УДК 004.657

© 2015 г. **Ю.А. Григорьев,** д-р техн. наук,
Е.В. Цвященко
(Московский государственный технический университет им. Н.Э. Баумана)

АНАЛИЗ НАДЕЖНОСТИ БАЗ ДАННЫХ NOSQL

В статье исследуется надежность баз данных NoSQL. Разработаны аналитическая и имитационная модели отказов для баз данных NoSQLc использованием теории массового обслуживания. Рассмотрены механизмы автоматического восстановления данных после отказа узла. Получена оценка времени восстановления узла. Выполнена оценка вероятности отказа в обслуживании при чтении записи из базы данных NoSQL. Приведен анализ полученных результатов модельных экспериментов.
Ключевые слова: база данных NoSQL, модель отказов, время наработки на отказ,вероятность отказа в обслуживании, надежность.

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) [1]. Эти системы обладают рядом преимуществ по сравнению с параллельными реляционными системами баз данных. К ним можно отнести высокую масштабируемость, возможность обработки неструктурированных данных, большое число реплик и др.

Благодаря широкому использованию репликации и хранению копий данных сразу на нескольких узлах базы данных NoSQLобладают очень высокой надежностью. Например, при настройках репликации [2] (N,W,R) = (5,1,1) система все еще будет отвечать на запросы пользователя, даже если четыре узла из пяти перестанут функционировать. Распределенные базы данных NoSQL могут быть развернуты на кластерах в тысячи узлов, т.е. обладают высокой горизонтальной масштабируемостью. Однако на больших кластерах узлы могут выходить из строя часто, а ремонтные бригады не всегда справляться с работой. Следовательно, существует определенная вероятность, что из распределенного хранилища не будет прочитана запись. Поэтому теоретическая оценка характеристик надежности кластеров большой размерности при наличии нескольких реплик каждой записи в базе данных является актуальной задачей.

В теории кворумов[3] рассматриваются византийские системы кворума, предполагающие наличие неисправных серверов в распределенной системе.

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

Пусть b > 0 – общее число неисправных серверов; B – множество неисправных серверов (которые не отвечают на запрос). Рассматриваются три случая. Во всех случаях предполагается, что существует Q∈D такое, что |B∩Q| = 0. Без этого ограничения неисправные серверы могут препятствовать работе системы, просто не отвечая, поскольку клиент, как правило, требует ответа от какого-либо полного кворума серверов, чтобы продолжить работу.

Случай 1. |Q∩Q'\B|> 0 для любых Q,Q'∈D. Это означает, что два кворума имеют по крайней мере один общий исправный сервер.

Случай 2. |Q∩Q'\B|>|Q'∩B|. Это означает, что пересечение кворумов содержит больше исправных серверов, чем неисправных в каком-либо кворуме.

Случай 3. |Q∩Q'\B|>|(Q'∩B)∪(Q'\Q)|.

Имеет место теорема[4].

Теорема 1. Пусть n = |Z| и пусть D – кворум над Z, тогда:

для случая 1 – b < n/3; для случая 2 – b < n/4; для случая 3 – b < n/5.

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. Поэтому при отказе сервера система кворумов в NoSQL автоматически перестраивается. Более того, классическая теория кворумов позволяет оценить только некоторые показатели надежности – и то на уровне нижних или верхних границ (см. теорему 1).

В [5] предлагается подход к настройке кворума, исходя из заданных вероятностей доступности узлов и соединяющих их каналов связи.Описываются два подхода к управлению репликацией данных через назначение весов (голосов) к отдельным копиям: неиерархическая схема и иерархическая.

1. *Неиерархическая схема.*

Определяется число реплик Ni узлы, где они располагаются, через уровни (вероятности) доступности узлов и связей между ними для обеспечения кворума R + W > N, W > N/2. Предлагается следующая эвристическая процедура. Рассчитывается округленный до целого вес каждого узла (v_i) через вероятность доступности узла и вероятности доступности каналов связи между этим узлом и другими узлами (сумма произведений вероятностей). N принимается равной сумме этих весов. Реплики назначаются узлам, у которых вес больше 0. Затем подсчитывается вероятность доступности всей системы. Для этого перебираются все возможные состояния системы, связанные с отказами узлов и каналов, оценивает-

ся вероятность каждого состояния и включается в итоговую сумму, если для этого состояния существует кворум чтения и записи (с учетом частоты использования кворума). Указывается, что при разрыве некоторых связей кворум записи не может быть обеспечен. Поэтому вводится процедура динамического изменения N. Такой подход к оценке надежности кластера с множеством реплик имеет существенные недостатки. Если узлы (каналы связи) имеют одинаковую вероятность доступности, то все серверы получат одинаковые веса и реплики записи следует располагать на всех узлах кластера. Более того, в реальности стратегию размещения реплик и доступа к серверам определяет не пользователь, а сама база данных NoSQL (на основе параметров N, W, R).

2. *Иерархическая схема*.

