Отже, можна зробити висновок, що існуючі алгоритми логічного виводу володіють кількома істотними недоліками, основним з яких є слабка цілеспрямованість дії, обумовлена наявністю евристичних елементів. Унаслідок цього в процесі аналізу логіко-лінгвістичної моделі формується великий обсяг проміжної інформації, що надалі не використовується, але різко збільшує витрати машинного часу. Необхідно поставити процес вироблення рішень в системах ситуаційного управління на строгу математичну основу, тим більше що він носить явно виражений комбінаторний характер.

Список використаних джерел:

1. Нильсон Н. Принципы искусственного интеллекта: Пер. с англ. // М.: Радио и связь, 1985. – 286 с.
2. Литвиненко А.Е., Кралина А.С. Математический метод оперативного планирования технического обслуживания воздушных судов в нештатных режимах роботы аэропорта. – Проблемы информатизации и управления // К.: НАУ – 2005, вып. 12. – С. 94-100.
3. Попов Э.В. Экспертные системы. (Решение неформализованных задач в диалоге с ЭВМ) // М.: Техническая кибернетика, 1987, № 5. – С. 5-18.
4. Поспелов Г.С. Системный анализ и искусственный интеллект // М.: ВЦ АН СССР, 1980. – 304 с.
5. Поспелов Д.А. Большие системы. Ситуационное управление // М.: Знание, 1975. – 64 с.

Маляренко М.А.
студентка,
Научный руководитель: **Лазурик В.М.**
старший преподаватель,
Харьковский национальный университет
имени В.Н. Каразина

ВЫБОР БАЗЫ ДАННЫХ TARANTOOL ДЛЯ СОВМЕСТНОЙ РАБОТЫ С MYSQL

Согласно статистике в настоящее время на проектах более распространены реляционный базы данных такие как: Oracle, MySql, PostgreSQL [1]. Реляционная модель данных хороша тем, что соответствует требованиям ACID [2], а также тем, что данные, с которыми вы работаете, структурированы, при этом структура не подвержена частым изменением. Часто, база данных перестает справляться с нагрузками и ее начинают масштабировать и оптимизировать. Проблема заключается в том, что для постоянного масштабирования необходимы серьезные средства на дополнительные серверы.

Высоконагруженный проект – это проект, в котором неэффективное решение или ошибка в системе приводит к тому, что инфраструктура проекта перестает справляться с нагрузкой или проект теряет большие суммы денег [3]. Хранение всех данных в оперативной памяти позволяет сделать их высоко доступными, а алгоритмы для работы с данными существенно упростить.

Таким образом, для экономии ресурсов, стоит обратить внимание на in memory базу данных. Для примера был выбран Tarantool, так как он соответствует требованиям ACID, позволяет откатывать изменения, а также имеет вторичные индексы и хранимые процедуры Lua-script. В отличие от Memcached или Redis, которые могут быть представлены только парами ключ / значение, объекты базы данных Tarantool позволяют использовать запросы реляционного стиля, которые могут быть написаны в Lua или SQL.

Выбор базы данных Tarantool

Tarantool – это быстрая база данных с открытым исходным кодом, которая хранит все свои данные в оперативной памяти. Благодаря такому способу хранению данных можно получить быстрый доступ к ним[4]. То, что Tarantool хранит их в оперативной памяти – не значит, что это небезопасно, и данные можно потерять. В Tarantool существует механизм сохранения данных – логи журналов и есть бинарные снимки состояний. Эти два механизма работают вместе, т.е. есть точки с сохраненными данными. Модель данных базы данных Tarantool – «ключ-значение», в которой каждое значение – это кортеж из произвольного количества полей. Например: (1. «name», «address», «country»), а первый элемент кортежа – это первичный ключ, по которому осуществляется доступ ко всему кортежу.

Для сравнения, Tarantool может выполнять один миллион запросов в секунду на одном ядре, тогда как для этого реляционной базе данных потребуется от 20 до 722 ядер.

