Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Системы обработки информации и управления

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Анализ работы метаграфого хранилища с использованием СУБД Tarantool для задач Big Data

---

Студент ИУ5-33М _____ Морозенков О.Н.
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ Гапанюк Ю.Е
(Подпись, дата) (И.О.Фамилия)

Консультант _____ _____
(Подпись, дата) (И.О.Фамилия)

2022 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой __ИУ5__
(Индекс)
_____ _Терехов В. И.
(И.О.Фамилия)
« _____ » _____ 2022 г.

ЗАДАНИЕ
на выполнение научно-исследовательской работы

по теме _Анализ работы метаграфого хранилища с использованием СУБД Tarantool для задач Big Data___

Студент группы __ИУ5-33М__

_____Морозенков Олег Николаевич_____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

_____

Источник тематики (кафедра, предприятие, НИР) _____

График выполнения НИР: 25% к ___ нед., 50% к ___ нед., 75% к __ нед., 100% к ___ нед.

Техническое задание ___исследовать и оценить параметры работы метаграфового_____
хранилища с использованием СУБД Tarantool для задач Big Data_____

Оформление научно-исследовательской работы:

Расчетно-пояснительная записка на _____ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____

Дата выдачи задания « __ » _____ 2022 г.

Руководитель НИР _____ ___Гапанюк Ю.Е. ____
(Подпись, дата) (И.О.Фамилия)

Студент _____ __Морозенков О.Н.___
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

Введение

Исследовательская работа на тему «Анализ работы метаграфого хранилища с использованием СУБД Tarantool для задач Big Data» посвящена исследованию метаграфов и возможных реализаций метаграфовых хранилищ. Необходимо происследовать данную графовую модель и вывести ее особенности.

Описание метаграфовой модели

Для построения интеллектуальных систем используется большое количество подходов: продукционные правила, нейронные сети, нечеткая логика, эволюционные методы и др. При этом можно отметить явную тенденцию к совместному использованию разных методов для решения различных классов задач. Это привело к появлению такого направления, как «гибридные интеллектуальные системы»

В настоящее время интеллектуальные системы, как правило, не разрабатываются отдельно, но встраиваются в виде модулей в традиционные информационные системы для решения задач, связанных с интеллектуальной обработкой данных и знаний. Такая комбинированая система называется гибридной интеллектуальной информационной системой (ГИИС).

Метаграфы – это модель, которая служит для описания сложных систем, для описания их семантики и прагматики, а также для описания гибридные интеллектуальных информационных систем.

Порождающее множество метаграфа – это множество переменных, встречающихся в ребрах метаграфа.

Ребро метаграфа содержит входную вершину (invertex) и выходную вершину (outvertex). Входная и выходная вершины могут содержать произвольное количество элементов. Различные элементы, принадлежащие входной (выходной) вершине, называются соответственно совходами (совыходами).

Тогда метаграф – это графовая конструкция, определяемая порождающим множеством X и множеством ребер E, при этом множество ребер определено на том же порождающем множестве. При этом если две или больше метавершин

соответствуют одному и тому же множеству вершин, то такие вершины считаются одинаковыми и рассматривается только одна из таких метавершин.

