Экспериментальная оценка ресурсной эффективности схем данных NoSQL в условиях заданной ИТ-инфраструктуры

Д. Ю. Ильин, Е.В. Никульчев

***Аннотация* — В статье представлен анализ эффективности двух схем NoSQL баз данных для сохранения потока слабоструктурированных данных с заданными характеристиками. Для экспериментальных исследований был подготовлен фреймворк на основе виртуальных машин, имитирующих работу веб-сервиса по сбору данных с классической трёхуровневой (three-tier) архитектурой. Рассмотрена ситуация, когда поток данных превышает ресурсные возможности веб-сервиса. На основе проведенных экспериментов проведен сравнительный анализ вычислительных ресурсов для заданных вариантов. Анализ результатов показал, что различия в схеме базы данных оказывают влияние на надёжность и эффективность работы системы в целом. В этой работе предлагается фреймворк и методика оценки эффективности и надёжности информационных систем с заданными схемами данных на примере MongoDB. Предложенная методика проведения экспериментов может быть использована с другими программными компонентами и системами управления базами данных. Важность исследований определяется необходимостью анализа влияния структур данных на эффективность и надежность работы вычислительных комплексов и систем обработки больших данных.**

***Ключевые слова* — NoSQL, схема данных, веб-сервис, эффективность, надёжность, эксперимент.**

I. Введение

В последнее десятилетие многие системы сбора данных представляют собой веб-приложения. Это открывает возможности подключения к вычислительным сервисам большего количества пользователей, независимо от используемого ими гаджета и операционных систем. Изменения в дизайне и архитектуре программного обеспечения и требованиях к приложениям определило требования хранения больших данных [1]. Реляционные базы данных не могут достаточно быстро масштабироваться для хранения больших данных на веб-серверах. Приложения для мобильных устройств также могут хранить данные на облачных серверах, а скорость доступа к ним является значимой, так как пользователи ожидают быстрого отклика [2]. Доступ к большому количеству пользователей определяет и развитие систем, ориентированных на задействование большого количества людей в сборе информации. Например, проведение популяционных веб-исследований в области психологии [3]. Большие данные создали реальные вызовы по обеспечению масштабируемости и производительности систем, основанных на реляционных базах данных [4]. Чтобы решить эти проблемы, компании, предоставляющие крупномасштабные веб-сервисы, разработали новые решения — это NoSQL базы данных, которые поддерживают отличную от реляционной (структурированной) форму хранения. Базы данных на основе документов позволяют хранить документы, состоящие из размеченных элементов, и они являются наиболее популярными NoSQL-технологиями.

MongoDB – это NoSQL система управления базами данных с открытым исходным кодом, которая хранит коллекции документов, не имеющие предопределённой схемы. Каждая запись или набор данных называется документом, совокупность которых объединяется в коллекции. MongoDB – решение, разработанное с целью повысить масштабируемость операций, выполняемых с крупномасштабными данными [5]. СУБД использует нотацию BSON (бинарный вариант нотации объектов JavaScript) в качестве модели документа данных, так как она удобна для программной обработки и предлагает высокую производительность. Документы могут иметь различную структуру. Индексы для некоторых атрибутов документов могут быть созданы для ускорения выполнения некоторых операций. MongoDB доказала свою масштабируемость на практике и часто используется для мобильных приложений, в управлении контентом, при анализе данных в реальном времени [6].

Однако имеют место значительные функциональные и нефункциональные требования [7], предъявляемые к структурированию и организации хранения данных. Их формируют заданная вычислительная архитектура, конфигурация вычислительной системы, возможности каналов связи. При высокой интенсивности запросов всё это может оказывать влияние на выбор как используемых технологий в вычислительном комплексе, так и используемых протоколов передачи данных.

Из практики известен ряд примеров, когда плохо проработанная система не справляется с загрузкой. В ходе

Статья получена 23 апреля 2020.
Работа выполнена при финансировании РФФИ, грант 17-29-02198.
Ильин Дмитрий Юрьевич, аспирант кафедры управления и моделирования систем, МИРЭА – Российский технологический университет; главный аналитик Дата Центра Российской академии образования (i@dmitryilin.com).
Никульчев Евгений Витальевич, доктор технических наук, профессор, профессор кафедры управления и моделирования систем, МИРЭА – Российский технологический университет; главный аналитик Дата Центра Российской академии образования (email: nikulchev@mail.ru).

проектирования цифровых платформ нами разрабатывается и применяется подход, заключающийся в экспериментальном исследовании рассматриваемых технологий в условиях, имитирующих функционирование взаимосвязанных модулей в имитационной виртуальной среде.

