# Модель оценки характеристик строгого согласования реплик в базах данных NoSQL и анализ ее адекватности

Ю.А. Григорьев, Е.В. Цвященко

**Аннотация.** В статье разработана новая модель строгого согласования реплик в базах данных NoSQL, позволяющая оценивать характеристики случайного времени ожидания запросов на чтение завершения операции обновления записи. Описывается процесс подготовки и проведения натурного эксперимента в облаке для адаптации модели и анализа ее адекватности. Приводятся результаты, подтверждающие адекватность модели на линейном участке зависимости времени ожидания начала чтения от интенсивности запросов на чтение.

**Ключевые слова:** база данных NoSQL, строгая согласованность, модель, адекватность, адаптация, время ожидания.

## Введение

В настоящее время львиная доля информации хранится в реляционных базах данных. Это объясняется внешней простотой реляционной модели, наличием такого мощного и в то же время простого для изучения языка доступа к данным, как SQL, а также наличием оптимизаторов выполнения транзакций и запросов к базе данных, обеспечивающих их «дробление» на более мелкие задания и параллельную реализацию этих работ на многопроцессорных или многомашинных комплексах.

Системы IBM DB2, Microsoft SQL Server и Oracle, которые в той или иной форме наследуют System R, созданную еще в 70-е годы, относятся к классу строчных реляционных СУБД. В них таблицы хранятся в виде блоков, записи в блоке располагаются последовательно одна за другой. Они полностью поддерживают ACID-транзакции (атомарность, согласованность, изоляция, долговечность), которые обеспечивают целостность базы данных. Это свойство СУБД – необходимое условие функционирования, например, финансовых систем: проводки между счетами, операции с карт-счетами клиентов банкоматов и т.д. реализуются в виде ACID-транзакций.

Но ядро этих реляционных СУБД фундаментально не менялось на протяжении 25 лет. В то же время характеристики аппаратных устройств резко изменились: мощности процессоров выросли в 5000-10000 раз, время передачи данных с диска в память уменьшилось в 100 раз, время поиска данных на диске – в 10 раз. Различие в приросте мощностей обрабатывающих ресурсов (10000–100–10 раз) значительно повлияло на рабочие нагрузки СУБД [1]. Наблюдался также дисбаланс между ростом емкости диска и скоростью передачи данных: 1) пропускная способность передачи информации по отношению к общему объему доступной информации уменьшилась на два порядка; 2) отношение скорости доступа к последовательно расположенным на диске данным и при случайной выборке выросло на порядок. Видно, что для СУБД необходимо было не только предотвращать случайный доступ к данным, но и уменьшать нагрузку на сам канал их передачи. Однако большинство строчных реляционных СУБД развивало именно технологии по-

следовательного доступа, несмотря на то, что объем данных возрастал и росла доля случайного доступа к ним. Таким образом, обработка больших объемов данных выполнялась все медленнее и медленнее.

Для решения возникших проблем Стоунбрейкер с коллегами предложили специализированную колоночную базу H-Store для работы в грид-средах, уже в первой реализации показавшую производительность на два порядка выше, чем традиционные строчные СУБД [2]. В настоящее время появилось много колоночных СУБД. Они поддерживают реляционную модель данных, включая язык SQL, но отличаются от строчных СУБД физической организацией данных: столбцы таблицы хранятся в отдельных блоках (колонках) в сжатом виде. Большая производительность достигается за счет большой степени сжатия однородных данных в колонках и выполнения операций не над записями, а над битовыми масками (отложенная материализация). Основное назначение этих СУБД – обработка аналитических запросов к большим базам данных. Например, колоночная СУБД MonetDB была использована в проекте широкомасштабного исследования изображений и спектров звезд и галактик с помощью 2,5-метрового широкоугольного телескопа в обсерватории Апачи-Пойнт в Нью-Мексико. Исследования начались в 2000 г., и тогда выяснилось, что, кроме MonetDB, ни одна база не может справиться с задачей накопления данных.

И строчные, и колоночные СУБД поддерживают реляционную модель данных, предполагающую хранение данных в виде таблиц и наличие схемы базы данных. И это ставит перед разработчиками больших баз банных трудно решаемые задачи:

1. Для реляционных баз данных характерна потеря соответствия. Проявляется она в том, что реляционные базы данных не позволяют хранить агрегаты. Например, чтобы отобразить всю информацию о покупателе и всех его покупках (агрегат), программист должен собрать в оперативной памяти данные из многих таблиц: покупатель, заказ, пункт заказа, цена и др. При значительном росте числа таблиц разработчик часто просто забывает назначение той или иной таблицы, осложняется связывание таблиц при выполнении запроса, т.е. существенно осложняется формирование агрегата. Появление множества библиотек для объектно-реляционного отображения, таких как Hibernate и iBATIS, не устранило проблему [3].