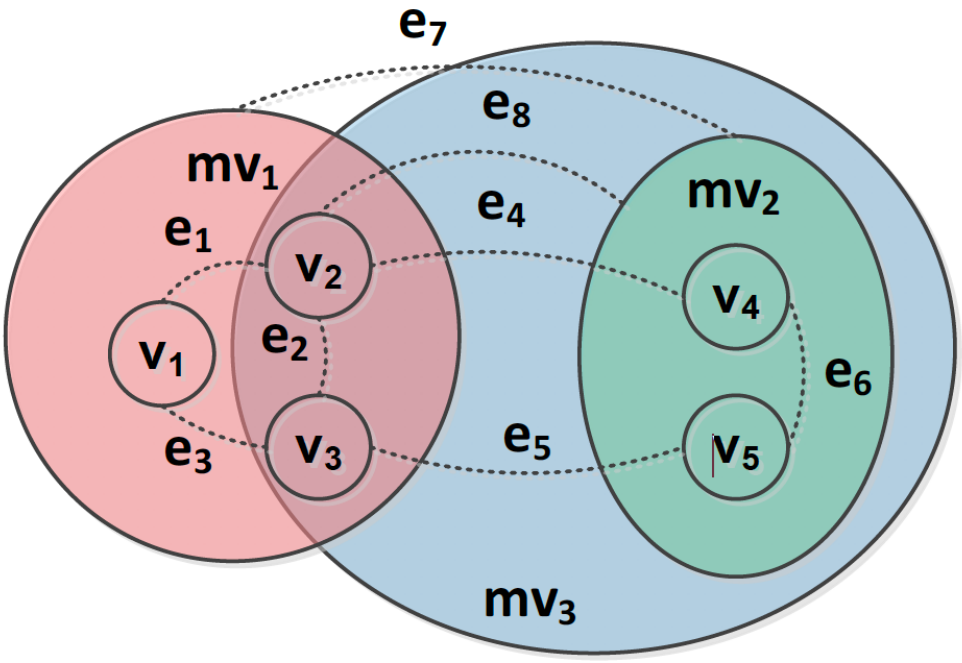

Рис 1. Визуальное представление метаграфа

Пример описания метаграфа показан на рис. 1. Данный метаграф содержит вершины, метавершины и ребра. На рис. 1 показаны три метавершины: mv1, mv2 и mv3. Метавершина mv1 включает вершины v1, v2, v3 и связывающие их ребра e1, e2, e3. Метавершина mv2 включает вершины v4, v5 и связывающее их ребро e6. Ребра e4, e5 являются примерами ребер, соединяющих вершины v2-v4 и v3-v5, включенные в различные метавершины mv1 и mv2. Ребро e7 является примером ребра, соединяющего метавершины mv1 и mv2. Ребро e8 является примером ребра, соединяющего вершину v2 и метавершину mv2. Метавершина mv3 включает метавершину mv2, вершины v2, v3 и ребро e2 из метавершины mv1, а также ребра e4, e5, e8, что показывает холоническую структуру метаграфа.

Метаграф можно охарактеризовать как «сеть с эмерджентностью», то есть фрагмент сети, состоящий из вершин и связей, может выступать как отдельное целое. У метаграфа есть ребра, вершины, а также метаребра и метавершины. Определения метавершины и метаребра являются рекурсивными, так как элементы могут быть, в свою очередь, метавершинами и метаребрами. Метавершина является формализмом описания данных, а метаребро – формализмом описания процессов.

Наличие у метавершин собственных атрибутов и связей с другими вершинами является важной особенностью метаграфов. Это соответствует принципу эмерджентности, то есть приданию понятию нового качества, несводимости понятия к сумме его составных частей. Фактически, как только вводится новое понятие в виде метавершины, оно «получает право» на собственные свойства, связи и т.д., так как в соответствии с принципом эмерджентности новое понятие обладает новым качеством и не может быть сведено к подграфу базовых понятий.

Анализ метаграфовой модели

Одна из приоритетных сфер применения метаграфового хранилища — это работа метаграфовых агентов. Реактивный метаграфовый агент может быть представлен в виде предикатного описания.

Приведем пример метаграфового агента. Описание правил содержит предикат «Rules». Правилу соответствует предикат «Rule». Стартовым правилом (параметр «start=true» предиката «Rule») является «правило 1» (остальные правила не показаны, чтобы не загромождать пример).

Условию правила соответствует предикат «Condition». Параметр «WorkMetagraph» содержит ссылку на проверяемую метавершину mv1. Условие проверяет, что метавершина mv1 содержит вершины v1 и v2, содержащие атрибуты k. Найденные значения атрибутов k помещаются в переменные $k1 и $k2. Вершины v1 и v2 должны быть соединены ребром, содержащим атрибут «flag=main».

