): N — количество узлов, на которые в конечном счете будет реплицирована запись (может быть с некоторой задержкой); W — количество узлов (реплик), 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 — количество узлов, от которых база данных ожидает ответа для успешного завершения чтения записи [Редмон, Уилсон, 2013]. Существуют различные виды согласованности: слабая и строгая (рис. 1).



(а) (б)

Рис. 1. Слабая (а) и строгая (б) согласованность реплик в базах данных NoSQL

При слабой согласованности ($R + W \le N$) гарантируется, что все реплики будут идентичны в конечном счете. При этом не гарантируется чтение последних обновлений. В работах [Bailis et al., 2012; Bermbach, Tai, 2011] упор делается на экспериментальное исследование свойств слабой согласованности. Предлагаемые математические модели носят упрощенный характер [Bailis et al., 2012]: в них не учитывается механизм распространения обновлений и его параметры, а также не учитывается интенсивность запросов на чтение.

Строгая согласованность ($W + R > N$) позволяет всегда получать актуальную версию записи, но это иногда приводит к большим задержкам ожидания завершения обновления W реплик базы данных и чтения записей из R реплик. Для большого класса приложений это имеет решающее значение. Например, для фирмы Amazon дополнительные задержки в 100 мс привели к 1 % падению продаж. В то же время 500 мс задержки в поисковой системе Google привели к снижению трафика на 20 % [Bailis et al., 2012].

Таким образом, актуальной является задача разработки математической модели строгой согласованности реплик, позволяющей прогнозировать значения таких задержек. Несмотря на важность этой проблемы [Григорьев, Цвященко, 2014а; Григорьев, Цвященко, 2014б; Редмон, Уилсон, 2013; Bailis et al., 2012; Bermbach, Tai, 2011], научных публикаций по этой тематике практически нет. Это связано со сложностью задачи, вызванной необходимостью учета механизмов репликации и параметров аппаратных ресурсов, задействованных в процессе тиражирования обновленных данных.

В [Григорьев, Цвященко, 2014б] была разработана модель строгой согласования реплик в базах данных NoSQL. Целью данной работы является доработка модели и доказательство ее адекватности. Для доказательства адекватности был выполнен натурный эксперимент на кла-

стере базы данных NoSQL Riak [Riak documentation] размером до 11 узлов. Приводятся спецификации прикладных программ для проведения эксперимента и анализа статистики, полученной от каждого клиента, для расчета времени ожидания требованием на чтение окончания обновления W реплик. Часть полученных результатов использовалась для адаптации модели, а часть — для оценки ее адекватности.

В работе [Григорьев, Цвященко, 2014б] разработана модель сильного согласования реплик. Было получено выражение для преобразования Лапласа–Стилтьеса (ПЛС) функции распределения вероятностей времени ожидания требованием на чтение окончания обновления W реплик ($W + R > N$):

$$\Omega(s) = W_q(0) + (1 - W_q(0)) \cdot \Phi(s), \tag{1}$$

$$\Phi(s) = -\frac{1 - \phi(s)}{s \cdot \phi^{(1)}(0)}, \tag{2}$$

$$\phi(s) = \prod_{i=1}^{W} \psi_i(s), \tag{3}$$

$$W_q(z) = \prod_{i=1}^{W} \psi_i(\lambda N(1 - z)). \tag{4}$$

Здесь $\Omega(s)$ — ПЛС времени ожидания начала чтения; $\Phi(s)$ — ПЛС времени до окончания обновления W реплик [Риордан, 1966, с. 36, 37]; $\Psi_i(s)$ — ПЛС времени обновления i-й реплики; $W_q(z)$ — производящая функция (ПФ) числа тех требований на чтение из N реплик, которые поступят за время обновления W реплик. — первая производная в нуле (1-й начальный момент). Учитывая вероятностный смысл производящей функции, описанный в [Ивченко и др., 1982], вероятность того, что за время обновления W реплик поступит хотя бы одно требование на чтение из N реплик, равна $1 - W_q(0)$. Также учитывался тот факт, что если в фиксированном интервале времени поступило конкретное число требований на чтение записи из реплик, то моменты поступления этих требований распределены по равномерному закону [Ивченко и др., 1982]. Это позволяет сделать вывод, что выражение (1) для ПЛС времени ожидания справедливо для всех требований, находящихся в очереди на чтение.