2. С увеличением объема хранимых данных возникает задача фрагментации таблиц базы данных по разным серверам, объединенных в кластер. Для поддержания работы такого кластера используются параллельные системы баз данных (Oracle RAC, Teradata и др.). Но настройка такого кластера сложна [4]. Более того, при выполнении сложных запросов производительность системы существенно уменьшается в результате межмашинного обмена данными между серверами кластера [5, 6]. Возникает проблема обеспечения надежности кластера.

3. Схема базы данных состоит из подсхем, каждая из которых отражает предметную область какого-либо подразделения организации (предприятия). Модификация подсхемы одного подразделения и реорганизация соответствующих таблиц приводит к вынужденной приостановке работы остальных подразделений. Сейчас в качестве альтернативы централизованной базы данных широко используются так называемые «базы данных приложений» [3], т.е. подразделений. Обеспечение взаимодействия этих баз данных возлагается на web-сервисы. Но поддержка целостности данных и оперативности такого взаимодействия является непростой задачей для программиста.

Как попытка решить накопившиеся проблемы реляционных баз данных появились альтернативные средства хранения и обработки данных, получившие название «базы данных NoSQL» [24]. Пионерами в этой области выступили две компании: Google и Amazon, которые опубликовали короткие документы о своих продуктах: BigTable и Dynamo [7, 8].

Внешне идея этих баз данных очень проста: данные хранятся в виде записей <ключ, значение>, схема базы данных отсутствует. Здесь изящно решены три перечисленные выше проблемы:

1. В поле «значение» может храниться агрегат, например, вся информация о покупателе и его покупках (решается 1-ая проблема).

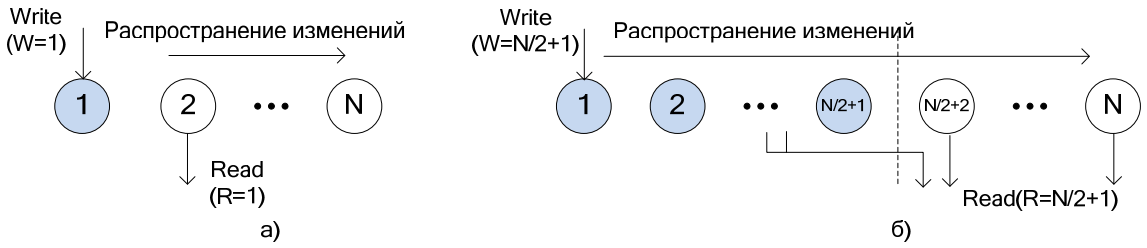



Рис. 1.Слабая (а) и строгая (б) согласованность реплик в базах данных NoSQL

2. Данные автоматически равномерно распределяются и реплицируются по серверам кластера. Процедура развертывания и настройки кластера в несколько тысяч узлов достаточно проста. Обработка запроса реализуется посредством механизма MapReduce, обеспечивающего обмен данными между узлами небольшими порциями (фаза shuffle). Объем данных, передаваемый от одного узла другим примерно равен $V/n^2$, V – объем данных, удовлетворяющих условию поиска; n – число серверов (узлов) в кластере [9]. Имеет место очень большая живучесть системы за счет мощного механизма репликации записей по узлам (решается 2-ая проблема).

3. Отсутствие схемы базы данных позволяет включать или удалять атрибуты на уровне отдельной записи, не затрагивая работу остальной части системы (решается 3-я проблема).

Но базы данных NoSQL обладают недостатками: в этих базах данных не поддерживается режим ведения транзакций и блокировок, поэтому возникает проблема согласования данных. Такие базы данных нельзя использовать, например, для реализации проводок в автоматизированных банковских системах. Для них характерна большая трудоемкость программирования заданий в среде MapReduce.

Итак, господствовавшая идея сведения всего разнообразия данных к таблицам устарела. Стоунбрейкер, в частности, отмечает [2]: «Сейчас у каждого из вертикальных рынков имеются свои проблемы, для решения которых требуются наиболее удобные средства, и нет нужды ограничиваться унаследованными из прошлого реляционными системами». Каждый тип СУБД имеет свою нишу, и они еще долго будут сосуществовать вместе.

Эта статья посвящена вопросу оценки временных характеристик согласованности данных в базах данных NoSQL, который является одним из ключевых для этого сегмента баз данных.

## Слабая и строгая согласованность данных в базах данных NoSQL

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 (м.б. 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 [10].

Существуют следующие виды согласованности: слабая (в конечном счете) и строгая (Рис. 1).

При слабой согласованности ($R+W\leq N$) гарантируется, что все реплики будут идентичны в конечном счете. При этом не гарантируется чтение последних обновлений.

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% [11].

Итак, увеличение задержки выполнения операций записи и чтения данных (для строгой согласованности) связны с большими экономическими потерями, но, с другой стороны, снижение задержки для согласованности в конечном счете (КС-согласованности) приводит к увеличению вероятности рассогласования данных: чем больше реплик участвуют в запросе чтения, тем меньше гарантий получить самые последние актуальные данные.

