УДК 004.652

П. А. Судеревский – магистрант кафедры моделирования вычислительных и электронных систем
В. С. Павлов (канд. техн. наук, доц.) – научный руководитель

ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ ИСПОЛЬЗОВАНИЯ ДОКУМЕНТНО-ОРИЕНТИРОВАННЫХ БАЗ ДАННЫХ В ВЕБ-ПРИЛОЖЕНИЯХ

За последние несколько лет появилось много веб-приложений с беспрецедентно высоким уровнем нагрузки, и большими объемами данных. Появление на свет и успех таких высоконагруженных проектов Facebook, Twitter, Amazon привел к необходимости поиска альтернатив традиционным реляционным СУБД, в виду предъявления к системам новых требований, с которыми традиционные реляционные СУБД уже не справляются.

Можно выделить три основных требования к современным веб-приложениям**.**

1. Работа с большими объемами данных: современные высоконагруженные веб-приложения обрабатывают беспрецедентно большие объемы данных, исчисляемые в петабайтах.
2. Огромное количество пользователей: миллионам пользователей должен быть обеспечен постоянный одновременный доступ к системе.
3. Гибкая модель данных: необходимость в использовании более простой и более гибкой модели данных. [3]

Таким образом, появляется задача сопоставления возможностей применения традиционных реляционных СУБД с возможными альтернативами для обеспечения выполнения новых требований. Мое предложение – использование документно-ориентированных баз данных.

На фоне предъявления к СУБД новых требований, в последние годы стало популярно и по-прежнему набирает обороты применение при разработке проектов NoSQL, или «Not Only SQL» баз данных.

Сторонниками концепции NoSQL подчёркивается, что она не является полным отрицанием языка SQL и реляционной модели, проект исходит из того, что SQL – это важный и весьма полезный инструмент, но при этом он не всегда может считаться универсальным. Основная цель подхода – расширить возможности БД там, где SQL недостаточно гибок, и не вытеснять его там, где он справляется со своими задачами.

NoSQL базы данных на данный момент подразделяются на 4 типа.

1. Хранилища типа «Ключ/Значение». Такие хранилища позволяют хранить в памяти по определенному ключу любые данные, это может быть просто число или текст, а может быть сериализованный объект.
2. Документно-ориентированные базы данных. Каждая запись хранится как отдельный документ, имеющий собственный набор полей, который может отличаться от документа к документу.
3. Колоночные Хранилища Данные хранятся в столбцах вместо привычного хранения в строках. Это выгодно для различных архивов информации и каталогов, в которых большая часть вычислений происходит над подобными выборками данных.
4. Графовые базы данных. Такие хранилища данных используют вершины и ребра графа для представления информации.

Мое предложение – использование документно-ориентированной модели данных, а именно базы данных MongoDB, как одной из наиболее перспективных и универсальных, а также имеющей наилучший баланс между производительностью и предлагаемой функциональностью.

MongoDB – это документно-ориентированная база данных с открытым кодом, разработанная для работы с веб-приложениями, имеющая гибкую модель данных, не ограниченную строгой схемой. MongoDB, по мнению разработчиков, должна заполнить разрыв между простыми хранилищами данных типа «ключ-значение», обладающими высокой производительностью и предлагающими хорошие возможности масштабирования базы данных, но предлагающими невысокую функциональность, и традиционными реляционным СУБД, обладающими высокой функциональностью и надежностью, но имеющими стро-

гую схему данных и необходимость соблюдения ACID-требований, что плохо сказывается на производительности и возможности выполнения горизонтального масштабирования.

Основные отличительные особенности документно-ориентированной базы данных MongoDB:

1. Документы хранятся в виде BSON-массивов. BSON – это расширение легковесного текстового формата хранения данных JSON, основанного на языке JavaScript. Этот формат считается языконезависимым и может использоваться практически с любым языком программирования.
2. Возможность агрегации данных с помощью технологии MapReduce. MapReduce – технология для параллельной обработки больших объемов данных. Работа MapReduce состоит из двух шагов: Map и Reduce. На Map-шаге происходит предварительная обработка входных данных. Преимущество MapReduce заключается в том, что он позволяет распределенно производить операции предварительной обработки и свертки. Операции предварительной обработки работают независимо друг от друга и могут производиться параллельно. Аналогично, множество рабочих узлов могут осуществлять свертку.
3. Хорошие возможности для выполнения горизонтального масштабирования. В случае использования традиционных реляционных баз данных, обычно используется вертикальное масштабирование, когда увеличение производительности системы достигается путем добавления ресурсов в единственный сервер. В контексте построения современных высоконагруженных веб-приложений, использование такого метода оказывается недостаточно: построение одного суперкомпьютера является экономически невыгодным и имеет предел производительности.[2] При горизонтальном масштабировании данные хранятся не на единственном сервере, а распределяются на какое-то количество узлов. Такой метод оказывается более экономически-выгодным, обеспечивает хорошую производительность, хорошую доступность системы, а также делает архитектуру разрабатываемой системы более гибкой для дальнейшего масштабирования.
4. Поддержка наборов репликаций. Репликация — это механизм синхронизации содержимого базы данных между двумя или несколькими узлами. С помощью наборов репликаций в MongoDB обеспечивается отказоустойчивость, а также повышается производительность за счет возможности параллельного чтения данных с разных реплик.

Подходящий и логичный способ дать оценку проблемам, связанным с обеспечением выполнения новых требований в веб-приложениях – это понять, как в таких системах сочетаются следующие три понятия: согласованность, доступность и устойчивость к разделению сети.

И на основе рассмотрения этих понятий можно сделать выводы о том, насколько оправдано предложенное решение проблемы обеспечения трех требований для высоконагруженных веб-приложений.

Согласованность данных, доступность и устойчивость к разделению сети – это три опоры, на которых строится теорема Брюера, которая лежит в основании мышления о транзакционной целостности в больших распределенных системах. Теорема Брюера говорит о том, что эти три свойства не могут быть одновременно достигнуты в распределенных системах. Нужно идти на компромиссы и жертвовать как минимум одним свойством из трех. Однако прежде чем идти на компромисс, важно понять, что означают эти три свойства.

Согласованность означает последовательную запись и чтение, так что параллельно-выполняющиеся операции видят одни и те же правильные данные, что как минимум означает, что никто не увидит устаревших данных. В ситуации с одним узлом, согласованность можеть быть достигнута при использовании правил ACID, но ситуация становится запутанней в случае распределенной системы.

Под доступностью имеется в виду, что система будет доступна для работы в любое время, когда это необходимо. Если система не готова обработать запрос в тот момент, когда это необходимо – она недоступна. Тем не менее, многие приложения могут жертвовать доступностью, и этот тот компромисс, на который они могут пойти.

Устойчивость к разделению означает возможность системы продолжать выполнять необходимые функции, если какой-то из узлов становится недоступным. [1]

Рис. 1.

Реляционные СУБД всегда делали акцент на согласованности и доступности данных, и плохо обеспечивали «устойчивость к разделению сети», необходимую для больших распределенных систем, использующих горизонтальное масштабирование.

MongoDB выбрала путь обеспечения устойчивости к разделению и обеспечения доступности системы, при этом жертвуя согласованностью. Но несмотря на то, что в MongoDB выбран путь компромисса не в пользу согласованности, она не полностью отсутствует. В MongoDB используется так называемая «Итоговая согласованность», когда в какой-то момент времени клиент может увидеть устарвшие данные, но в конце концов все данные в системе примут согласованное непротиворечивое состояние. Как MongoDB обеспечивает выполнение этих свойств? На рисунке 2 показан набор репликаций, настроенный на обеспечение итоговой согласованности: запись данных выполняется только на главный мастер-сервер, а чтение данных возможно со всех реплик. В случае отказа мастер-сервера, один из слейв-серверов автоматически будет выбран в качестве мастер-сервера. Неисправный мастер-сервер, в свою очередь, после устранения неполадок автоматически будет выбран в качестве слейв-сервера. Также MongoDB может быть настроена на обеспечение сильной согласованности, когда и запись, и чтения будут выполняться только на главную реплику.