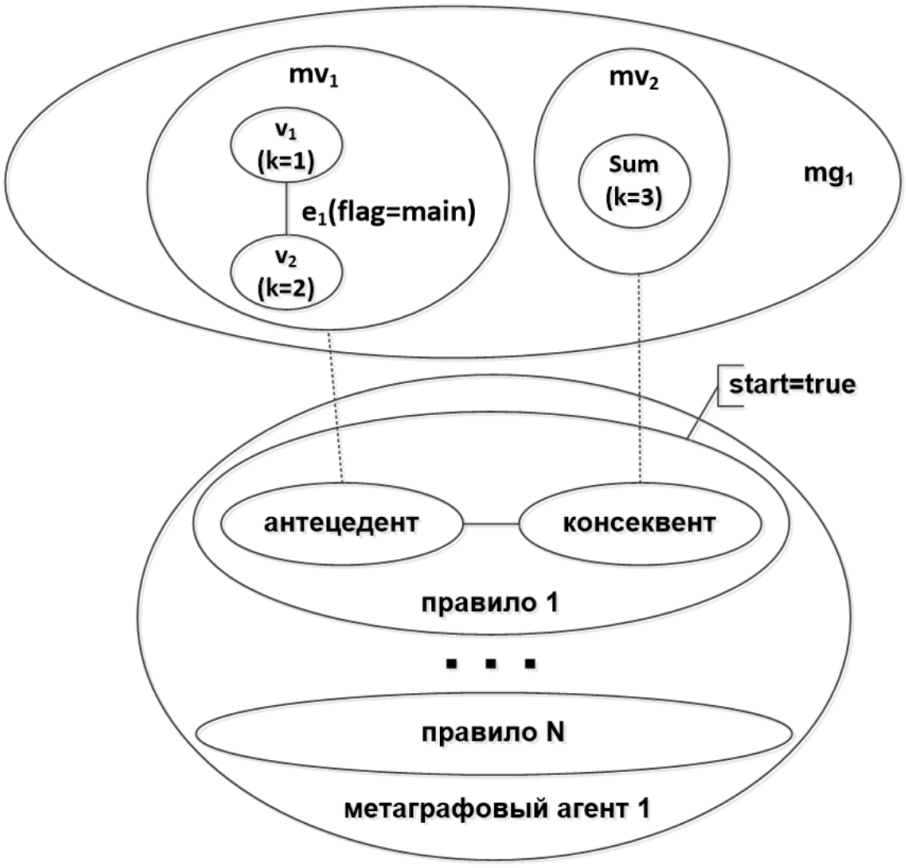

Рис 2. Представление реактивного метаграфового агента.

```
Agent(Name='метаграфовый агент 1',
  Rules(
    Rule(Name='правило 1', start=true,
      Condition(WorkMetagraph=mv1,
        Vertex(Name=v1, Attribute(k, $k1)),
        Vertex(Name=v2, Attribute(k, $k2)),
        Edge(v1, v2, Attribute(flag, main))
      )
      Action(WorkMetagraph=mv2,
        Add(Vertex(Name=Sum,
          Attribute(k, Eval($k1+$k2))))
      )
    ),
    Rule…() … ))

```

Листинг 1. Предикатное описание реактивного метаграфового агента.

Для производительной работы метаграфовых агентов можно выразить следующие гипотезы по поводу хранилища:

- Соотношение чтений из хранилища ко вставкам в хранилище будет примерно одинаково.
- Удалений из хранилища будет меньше вставок в хранилище.

С этими вводными начинаем исследование хранилищ с достаточной пропускной способностью одновременно, как на чтение, так и на запись.