Несмотря на важность проблемы рассогласования реплик в кластерной архитектуре баз данных NoSQL [3, 10-12], научных публикаций по этой тематике немного в связи со сложностью теоретического решения задачи оценки показателей согласования реплик. Это связано с необходимостью учета сложных механизмов репликации, а также параметров аппаратных ресурсов, задействованных в процессе тиражирования обновленных данных.

До настоящего времени теоретической основой исследования проблемы согласования данных является теория кворумов [13].

Пусть дано множество серверов U. Системой кворума над U называется множество подмножеств D={Q1,...,Qm} такое, что каждое Qi $\subseteq$U и $|Q \cap Q'| > 0$ для всех Q,Q' $\in$ D. Каждое Qi называется кворумом. Некоторые процессы пишут в кворумы (обновляют записи), другие процессы читают из кворумов. Условие $|Q \cap Q'| > 0$ гарантирует, что процесс чтения будет получать актуальные записи по крайней мере с одного сервера, т.е. система будет согласованной.

В теории кворумов рассматриваются византийские системы кворума, предполагающие наличие неисправных серверов. Рассматриваются различные варианты распределения неисправных серверов по кворумам (возможно с некоторой вероятностью) и оценивается верхняя граница общего числа неисправных серверов в зависимости от общего числа серверов n=|U|.

В теории также решается задача поиска нижней границы загрузки системы кворума L(D) – вероятности, с которой осуществляется доступ к самому занятому серверу для лучшей стратегии, т.е. для наиболее оптимального распределения вероятностей кворумов P(Q) ($\sum_{Q \in D} p(Q)=1$).

Интересной является задача минимизации задержки доступа к кворуму. Задача формализуется следующим образом. Пусть отображение $\delta : V \times 2^V \rightarrow R+$ определяется как $\delta(v,Q)=\max_{v' \in Q} d(v,v')$, где d – расстояние между двумя узлами-серверами из V. Многие авторы показывают, как построить систему кворума на V, чтобы минимизировать среднее: $\text{avg}_{v \in V} \min_{Q \in D} \delta(v,Q)$.

Но использование теории кворумов применительно к базам данных NoSQL наталкивается на серьезные препятствия. В NoSQL отказ сервера не приводит к потере реплики (копии): если узел отказывает, то реплика автоматически создается на другом узле. Стратегию доступа к серверам определяет не пользователь, а сама база данных NoSQL (на основе параметров N, W, R). База данных NoSQL функционирует в рамках локальной вычислительной сети, и расстояние d(v,v') между узлами не играет большой роли. По этой причине предлагаются другие модели согласования данных в базах данных NoSQL.

В [11] предлагается подход к количественному измерению показателей КС-согласованности реплик в NoSQL 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) [11]. Здесь N, W, R – параметры, регулирующие требуемый уровень согласованности (см. выше).

В формуле

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

авторы попытались учесть распространение обновлений во времени [11]. Но они предлагают получить функцию $P_w(c,t)$ с помощью имитационного моделирования или измерений, что на наш взгляд является нереальным (в [11] эта функция так и не была получена). В формулах (1), (2) не учитывается интенсивность запросов на чтение.

В работе [12] 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 секунд, чтение – раз в 10 миллисекунд. Каждые 10 минут 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 значение 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 зарубежных источниках отсутствуют упоминания о разработке математических моделей для случая строгого согласования реплик. В работах [14, 15] авторами данной статьи была разработана такая модель, но натурные эксперименты показали, что она не является адекватной. Ниже разрабатывается новая модель для этого варианта согласования и анализируется ее адекватность.

## Модель оценки временных характеристик строгого согласования реплик

Как отмечалось в предыдущем разделе, гарантии строгой согласованности достигаются путем выбора параметров W и R таким образом, чтобы их сумма превышала N. В этом случае множество реплик, с которых необходимо выполнить чтение записи БД для завершения операции, и множество реплик, на которых необходимо выполнить обновление записи, чтобы считать эту операцию завершенной, всегда будут иметь непустое множество в своем пересечении. Следовательно, гарантируется чтение актуальной записи хотя бы из одной реплики, так как координатор не будет читать записи из реплик в этом пересечении пока не завершится их обновление. При этом степень согласованности растет. Но растет и задержка чтения записи БД из согласованной реплики: требование на чтение должно ожидать окончания обновления W реплик.