В системах сбора больших данных возможности обработки запросов в режиме реального времени ограничены ресурсами вычислительной инфраструктуры – ресурсами сервера, каналами связи. Когда число запросов превышает возможности сервера, они добавляются в очередь на обработку и ожидают высвобождения ресурсов. Если очередь становится слишком большой, случается ситуация «отказа обслуживания» (Denial of Service). При ограниченном объёме оперативной памяти (выделенной для процесса или доступного операционной системе) произойдёт отказ работы серверного программного обеспечения. Поступившие данные, оставшиеся в оперативной памяти, при таком сбое будет потеряны. Поэтому выбор в пользу той или иной системы организации данных является одной из важных задач.

Одной из причин увеличения очереди запросов может быть недостаточная пропускная способность подсистемы ввода-вывода или программных компонентов взаимодействия с ней. В условиях, когда производительность является критически важной [8], необходимо оценить ресурсную эффективность операций чтения-записи, на которой основаны системы NoSQL [9]. Исследования в области производительности NoSQL связаны с физическим уровнем и зависят от объема обрабатываемых данных [10]. Таким образом, эффективность структуры хранения данных напрямую определяет объем данных, который может быть физически обработан в единицу времени.

В целом, задача выбора структуры требует в каждом конкретном случае проведения специального ресурсного исследования [11]. Выявлено, что современные методы проектирования баз данных не учитывают нефункциональные требования [12]. В работе на примере рассматриваемой системы разрабатывается методика проведения экспериментальных исследований, направленных на получение адекватных оценок в заданных условиях функционирования. Необходимо заметить, что полученные результаты и выводы применимы только для рассматриваемой системы в условиях заданного количества пользователей и конфигурации инфраструктуры. Однако полученная методика может быть полезна разработчикам информационных систем для решения проблемы выбора способа организации хранения слабоструктурированных данных.

II. Методы исследования и эксперименты

Исследования проведены для разработки конкретной цифровой платформы в конкретной конфигурации вычислительной инфраструктуры. Подразумевается, что система собирает данные по каналам связи с одновременно более чем 20000 участников в условиях ограниченности вычислительных ресурсов. Данные первоначально собираются предварительно загруженными клиентскими веб-интерфейсами с использованием средств браузера. После полного или частичного сбора данных полученные результаты проходят предварительную обработку на клиенте и передаются на сервер в слабоструктурированном формате JSON по протоколу HTTP. Затем они сохраняются в документоориентированную СУБД MongoDB 4 версии с движком хранилища WiredTiger. К особенностям СУБД относится возможность хранения слабоструктурированных данных, что может быть подходящим инструментом для денормализации базы данных.

Исходя из предполагаемого характера обмена данными, обеспечение производительности операций записи становится приоритетным. Соотношение операций чтения и записи таково, что запись будет проводиться тысячами пользователей, а чтение - единицами. Важно не потерять данные и оперативно их сохранить. Время последующей выборки и анализа данных не столь существенно.

Для поставленной задачи важен выбор между возможными вариантами структур данных, которые необходимо сохранять в базу данных. Для рассмотрения возьмём два возможных варианта:

1) сохранение результатов исследования каждого участника в качестве документа в одной коллекции данных в виде целостного документа (это также можно назвать денормализованным представлением данных);

2) сохранение результатов исследования каждого участника с разделением на несколько документов в отдельные коллекции данных (при этом в той или иной степени могут быть применены принципы нормализации базы данных, например, для снижения структурной сложности данных).

MongoDB — это база данных NoSQL, написанная на C++, где данные хранятся не в таблицах, а в виде плоских файлов в формате BSON (двоичная нотация объектов JavaScript) [13]. MongoDB имеет децентрализованную структуру, которая позволяет использовать распределенные ресурсы. Эта функция обеспечивает горизонтальную масштабируемость.

MongoDB для развертывания в кластере задействует следующие компоненты:

- Mongod: основной процесс (демон) для управления доступом к данным. Это сервер базы данных.
- Mongos: процесс, отвечающий за маршрутизацию между пользовательским приложением и базой данных MongoDB.
- config.server: хранит метаданные о размещении документов на серверах базы данных для эффективной обработки запросов пользовательского приложения.

Сервера базы данных могут быть сгруппированы следующим образом:

- Replica set: группа процессов Mongod, которые хранят одинаковые копии данных. Первичная реплика хранит основные копии документов. Вторичные реплики дублируют первичную. В случае сбоя первичной реплики запускается процесс голосования, определяющий новую первичную реплику большинством голосов.
- Shard: это набор реплик, который хранит часть базы данных. Используется для распределения фрагментов базы данных на несколько серверов.