Описание СУБД PostgreSQL и применяемой структуры данных B-дерево

PostgreSQL — это СУБД, созданная с упором на расширяемость и соответствие SQL. PostgreSQL поддерживает транзакции со свойствами атомарности, согласованности, изоляции, долговечности (ACID), автоматически обновляемыми представлениями, материализованными представлениями, триггерами, внешними ключами и хранимыми процедурами. Он предназначен для обработки широкого спектра рабочих нагрузок, от отдельных компьютеров до хранилищ данных или веб-служб с большим количеством одновременных пользователей.

Сильными сторонами PostgreSQL считаются:

- высокопроизводительные и надёжные механизмы транзакций и репликации;
- расширяемая система встроенных языков программирования: в стандартной поставке поддерживаются PL/pgSQL, PL/Perl, PL/Python и PL/Tcl; дополнительно можно использовать PL/Java, PL/PHP, PL/Py, PL/R, PL/Ruby, PL/Scheme, PL/sh и PL/V8, а также имеется поддержка загрузки модулей расширения на языке C[10];
- наследование;
- возможность индексирования геометрических (в частности, географических) объектов и наличие базирующегося на ней расширения PostGIS;
- встроенная поддержка слабоструктурированных данных в формате JSON с возможностью их индексации;
- расширяемость (возможность создавать новые типы данных, типы индексов, языки программирования, модули расширения, подключать любые внешние источники данных).

Ключевая структура данных используемая для хранения данных в PostgreSQL — это B-дерево.

B-дерево — структура данных, дерево поиска. С точки зрения внешнего логического представления — сбалансированное, сильно ветвистое дерево. Сбалансированность означает, что длины любых двух путей от корня до листьев различаются не более, чем на единицу. Ветвистость дерева — это свойство каждого узла дерева ссылаться на большое число узлов-потомков. С точки зрения физической организации B-дерево представляется как мультисписочная структура страниц памяти, то есть каждому узлу дерева соответствует блок памяти (страница). Внутренние и листовые страницы обычно имеют разную структуру.