При разработке модели необходимо учитывать, что время выполнения операций в процессоре, передачи данных между устройствами и оперативной памятью, время передачи данных по сети и т.д. – являются случайными величинами. Оперирование большим количеством случайных величин и построение результирующе функции распределения F(t) времени ожидания или обработки запроса к базе данных является трудоемкой задачей. Переход от оригинала F(t) к его изображению – преобразованию Лапласа-Стилтьеса (ПЛС) – позволяет существенно упростить получение моментов результирующей случайной величины. В общем виде ПЛС имеет следующий вид:

$$\varphi(s) = \int_0^\infty e^{-st} dF(t). \quad (3)$$

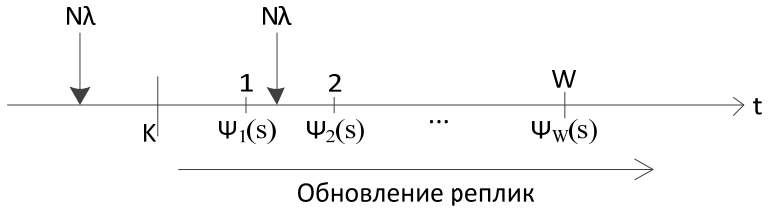



Рис. 2. Модель сильной согласованности реплик

В дальнейшем будем говорить о ПЛС функции распределения вероятностей случайной величины или просто о ПЛС случайной величины для сокращения.

Моменты соответствующей случайной величины $\xi$ можно рассчитать, дифференцируя (3) в нуле:

$$M_\xi = -\varphi^{(1)}(0),\ M_{\xi^2} = \varphi^{(2)}(0),\qquad \sigma_\xi^2 = M_{\xi^2} - M_\xi^2. \tag{4}$$

ПЛС обладает важным свойством: пусть $\xi = \sum_{i=1}^{k} \xi_i$ - сумма независимых случайных величин, тогда имеет место равенство $\varphi(s) = \prod_{i=1}^{k} \varphi_i(s)$.

На Рис. 2 представлена модель согласования данных для рассматриваемого случая. Входящий поток требований на чтение из одной реплики принимается пуассоновским с параметром λ. Обновляемые реплики обозначаются 1, 2, …, W; буквой K обозначен координатор. $\Psi_i(s)$ – преобразование Лапласа-Стилтьеса функции распределения вероятностей времени обновления i-й реплики (см. ниже). Требования на чтение записи поступают с суммарной интенсивностью Nλ. Требования могут поступить как в процессе обновления W реплик, так и до начала обновления. Задача состоит в том, чтобы оценить время ожидания требованием на чтение (вторая отметка Nλ на Рис 2) возможного окончания обновления W-й реплики.

Для решения поставленной задачи сначала предлагается получить производящую функцию (ПФ) $W_q(z)=\sum p_i z^i$ числа тех требований на чтение записи БД из N реплик, которые поступят за время обновления W реплик. Из определения ПЛС (3) следует, что требуемая ПФ числа требований на чтение из N реплик, поступивших за время обновления i-й реплики, равна $\Psi_i(\lambda N(1-z))$. Так как требования на чтение записи БД из каждой реплики поступают независимо друг от друга, ПФ числа требований на чтение из N реплик, поступивших за время обновления W реплик, будет равна:

$$W_q(z) = \prod_{i=1}^{W} \Psi_i(\lambda N(1-z)). \tag{5}$$

Тогда, согласно вероятностному смыслу производящей функции [16], вероятность того, что за время обновления W реплик не поступит ни одного требования на чтение, равна $W_q(0)$. В [16, стр. 23, 24] доказывается, что если за некоторый временной интервал поступило $n \geq 1$ требований, моменты поступления этих требований внутри данного интервала независимы и распределены по равномерному закону. Это позволяет сделать вывод, что любое из пришедших требований имеет одну и ту же функцию распределения вероятностей времени ожидания окончания обновления W реплик.

Рассмотрим интервал между началом и окончанием обновления W реплик как интервал между событиями потока событий $\varsigma_1, \varsigma_2, \varsigma_3 \ldots$ с одинаковыми функциями распределений вероятностей интервалов между этими событиями. Тогда задача сводится к тому, чтобы оценить время $t_0$ между произвольным моментом (в силу равномерности поступления требований внутри интервала) и моментом наступления следующего события $\varsigma_i$ (Рис. 3).

ПЛС Φ(s) функции распределения временного промежутка $t_0$ выводится в работе [17, стр. 36, 37] и имеет следующий вид:

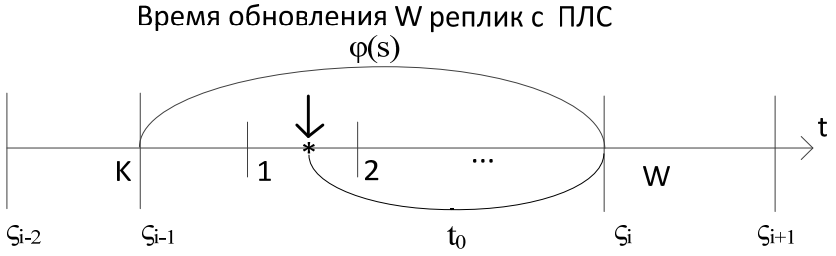



Рис. 3. Схема оценки времени ожидания $t_0$

$$\Phi(s) = \frac{1}{-s \cdot \varphi^{(1)}(0)}(1 - \varphi(s)), \quad (6)$$

где $(-1)\cdot(1/\varphi^{(1)}(0))$ – среднее количество событий потока в единицу времени, φ(s) – ПЛС функции распределения вероятностей времени между событиями потока $\varsigma_i$. В нашем случае:

$$\varphi(s) = \prod_{i=1}^{W} \psi_i(s) \quad (7)$$

где $\psi_i(s)$ – преобразование Лапласа-Стилтьеса функции распределения вероятностей времени обновления i-й реплики.

