1. zabbix [Электронный ресурс]. –Режим доступа:https://www.zabbix.com. – Дата доступа: 04.04.2018.
2. ООО "Модуль-Проекты" [Электронный ресурс]. –Режим доступа:http://onreader.mdl.ru. – Дата доступа: 03.04.2018
3. Zabbix и UbuntuSnappyCore[Электронный ресурс]. –Режим доступа:http://vasilisc.com/zabbix-ubuntu-snappy-core. – Дата доступа: 25.03.2018

ОПТИМИЗАЦИЯ ЗАПРОСОВ К БАЗЕ ДАННЫХ С ПОМОЩЬЮ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Шараев Е.В.

Луцик Ю.А. – к.т.н.,доцент

В наше время существует большое число решений в области систем управления базами данных. Несмотря на растущую сложность реализации и большое обилие алгоритмов, обеспечивающих бóльшую производительность, по-прежнему существует некоторое число проблем, связанных с производительностью запросов. В то время как одна часть проблем с производительностью решается с помощью увеличения технических характеристик машин, на которых развернуты базы данных, создания реплик, тонкой настройки конфигураций СУБД либо других оптимизаций, существует так же возможность использования алгоритмов машинного обучения для выбора наиболее оптимальных стратегий для взаимодействия с хранящимися в базе данными.

Идея применения алгоритмов машинного обучения не является новой и уже довольно давно применяется для оптимизации работы различных действий.

Одной из наиболее продвинутых РСУБД в наши дни является OracleDatabase, которая имеет внушительное количество конфигураций, а также реализованных алгоритмов оптимизации производительности, включая алгоритмы машинного обучения. Для оптимизации запросов в Oracle (как и в большинстве других РСУБД) применяется стоимостной оптимизатор (CBO, CostBasedOptimizer) [1]. Стоимость запроса рассчитывается как совокупная стоимость каждой операции в запросе и определяется в зависимости от планируемого числа записей, которые будут извлечены из базы (значение *cardinality*). Приблизительное значение числа записей извлекается из статистики, которую собирает база. При этом отдельно учитывается стоимость для различных операций: чтения и записи на диск (IO_COST), стоимость действий на процессоре (CPU_COST) и так далее. Для решения проблемы производительности при взаимодействии с данными, которые имеют зависимость между собой (например, атрибуты *военнообязанный* и *возраст* у таблицы *пользователи*) применяются алгоритмы, позволяющие подбирать наилучший план. Данная реализация имеет название AdaptiveQueryOptimization [2]. Она позволяет выбирать наиболее производительный план при повторном выполнении запросов одного типа, обучаясь после каждого запроса. Несмотря на все преимущества, существует так же и недостаток – платная и довольно дорогостоящая лицензия на использование OracleDatabase.

Так же существует аналогичная экспериментальная реализация адаптивных запросов для PostgreSQL [3], которая, в свою очередь, имеет бесплатную лицензию. Как и в OracleDatabase здесь используется метод подбора значения cardinality для схожих запросов. Однако данная реализация так же имеет несколько недостатков. В качестве признаков здесь используются значения селективности (процент кортежей, удовлетворяемых конкретному условию), извлекаемые из статистики, что может работать не совсем корректно, так как неэквивалентные условия в запросе могут превращаться в эквивалентные по значению признаки. Например, если положить, что условия *age< 15* и *age> 25* имеют одинаковое значение селективности 0.05, то условия *age< 15 ANDreservistISTRUE* и *age> 25 ANDreservistISTRUE* будут иметь эквивалентное предсказание значения *cardinality*, что не является верным, так как в действительности первый запрос, в отличие от второго, не вернет кортежей (предполагается, что не существует военнообязанных моложе 15 лет). Кроме того, существуют так же и технические недочеты, такие как отсутствие оптимизации на slave-репликах и так далее. Хотя упомянутая наработка не является наиболее оптимальной, важно отметить, что выбранное направление в действительности имеет потенциал, однако нуждается в пересмотре некоторых решений.

Таким образом, на высоком уровне абстракции планируемый подход заключается в следующем: для множества однообразных запросов создать множество соответствующих им наборов параметров модели машинного обучения, при поступлении нового запроса использовать соответствующий набор параметров модели для предсказания, а затем, после выполнения самого запроса, с учетом предсказания сравнить с получившимся результатом и, таким образом, настроить параметры модели. Однообразность запросов определяется посредством схожести их условий (например, *col1 > 1 ANDcol2 > 1* и *col1 > 2 ANDcol2 > 2* будем считать однообразными). Модель представляет собой реализацию определённого алгоритма машинного обучения (например, перцептрон, набором параметров которого будет массив весов). В качестве признаков используются операнды условий. Предсказанием является число, отражающее ожидаемое количество записей, которые удовлетворяют соответствующему условию. Используя число записей, которое было предсказано, а также число записей, которое было получено после выполнения запроса, мы можем осуществить оптимизацию нашей модели и иметь более точное предсказание при последующих запросах.