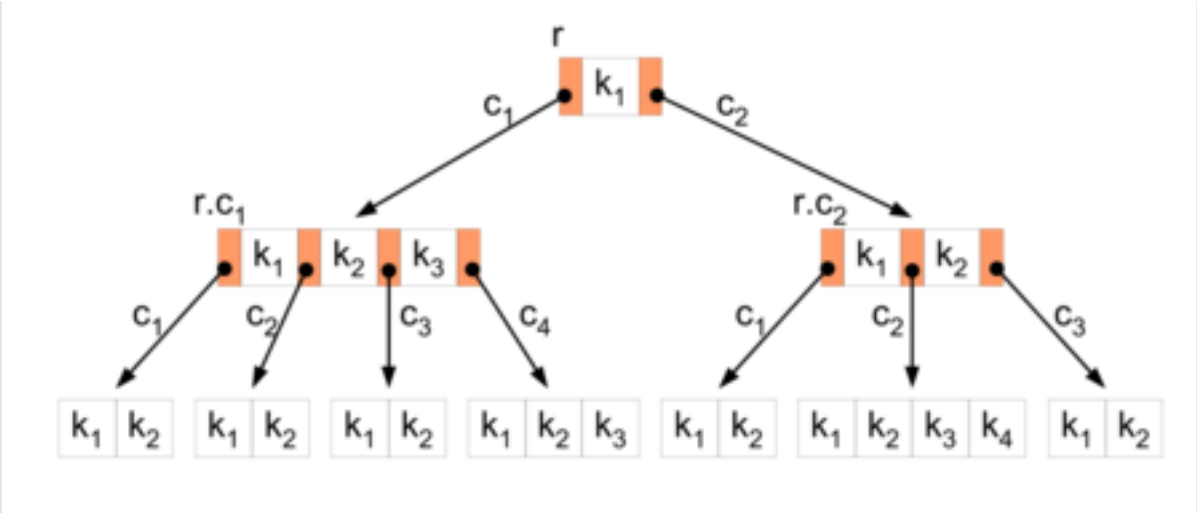

Рис. 2. Пример B-дерева со степенью 3

B-дерево применяется для индексирования информации на жёстком диске. Время доступа к произвольному блоку на жёстком диске очень велико (порядка миллисекунд), поскольку оно определяется скоростью вращения диска и перемещения головок. Поэтому важно уменьшить количество узлов, просматриваемых при каждой операции. Использование поиска по списку каждый раз для нахождения случайного блока могло бы привести к чрезмерному количеству обращений к диску вследствие необходимости последовательного прохода по всем его элементам, предшествующим заданному, тогда как поиск в B-дереве, благодаря свойствам сбалансированности и высокой ветвистости, позволяет значительно сократить количество таких операций.

Описание СУБД Tarantool и применяемой структуры данных LSM-дерево

Tarantool — это платформа in-memory вычислений с гибкой схемой данных для эффективного создания высоконагруженных приложений. Включает в себя базу данных и сервер приложений на Lua. Обладает высокой скоростью работы по сравнению с традиционными СУБД, обладая теми же свойствами: персистентности, транзакционности ACID, репликации master-slave, master-master.

Tarantool имеет два движка хранения данных:

- движок хранения данных в оперативной памяти;
- движок хранения данных на жестком диске;

В данной работе обратим фокус на движок хранения данных на жестком диске.

Ключевая структура данных используемая для хранения данных в Tarantool — это LSM-дерево.

LSM-дерево — структура данных, предоставляющая быстрый доступ по индексу в условиях частых запросов на вставку.

Простая версия LSM-дерева — двухуровневое дерево — состоит из двух древоподобных структур C0 и C1. C0 меньше по размеру и хранится целиком в оперативной памяти, а C1 находится в энергонезависимой памяти. Новые записи вставляются в C0. Если после вставки размер C0 превышает некоторое заданное пороговое значение, непрерывный сегмент удаляется из C0 и объединяется с C1 для хранения на жестком диске. Хорошая производительность достигается за счёт того, что деревья оптимизированы под своё хранилище, а слияние осуществляется эффективно и группами по нескольку записей, используя алгоритм, напоминающий сортировку слиянием.

Большинство LSM-деревьев, используемых на практике, реализует несколько уровней. Уровень C0 (назовём его MemTable) хранится в оперативной памяти и может быть представлен обычным деревом. Данные на жестком диске хранятся в виде отсортированных по ключу таблиц (SSTable).

Таблица может храниться в виде отдельного файла или набора файлов с непересекающимися значениями ключей. Для поиска конкретного ключа нужно проверить его наличие в MemTable, а затем — пройти по всем SSTable на устройстве постоянного хранения.

Схема работы с LSM-деревом:

- индексы SSTable всегда загружены в оперативную память;
- запись производится в MemTable;
- при чтении сначала проверяется MemTable, а затем, если надо, — SSTable на жестком диске;
- периодически MemTable сбрасывается на жесткий диск в виде SSTable;
- периодически SSTable объединяются.