Используя формулу полной вероятности, получим ПЛС Ω(s) времени ожидания требованием на чтение окончания обновления W реплик:

$$\Omega(s) = W_q(0) + (1 - W_q(0)) \cdot \Phi(s), \quad (8)$$

где $W_q(z)$ и Φ(s) определяются соответственно выражениями (5) и (6).

Выражение (8) читается следующим образом: с вероятностью $W_q(0)$ того, что за время обновления W реплик не поступит ни одного требования на чтение из N реплик, ПЛС равно единице (время ожидания равно нулю); с вероятностью $(1\text{-}W_q(0))$ того, что за время обновления W реплик поступит хотя бы одно требование на чтение, ПЛС времени ожидания окончания обновления реплик любым из них равно ПЛС, рассчитанному по формуле (6).

Учитывая выражение для $\Phi^{(1)}(0)$ [17, стр. 37], получим математическое ожидание времени ожидания начала чтения записи из реплики (задержки чтения):

$$M = -\Omega^{(1)}(0) = -(1 - W_q(0)) \cdot \frac{\varphi^{(2)}(0)}{2 \cdot \varphi^{(1)}(0)}, \quad (9)$$

где $\varphi^i(0)$ – i-ая производная φ(s) в нуле (i-й начальный момент, умноженный на $(-1)^i$). Моменты функции φ(s) можно оценить, используя методы численного дифференцирования [18] (напрямую к (8) нельзя применить методы численного дифференцирования из-за возникающей неопределенности вида 0/0).

Получим теперь $\psi_i(s)$ – преобразование Лапласа-Стилтьеса функции распределения вероятностей времени обновления i-й реплики. Предложенное в [15] выражение для ПЛС $\psi_i(s)$ времени обновления реплики имеет составляющие следующего вида:

$$\beta(s) = \left(\frac{\mu}{\mu + s}\right)^n. \quad (10)$$

Квадрат коэффициента вариации соответствующей случайной величины рассчитывается по формуле:

$$\alpha^2 = \frac{D}{M^2} = \frac{n/\mu^2}{(n/\mu)^2} = \frac{1}{n}. \quad (11)$$

При большом n квадрат коэффициента вариации стремится к нулю. Это означает, что выражение (10) описывает более детерминированную случайную величину. Поэтому предлагается другая модель расчета времени обновления реплики.

Обозначим через Λ(s,r,t) ПЛС сетевой составляющей, а через Θ(s) - ПЛС локальной составляющей времени обновления реплики. Имеем

$$\Lambda_i(s,t,r) = (\varphi_{net}(s,r))^t \cdot (\varphi_{nm}(s))^{1-t}, \quad (12)$$

$$\varphi_{net}(s,r) = \frac{\mu_{net}(r)/(k+v)}{\mu_{net}(r)/(k+v) + s}, \quad (13)$$

$$\frac{1}{\mu_{net}(r)} = \frac{2}{\mu_m} + \frac{2}{\mu_n} + \frac{r}{\mu_{ns}},$$

$$\varphi_{nm}(s) = \frac{\mu_m/(k+v)}{\mu_m/(k+v) + s}. \quad (14)$$

Здесь параметр t=1, если узел, содержащий i-ую реплику, не совпадает с координатором (имеет место передача данных по сети), 0 – в противном случае (имеет место передача данных в памяти). Параметр r=1, если узел, содержащий i-ую реплику, находится в подсети, не содержащей координатор, 0 – в противном случае.

$\varphi_{net}(s,r)$ – ПЛС суммарного времени передачи записи по сети; $\mu_m$ – интенсивность чтения байтов данных из ОП ($2/\mu_m$ - 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); $\mu_n$ – 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; $\mu_{ns}$ – 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:

$$\Theta(s) = \frac{\mu_{crc}}{\mu_{crc}+s} \cdot \frac{\mu_{md}}{\mu_{md}+s} \cdot \frac{\mu_d}{\mu_d+s} \cdot \varphi_{kd}(s). \quad (15)$$

Здесь $\mu_{crc}$ – интенсивность расчета контрольной суммы записи:

$$\frac{1}{\mu_{crc}} = (k+v+12)\left(\frac{4}{\mu_p}+\frac{1}{\mu_m}\right), \quad (16)$$

где $\mu_p$ – количество операций, выполняемых процессором в секунду; 12 – число байт, используемых для расчета контрольной суммы (временная метка, ...); 4 – учитывает, что для расчета контрольной суммы требуется 4 процессорных операции на один байт данных.