Учитывая выражение для $\Phi^{(1)}(0)$ [Риордан, 1966, с. 37], получим математическое ожидание времени ожидания начала чтения записи из реплики (задержки чтения):

$$M = -\Omega^{(1)}(0) = -(1 - W_q(0)) \cdot \frac{\phi^{(2)}(0)}{2 \cdot \phi^{(1)}(0)}, \tag{5}$$

где $\phi^{(i)}(0)$ — i-я производная $\phi(s)$ в нуле (i-й начальный момент). Моменты функции можно оценить, используя методы численного дифференцирования [Бахвалов, 1973].

Предложенное в [Григорьев, Цвященко, 2014а] ПЛС $\Psi_i(s)$ времени обновления реплики имеет составляющие следующего вида:

$$\beta(s) = \left(\frac{\mu}{\mu + s} \right)^n. \tag{6}$$

Квадрат коэффициента вариации этой случайной величины рассчитывается по формуле

$$\alpha^2 = \frac{D}{M^2} = \frac{n / \mu^2}{(n / \mu)^2} = \frac{1}{n}. \tag{7}$$

При большом n квадрат коэффициента вариации стремится к нулю. Это означает, что выражение (6) описывает более детерминированную случайную величину. Поэтому предлагается другая модель расчета времени обновления реплики.

Обозначим через $\Lambda(s, r, t)$ ПЛС сетевой составляющей, а через $\Theta(s)$ — ПЛС локальной составляющей времени обновления. Имеем

$$\Lambda_{i+1}(s,t,r) = \left(\phi_{net}(s,r)\right)^{t} \cdot \left(\phi_{nm}(s)\right)^{1-t}, \tag{8}$$

$$\phi_{net}(s,r) = \frac{\mu_{net}(r)/(k+v)}{\mu_{net}(r)/(k+v)+s}, \qquad \frac{1}{\mu_{net}(r)} = \frac{2}{\mu_m} + \frac{2}{\mu_n} + \frac{r}{\mu_{ns}}, \tag{9}$$

$$\phi_{nm}(s) = \frac{\mu_m/(k+v)}{\mu_m/(k+v)+s}. \tag{10}$$

Здесь параметр $t = 1,$ если узел, содержащий $i+1$ реплику, не совпадает с координатором (передача данных по сети), 0 — иначе (передача данных в памяти). Параметр $r = 1,$ если узел, содержащий $i+1$ реплику, находится в подсети, не содержащей координатор, 0 — иначе.

$\varphi_{net}(s,r)$ — ПЛС суммарного времени передачи записи по сети.

μ_m — интенсивность чтения байтов данных из ОП (2/μm — учитывает передачу данных из ОП в буфер сетевого адаптера (СА) до передачи по сети и передачу данных из буфера СА в буфер ОП после передачи по сети). μ_n — интенсивность передачи байтов данных по локальной сети между станцией и коммутатором ($2/\mu_n$ — предполагается, что коммутатор работает с буферизацией). Будем считать, что подсети имеют одинаковую пропускную способность. μ_{ns} — интенсивность передачи байтов данных по сети, соединяющей подсети.

k — размер ключа записи (20-байтное число — RIAK, 16-байтное — Cassandra, Dynamo (MD5));

v — размер поля «значение» записи.

ПЛС локальной составляющей имеет вид

$$\Theta(s) = \frac{\mu_{crc}}{\mu_{crc}+s} \cdot \frac{\mu_{md}}{\mu_{md}+s} \cdot \frac{\mu_d}{\mu_d+s} \cdot \phi_{\text{kd}}(s). \tag{11}$$

Здесь μ_{crc} — интенсивность расчета контрольной суммы записи:

$$\frac{1}{\mu_{crc}} = (k+v+12)\left(\frac{4}{\mu_p} + \frac{1}{\mu_{\text{m}}}\right), \tag{12}$$

где μ_p — количество операций, выполняемых процессором в секунду; 12 — длина временной метки + длина ключа + длина значения; 4 — учитывает, что для расчета контрольной суммы требуется 4 процессорных операции на один байт данных. μ_{md} — интенсивность передачи записи из ОП в буфер диска:

$$\frac{1}{\mu_{md}} = (k+v+16)\frac{1}{\mu_m}, \tag{13}$$

где 16 — длина контрольной суммы + длина временной метки + длина ключа + длина значения. μ_d — интенсивность чтения + записи блока на диск:

$$\frac{1}{\mu_d} = wb\frac{1}{\mu_{d1}}, \tag{14}$$

где μ_{d1} — интенсивность чтения/записи байтов на диск; b — длина блока диска в байтах (может быть задана произвольно, так как величина μ_{d1} адаптируется); w — учитывает время на чтение блока и время обязательной записи блока на диск после обновления записи (режим DW). В случае включенной опции DW $w = 2,$ иначе — 1.