Клиент (сервер приложений) подключается к маршрутизатору базы данных (mongos). Затем с помощью конфигурационного сервера (config.server) происходит определение, где находятся запрашиваемые данные или куда записывать новые. Mongos подключается к Shard (множеству реплик), отвечающему за диапазон значений для некоторых индексированных значений в базе данных, чтобы получить запрошенные данные и затем вернуть их в приложение пользователя. Требуется настроить индекс уникального значения в коллекции, используемый для балансировки загрузки данных между несколькими наборами реплик.

Наборы реплик (shard) используются для обеспечения масштабирования базы данных. Они предоставляют возможность распределения фрагментов данных между несколькими серверами MongoDB. Каждый из них имеет основную реплику mongod (сервер MongoDB) и ноль или несколько вторичных mongod. В MongoDB также работает балансировщик для распределения данных по наборам реплик в режиме реального времени. MongoDB поддерживает разные конфигурации.

В MongoDB существует два типа схем репликации: Master-Slave Replication и Replica-Set Replication.

Replica-Set обеспечивает лучшую автоматизацию и обработку при сбоях. Независимо от числа реплик в шарде только одна реплика выступает в качестве основной (первичной). Все операции записи и чтения отправляются первичной реплике, а затем они дублируются на вторичные реплики.

Исходный набор слабоструктурированных данных имеет объём ~1.2 ГБ. В нём имеются 34132 документа ResearchSubject и 16883 документа ResearchResult. Документы ResearchResult содержат вложенные документы ResearchResultAnswer, количество которых составляет 4934887 экземпляров.

Для постановки эксперимента на локальной виртуальной инфраструктуре использовались следующие инструменты:

- VirtualBox – система виртуализации;
- Vagrant – инструмент управления виртуальной инфраструктурой;
- Ansible – система провизии;
- Ubuntu Xenial – гостевая операционная система;
- Atop – инструмент мониторинга производительности;
- Gulp.js – инструмент автоматического выполнения часто используемых задач.

Схема подготовки инфраструктуры с применением перечисленных инструментов показана на рисунке 1. Развертывание инфраструктуры производилось на следующем аппаратном обеспечении:

- ЦПУ – AMD Ryzen 7 3700X 8-Core Processor, 3600 МГц, 8 физических ядер, 16 логических ядер.
- ОЗУ – 32 Гб DDR4, частота 1600 МГц, двухканальный режим.
- Дисковая подсистема – Samsung SSD 970 EVO Plus.

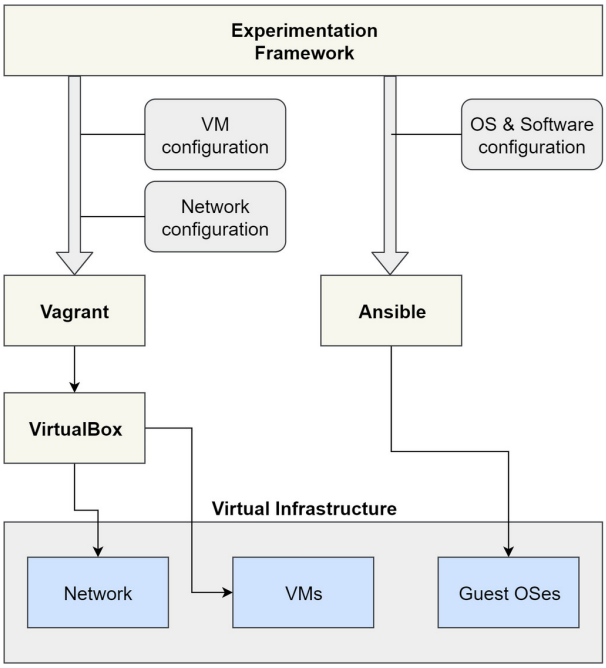

Рис. 1. — Схема подготовки виртуальной инфраструктуры

Подготовленному набору виртуальных машин задана конфигурация, показанная в таблице 1.

TABLE I
КОНФИГУРАЦИИ ВИРТУАЛЬНЫХ МАШИН

Роль	Количество ядер ЦПУ	Объём ОЗУ	Ограни-чение ЦПУ	Ограничение скорости дискового ввода/вывода
Генератор клиентских запросов	4	8 ГБ	–	–
Сервер	2	2 ГБ	–	–
СУБД	2	2 ГБ	50%	25 МБ/сек

Эксперимент проводился на следующем стеке технологий:

1. Node.js версии 13.x – платформа для исполнения кода генератора запросов и серверного приложения.
2. Loopback.io 3 – фреймворк серверного приложения, для взаимодействия с СУБД использовался коннектор «loopback-connector-mongodb» версии 3.3.1.
3. MongoDB версии 4.2.3 Community – документоориентированная СУБД для сохранения переданных данных.