$\mu_{md}$ – интенсивность передачи записи из ОП в буфер диска:

$$\frac{1}{\mu_{md}} = (k+v+16)\frac{1}{\mu_m}, \quad (17)$$

где 16 - число байт, выводимых на диск (контрольная сумма, временная метка, ...);

$\mu_d$ – интенсивность чтения+записи блока на диск:

$$\frac{1}{\mu_d} = wb\frac{1}{\mu_{dl}}, \quad (18)$$

где $\mu_{dl}$ – интенсивность чтения/записи байтов на диск; b – размер блока диска в байтах (может быть задан произвольно, т.к. величина $\mu_{dl}$ 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 w=1.

$\varphi_{kd}(s)$ – ПЛС времени обновления хеш-таблицы (keydir) в ОП; не зависит от размера поля «значение» записи, т.к. хеш-таблица строится по ключу размером k, значение которого соответствует структуре, размером в 20 байт, хранящей информацию о местонахождении данных на диске.

$$\varphi_{kd}(s) = \frac{\mu_m/(20+k)}{\mu_m/(20+k)+s} \cdot \frac{\mu_p/16}{\mu_p/16+s}, \quad (19)$$

где 16 - учитывает, что количество процессорных операций, необходимых для подсчета хеша, примерно равно 16.

Таким образом, ПЛС времени обновления реплики можно определить как:

$$\Psi_i(s) = \Lambda_i(s,r,t) \cdot \Theta(s), \quad (20)$$

где $\Lambda_i(s,r,t)$ и $\Theta(s)$ определяются выражениями (12) и (15).

## Анализ адекватности модели

Была выполнена серия натурных экспериментов, цель которых заключалась в оценке адекватности модели расчета среднего времени ожидания требованием на чтение окончания обновления W реплик, т.е. адекватности оценки (9).

*Описание экспериментальной установки*. На рынке существует много компаний, предоставляющих облачные вычислительные ресурсы, которые могут быть использованы для реализации натурных экспериментов. В нашем эксперименте использовались виртуальные узлы, предоставленные компанией *DigitalOcean* (DO) [19]. Т.к. 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 виртуаль-

ные 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. 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 $\mu_{ns}$ – 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 [20], предустановленная поставщиком облачных услуг. 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 системы, описанных в [21], была выполнена один раз на одном узле. Далее был сделан снимок узла, который впоследствии восстанавливался на остальных узлах.

Т.к. исследовалась строгая 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 и проведение 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 независимо поступать к каждой из 1…N реплик с интенсивностью λ. Интенсивность поступления требований на обновление записи была принята равной 1,25 (1/с). 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 выполнялись непосредственно на узлах. В явном виде сложно оценить время ожидания требованием на чтение окончания обновления W реплик, поэтому это время 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 отсутствии обновления записи БД. Читающие процессы фиксируют в журнале время, затрачиваемое на выполнение операции чтения. Далее файлы журналов передаются на локальную машину для анализа. 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». Для оценки среднего времени чтения «без обновления» использовалось 3000 итераций.

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:

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

Элементы этих множеств описаны выше при описании ПЛС времени обновления реплики. Значения общих фиксированных параметров C для всех серий экспериментов имели следующие значения: K=20 байтов – размер ключа изменяемой записи; V=1024 байта – размер значения изменяемой записи; r = 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.

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 отсутствии 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 было установить минимальный объем выборки N_ITER_W. На основании теоремы Ляпунова [22] было получено N_ITER_W=4000. Были выполнены две серии экспериментов.

В первой серии было проведено 16 экспериментов: N=5, W=R=3, λ изменялось от 3 до 10 (1/с) с шагом 1, далее N=7, W=R=4, λ также изменялось от 3 до 10 (1/с) с шагом 1. Во второй серии было также проведено 16 экспериментов: N=9, W=R=5, λ изменялось от 3 до 10 (1/с) с шагом 1, далее N=11, W=R=6, λ также изменялось от 3 до 10 (1/с) с шагом 1. Половина экспериментов (8 из 16 для каждой серии) была использована для адаптации модели, остальные – для оценки ее адекватности.

*Адаптация и анализ адекватности модели строгого согласования реплик.* Задача адаптации модели (9) решалась методом наименьших квадратов [23]:

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

где $M(C,X_i,Y)$ – аналитическое выражение (9); $M_i$ – оценка среднего времени ожидания требованием на чтение окончания обновления реплик, полученное при проведении i-ого эксперимента; L=8 – число экспериментов, по которым проводилась адаптация модели для каждой серии экспериментов. Как видно из (9), M зависит от значений производных $\varphi(s)$ в нуле; $\varphi(s)$ в свою очередь определяется выражением (7), куда входят ПЛС $\psi_i(s)$, зависящие от множества адаптируемых параметров $Y = \{\mu_n, \mu_m, \mu_{dl}\}$ (формулы (20), (12)-(19)).