$\varphi_{kd}(s)$ — ПЛС времени обновления хеш-таблицы (keydir) в ОП; не зависит от размера поля «значение» записи, так как хеш-таблица строится по ключу, значение которого соответствует

структуре «идентификатор файла, размер значения, смещение значения, временная метка». Размер этой структуры равен 20 байтам

$$\varphi_{kd}(s) = \frac{\mu_{\text{m}}/(20+k)}{\mu_m/(20+k)+s} \cdot \frac{\mu_p/16}{\mu_p/16+s}, \tag{15}$$

где $20+k$ учитывает передачу описанной структуры и ключа из памяти в кеш процессора для подсчета контрольной суммы; 16 — учитывает, что количество процессорных операций, необходимых для подсчета хеша, примерно равно 16.

Таким образом, ПЛС времени обновления реплики можно определить как

$$\Psi_{i+1}(s) = \Lambda_{i+1}(s,r,t) \cdot \Theta(s), \tag{16}$$

где $\Lambda(s, r, t)$ и $\Theta(s)$ определяются выражениями (8) и (11).

Цель натурных экспериментов — это оценить адекватность модели расчета среднего времени ожидания требованием на чтение окончания обновления W реплик, т. е. адекватность выражения (5). Обозначим через C множество измеренных неизменяемых параметров системы, X — настраиваемые параметры, т.е. параметры, которые меняются от эксперимента к эксперименту, Y — адаптируемые параметры. Ниже представлены параметры C, X и Y:

$$C = \{k, v, \mu_{ns}, \mu_p\}, \quad X = \{N, W, \lambda\}, \quad Y = \{\mu_n, \mu_m, \mu_{d1}\}.$$

Элементы этих множеств описаны выше при описании ПЛС времени обновления реплики.

Описание экспериментальной установки

На рынке существует много компаний, предоставляющих облачные вычислительные ресурсы. Ресурсы бывают виртуальными и выделенными. Выделенными облачными ресурсами являются узлы, которым соответствуют физически отдельные серверы. В отличие от выделенных узлов виртуальные узлы — это виртуальные машины. Выделенные серверы стоят существенно дороже, поэтому в нашем эксперименте использовались виртуальные узлы, предоставленные компанией DigitalOcean (DO) [Digital Ocean]. Так как время ожидания требованием на чтение оценивается при работе клиента с одной записью «ключ/значение», то база данных NoSQL Riak не нагружает оперативную память, что позволило арендовать недорогие серверы с малым объемом оперативной памяти.

Все узлы, предоставляемые поставщиком облачных ресурсов, основаны на многопроцессорных машинах с SSD-дисками. Использование многопроцессорных машин позволяет сделать следующий вывод: вероятность того, что другие клиенты DO, которым предоставлены виртуальные узлы на том же физическом сервере, будут использовать именно тот процессор, на котором работает Riak, мала. Следовательно, производительность виртуального узла существенно не зависит от фоновой загрузки процессора, что позволяет отнести параметр μ_p — количество процессорных циклов, выполняемых в секунду, — к фиксированным параметрам (т. е. к множеству 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 отсут-ствие 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 [Ubuntu], предустановленная поставщиком. Для установки и настройки Riak необходимо выполнить ряд действий на каждом из N узлов. Установка базы данных Riak с нуля требует много времени. Поэтому для ускорения подготовки кластера общая часть действий по установке сис-