Возникает очевидный вопрос, существует ли способ управлять данными репликации, который не требует, чтобы $R + W > N$? Строится дерево с числом уровней m (от 0 до m – 1). Уровень (m – 1) соответствует множеству узлов. Узлы остальных уровней являются виртуальными группами реальных узлов. Дерево строится таким образом, что каждый узел i-го уровня имеет одинаковое число l_{i+1} дочерних узлов. Каждому узлу дерева назначается вес, равный 1. Размеры кворумов R_i и W_i на каждом уровне выбираются так, чтобы выполнялись неравенства $R_i + W_i > l_i$, $W_i > l_i/2$. Read-кворум для листовых узлов (физических) формируется следующим образом. Выбирается корневой узел, потом отмечаются R_1 узлов на 1-м уровне. Для каждого отмеченного узла из R_i отмечаются R_{i+1} узлов на (i+1)-м уровне и т.д. Узлы, отмеченные на листовом уровне, и есть кворум по чтению. Кворум по записи определяется аналогично.

Если нельзя отметить R_{m-1} узлов (так как некоторые узлы являются неработоспособными), то выполняется возврат на предыдущий уровень дерева, соответствующий родительский узел вычеркивается из рассмотрения и назначается новый узел на уровне (m – 2). Если узел нельзя назначить (остались только вычеркнутые узлы), то выполняется возврат на уровень выше и т.д.

Преимущество описанной схемы заключается в том, что кворумы могут динамически меняться в зависимости от работоспособности узлов. Но иерархическая схема более сложная, и в работе [5] не проводится оценка показателей качества такой схемы. Ее сложно применить в NoSQL.

Обе схемы, неиерархическая и иерархическая, не учитывают возможности отсутствия кворума в NoSQL (согласованность в конечном счете [2]).

Как видно из приведенного выше анализа, модели оценки характеристик надежности, основанные на кворуме, мало подходят для баз данных NoSQL. В статье разрабатывается новая модель, учитывающая особенности функционирования NoSQL.

Одной из важных таких особенностей NoSQL является то, что при невозможности чтения или обновления реплики какой-либо записи на сервере база данных NoSQL автоматически создает реплику на другом работоспособном узле (при чтении такой реплики в ней сохраняется только значение ключа). Поэтому в системе всегда присутствует N реплик записи.

При чтении записи клиент получит только значение ключа (отказ в обслу-

живании), если одновременно отказали все N серверов, где хранились реплики этой записи.

Целью данной работы является разработка модели отказа в обслуживании в базе данных NoSQL для оценки вероятности того, что при выполнении запроса на чтение клиент получит только значение ключа записи, т.е. вероятности одновременного отказа N серверов.

Аналитическая модель оценки вероятности отказа в обслуживании запросов в базе данных NoSQL

Рассмотрим кластер, состоящий из U узлов. Пусть каждый узел выходит из строя с интенсивностью δ (1/δ – среднее время наработки на отказ одного узла). Узел обслуживается ремонтной бригадой с интенсивностью μ (1/ μ – среднее время восстановления одного узла). В дальнейшем узел будем называть заявкой. Если все бригады заняты, то заявка становится в очередь, K – число обслуживающих аппаратов, т.е. ремонтных бригад. На рис. 1 схематически представлен процесс отказов и восстановления узлов кластера.

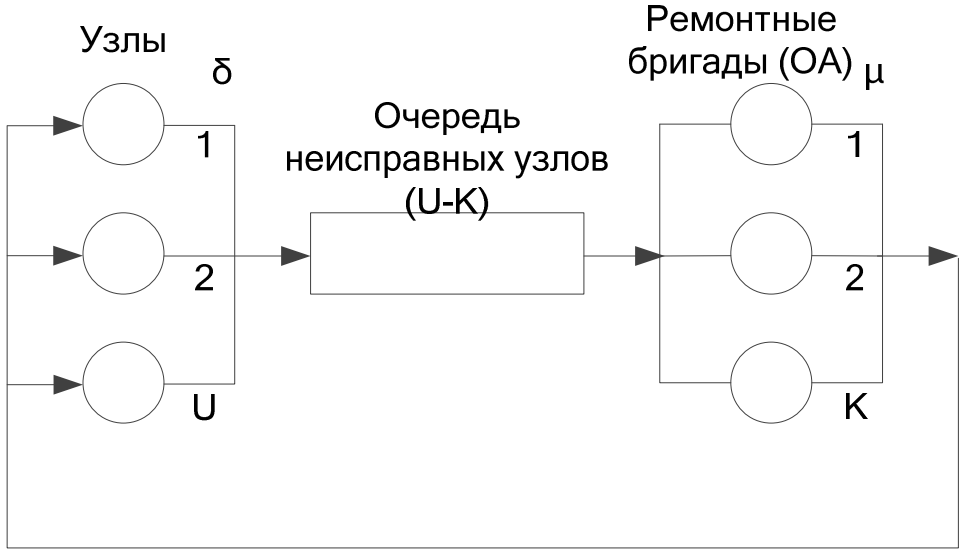

Рис. 1. Процесс выхода из строя и восстановления узлов.