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: для первой серии: $\mu_n$=441.4 Мбит/с, $\mu_m$=7460 Мбайт/с, $\mu_{dl}$=201 Мбайт/с; для второй серии: $\mu_n$=369 Мбит/с, $\mu_m$=8560 Мбайт/с, $\mu_{dl}$=152 Мбайт/с. Средняя ошибка моделирования по формуле (9) по двум сериям экспериментов составила 7,42%.

На Рис. 4 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(λ). На рисунке «мод» и «экс» обозначают графики, построенные соответственно по модели (9) и экспериментальным данным.

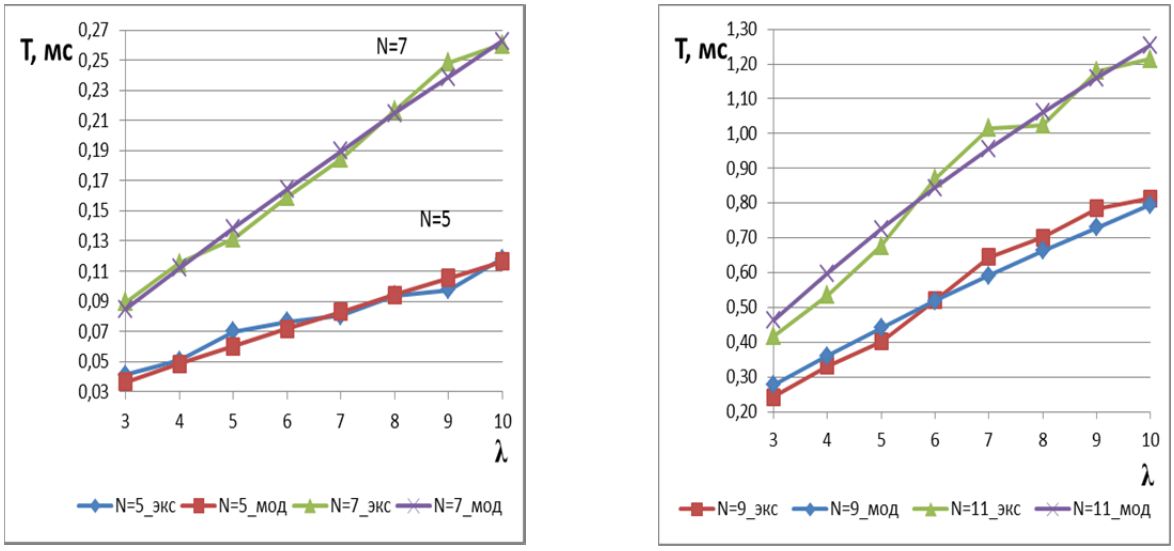
Из рисунка видно, что модель близка к линейной на указанном (рабочем) интервале изменения λ. Сама модель не является линейной. Из формул (9), (5), (20) и (12)-(19) следует, что M=T(λ) стремится к константе при дальнейшем увеличении λ ($W_q(0) \to 0$). Следует также отметить ограниченные возможности натурного эксперимента. Регрессионные зависимости, построенные по результатам натурных экспериментов, можно использовать только для тех значений параметров C, Y, которые использовались при построении этих зависимостей, а также для X из указанного диапазона. Адекватную же математическую модель можно применять для значений параметров C, X и Y, изменяемых в широком диапазоне (например, на этапе проектирования системы на основе базы данных NoSQL).

## Заключение

1. Выполнено сравнение баз данных NoSQL с реляционными базами данных. На основе публикаций выявлено, что каждый тип БД имеет свою нишу, и они еще долго будут сосуществовать вместе.

2. Разработана новая математическая модель оценки временных характеристик строгого согласования реплик в базах данных NoSQL, учитывающая случайный характер поступления запросов на чтение, а также параметры репликации (N, W) и параметры аппаратно-программного комплекса.

3. По результатам натурного эксперимента, выполненного с базой данных NoSQL Riak в облачной среде компании Digital Ocean на кластере




а) б)

Рис. 4. Зависимости среднего времени ожидания T= M(λ) при различных значениях N

а) первая серия экспериментов; б) вторая серия экспериментов

с числом узлов до 11, доказана адекватность разработанной модели (9) на линейном участке зависимости M от λ. Средняя относительная погрешность моделирования составил а 7,42%.