темы, описанных в [6], была выполнена один раз на одном узле. Далее был сделан снимок узла, который впоследствии восстанавливался на остальных узлах. Индивидуальная часть действий по настройке Riak, описанная в [Riak documentation], была сохранена в виде bash-скрипта. Данные скрипты являются Linux-аналогом bat-скриптов для ОС Windows. После установки и настройки системы все узлы необходимо соединить 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 выполняется на каждом узле, кроме одного, к которому присоединяются остальные узлы (ip_first_node). После выполнения команды 1 необходимо проверить конфигурацию кластера на любом из узлов командой 2, после чего сохранить изменения командой 3. После выполнения всех команд кластер будет готов к работе, однако необходимо некоторое время для переноса уже хранящихся данных из первого узла во все остальные. В течение этого времени система может отвечать not found на запросы клиентов. Так как исследуется сильная согласованность, то настройки согласованности (N, W, R) для каждого эксперимента задавались следующим образом: N, $(N + 1)/2$, $(N + 1)/2$. Для всех экспериментов N — нечетное число. На одном узле с репликой запускались два процесса — на обновление записи и ее чтение, на остальных $(N - 1)$ узлах с другими репликами запускались только процессы чтения (по одному на каждый узел).

Подготовка эксперимента

В теории систем массового обслуживания [Ивченко и др., 1982, с. 24–29] доказывается, что суммирование независимых однородных событий (от отдельных клиентов) близко к пуассоновскому потоку, поэтому в модели входящий поток требований на чтение к одной реплике принимался пуассоновским с параметром λ. Натурный эксперимент выполняется для доказательства адекватности модели, поэтому важно, чтобы условия эксперимента соответствовали модели: требования на чтение должны независимо поступать к каждой из $1 \ldots N$ реплик с интенсивностью λ. Также важно отношение интесивности поступления требований на обновление к интенсивности поступления требований на чтение. В эксперимintах интенсивность поступления требований на обновление записи была принята равной 1.25 (1/с), но варьировалась интенсивность поступления требований на чтение записей для каждого числа реплик N.

Для выполнения обновления и чтения записи базы данных были разработаны соответствующие прикладные программы. 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 в среде ОС Windows.

Все процессы чтения и обновления выполняются непосредственно на узлах. В явном виде сложно оценить время ожидания требованием на чтение окончания обновления W реплик, поэтому это время ожидания оценивалось косвенно. Общая идея заключается в следующем. Все процессы запускаются одновременно, но процесс записи начинает обновлять запись базы данных с задержкой, в течение которой читающие процессы производят чтение в отсутствие обновления записи БД. Среднее время ожидания требованием на чтение окончания обновления W реплик рассчитывается как разница между средним временем чтения из БД «на фоне обновления» и средним временем чтения «без обновления». Ниже приведены спецификации алгоритмов. Для оценки среднего времени чтения «без обновления» использовалось 3000 итераций.

Алгоритм процесса записи, выполняемого на одном из N узлов

Вход: N_ITER_W — количество итераций обновления записи БД (см. ниже); KEY — ключ обновляемой записи; REC_VAL — счетчик, используемый в качестве «значения» записи; λ — интенсивность поступления требований на чтение.

Алгоритм:
REC_VAL = 1
DELAY = 3000 / λ
Задержать выполнение на время DELAY
ЗАПИСАТЬ в журнал CURR_TIME // TIME_W
ЦИКЛ по N_ITER_W
    TIME_STAMP = CURR_TIME
    ЗАПИСАТЬ в БД запись <KEY/REC_VAR>
    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.

Алгоритм процесса чтения, выполняемого на каждом из узлов 1…N

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

Алгоритм:
N_ITER_R = N_ITER_W / 1.25 · λ + 3000
ЦИКЛ по N_ITER_R
    TIME_STAMP = CURR_TIME
    TIME_START_READ = TIME_NANO
    СЧИТАТЬ из БД запись <KEY/REC_VAL>
    ЗАПИСАТЬ в журнал
    <TIME_STAMP, TIME_NANO-TIME_START_READ>
    DELAY = EXPONENTIAL(λ)
    DELAY –= (CURR_TIME – TIME_STAMP)
    ЗАДЕРЖАТЬ выполнение на время DELAY
КОНЕЦ ЦИКЛА

Функция TIME_NANO возвращает системное время в наносекундах. Данное время не является астрономическим — оно привязано к времени старта операционной системы и используется только для профилирований программных продуктов. Использование времени в наносекундах позволяет достичь высокой точности при оценке времени чтения из хранилища данных NoSQL.

После завершения процессов чтения и записи на каждом узле журналы должны быть скопированы на локальную машину для оценки среднего времени ожидания начала чтения из БД, полученного экспериментальным путем. Ниже приведен алгоритм оценки среднего времени ожидания.