Пусть время наработки на отказ узла и время его обслуживания (восстановления) распределены по экспоненциальному закону соответственно с параметром 1/δ и μ. Тогда данный процесс можно описать в виде системы массового обслуживания M/M/K/U/U (модель «ремонтника» с многоканальным обслуживанием). Для этой модели существует аналитическое решение [6, с. 126, 127]. Вероятность p_i того, что в системе (в очереди на обслуживание или в ОА) находятся i заявок, равна:

$$p_i = \begin{cases} p_0 \cdot \left(\frac{\delta}{\mu}\right)^i \cdot C_U^i, & \text{если } i < K, \\ p_0 \cdot \left(\frac{\delta}{\mu}\right)^i \cdot C_U^i \cdot \frac{i!}{K!} \cdot K^{K-i}, & \text{если } i \geq K, \end{cases} \qquad (1)$$

$$p_0 = \left[1 + \sum_{i=1}^{K-1} \left(\left(\frac{\delta}{\mu} \right)^i \cdot C_U^i \right) + \sum_{i=K}^{U} \left(\left(\frac{\delta}{\mu} \right)^i \cdot C_U^i \cdot \frac{i!}{K!} \cdot K^{K-i} \right) \right]^{-1}. \quad (2)$$

Обозначим через p_{Ni} вероятность того, что N узлов с репликами какой-либо записи находятся на обслуживании при условии, что в системе находятся i заявок ($i \geq N$):

$$p_{Ni} = \frac{C_{U-N}^{i-N}}{C_U^i}. \quad (3)$$

Тогда вероятность Pr_0 того, что на обслуживании (в очереди или в ОА) находятся все узлы, где хранятся N реплик записи, можно определить по формуле полной вероятности:

$$Pr_0 = \sum_{i=N}^{U} p_i p_{Ni}, \quad (4)$$

т.е. это вероятность, что при обработке запроса на чтение клиент получит только значение ключа записи (вероятность отказа в обслуживании).

Имитационная модель оценки вероятности отказа в обслуживании запросов в базе данных NoSQL

В рассмотренной выше аналитической модели время восстановления узла распределено по экспоненциальному закону. Но это время может обладать большой дисперсией (см. ниже). В таком случае необходимо использовать модель M/G/K/U/Uc произвольной функцией распределения вероятностей времени восстановления. Для такой модели не существует аналитического решения. Здесь необходимо использовать имитационную модель. Но как в этом случае оценить вероятность отказа Pr_0? Для решения данной задачи был разработан специальный прием, который описан ниже при рассмотрении структуры имитационной модели на языке GPSS,представленной на рис. 2.

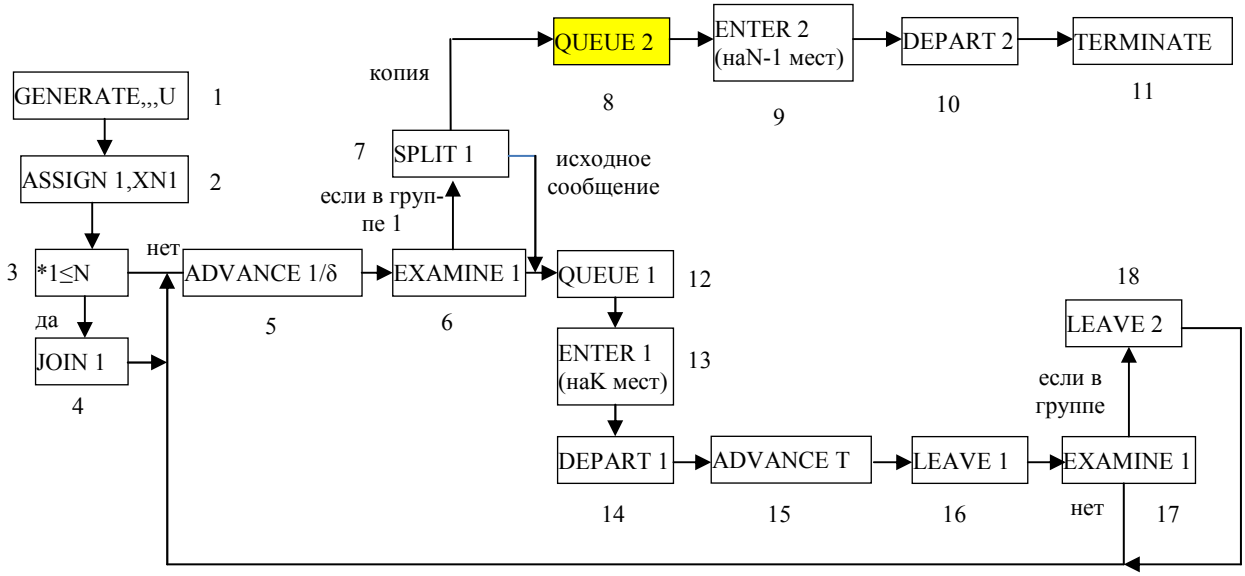

Рис. 2. Структура имитационной модели на языке GPSS.

Генерируются U транзактов (узлов) (см. метку 1); в параметре 1 транзакта сохраняется номер транзакта (метка 2); первые N тразактов объединяются в группу 1 (метки 3, 4).

Примечание. Группа 1 – узлы, где хранятся N реплик какой-либо записи. Это первые N транзактов (в силу симметричности модели номера узлов не играют роли).

Далее транзакты задерживаются на время безотказной работы (экспоненциальное распределение со средним $1/\delta$) (см. метку 5). После отказа узла определяется, принадлежит ли транзакт группе 1 (метка 6). Если «да» (отказал узел с репликой), то генерируется копия транзакта (метка 7).

Копия передается на вход очереди 2 (метка 8), затем на вход памяти (метка 9) (емкость памяти равна N – 1), далее очередь 2 уменьшается (метка 10) и копия удаляется из системы (метка 11).