## Литература


1. DeWitt, David. From 1 to 1000 mips, November 2009. – 2009, PASS Summit 2009 Keynote.
2. Черняк, Л. Серьезно о технологиях для Больших Данных // Открытые системы. СУБД. - 2014. – №01. – с. 12–15.
3. Мартин Фаулер, Прамодкумар Дж. Садаладж. NoSQL. Новая методология разработки нереляционных баз данных. - М.: И.Д.Вильямс, 2013. 192 с.
4. Эндрю Павло, Эрик Паулсон, Александр Разин, Дэниэль Абади, Дэвид Девитт, Сэмюэль Мэдден, Майкл Стоунбрейкер. Сравнение подходов к крупномасштабному анализу данных. Пересказ Сергея Кузнецова. [Электронный ресурс] [http://citforum.ru/database/articles/mr_vs_dbms/] Проверено 08.07.2015.
5. Григорьев, Ю.А. Плутенко, А.Д. Анализ процесса выполнения запроса на соединение таблиц в строчной параллельной СУБД // Информатика и системы управления. – 2013. - № 4. – С. 3-15.
6. Григорьев, Ю.А. Плутенко, А.Д. Анализ времени соединения таблиц в строчной параллельной системе баз данных и по технологии MapReduce // Информатика и системы управления. – 2014. – № 2. – С. 3–11, URL: http://ics.khstu.ru/media/2014/N40_01.pdf
7. Fay Chang, JeffreyDean, Sanjay Ghemawat,WilsonC. Hsieh,Deborah A. Wallach MikeBurrows, TusharChandra, AndrewFikes, Robert E. Gruber. Bigtable: A Distributed Storage System for Structured Data // Google,Inc, OSDI 2006, - [Электронный ресурс] [http://www.cs.utexas.edu/~dahlin/Classes/GradOS/papers/chang06bigtable.pdf] Проверено 08.07.2015.
8. Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels. Dynamo: Amazon's Highly Available Key-value Store // SOSP'07, October 14–17, 2007, Stevenson, Washington, USA, pp. 205-220.
9. Григорьев, Ю.А. Плутенко, А.Д. Оценка времени соединения таблиц в базе данных NoSQL по технологии Mapreduce // Информатика и системы управления. – 2014. - № 1. – С. 3-16.
10. 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.
11. Peter Bailis, Shivaram Venkataraman, Michael J. Franklin, Joseph M. Hellerstein, Ion Stoica. Probabilistically Bounded Staleness for Practical Partial Quorums, 2012: [Электронный ресурс]. – режим доступа: http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-4.pdf (дата обращения 08.07.2015).
12. David Bermbach, Stefan Tai. Eventual Consistency: How soon is eventual? // ACM MW4SOC '11, December 12, 2011, Lisboa, Portugal. - [Электронный ресурс] [http://dl.acm.org/citation.cfm?id=2093186] Проверено 08.07.2015.
13. M. Merideth and M. Reiter. Selected results from the latest decade of quorum systems research. In Replication, volume 5959 of LNCS, pages 185–206. Springer, 2010.
14. Григорьев Ю.А. Цвященко Е.В. Сильная и слабая согласованность в базах данных NoSQL // Информатика и системы управления. – 2014. - №4. – С. 14-23.
15. 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.
16. Ивченко Г.И., Каштанов В.А., Коваленок И.Н. Теория массового обслуживания. –М.: Высшая школа, 1982. – 256 с.
17. Риордан Дж. Вероятностные системы обслуживания. – М.: Связь, 1966. – 184 с.
18. Бахвалов Н.С. Численные методы (анализ, алгебра, обыкновенные дифференциальные уравнения). - М.: Наука, 1973. - 631 с.
19. Digital Ocean. [Электронный ресурс] [https://www.digitalocean.com]. Проверено 08.07.2015.
20. Ubuntu OS 14.04. [Электронный ресурс] [http://releases.ubuntu.com/14.04] Проверено 08.07.2015.
21. Riak documentation. [Электронный ресурс] [http://docs.basho.com/index.html] Проверено 08.07.2015.
22. Теорма Ляпунова. [Электронный ресурс] [https://ru.wikipedia.org/wiki/Теорема_Ляпунова]. Проверено 08.07.2015.
23. Метод наименьших квадратов. [Электронный ресурс] [https://ru.wikipedia.org/wiki/Метод_наименьших_квадратов]. Проверено 08.07.2015.
24. NoSQL. [Электронный ресурс] [http://ru.wikipedia.org/wiki/NoSQL] Проверено 08.07.2015.

**Григорьев Юрий Александрович.** Профессор МГТУ им. Н.Э. Баумана. Окончил МГТУ им. Н.Э. Баумана в1975 году. Доктор технических наук. Автор более 100 научных публикаций. Область научных интересов: разработка методов исследования параллельных систем баз данных. E-mail: grigorev@bmstu.ru

**Цвященко Евгений Васильевич**. Аспирант МГТУ им. Н.Э. Баумана. Окончил МГТУ им. Н.Э. Баумана в 2013 году. Автор 8 публикаций. Область научных интересов: анализ процессов согласования реплик в базах данных NoSQL.
E-mail: eugene.tsviashchenko@gmail.com