Алгоритм вычисления среднего времени ожидания

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

Алгоритм:
COUNT_A = COUNT_B = 0
TIMES_A = TIMES_B = 0
СЧИТАТЬ время TIME_W из журнала LOG_W
ДЛЯ каждого журнала LOG_I_R из LOGS_R
    ДЛЯ каждой записи <TIME_1,TIME_2 > из LOG_I_R
        TIME_2 = TIME_2 · 10^6

```
                ЕСЛИ TIME_1 < TIME_W
                        COUNT_A++           // до начала обновления
                        TIMES_A += TIME_2
                ИНАЧЕ
                        COUNT_B++           // после начала обновления
                        TIMES_B += TIME_2
                КОНЕЦ ЕСЛИ
        КОНЕЦ ДЛЯ
КОНЕЦ ДЛЯ
ВЕРНУТЬ TIMES_B / COUNT_B - TIMES_A / COUNT_A
```

Умножение переменной TIME_2 на 10^6 обусловлено тем, что время выполнения операции чтения каждым клиентом измеряется в наносекундах.

Проведение эксперимета

Число процессов чтения записи в эксперименте изменялось от 5 до 11, на один процесс приходилось 3000 итераций чтения. Поэтому общий объем выборки операций чтения в отсутствие обновления составил от 15000 до 33000. Когда система не нагружена и отсутствуют обновления, время чтения записи из БД имеет небольшую дисперсию, о чем нельзя сказать для режима чтения «на фоне обновления». Для этого режима необходимо установить минимальный объем выборки N_ITER_W. Для этого воспользуемся теоремой Ляпунова [Теорма Ляпунова]. 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

$$\Pr\left(\left| \frac{\sum_{i=1}^{k}\xi_i}{k} - M(\xi) \right| < \delta \right) \approx \gamma = 2\Phi(x) = 2\frac{1}{\sqrt{2\pi}}\int_0^x e^{-\frac{z^2}{2}}dz\,, \tag{17}$$

$$x = \frac{\delta\sqrt{k}}{\sigma}\,, \tag{18}$$

откуда получаем

$$\delta = \frac{x\sigma}{\sqrt{k}}\,. \tag{19}$$

Здесь σ — оценка среднеквадратического отклонения случайной величины ξ_i. Значение x зависит от надежности γ: $x = 1{,}96$ (для $\gamma = 0{,}95$), $x = 2{,}58$ (0,99), $x = 3{,}29$ (0,999).

В нашем случае случайная величина времени чтения записи ξ_i измеряется в несколько миллисекунд, примем $\sigma = 1{,}0$ мс. Пусть необходимо получить точность оценки среднего значения, определяемого выражением (5), с надежностью $\gamma = 0{,}95$. Тогда из (19) получим

$$k \geq (x\sigma/\delta)^2 = \frac{1{,}96^2 \cdot (1{,}0)^2}{\delta^2} = \frac{3{,}84}{\delta^2}. \tag{20}$$

Рассчитаем объем выборки k по выражению (20): при $\delta = 0{,}01$ (мс) получаем $k \geq 38400$ (общее число чтений). Учитывая, что минимальная интенсивность чтений из всех реплик равна $(N = 5)\cdot(\lambda = 3) = 15$ (1/с), получим значение N_ITER_W $\geq (k = 38400)\cdot 1{,}25/15 = 3200$. В экспериментах было принято N_ITER_W = 4000.

Было выполнено две серии экспериментов. В таблице 1 приведены значения изменяемых параметров $X(N, W, \lambda)$ и результаты проведения первой серии экспериментов.

Таблица 1. Первая серия экспериментов.

N	W	$\lambda(1/c)$	Среднее время ожидания, мс	N	W	$\lambda(1/c)$	Среднее время ожидания, мс
5	3	3	0,041	7	4	3	0,090
		4	0,050			4	0,115
		5	0,067			5	0,131
		6	0,077			6	0,160
		7	0,080			7	0,184
		8	0,094			8	0,217
		9	0,098			9	0,248
		10	0,118			10	0,260

В таблице 1 серым выделены те результаты экспериментов, которые были использованы для адаптации модели. В таблице 2 приведены значения изменяемых параметров X и результаты проведения второй серии экспериментов.