Алгоритм эксперимента состоит из ряда шагов:

1. Инициализация виртуальных машин.
2. Загрузка массива данных в оперативную память виртуальной машины генератора клиентских запросов.
3. Последовательная запись инвариантного набора данных (ResearchSubject).

4. Последовательная запись тестового набора данных (ResearchResult, ResearchResultAnswer).
5. Получение результатов о количестве сохранённых записей.
6.

Взаимодействие компонентов экспериментального стенда в процессе инициализации виртуальной инфраструктуры и исполнения алгоритма показано на рисунке 2.

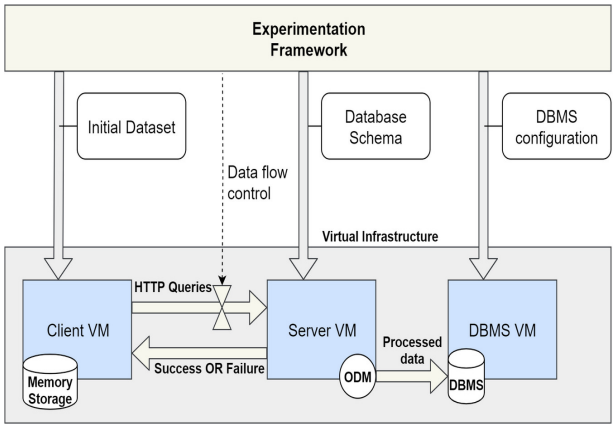

Рис. 2. – Подготовка системы управления базами данных, схемы и данных

Для хранения данных предложены две структуры, представленные на рисунках 3 и 4. Структура коллекции ResearchSubject инвариантна и не требует детального рассмотрения, тогда как структура коллекции ResearchResult подразумевает два альтернативных решения: сохранение в одну коллекцию данных или сохранение с разделением на две коллекции (ResearchResult и ResearchResultAnswer), имеющие логическую связь «1 ко многим».

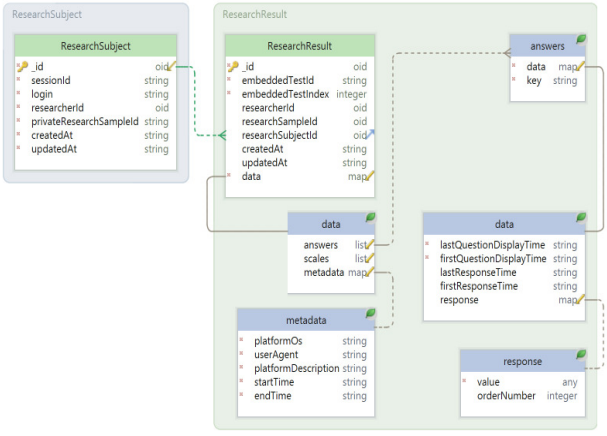

Рис. 3. – Структура документов в MongoDB №1

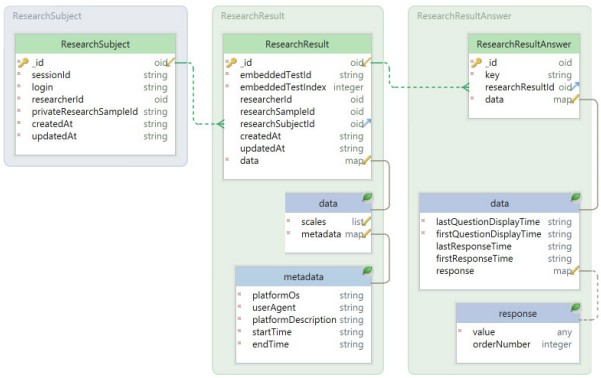

Рис. 4 — Структура документов в MongoDB №2

Для записи данных в СУБД использовались два алгоритма: алгоритм №1 сохранял денормализованные данные в коллекцию *ResearchResult*, алгоритм №2 – данные, разделённые на две коллекции *ResearchResult* и *ResearchResultAnswer* (рисунки 5 и 6 соответственно).

```
function saveResult(result) {
  const { ResearchSubject } = this.app.models;

  const rrs = _.omit(result, []);

  return ResearchSubject.findById(result.researchSubjectId)
    .then(() => ResearchResult.create(rrs));
}
```