Список использованных источников:
1. Jonathan Lewis, Cost-Based Oracle Fundamentals;
2. Tariq Farooq, Charles Kim, Nitin Vengurlekar, Sridhar Avantsa, Guy Harrison, Syed Jaffar Hussain, Oracle Exadata Expert's Handbook;
3. Oleg Ivanov, Sergey Bartunov, Adaptive Cardinality Estimation.

СЕРВИСЫ РАСШИРЕНИЯ ФУНКЦИОНАЛА В ОБЛАЧНЫХ ТЕХНОЛОГИЯХ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Янчевский В.И.

Селезнёв И.Л. – к. техн. н, доцент

В докладе рассматривается построение локального облака (private cloud) с обеспечением высокого уровня безопасности данных, предоставляющего функционал веб-сервиса, которое используются в системах с SOA (сервис-ориентированной архитектурой); локальное облако предоставляет услуги через программные интерфейсы SOAP и REST другим веб-сервисам или приложениям-клиентам. Предлагается принцип построения многосервисной облачной структуры, как совокупности набора элементов, представляющих функционально-независимые многопользовательские облачные сервисы, отвечающие требованиям архитектуры SOA.

В процессе развития облачных технологий стало ясно, что концепция облака гораздо шире, чем просто использование виртуальных вычислительных ресурсов. Следуя за спросом, требуется разрабатывать более широкий функционал решений для облачных технологий, включая инструменты для полноценного управления виртуальными машинами и базовыми сценариями.

SOA (сервис-ориентированная архитектура) рассматривается в виде совокупности веб-сервисов, часто строящихся как распределённые системы и работающие на разных платформах. Веб-сервисы могут взаимодействовать как друг с другом, так и с приложениями, созданными на основе SOA, посредством сообщений. Эти сообщения передаются стандартными протоколами, которые получили наибольшее распространение – SOAP (протокол обмена xml-данными) и REST (стиль архитектуры программного обеспечения для распределенных систем), в формате (языка разметки) XML и/или JSON.

В качестве транспорта сообщений используют протокол HTTP. В SOAP применяется специальный язык описания веб-сервисов и доступа к ним (WSDL) в формате XML. В REST нет такого описания типа, так как используется фиксированный набор методов доступа к веб-сервисам: GET, POST, PUT, DELETE. Указанные веб-сервисы основаны на открытых международных стандартах, которые поддерживаются современными операционными системами.

Программное обеспечение как сервис (Software software as a service, Saas) является перспективным направлением в облачных технологиях стимулирующим развитие архитектуры SOAP.

SaaS предполагает, что пользователь получает в распоряжение ПО, функции которого доступны через веб-интерфейс, в то время, как основная программная часть приложения находится на сервере разработчика. (Рис.1)

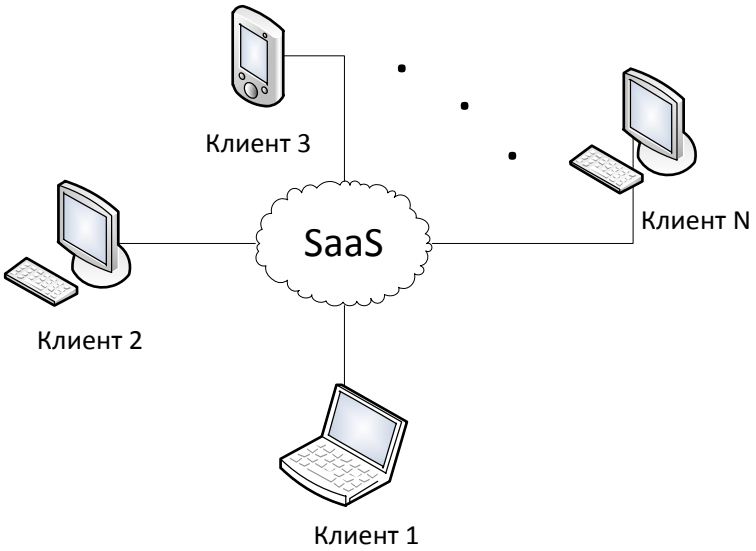

Рисунок 1 – Однооблачная структура сервиса SaaS

Наиболее перспективными средствами интеграции приложений SaaS в системы с архитектурой SOA