Таблица 2. Вторая серия экспериментов

N	W	$\lambda(1/c)$	Среднее время ожидания, мс	N	W	$\lambda(1/c)$	Среднее время ожидания, мс
9	5	3	0,242	11	6	3	0,416
		4	0,332			4	0,535
		5	0,401			5	0,676
		6	0,520			6	0,870
		7	0,645			7	1,015
		8	0,700			8	1,025
		9	0,784			9	1,180
		10	0,812			10	1,215

Адаптация модели

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

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

Задача адаптации модели (5) решалась методом наименьших квадратов [Метод наименьших квадратов]:

$$\sum_{i=1}^{L} (M(C, X_i, Y) - M_i)^2 \xrightarrow[Y]{} \min, \tag{21}$$

где $M(C, X_i, Y)$ — аналитическое выражение (5); M_i — оценка среднего времени ожидания требованием на чтение окончания обновления реплик, полученное при проведении i-го эксперимента (см. таблицы 1 и 2); L — число экспериментов, по которым проводилась адаптация модели (они отмечены в таблицах 1 и 2 серым цветом, остальные эксперименты были использованы при оценке адекватности модели). Оптимальные значения адаптируемых параметров Y были получены методом наискорейшего спуска.

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 серии экспериментов. После решения оптимизационной задачи (21) были получены адаптируемые параметры, представленные в таблице 3.

Таблица 3. Адаптируемые параметры

Первая серия		Вторая серия	
Параметр	Значение	Параметр	Значение
μ_n	441,4 Мбит/с	μ_n	369 Мбит/с
μ_m	7460 МБайт/с	μ_m	8560 МБайт/с
μ_{dl}	201 МБайт/с	μ_{dl}	152 МБайт/с

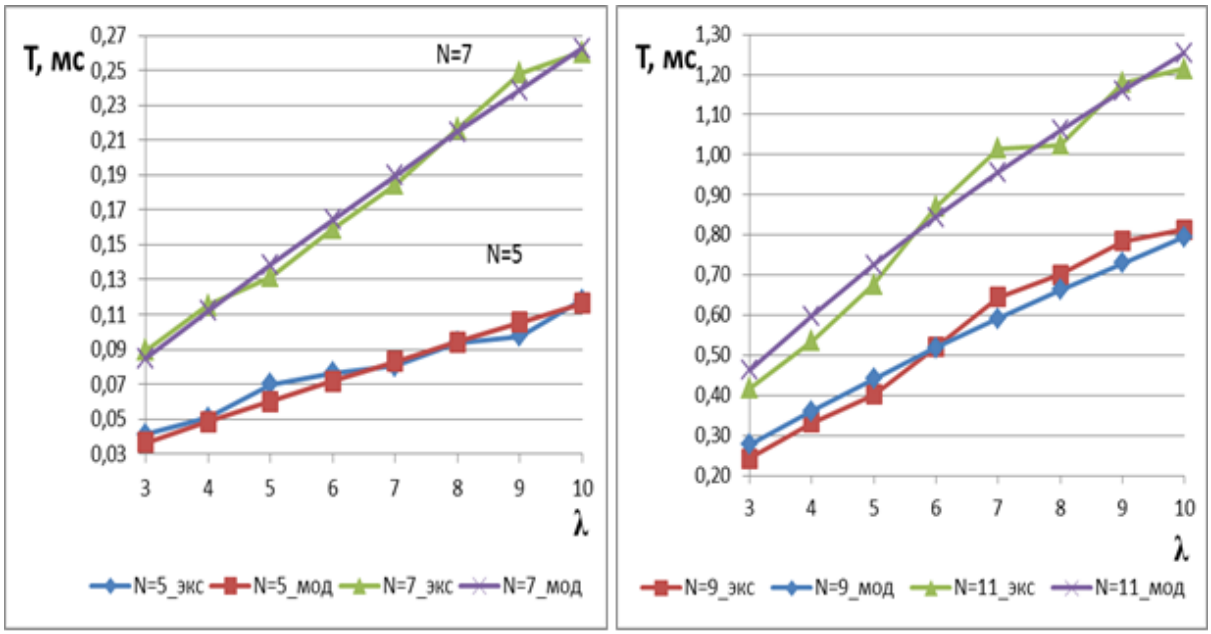
Анализ адекватности модели строгого согласования реплик