Рис. 5. – Алгоритм записи данных в MongoDB в одну коллекцию (алгоритм №1)

```
function saveResult(result) {
  const { ResearchSubject, ResearchResultAnswer } = this.app.models;

  const rrs = _.omit(result, ['data.answers']);
  const rras = _.map(
    result.data.answers,
    (answer) => _.assignIn({}, answer, { researchResultId: result.id })
  );

  return ResearchSubject.findById(result.researchSubjectId)
    .then(() => ResearchResult.create(rrs))
    .then(() => ResearchResultAnswer.create(rras));
}
```

Рис. 6. – Алгоритм записи данных в MongoDB с разделением на 2 отдельные коллекции (алгоритм №2)

Разработана схема методики экспериментального исследования для алгоритмов и структур данных №1 и №2, представленная на рисунках 7 и 8.

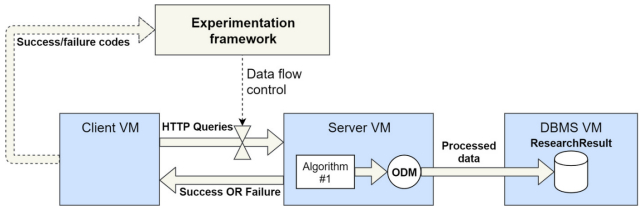

Рис. 7 – Схема эксперимента с использованием алгоритма №1 и структуры данных №1

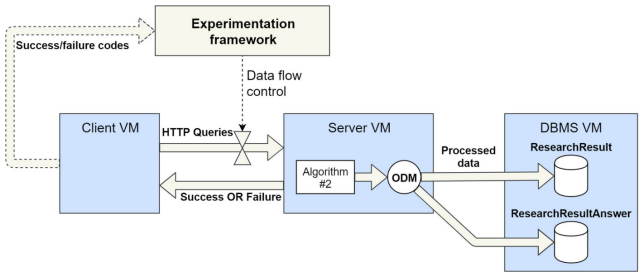

Рис. 8. – Схема эксперимента с использованием алгоритма №2 и структуры данных №2

Отправка запросов производилась в 4 потока, каждый из которых исполнялся на отдельном ядре ЦПУ. Одновременно в каждом потоке совершалось 1000 конкурентных запросов (итого до 4000 одновременных подключений к серверу) с заданным временем ожидания 10 секунд. После выполнения 4000 запросов (получения ответа от сервера, истечения времени ожидания ответа) устанавливались дополнительные 10 секунд ожидания, прежде чем начать следующую итерацию отправки запросов. Между этапами эксперимента также вводилось дополнительное ожидание – 60 секунд.

III. Результаты

A. *Показатели виртуальной машины СУБД*

В ходе сохранения инвариантного набора данных (рис. 9) различий в нагрузке на ЦП виртуальной машины СУБД нет (часть графика до первого вертикального разделителя, примерно 180-я секунда). Графики нагрузки на ЦП во время записи отличающихся структур данных разнятся (рис. 9). Неполная загруженность ресурсов ЦП возможна ввиду того, что в ходе второго эксперимента произошло 3 перезагрузки серверного приложения. Также следует обратить внимание на пик нагрузки после завершения отправки клиентских запросов (второй вертикальный разделитель, примерно 360-я секунда). Ему соответствует длительная и ресурсоёмкая обработка запросов, которые были отправлены последними, но не повлекли за собой перезагрузку серверного приложения и, как следствие, потерю данных.

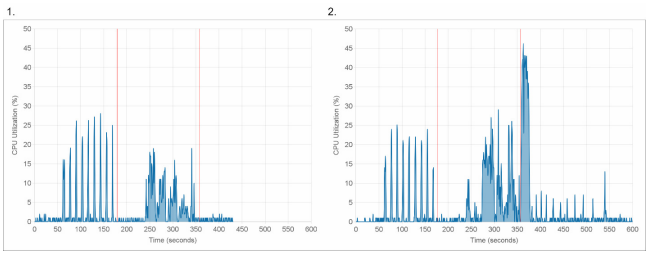

Рис. 9 – График нагрузки на ЦП для виртуальной машины СУБД

Подсистема дискового ввода/вывода не была значительно нагружена ни в одном из экспериментов, поэтому рассматривать этот показатель нецелесообразно.

Объём свободной оперативной памяти сокращается как в первом, так и во втором эксперименте (рис. 10). Как известно, в стандартной конфигурации MongoDB кэширует часть данных в оперативной памяти. Больший объём занятого пространства в первом эксперименте согласуется с тем, что в СУБД было сохранено больше данных. Иных значительных отличий по этому показателю в экспериментах не наблюдается.