Примечание. Копия транзакта (реплика записи) задерживается в очереди 2 (метка 8) только в случае, если в системе уже присутствуют неисправные узлы с N – 1 репликами. В таком случае время ожидания транзакта в этой очереди равно времени пребывания всех узлов с N репликами в системе (отказали все узлы с репликами). Доля суммарного времени ожидания от всего времени моделирования и есть искомая вероятность Pr_0. Эта доля равна значению, которое хранится в СЧА QA2 (средняя длина очереди 2 – в очереди может быть или 0 транзактов, или 1).

Основной транзакт группы 1 или транзакт, не принадлежащий группе, передается на вход очереди 1 отказавших узлов (метка 12). Далее он занимает место в памяти 1 (метка 13), если имеется свободное место (это многоканальное устройство – его емкость равна числу ремонтных бригад K). Затем очередь 1 уменьшается (метка 14), т.е. узел принимается на обслуживание ремонтной бригадой (метка 15). T – случайное время восстановления узла. Далее ремонтная бригада освобождается (метка 16). Если транзакт не входит в группу исследуемых реплик (метка 17), то он возвращается в блок с меткой 5. Если транзакт входит в группу 1, то перед этим он уменьшает память 2 на 1 (метка 18) – реплика восстановлена.

Если N = 1 (нет репликации), то в имитационной модели надо убрать очередь 2 (т.е. блоки QUEUE 2 и DEPART 2 – см. рис. 2) и установить объем памяти 2, равный 1. Тогда $Pr_0 = NA2$ (коэффициент использования памяти 2).

Данная модель была реализована в среде GPSS [7]. В качестве функции распределения вероятностей времени восстановления узла использовалась следующая дискретная функция GPSS ($P_2 > P_3$):

RECV FUNCTION RN3, D3

$P_3,(T_{F3}+T_{C3}) / (P_3+ P_2), (T_{F2}+T_{C2})/1, (T_{F1}+T_{C1})$. (5)

Выражения для оценки величин P_i, T_{Fi}, T_{Ci} приведены в следующем разделе.

Оценка времени восстановления узла

Для оценки времени восстановления узла с репликой записи определим три варианта выхода узла из строя.

A1. Случайный сбой. Для устранения неисправности достаточно просто пе-

резапустить операционную систему. Обозначим вероятность такого события через P_1.

А2. Выход из строя системного блока, но диски не повреждены. Неисправность устраняется на месте или станция заменяется из ЗИП (запасные изделия прилагаемые), но со старыми дисками. Обозначим вероятность такого события через P_2.

А3. Повреждены диски. Станция заменяется из ЗИП с новыми дисками. Обозначим вероятность такого события через P_3.

Тогда P_i, i = 1,2,3 – вероятность i-го события в момент отказа и $P_1+P_2+P_3 = 1$.

В базах данных NoSQL используются следующие технологии автоматического копирования данных на восстановленный узел.

В1. Временная передача ответственности (*hintedhandoff*) [8].

Если узел временно выключен или недоступен во время операции записи, то реплика, которая обычно находилась бы на узле A, теперь будет направлена на узел D. Это делается для того, чтобы сохранить требуемую доступность и гарантии надежности. Реплика, отправленная на узел D, содержит метку «*hint*» в своих метаданных, в которой указывается, какой узел был предполагаемым получателем реплики (в данном случае узел A). Узлы, которые получают помеченные реплики, хранят их в отдельной локальной базе данных, которая периодически сканируется по протоколу сплетен [9]. Если узел A восстановит работоспособность, то узел D будет пытаться доставить помеченную реплику на узел A. После успешной передачи узел D может удалить объект из локального хранилища, не уменьшая при этом общего количества реплик в системе.

В2. Обработка долговременных сбоев (*permanentfailures*): синхронизация реплик.

Механизм временной передачи ответственности отлично работает, если из строя выходят небольшое число узлов и их работа восстанавливается через некоторое время. Есть сценарии, при которых помеченные реплики становятся недоступными, прежде чем они вернутся на исходный для данной реплики узел. Обозначим через $T^{ГР}$ время, после которого временная передача ответственности перестает работать.

Чтобы справиться с подобными угрозами, в системах NoSQL для повышения надежности реализован протокол для противодействия энтропии (протокол синхронизации реплик, *activeanti-entropy, AAE*) [10]. С целью ускорить обнаружение несоответствий между репликами и минимизации объема передаваемых данных используют деревья Меркле (*Merkletrees*) (в дальнейшем – просто деревья).

В3. Обработка сбоев без использования деревьев. То же самое, что В2, но копирование выполняется на пустые диски восстановленного узла (так как станция заменяется из ЗИП с новыми дисками).

Далее оцениваются характеристики случайного времени T восстановления узла (см. μ на рис. 1 и T на рис. 2, метка 15).Среднее значение этого времени определим так:

$$T_{ср} = 1/\mu = P_1(T_{F1} + T_{C1}) + P_2(T_{F2} + T_{C2}) + P_3(T_{F3} + T_{C3}), \qquad (6)$$

где T_{Fi} – среднее время устранения неисправности для варианта Ai выхода узла из

строя; T_{Ci} – среднее время копирования данных на восстановленный узел для варианта Ai выхода узла из строя. Рассмотрим оценки T_{Fi} и T_{Ci} для различных вариантов.

