1. Вознаграждение в обучающем процессе или не предусматривается или предполагается после завершения проекта. В реальном процессе команда работает не на перспективу, а получает быстрый отклик на свою работу.
2. Невозможность сэмулировать процесс полностью. Ежедневный митинг невозможен в принципе, поскольку занятия проходят раз в неделю. Также невозможно воссоздать полный рабочий день, поскольку учебной программой предусмотрено времени гораздо меньше.

В общем случае можно уменьшить влияние отрицательных отличий. Способы уменьшения были выявлены опытным путём в процессе разработки проектов.

Одним из способов уменьшения влияния является привязка графика работы к расписанию занятий. Таким образом нивелируется сложность воссоздания рабочего процесса, а у команды вырабатывается ритм работы и нет соблазна отложить разработку проекта на конец спринта.

Снизить влияние отрицательных отличий позволяет использование средств удалённого общения и выбор проектов небольшого размера или проектов, с которыми команды уже работали.

Таким образом, используя все положительные отличия и уменьшая влияние отрицательных, возможно изучать Scrum в “реальном” процессе разработки проектов. В итоге студенты получают практические навыки работы в рамках данного фреймворка, что в будущем поможет им быстрее влиться в рабочий процесс.

Список использованных источников:

1. ScrumAlliance: The Scrum Guide [Электронный ресурс]. – Режим доступа: https://www.scrumalliance.org/learn-about-scrum/the-scrum-guide – Заглавие с экрана. – (Дата обращения: 10.04.2018).

2. 11th Annual State of Agile Report [Электронный ресурс]. – Режим доступа: https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2 – Заглавие с экрана. – (Дата обращения: 12.04.2018).

3. Искра, Н. А. Технологии разработки и тестирования программного обеспечения [Электронный ресурс] / Н. А. Искра // Учебная программа учреждения высшего образования по учебной дисциплине для специальности 1-40 02 01 Вычислительные машины, системы и сети – 2015. – Режим доступа: https://libeldoc.bsuir.by/handle/123456789/30530. – (Дата обращения: 12.04.2018).

СИСТЕМА КОНТРОЛЯ СОСТОЯНИЯ КОМПОНЕНТОВ ЛОКАЛЬНОЙ СЕТИ

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

ШантаровичВ.Д.

Ганжа В.А. – к. ф.-мат. н,доцент

Концепция Internet of Things, как множество физических объектов (вещей), взаимодействующих с друг другом и средой, является нашим будущим и от него не убежать.Вещи вокруг нас становятся умнее с каждым днём и многие используемые задачи должны быть автоматизированы. Так как наша зависимость от IoT растёт с каждым днём, мы должны быть уверены, что такие устройства работают надёжно, безопасно и выполняют задачи, возложенные на них. Реализация такой системы контроля требует безопасную операционную систему для коммуникации с вещами, так как они могут пересылать персональную информацию, и, одновременно, эффективную, так как аппаратные ресурсы IoT скудны. Вот где Zabbixвступают в игру. Zabbix – высоко интегрированное решение мониторинга сети, которое предлагает множество функций в одном пакете.

Zabbix– это программное обеспечение для мониторинга многочисленных параметров сети, жизнеспособности и целостности серверов. Zabbix использует гибкий механизм оповещений, что позволяет пользователям конфигурировать уведомления основанные на e-mail практически для любого события. Это позволяет быстро реагировать на проблемы с серверами. Zabbix предлагает отличные функции отчетности и визуализации данных основанные на данных истории. Это делает Zabbix идеальным для планирования мощности[1].

Zabbix предлагает крошечного, скромного к ресурсам, агента для таких устройств: датчики в доме, торговый аппарат, электронное устройство и так далее. Данный агент собирает данные о производительности, доступности, статусе данных и других, различных метриках, передавая их другим устройствам или в облако. Создаётся истинная синергия между вещами IoT.

Zabbix состоит из нескольких основных программных компонентов.

Zabbix-сервер является основным компонентом, которому агенты сообщают информацию и статистику о доступности и целостности. Сервер является главным хранилищем, в котором хранятся все данные конфигурации, статистики, а также оперативные данные.

Как таковая вся информация о конфигурации, а также данные собранные Zabbix, хранятся в базе данных.