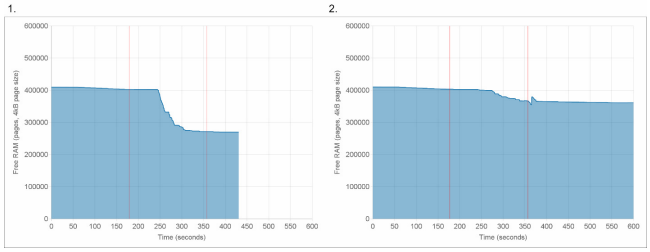

Рис. 10. – График свободной оперативной памяти виртуальной машины СУБД

B. *3. Показатели виртуальной машины серверного приложения*

В ходе сохранения инвариантного набора данных (рис. 12) различий в нагрузке на ЦП виртуальной машины серверного приложения нет (часть графика до первого вертикального разделителя, примерно 180-я секунда). На обоих графиках (рис. 12) видно, что во время сохранения основного объёма данных нагрузка на ЦП виртуальной машины серверного приложения максимальна. Однако если в ходе первого эксперимента есть промежутки, когда она спадает практически полностью, то во втором она остаётся неизменно высокой. Более того, во втором эксперименте видна длительная и ресурсоёмкая обработка запросов, которые были отправлены последними.

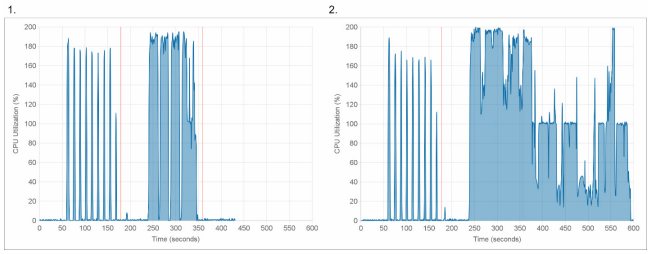

Рис. 11. – График нагрузки на ЦП для виртуальной машины серверного приложения

На графике второго эксперимента (рис. 12) отчётливо видны 3 перезагрузки серверного приложения, характеризующиеся интенсивными обращениями к системе дискового ввода/вывода. В остальном полученные результаты ожидаемы: обращения к диску практически отсутствуют.

Для второго эксперимента (рис. 13) отчётливо видны 3 пика, когда происходило освобождение оперативной памяти в силу перезагрузки серверного приложения. Это полностью согласуется с результатами на графике, представленном на рисунке 12. В остальном видна ожидаемая закономерность: объём свободной памяти снижается после получения клиентских запросов, а после

их обработки и запуска процесса сборки мусора – увеличивается.

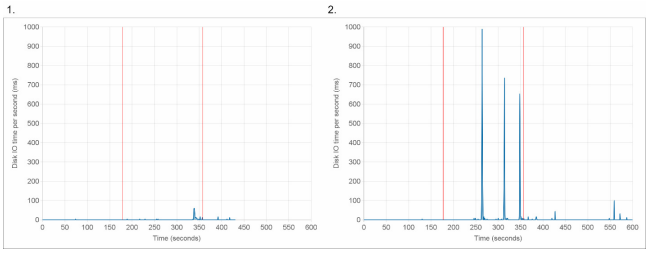

Рис. 12. – График затраченного времени на операции дискового ввода/вывода для виртуальной машины серверного приложения

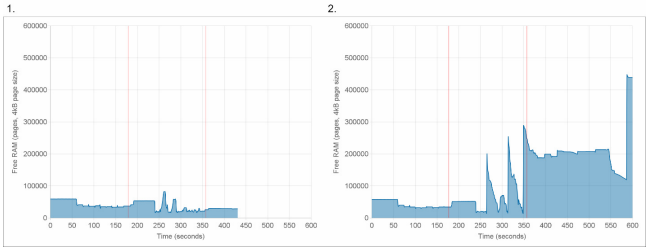

Рис. 13. – График свободной оперативной памяти виртуальной машины серверного приложения

В табл. 2 и 3 приведены численные значения по экспериментальным исследованиям.

Представленные в таблице 2 результаты и идентичные показатели задействования ресурсов (рисунки 9-13, части графиков до первого вертикального разделителя, примерно 180-я секунда) подтверждают равные условия проведения экспериментов.