1. Вариант отказа A1 (перезапуск операционной системы). Время устранения неисправности$T_{F1}<T^{ГР}$. Имеет место способ B1 копирования записей на восстановленный узел (временная передача ответственности).

$$T_{C1} = T_C(T_{F1}) = [V/\mu_{DR} + (V + (N-1)\cdot V)(1/\mu_N + 1/\mu_{DW})](1 - e^{-\lambda T_{F1}}), \qquad (7)$$

где V – средний объем записей (документов), хранимых на одном физическом узле (основные реплики записей узла); N – среднее число реплик на одну запись; (N – 1)V – объем записей-реплик на данном узле от (N – 1) узлов по кольцу против часовой стрелки; λ – интенсивность обновления одной записи базы данных, 1/с; $(1 - e^{-\lambda T_{F1}})$ – вероятность, что за время устранения неисправности поступит хотя бы одно обновление какой-либо записи базы данных (БД); μ_{DR} – интенсивность чтения данных с диска, байт/с (неразделяемый ресурс для разных узлов, откуда выполняется копирование; чтение с диска – параллельно); μ_N – интенсивность передачи данных по сети, байт/с (разделяемый ресурс); μ_{DW} – интенсивность копирования (записи) данных на восстановленный диск, байт/с (разделяемый ресурс).

Докажем (7). Среднее время копирования одной записи равно

$$t_1 = [L/\mu_{DR} + L(1/\mu_N + 1/\mu_{DW})](1 - e^{-\lambda T_{F1}}), \qquad (8)$$

где L – длина записи (байт). Суммируя (8) по всем записям восстановленного узла с учетом наличия неразделяемых и разделяемых ресурсов, получим (7).

2. Вариант отказа A2 (неисправность устраняется на месте или станция заменяется из ЗИП со старыми дисками). Если время устранения неисправности $T_{F2} < T^{ГР}$, имеет место способ B1 копирования записей на восстановленный узел: $T_{C2} = T_C(T_{F2})$, иначе восстановление происходит согласно способу B2 (синхронизация реплик):

$$T_{C2} = V_H/\mu_{DR} + (V_H + (N-1)\cdot V_H)/\mu_N + T_M + T_C(T_{F2}), \qquad (9)$$

где V_H – средний объем дерева Меркле на один виртуальный узел (v-узел); N – среднее число реплик на одну запись; $(N – 1)V_H$ – объем деревьев Меркле на данном физическом узле от (N – 1) узлов по кольцу против часовой стрелки; T_M – среднее время сравнения хешей деревьев Меркле в ОП узла; $T_C(T_{F2})$ – время чтения, передачи и сохранения новых обновлений записей, пришедших за время устранения неисправности

$$V_H = L_H(Q_H + \frac{Q_H}{2} + \frac{Q_H}{2^2} + \ldots + \frac{Q_H}{2^{\log_2 Q_H}}) = L_H(2Q_H - 1), \qquad (10)$$

где L_H – длина поля хеша, байт; $Q_H = Q/S$ – число записей базы данных в одном v-узле (секции); Q – число записей (документов), хранимых во всей базе данных (основные реплики записей); S – число v-узлов; в выражении (10) в скобках указано число узлов дерева Меркле на каждом уровне.

$$T_M = (2Q_H - 1 + (N-1)(2Q_H - 1))\cdot(3/\mu_p) = N(2Q_H - 1)\cdot(3/\mu_P) \qquad (11)$$

здесь сравниваются пары значений хешей на всех уровнях дерева ($2Q_H$-1) для ка-

ждого v-узла физического узла, включая секции реплик других узлов (N – 1); 3 – учитывает сравнение пары хешей и перемещение к следующей паре; μ_P – число циклов, выполняемых процессором в секунду.

3. Вариант отказа А3 (станция заменяется из ЗИП с новыми дисками).Время устранения неисправности – T_{F3}. Восстанавливаются все реплики путем их копирования из других узлов

$$T_{C3} = V/\mu_{DR} + (V + (N-1)\cdot V)(1/\mu_N + 1/\mu_{DW}). \quad (12)$$

Процесс получения формулы (12) аналогичен (7), но восстанавливаются все записи узла.

Пример анализа характеристик надежности базы данных NoSQL

Разработанные модели были использованы для оценки вероятности отказа в обслуживании при чтении записи из базы данных NoSQL.

Прежде всего необходимо оценить вероятности P_1, P_2, P_3 (случайный сбой, выход из строя системного блока, повреждены диски) при условии, что сервер отказал. В [11] приводится сравнение кластера и «обычного» сервера. В этой работе описан расчет вероятности отказа сервера за год, которая складывается из вероятностей отказа комплектующих. На основе этих данных можно сделать вывод, что вероятность P_2 превышает вероятность P_3примерно на порядок, поскольку часто выходят из строя блоки питания, элементы охлаждения и т.д. Но аппаратное обеспечение выходит из строя гораздо реже, чем программное, поэтому положим $P_1 = 5(P_2+P_3)$, $P_2 = 10P_3$. Так как $P_1+P_2+P_3 = 1$, определим вероятности следующим образом: $P_1 = 0.835$; $P_2 = 0.15$; $P_3 = 0.015$.