Для легкого доступа к Zabbix из любого места и с любой платформы, поставляется интерфейс на основе Веб. Интерфейс является частью Zabbix сервера и обычно (но не обязательно) работает на том же самой физической машине, что и сервер.

Zabbix прокси может собирать данные о производительности и доступности от имени Zabbix сервера. Прокси является опциональной частью Zabbix; однако он может быть полезен, чтобы распределить нагрузку одного Zabbix сервера.

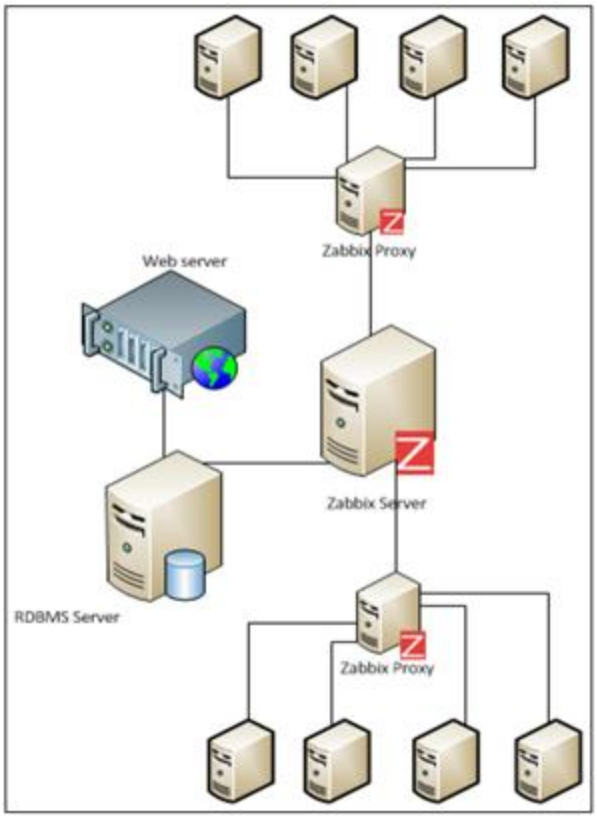

Рис. 1 –инфраструктура Zabbix

Zabbix агенты разворачиваются на наблюдаемых системах для активного мониторинга за локальными ресурсами и приложениями, и для отправки собранных данных Zabbix серверу или прокси.

Кроме того, важно сделать шаг назад и взглянуть на весь поток данных в Zabbix. Для того чтобы создать элемент данных, который будет собирать данные, вы должны сначала создать узел сети. Перемещаясь в другой конец спектра Zabbix, у вас должен быть элемент данных, чтобы создать триггер. У вас должен быть триггер, чтобы создать действие.

Таким образом, если вы хотите получать оповещения о слишком высокой загрузке CPU на Сервере X, вы сначала должны создать запись о узле сети для Сервера X, затем элемент данных для наблюдения за CPU, затем триггер, который сработает, если загрузка CPU будет слишком высокой, а затем действие, которое отправит вам email.

Zabbix можно определить как распределённую систему мониторинга с централизованным веб-интерфейсом (в котором мы можем управлять практически всем). Наряду с основными функциями, мы выделим следующие:

- Zabbix имеет централизованный веб-интерфейс;
- его сервер может выполняться на большинстве Unix-подобных операционных систем;
- система мониторинга имеет внутренне присутствующих агентов для большинства операционных систем Unix, Unix-подобных и Microsoft Windows;
- данная система легко интегрируется в другие системы благодаря своему API, доступному во многих различных языках программирования и параметрах, поддерживаемых Zabbix самой по себе;
- Zabbix может осуществлять мониторинг через SNMP (v1, v2 и v3), IPMI, JMX, ODBC, SSH, HTTP(s), TCP/UDP, а также Telnet;
- Эта система мониторинга даёт нам возможность создания индивидуальных элементов и графиков, а также интерпретации данных;
- Система просто настраивается под персональные требования[2].

Одно из наиболее важных свойств Zabbix состоит в его ёмкости для хранения данных истории. Это свойство является жизненно важным в процессе предсказания тенденций. Предсказание наших тенденций не является простой задачей и является важным для рассмотрения того бизнеса, который мы обслуживаем, а когда мы рассматриваем историю данных, нам следует увидеть будут ли присутствовать повторяющиеся периоды, либо существует некоторая формула, которая может описать нашу тенденцию [3].

Список использованных источников:

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.