ТАБЛИЦА II
РЕЗУЛЬТАТИВНОСТЬ ЗАПИСИ ПРИ ИНВАРИАНТНОЙ СТРУКТУРЕ ДАННЫХ

Показатель	Результаты для эксперимента №1	Результаты для эксперимента №2
Время эксперимента, секунд	179	177
Сохранено документов ResearchSubject	34132 (100%)	34132 (100%)
Сообщения сервера об успешной записи документов ResearchSubject	34132 (100%)	34132 (100%)
Сообщения сервера об ошибке при записи документов ResearchSubject	0 (0%)	0 (0%)
Превышение времени ожидания ответа сервера при записи документов ResearchSubject	0 (0%)	0 (0%)

ТАБЛИЦА III
РЕЗУЛЬТАТИВНОСТЬ ЗАПИСИ ЭКСПЕРИМЕНТАЛЬНЫХ ДАННЫХ

Показатель	Результаты для эксперимента №1	Результаты для эксперимента №2
Время эксперимента, секунд	252	423
Сохранено документов ResearchResults	7034 (41.66%)	2497 (14.79%)
Сохранено документов ResearchResultAnswers	– (41.66%)	398063 (8.07%)
Сообщения сервера об успешной записи документов ResearchResult и ResearchResultAnswer	3619 (21.44%)	21 (0.12%)
Сообщения сервера об ошибке при записи документов ResearchResult и ResearchResultAnswer	0 (0%)	0 (0%)
Превышение времени ожидания ответа сервера при записи документов ResearchResult и ResearchResultAnswer	13264 (78.56%)	16862 (99.88%)

IV. ОБСУЖДЕНИЕ РЕЗУЛЬТАТОВ И ЗАКЛЮЧЕНИЕ

В работе сформирована методика анализа вычислительных ресурсов для двух вариантов схем хранения слабоструктурированных данных в документоориентированной NoSQL-системе MongoDB.

В сформированных имитационных условиях большого потока данных и ограниченности вычислительных ресурсов получены экспериментальные результаты.

Результаты экспериментов (см. табл. 3), демонстрируют следующее:

- запись данных с разделением на 2 коллекции занимает больше времени, чем запись объектов в одну коллекцию;
- потеря данных при записи в одну коллекцию значительно меньше, кроме того – исключается вероятность неполной записи данных;
- качество обслуживания клиентских запросов при разделении данных для записи в 2 коллекции ниже, чем при записи в одну коллекцию.

Неполная запись данных для второго эксперимента, вероятно, связана с тем, что сохранение данных в две коллекции не является атомарной операцией. При отказе серверного программного обеспечения часть данных была утеряна. Важно учитывать это при разработке систем, основанных на СУБД и/или на программных интерфейсах, не поддерживающих транзакции при работе с данными.

Можно заключить, что различия в используемых структурах данных, определяющие применяемые к ним операции, могут оказывать влияние на надёжность и эффективность работы программных систем.

БИБЛИОГРАФИЯ

[1] W. Hendricks, “Review of NoSQL Data Stores: Using a reactive three-tier application for software developers to achieve a high availability application design architecture,” In *2019 Open Innovations (OI)*, 2019, pp. 71-77.

[1] A. Corbellini, C. Mateos, A. Zunino, D. Godoy and S. Schiaffino, "Persisting big-data: The NoSQL landscape," *Information Systems*, vol. 63, pp. 1-23, 2017.

[2] P. Kolyasnikov et al., "Analysis of software tools for longitudinal studies in psychology," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 8, pp. 21– 33, 2019.

[3] N.Q. Mehmood, R. Culmone and L. Mostarda, "Modeling temporal aspects of sensor data for MongoDB NoSQL database," *Journal of Big Data*, 4(1), 82017

[4] Roy-Hubara, N., Shoval, P., & Sturm, A. "A Method for Database Model Selection," In *Enterprise, Business-Process and Information Systems Modeling*, 2017, pp. 261-275.

[5] Mehmood, E., & Anees, T. (2019). Performance Analysis of Not Only SQL Semi-Stream Join Using MongoDB for Real-Time Data Warehousing. IEEE Access, 7, 134215-134225.

[6] Chopade, R., & Pachghare, V. (2020). MongoDB Indexing for Performance Improvement. In ICT Systems and Sustainability (pp. 529-539). Springer, Singapore.

[7] B.M. Basok, A.N. Rozhanskaya and S.L. Frenkel, "On web-applications usability testing," *Russian Technological Journal,* vol. 7, no. 6, p. 9-24, 2019. DOI: 10.32362/2500-316X-2019-7-6-9-24

[8] R. Cattell, "Scalable SQL and NoSQL data stores," *Acm Sigmod Record*, vol. 39, no. 4, pp. 12-27, 2011.

[9] V. Herrero, A. Abelló and O. Romero, "NOSQL design for analytical workloads: variability matters," In *International Conference on Conceptual Modeling*, 2016, pp. 50-64.