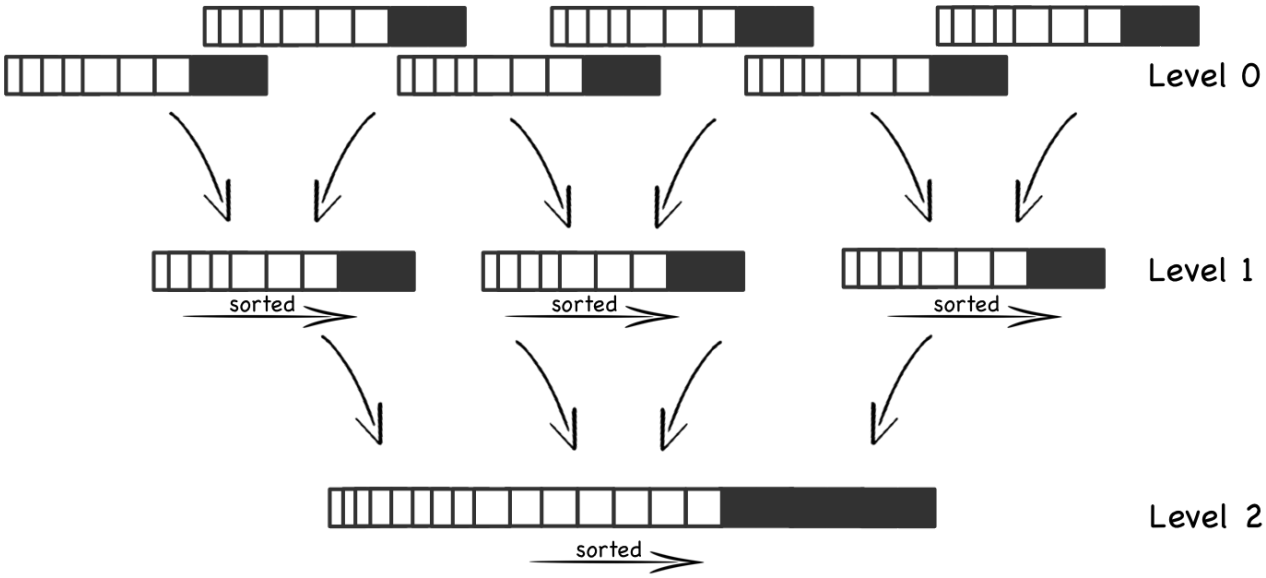

Рис 3. Пример LSM-дерева

Сравнение структур данных B-дерево и LSM-дерево

Считается, что B-деревья более эффективны для чтения, а LSM-деревья – для записи. Тем не менее, с распространением SSD-дисков, у которых в несколько раз выше производительность чтения по сравнению с производительностью записи, преимущества LSM-деревьев стали явнее в большинстве сценариев.

Сравним работу B-деревьев и LSM-деревьев. Представим ситуацию, возьмем B-дерево на 100 000 000 узлов и предположим, что размер блока равен 4096 байтов, а размер элемента – 100 байтов. Таким образом, в каждом блоке можно будет разместить до 40 элементов с учетом накладных расходов, а в B-дереве будет около 2 570 000 блоков, пять уровней, при этом первые четыре займут по 256 МБ, а последний – до 10 ГБ. На современном компьютере все уровни, кроме последнего, успешно попадут в кэш файловой системы, и фактически любая операция чтения будет требовать не более одной операции ввода-вывода.

Ситуация выглядит иначе при записях в B-дерево. Предположим, что необходимо обновить один элемент дерева. Так как операции с B-деревьями работают через чтение и запись целых блоков, приходится прочитать 1 блок в память, изменить 100 байт из 4096, а затем записать обновленный блок на диск. Таким образом, нам пришлось записать в 40 раз больше, чем реальный объем измененных данных. Принимая во внимание, что внутренний размер блока в SSD-дисках может быть 64 КБ и больше, и не любое изменение элемента меняет его целиком, объем «паразитной» нагрузки на диск может быть еще выше.

Феномен таких «паразитных» чтений в литературе называется read amplification (усложнение чтения), а феномен «паразитной» записи – write amplification (усложнение записи). Коэффициент усложнения, то есть коэффициент умножения, вычисляется как отношение размера фактически прочитанных (или записанных) данных к реально необходимому (или измененному) размеру. В нашем примере с B-деревом коэффициент составит около 40 как для чтения, так и для записи.

Объем «паразитных» операций ввода-вывода при обновлении данных является одной из основных проблем, которую решают LSM-деревья.