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

Таблица 4. Анализ адекватности модели

N	W	$\lambda(1/c)$	Среднее время ожидания, мс		Относительная погрешность
			Эксперимент	Модель (5)	
5	3	3	0,041	0,036	0,115
		5	0,070	0,060	0,140
		7	0,080	0,083	0,032
		9	0,097	0,105	0,082
7	4	3	0,090	0,085	0,051
		5	0,132	0,139	0,053
		7	0,184	0,190	0,029
		9	0,249	0,239	0,039
9	5	3	0,242	0,277	0,141
		5	0,402	0,441	0,098
		7	0,645	0,592	0,082
		9	0,784	0,730	0,069
11	6	3	0,416	0,462	0,112
		5	0,676	0,725	0,072
		7	1,015	0,956	0,058
		9	1,180	1,160	0,016

Таким образом, средняя ошибка по двум сериям экспериментов составила 7,42 %. На рис. 2 показаны графики зависимости среднего времени ожидания требованием на чтение окончания обновления W реплик от интенсивности λ при различных значениях N для первой и второй серии экспериментов, $T = M(\lambda)$. На рисунке «мод» и «экс» обозначают графики, построенные соответственно по модели (5) и экспериментальным данным. До сих пор речь шла о безусловном среднем времени ожидания M, которое определяется выражением (5). В таблице 5 приведено среднее время ожидания при условии, что требование на чтение действительно ожидает завершения обновления W реплик, $T = M / (1 - W_q(0)) = -\Phi^{(1)}(0)$ (см. (1) и (5)). Это время не зависит от λ.



(а) (б)

Рис. 2. Зависимости среднего времени ожидания $T = M(\lambda)$ при различных значениях N: (а) первая серия экспериментов; (б) вторая серия экспериментов

Таблица 5. Условное среднее время ожидания

N	5	7	9	11
T (мс)	1,30	1,60	2,55	3,00

Из таблицы 5 и рис. 2 видно, что условное среднее время ожидания больше безусловного среднего времени в несколько раз.

Заключение

1. По результатам натурного эксперимента, выполненного с базой данных NoSQL Riak в облачной среде компании Digital Ocean на кластере с числом узлов до 11, доказана адекватность разработанной модели (5) (см. таблицу 4). Средняя относительная погрешность составила 7,42 %.

2. Задача (21) адаптации модели решалась методом наименьших квадратов. При этом была использована только часть полученных экспериментальных данных, другая часть использовалась для доказательства адекватности модели.

3. Приведены алгоритмы работы прикладных программ для проведения натурных экспериментов для режима сильного согласования реплик и оценки среднего времени ожидания требованием на чтение окончания обновления W реплик. На основе теоремы Ляпунова обоснован объем выборки, необходимый для достижения надежности оценки, равной 0,95.

Список литературы

Бахвалов Н. С. Численные методы (анализ, алгебра, обыкновенные дифференциальные уравнения). — М.: Наука, 1973. — 631 с.

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

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

Ивченко Г. И., Каштанов В. А., Коваленок И. Н. Теория массового обслуживания. — М.: Высшая школа, 1982. — 256 с.

Метод наименьших квадратов. URL: https://ru.wikipedia.org/wiki/Метод_наименьших_квадратов (дата обращения: 01.07.2015).

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.

Риордан Дж. Вероятностные системы обслуживания. — М.: Связь, 1966. — 184 с.

Теорма Ляпунова. URL: https://ru.wikipedia.org/wiki/Теорема_Ляпунова (дата обращения: 01.07.2015).

Bailis P., Venkataraman Sh., Franklin M. J., Hellerstein J. M., I. Stoica I. Probabilistically Bounded Staleness for Practical Partial Quorums, 2012: URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-4.pdf (дата обращения 01.07.2015).

Bermbach D., Tai S. Eventual Consistency: How soon is eventual? // ACM MW4SOC ’11, December 12, 2011, Lisboa, Portugal. URL: http://dl.acm.org/citation. cfm?id=2093186 (дата обращения: 01.07.2015).

Digital Ocean. URL: https://www.digitalocean.com (дата обращения: 01.07.2015).

NoSQL. URL: http://ru.wikipedia.org/wiki/NoSQL (дата обращения: 01.07.2015).

Riak documentation. URL: http://docs.basho.com/index.html (дата обращения: 01.07.2015).

Ubuntu OS 14.04. URL: http://releases.ubuntu.com/14.04 (дата обращения: 01.07.2015).