[10] N. Roy-Hubara and A. Sturm, "Design methods for the new database era: a systematic literature review," *Software & Systems Modeling*, vol. 19, pp. 297–312, 2020.

[11] Atzeni, P., Bugiotti, F., Cabibbo, L., & Torlone, R. (2020). Data modeling in the NoSQL world. Computer Standards & Interfaces, 67, 103149.

[12] J. K. Chen and W. Z. Lee, "An Introduction of NoSQL Databases based on their categories and application industries," *Algorithms*, vol. 12, no. 5, p. 106, 2019

[13] Mahruqi, R. S. A., Alalfi, M. H., & Dean, T. R. (2019, November). A semi-automated framework for migrating web applications from SQL to document oriented NoSQL database. In Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering (pp. 44-53).

Experimental Evaluation of the Resource Efficiency of NoSQL Data Schemes in a Given IT-Infrastructure

D. Ilin, E. Nikulchev

***Abstract*— The paper presents an analysis of the effectiveness of two NoSQL database schemas for saving a stream of semi-structured data with specified characteristics. A framework based on virtual machines was prepared for the experimental research. It simulates the operation of the web service with a classic three-tier architecture for collecting data. The situation when the data flow exceeds the resource capabilities of the web service is considered. Based on the experiments, a comparative analysis of computing resources utilization for given database schemas is carried out. The analysis of the results showed that differences in the database schema affect the reliability and overall performance of the system. In this paper, we propose a framework and methodology for evaluating the effectiveness and reliability of information systems with given data schemes using the example of MongoDB. The proposed experiment methodology can be used with other software components and database management systems. The importance of the research is determined by the need to analyze the influence of data structures on the efficiency and reliability of computing systems and big data processing systems.**

Keywords - NoSQL, database schema, web service, efficiency, reliability, experiment methodology.

REFERENCES

[1] W. Hendricks, "Review of NoSQL Data Stores: Using a reactive three-tier application for software developers to achieve a high availability application design architecture," In *2019 Open Innovations (OI)*, 2019, pp. 71-77.

[2] A. Corbellini, C. Mateos, A. Zunino, D. Godoy and S. Schiaffino, "Persisting big-data: The NoSQL landscape," *Information Systems*, vol. 63, pp. 1-23, 2017.

[3] P. Kolyasnikov et al., "Analysis of software tools for longitudinal studies in psychology," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 8, pp. 21– 33, 2019.

[4] N.Q. Mehmood, R. Culmone and L. Mostarda, "Modeling temporal aspects of sensor data for MongoDB NoSQL database," *Journal of Big Data*, 4(1), 82017

[5] Roy-Hubara, N., Shoval, P., & Sturm, A. "A Method for Database Model Selection," In *Enterprise, Business-Process and Information Systems Modeling*, 2017, pp. 261-275.

[6] Mehmood, E., & Anees, T. (2019). Performance Analysis of Not Only SQL Semi-Stream Join Using MongoDB for Real-Time Data Warehousing. IEEE Access, 7, 134215-134225.

[7] Chopade, R., & Pachghare, V. (2020). MongoDB Indexing for Performance Improvement. In ICT Systems and Sustainability (pp. 529-539). Springer, Singapore.

[8] B.M. Basok, A.N. Rozhanskaya and S.L. Frenkel, "On web-applications usability testing," *Russian Technological Journal,* vol. 7, no. 6, p. 9-24, 2019. DOI: 10.32362/2500-316X-2019-7-6-9-24

[9] R. Cattell, "Scalable SQL and NoSQL data stores," *Acm Sigmod Record*, vol. 39, no. 4, pp. 12-27, 2011.

[10] V. Herrero, A. Abelló and O. Romero, "NOSQL design for analytical workloads: variability matters," In *International Conference on Conceptual Modeling*, 2016, pp. 50-64.

[11] N. Roy-Hubara and A. Sturm, "Design methods for the new database era: a systematic literature review," *Software & Systems Modeling*, vol. 19, pp. 297–312, 2020.

[12] Atzeni, P., Bugiotti, F., Cabibbo, L., & Torlone, R. (2020). Data modeling in the NoSQL world. Computer Standards & Interfaces, 67, 103149.

[13] J. K. Chen and W. Z. Lee, "An Introduction of NoSQL Databases based on their categories and application industries," *Algorithms*, vol. 12, no. 5, p. 106, 2019

[14] Mahruqi, R. S. A., Alalfi, M. H., & Dean, T. R. (2019, November). A semi-automated framework for migrating web applications from SQL to document oriented NoSQL database. In Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering (pp. 44-53).