Миграция из MySQL в Tarantool

Для имитации нагруженного проекта, использовалось тестовое приложение месенджер. Для его написания использовался язык Java 1.8 и фреймворк JUnit 4.11 для написания тестов. Для достижения высокой нагрузки, использовался инструмент с открытым исходным кодом – Apache JMeter 4.0. Также использовался MySQL 5.7 Server Master и Tarantool 1.7.

Основные функции системы обмена сообщениями в приложении – отправка уведомлений / сообщений пользователю или группе пользователей. Группы пользователей могут быть сформированы по некоторым параметрам: таким как права доступа, географическое расположение пользователей и т. д. (рис. 1).

Система позволяет получателям отвечать на сообщения. Она также отслеживает, кто прочитал сообщение, а кто нет. Кроме того, система имеет встроенный механизм напоминания, который позволяет отправителю создавать напоминание, а затем отправляет напоминание всем получателям соответственно.

Для нагрузочного тестирования использовалась виртуальная машина с 1GB памяти, скоростью чтения SSD в 4447042145 байт на секунду и процессорным ядром со следующими параметрами:

- processor: 2.5 GHz Intel Core i7;
- cpu MHz: 1600;
- cache size: 10302 MB.

Выполнялись запросы к API на создание сообщений, на их чтение, на проверки статуса просмотра сообщений, на создание и прочтение напоминаний.

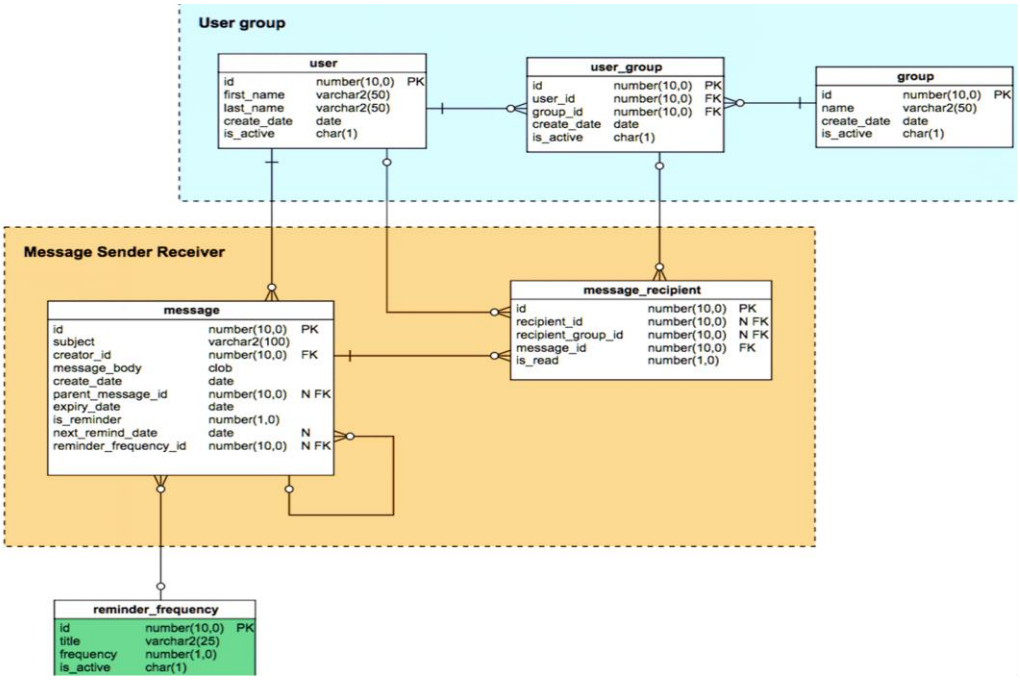

Рис. 1. Модель базы данных

Так как MySQL и Tarantool ничего не знают друг о друге, используется репликатор, который получает bin logs с MySQL Master, и всё это записывает в Tarantool. Репликатор – это не отдельный модуль в Tarantool, а отдельный Daemon – инструмент, независим от Tarantool и MySQL.