В [11] вероятность отказа P_{ph} сервера за период t =12 месяцев равна 0.457; так как вероятность P безотказной работы за период t рассчитывается по формуле $P = \exp(-\delta_{ph}\cdot t)$, где δ_{ph} – интенсивность отказа, то имеем $\delta_{ph} = -\ln(1 - 0.457)/12 = 0.05$ – интенсивность отказа физических компонентов узла. Вероятность отказа программного обеспечения (операционной системы), согласно предположениям, в пять раз превышает вероятность отказа комплектующих, поэтому примем интенсивность отказа узла $\delta = 6\delta_{ph} = 0.3$. Получаем среднее время наработки на отказ $1/\delta = 3.33$ месяца. Для расчетов примем $1/\delta = 3$ мес.

В расчетах по умолчанию были использованы следующие исходные данные: U – число узлов кластера – 3000; N – число реплик – 3; K – число ремонтных бригад – 2; V – объем основных реплик записей узла – 1 Гбайт; S – число виртуальных узлов – 8192 (в каждом узле располагаются S/U виртуальных узлов); Q – число записей БД – 1млрд; λ – интенсивность обновления какой-либо записи в БД – 0.1 (1/с); T_{F1} – время устранения неисправности по варианту отказа А1, т.е. время перезапуска операционной системы – 600 с; T_{F2} –время устранения неисправности по варианту отказа А2, т.е. время устранения неисправности на месте или восстановления узла из ЗИП но со старыми дисками – 4 часа; согласно документации Cassandra, временные реплики хранятся 3 часа, примем это значение за $T^{ГР}$, тогда $T_{F2} > T^{ГР}$ и имеет место В2 способ автоматического восстановления данных узла; T_{F3} – время устранения неисправности по варианту отказа А3, т.е. время восстановления дисков из ЗИП и время восстановления программного обеспече-

ния – 5 часов; L_H – размер хеша в деревьях Меркле – 25 байт.

Значения интенсивностей обработки данных в устройствах получены при помощи программы синтетических тестов AIDA[12]: μ_N – интенсивность передачи данных по сети – $12.5 \cdot 10^6$байт/с; μ_P – производительность процессора – 2 млрд. операций в секунду; μ_{DR} – интенсивность чтения байтов с диска – 80 Мбайт/с; μ_{DW} – интенсивность вывода байтов на диск – 30 Мбайт/с.

Результаты модельных экспериментов представлены на рис. 3 – 7. При построении графиков для вероятности отказа в обслуживании Pr_0 принят логарифмический масштаб. На графиках «расчет» обозначает зависимости, построенные по формуле (4), «имит» – зависимости, полученные по результатам имитационного моделирования.

При очень малых вероятностях имитационная модель при тех же исходных данных дает нулевые значения ($Pr_0 = 0$ на рис. 3 – 5), в этом случае лучше использовать аналитические данные. Из рис. 3 видно, что сначала вероятность отказа растет медленно при увеличении числа узлов в кластере до 2000 (при увеличении числа узлов в 2000/100 = 20 раз вероятность возросла только в 4,2E–10/1,2E–10 = =3,5 раза). При увеличении числа узлов с 3000 до 5000 вероятность возрастает на 5 порядков.

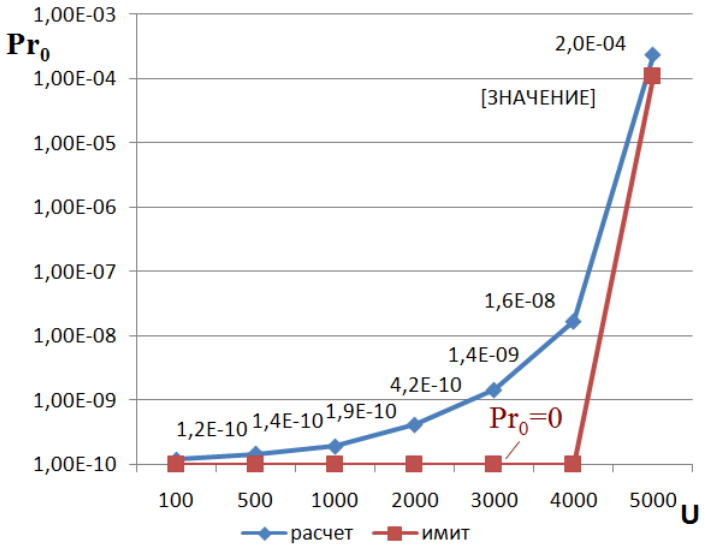

Рис. 3. Зависимость вероятности Pr_0 от числа узлов U.

Необходимо подчеркнуть, что вероятность отказа в обслуживании (чтении записи) должна быть очень небольшой. Согласно [13], число ежедневных поисковых запросов в Facebook равно более 1 млрд. (данные на декабрь 2014). Поэтому число читаемых записей также равно не менее 1 млрд. Если вероятность Pr_0 отказа в чтении одной записи равна порядка 10^{-10}, то среднее число отказов в обслуживании равно примерно $10^{+9} \cdot 10^{-10} = 0{,}1$ в день или 36,5 в год (среднее суммы случайных величин равно сумме средних независимо от степени коррелированности этих величин). При $Pr_0 \sim 10^{-4}$ число отказов в день возрастет до 10^5 – это потенциальное количество жалоб в адрес компании.