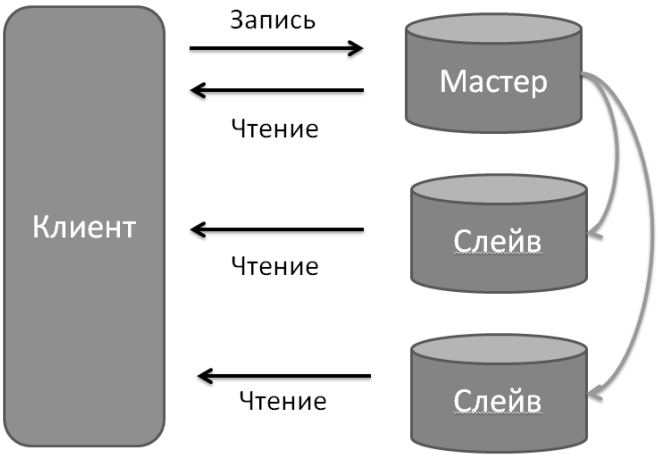

Рис. 2 .

На рисунке 3 проиллюстрирован пример архитектуры системы, содержащий четыре набора репликаций, каждый из которых содержит по три реплики. Метаданные, содержащие информацию о том, на каких наборах реплик какие содержаться данные, хранятся на серверах конфигураций.

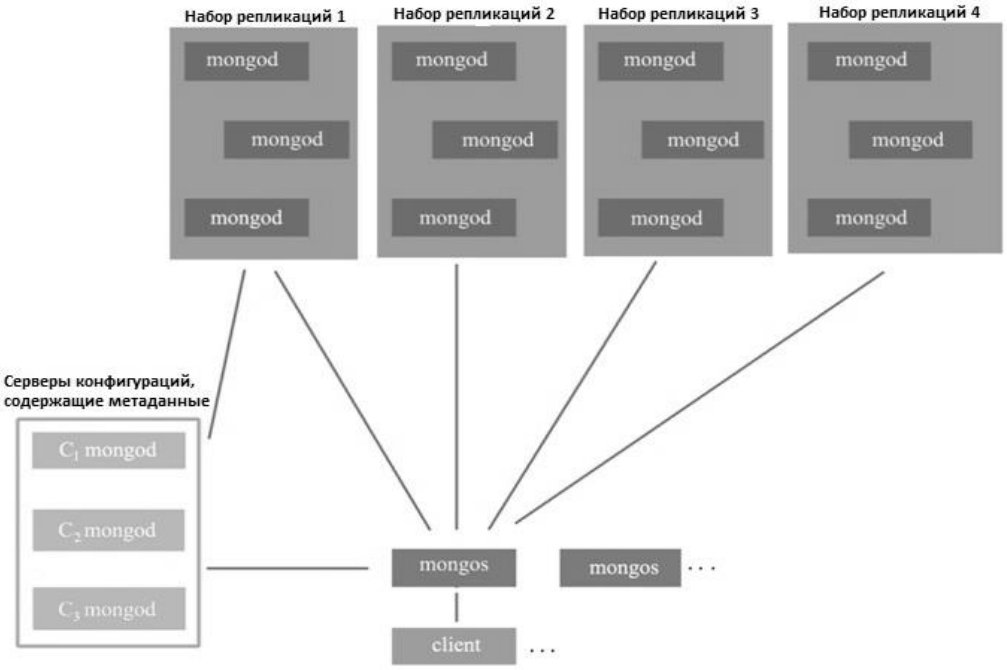

Рис. 3.

Таким образом, можно сказать, что документно-ориентированная база данных MongoDB, жертвуя согласованностью, не столь необходимой в большинстве современных высоконагруженных веб-приложений, обеспечивает выполнение свойств доступности и устойчивости к разделению сети, что важно в контексте таких проектов. MongoDB имеет хорошие возможности для хранения и обработки больших объемов данных, может обеспечить постоянный одновременный доступ к системе огромного количества пользователей, и имеет гибкую модель данных, которая, в отличие от традиционных реляционных СУБД, не ограничена строгой схемой данных. Предложенное программное обеспечение может быть рассмотрено в качестве основной базы данных при разработке высоконагруженных веб-приложений, не требующих обеспечения выполнения ACID-требований.

Библиографический список

1. Tiwari S. Professional NoSQL / S.Tiwari : John Willey & Sons. Inc, 2011, 361p.
2. Banker K. MongoDB in Action / K.Banker : Manning Sheltor Island, 2011, 287p.
3. Олег Дмитрович Универсальное NoSQL - введение в теорию // Компьютерные Вести. 2010