Тест структур данных B-дерево и LSM-дерево

Тест создает базу данных путем последовательной вставки 100 миллионов пар ключ/значение с размером значения 100 байт. Генерирует около 10 ГБ данных. Тест запускается сначала в однопоточном режиме, а затем в многопоточном режиме.

Переменными между запусками являются: ограничено ли количество обновлений в секунду или нет; количество потоков, выполняющих запросы.

Оборудование перечислено в таблице 1.

Результат теста на рис 4.

Компонент	Спецификация
CPU	4 x Intel Xeon CPU X5650 @ 2.67GHz (24 cores total)
Memory	144 GB RAM
Disk	400GB Intel S3700 SSD

Таблица 1. Оборудование для теста.

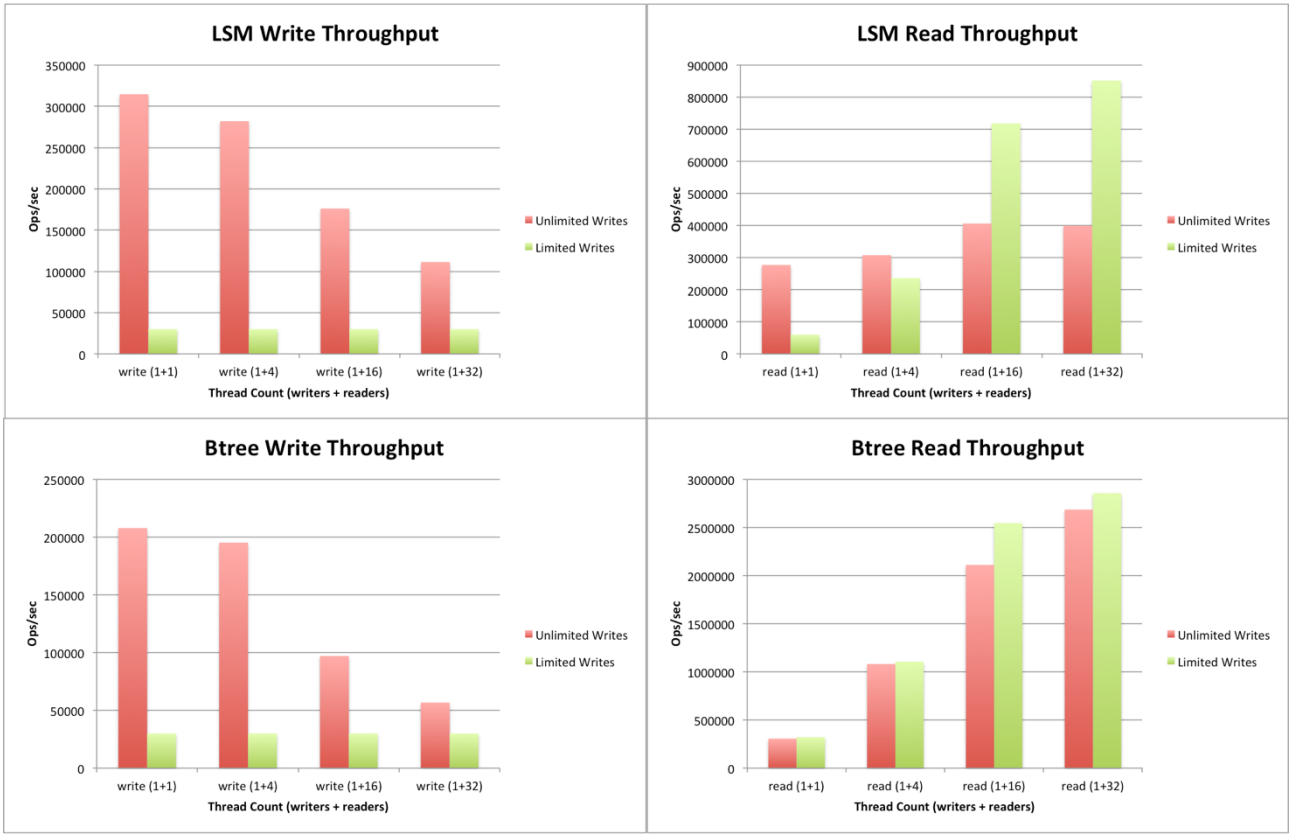

Рис. 4. Сравнение производительности B-дерева и LSM-дерева

Сравним пропускную способность записи LSM-дерева и B-дерева. Обе структуры способны поддерживать 30 000 вставок в секунду, требуемых нагрузкой с ограниченной скоростью. LSM-дерево обеспечивает пропускную способность вставки в 1,5-2 раза выше, чем B-дерево. Это ожидаемо — LSM-дерево оптимизирован для работы с большими объемами записи. Также интересно, что когда существует только один поток записи, то по мере увеличения количества потоков чтения, пропускная способность записи снижается. Эффект весьма значителен при 32 потоках чтения, что вполне ожидаемо, поскольку машина, на которой выполняется эталон, имеет 24 ядра процессора — наличие 32 потоков приложений приводит к переключению контекста. Снижение пропускной способности при записи с 16 потоками чтения требует дальнейшего изучения.

Сравним пропускную способность чтения LSM-дерева и B-дерева. Ключевое наблюдение между двумя графиками заключается в том, что пропускная способность чтения с использованием дерева B-дерева составляет от 1,5x до 3x по сравнению с LSM-деревом. Разница становится все более заметной по мере добавления потоков чтения. Первый набор результатов для LSM-дерева является побочным эффектом того, как выполняется эталон. Эталон запускает фазу заполнения, за которой сразу же