У команды Mail.Ru Group есть репликатор, который использует libslave, разработку с открытым исходным кодом для чтения событий с сервера MySQL Master [5]. Репликатор, полностью собирается статически и не использует системных MySQL библиотек. Если нужно, чтобы у одного Master'а было 5 Slave'ов, то придётся запускать 5 Daemon'ов. Один репликатор может реплицировать только в один Tarantool. Когда репликатор запускается, он захватывает все данные от master на основе файла конфигурации, который указывает, какие базы данных / таблицы необходимо реплицировать. Тем не менее, для запуска репликатора требуется просто экземпляр Tarantool с пустым пространством. Для сравнения результатов было создано 3 MySQL Slave, а также было поднято 3 экземпляров Tarantool, после подключения репликатора нагрузка на MySQL Slave'ах резко упала. После, этого можно не использовать MySQL Slave-ы, так как Tarantool будет стабильно справляйся с нагрузкой.

Главное здесь, что каждый поток работает параллельно и не мешает другим потокам выполнять свою работу. Чем больше параллельная и тяжелая рабочая нагрузка, тем меньше системных вызовов для каждого запроса и больше запросов в секунду, которые может обрабатывать система.

При сравнении нагрузки на систему на виртуальной машине, был выявлен спад нагрузки в 2.5 раза при подключении Tarantool для чтении данных на разогретом кэше. Такой подход репликации лучше использовать в системах, где большинство запросов к базе данных на чтение. Так как Tarantool in-memory база данных, лучше использовать ее, в тех случаях, когда незначительная потеря данных не так критична.

Репликация в Tarantool, которая получилась, работает быстрее, чем в MySQL. Данная работа показывает, что в условиях Highload будет более экономичнее использовать пару серверов с инстансами Tarantool вместо десятков серверов с MySQL Slave-ами.

Список использованных источников:

1. Knowledge Base of Relational and NoSQL Database Management Systems. [Электронный ресурс]. – Режим доступа: https://dbengines.com/en/ranking.
2. ACID: Concurrency Control with Transactions. [Электронный ресурс]. – Режим доступа: https://mariadb.com/kb/en/library/acid-concurrency-control-with-transactions/.
3. Что такое Highload. [Электронный ресурс]. – Режим доступа: https://ruhighload.com/Что+такое+highload.
4. Tarantool 1.7 manual. [Электронный ресурс]. – Режим доступа: https://tarantool.io/en/doc/1.7/.
5. MySQL slave replication daemon for Tarantool. [Электронный ресурс]. – Режим доступа: https://github.com/tarantool/mysql-tarantool-replication.

Общта А.Ф.
доктор технічних наук, професор;
Руда М.В.
кандидат технічних наук, асистент;
Сорока І.Й.
старший викладач;
Мудрак А.В.
студент,
Національний університет «Львівська політехніка»

ЕКОЛОГІЧНА ОЦІНКА НАВКОЛИШНЬОГО СЕРЕДОВИЩА ТА СТІЙКІСТЬ ЕКОЛОГІЧНИХ СИСТЕМ

У сучасному соціально-економічному середовищі матеріальні потоки і процеси відбуваються за лінійною схемою. Але, на нескінченному відрізку часу матеріали, що пройшли через техносферу, заново повертаються у навколишнє середовище як сировина. Концепція життєвого циклу розглядає продукти/послуги з початку їх фізичного виникнення і до моменту припинення їх функціонування.

Вихідні потоки енергії можуть бути як відходами досліджуваної системи, так і слугувати ресурсами (вхідними потоками) в іншу систему [1]. На всіх стадіях життєвого циклу консорційних екотонів захисного типу (тут і далі КЕЗТ) має місце певне забруднення, використовуються енергія та матеріали.

Для визначеня екологічного індексу КЕЗТ необхідно здійснити: інвентаризаційну → екологічного впливу → можливостей поліпшення. Всі три