Из рис. 4 видно, что одна ремонтная бригада не справляется с интенсивностью отказов от U=3000 станций ($Pr_0 \sim 10^{-2}$). Для обеспечения вероятности отказа порядка 10^{-10} требуется 3, 4 бригады (для N = 3). Причем дальнейшее увеличение

числа ремонтных бригад не приводит к существенному уменьшению вероятности Pr_0.

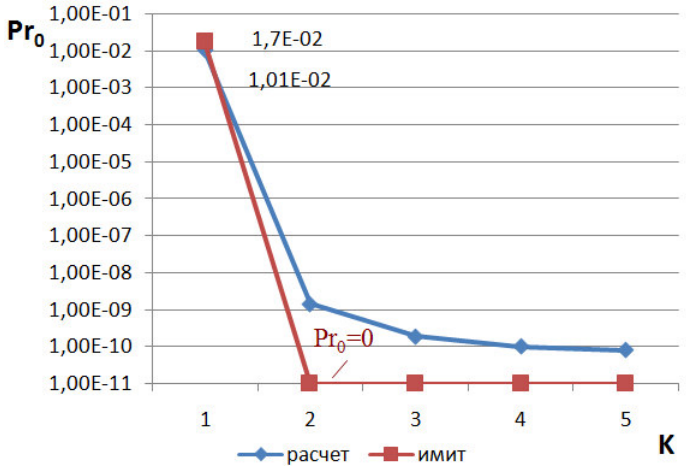

Рис. 4. Зависимость вероятности Pr_0 от числа ремонтных бригад K.

График на рис. 5 показывает, что при увеличении числа реплик записи вероятность отказа Pr_0 уменьшается по показательному закону: при увеличении числа реплик только на 1 вероятность уменьшается на 2 порядка. Для обеспечения надежности $Pr_0 \sim 10^{-11}$ при K = 2 требуется назначать число реплик не менее 4.

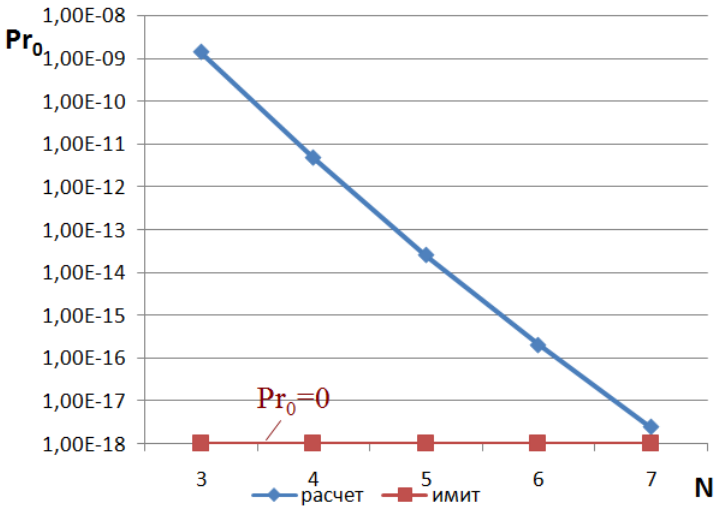

Рис. 5. Зависимость вероятностиPr_0 от параметра N.

Анализ результатов аналитического моделирования показал, что при изменении среднего времени наработки на отказ существует время $T_{кр} = 1/\delta_{кр}$, меньше которого Pr_0 возрастает сразу на несколько порядков. На рис. 6 показаны зависимости $T_{кр}$ от U при различных значениях числа ремонтных бригад K (N = 3).

Интересно, что графики имеют очень пологий участок(и)при каждом K, т.е. $T_{кр}$ практически не меняется на этом участке.

На рис. 7 представлена зависимость вероятности Pr_0 от числа узлов U при N =1 (отсутствует репликация). В этом случае надежность системы нельзя признать приемлемой для режима интенсивного чтения записей из базы данных ($Pr_0 \sim 10^{-4} \div 10^{-2}$).

Следует отметить, что разница между результатами аналитического и имитационного моделирования невелика (при не очень маленьких значениях вероят-

ности $Pr_0 \sim 10^{-4} \div 10^{-2}$ – см. рис. 3, 4, 7): сохраняется порядок значения вероятности и соответствующие значения отличаются не более чем в 2 раза.

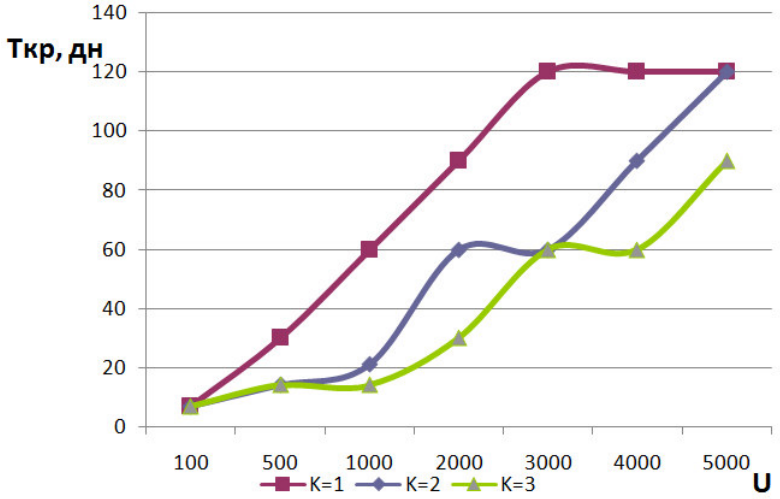

Рис. 6. Зависимость времени $T_{кр}$ от числа узлов U при различных значениях K.