следует последовательность фаз чтения. В дереве LSM фаза заполнения приведет к тому, что дерево окажется в состоянии, когда в нем будет много фонового обслуживания, которое может быть выполнено. Поэтому набор результатов чтения (1+1) будет перекошен, пока дерево LSM "успокаивается". Эта работа по оседанию помогает улучшить производительность чтения в последующих конфигурациях. B-деревья не требуют фонового обслуживания, поэтому это не влияет на их производительность.

Заключение

В ходе курсовой работы была проанализирована работы метаграфого хранилища с использованием СУБД Tarantool для задач Big Data. Была изучена теория графовых моделей. Была изучена внутренняя работа СУБД PostgreSQL и СУБД Tarantool. Изучив структуры данных B-дерева и LSM-дерева, было проведено тестирование алгоритмов под высокой нагрузкой и анализ получившихся результатов.

Список литературы

1. Конспект лекций по спецкурсу «Гибридные интеллектуальные информационные системы на основе метаграфового подхода»: Учебно-методическое пособие. – М.: Издательство «Спутник +», 2018. – 53с., ил.
2. Chernenkiy V.M., Gapanyuk Yu.E., Revunkov G.I., Terekhov V.I., Kaganov Yu.T. Metagraph approach for hybrid intelligent information systems description. Prikladnaya Informatika – Journal of Applied Informatics, 2017.
3. PostgreSQL // Википедия. [2022]. Дата обновления: 30.10.2022. URL: https://ru.wikipedia.org/?curid=39423&oldid=126362893 (дата обращения: 1.10.2022).
4. Tarantool // Википедия. [2022]. Дата обновления: 20.05.2022. URL: https://ru.wikipedia.org/?curid=7446100&oldid=122433459 (дата обращения: 1.10.2022).
5. Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ, 3-е издание = Introduction to Algorithms, Third Edition. — М.: «Вильямс», 2013. — 1328 с.
6. Хранение данных с помощью vinyl. URL: https://www.tarantool.io/ru/doc/latest/concepts/engines/vinyl/ (дата обращения: 1.10.2022).
7. Btree vs LSM // GitHub. URL: https://github.com/wiredtiger/wiredtiger/wiki/Btree-vs-LSM (дата обращения: 1.10.2022).