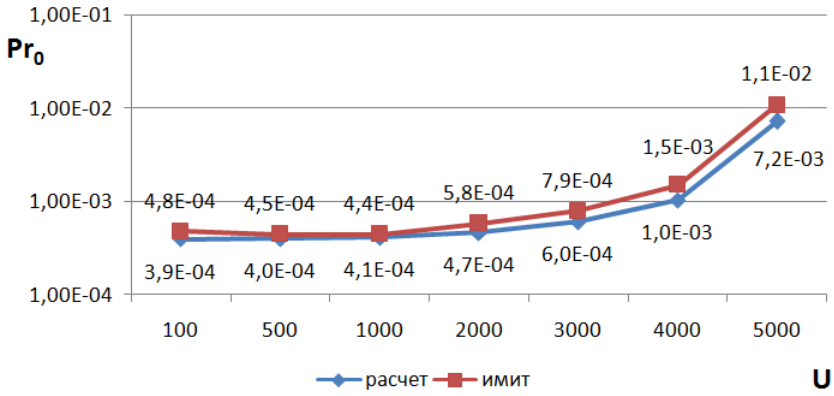

Рис. 7. Зависимость вероятности Pr_0 от числа узлов U при N = 1 (K = 2, 1/δ = 3 мес.).

Заключение

Разработаны аналитическая и имитационная модели оценки вероятности отказа в обслуживании при чтении записи из базы данных NoSQL, учитывающие наличие N реплик записи.

Рассмотрены три варианта выхода узла кластера из строя (случайный сбой, неисправность системного блока, повреждение дисков) и три варианта автоматического копирования данных на восстановленный узел (временная передача ответственности, обработка долговременных сбоев с использованием деревьев Меркле, копирование на пустые диски). Получены выражения для среднего времени копирования для каждого варианта отказа.

Выполнены расчеты с использованием разработанных моделей. Вероятность отказа возрастает на 5 порядков при увеличении числа узлов с 3000 до 5000. Показано, что при большом числе поисковых запросов в день вероятность отказа в обслуживании должна быть очень маленькой (порядка 10^{-10}). Для достижения этой вероятности необходимо либо увеличивать число ремонтных бригад (K = 3÷4 при N = 3), либо увеличивать число реплик записи (N = 4 при K = 2). По-

строены зависимости среднего времени наработки на отказ, меньше которого вероятность отказа возрастает на несколько порядков от числа узлов U в кластере. Показано, что при отсутствии репликации надежность системы NoSQL нельзя признать приемлемой для режима интенсивного чтения записей из базы данных ($Pr_0 \sim 10^{-4} \div 10^{-2}$).

При сравнении результатов аналитического и имитационного моделирования было выявлено, что порядок значений вероятности отказа в обслуживании сохраняется и соответствующие значения отличаются не более чем в 2 раза.

ЛИТЕРАТУРА

1. NoSQL. [Электронный ресурс] [http://ru.wikipedia.org/wiki/NoSQL] Проверено 01.07.2015.
2. *Редмон Э., Уилсон Д.Р.* Семь баз данных за семь недель. Введение в современные базы данных и идеологию NoSQL. – М.: ДМК Пресс, 2013.
3. *Merideth M., Reiter M.* Selected results from the latest decade of quorum systems research // In-Replication –2010. – Vol. 5959 ofLNCS. – P. 185-206.
4. *Malkhi D., Reiter M.* Byzantine quorum systems // Distributed Computing. – 1998. – № 11(4). – P. 203-213.
5. *Koren Israel, Krishna C. Mani.* Fault tolerant systems. – Elsevier, 2007.
6. *Клейнрок Л.* Теория массового обслуживания. – М.: Машиностроение, 1979.
7. GPSS WorldReferenceManual. [Электронный ресурс] [http://www.minutemansoftware.com/reference/reference_manual.htm] Проверено 01.07.2015.
8. Datastax. Hintedhadeoff. [Электронный ресурс] [http://www.datastax.com/dev/blog/modern-hinted-handoff]. Проверено 01.07.2015.
9. Gossipprotocol. [Электронный ресурс] [https://en.wikipedia.org/wiki/Gossip_protocol]. Проверено 01.07.2015.
10. Activeanti-entropy. [Электронный ресурс] [http://kellabyte.com/2013/07/11/active-anti-entropy/] Проверено 01.07.2015.
11. Сравнение кластера надежности и обычного сервера. [Электронный ресурс] [http://www.team.ru/server/stbl_compare.shtml] Проверено 01.07.2015.
12. AIDA64 ExtremeEdition. [Электронный ресурс] [http://www.aida64.com/product/aida64-extreme-edition/overview] Проверено 22.03.2014.
13. Фролова Е. Фейсбуквцифрах 2015.[Электронный ресурс] [http://www.pro-smm.com/facebook-2015/] Проверено 18.07.2015.

E-mail:
Григорьев Юрий Александрович – grigorev@bmstu.ru;
Цвященко Евгений Васильевич – eugene.tsviashchenko@gmail.com.