

УДК 681.3

Алиев А.А., д.т.н.,
Самедов Р.Б.

АЛГОРИТМ СИСТЕМЫ MVAAS ДЛЯ МАССОВОЙ ОТПРАВКИ PUSH-УВЕДОМЛЕНИЙ GOOGLE GCM С ИСПОЛЬЗОВАНИЕМ МНОГОПОТОЧНОСТИ

Бакинский Государственный Университет

aaliyev@mail.ru
ramin.samedov@gmail.com

Представлены современные технологии отправки уведомлений в облачных приложениях. Предложен алгоритм отправки Push-уведомлений при помощи использования многопоточности. В результате использования предложенного алгоритма получен выигрыш в передаче Push-уведомлений в несколько раз. Проведены эксперименты, результаты которых приведены в таблице и на графике

Ключевые слова: Облачные вычисления, разработка мобильных приложений, распределенная система, mVaas, push-уведомления, GCM, многопоточность

Введение

Впервые термин облачные вычисления был озвучен Джозефом Карлом Робнетт Ликлайдером, в 1970 году. В эти годы он был ответственным за создание ARPANET (Advanced Research Projects Agency Network). Его идея заключалась в том, что каждый человек на земле будет подключен к сети, из которой он будет получать не только данные, но и программы. В те времена также была идея о том, что вычислительные мощности будут предоставляться пользователям как услуга (сервис). На этом развитие облачных технологий было приостановлено до 90-х годов, после чего ее развитию поспособствовал ряд факторов:

Основным фактором стало расширение пропускной способности Интернета. В 90-е годы интернет не позволял получить значительный скачок в развитии в облачной технологии, так как практически ни одна компания и технологии того времени не были готовы к этому. Однако сам факт ускорения Интернета дал толчок скорейшему развитию облачных вычислений.

Одним из наиболее значимых событий в данной области было появление Salesforce.com в 1999 году. Данная компания стала первой компанией предоставившей доступ к своему приложению через сайт, по сути данная компания стала первой компа-

нией предоставившей свое программное обеспечение по принципу – программное обеспечение как сервис (SaaS).

Следующим шагом стала разработка облачного веб-сервиса компанией Amazon в 2002 году. Данный сервис позволял хранить, информацию и производить вычисления.

В 2006, Amazon запустила сервис под названием Elastic Compute cloud (EC2), как веб-сервис который позволял его пользователям запускать свои собственные приложения. Сервисы Amazon EC2 и Amazon S3 стали первыми доступными сервисами облачных вычислений.

Другая веха в развитие облачных вычислений произошла после создания компанией Google, платформы Google Apps для веб-приложений в бизнес секторе.

Значительную роль в развитии облачных технологий сыграли технологии виртуализации, в частности программное обеспечение позволяющее создавать виртуальную инфраструктуру.

Развитие аппаратного обеспечения способствовало не столько быстрому росту облачных технологий, сколько доступности данной технологии для малого бизнеса и индивидуальных лиц. Что касается технического прогресса, то значительную роль в этом сыграло создание многоядерных процессоров и увеличения емкости накопите-

лей информации.

Облачные вычисления (англ. cloud computing) в настоящее время — это технология распределённой обработки данных, в которой компьютерные ресурсы и мощности предоставляются пользователю как Интернет-сервис. Предоставление пользователю услуг как Интернет-сервис является ключевым. Однако под Интернет-сервисом не стоит понимать доступ к сервису только через Интернет, он может осуществляться также и через обычную локальную сеть с использованием веб-технологий [1].

В данной работе рассмотрена платформа BaaS - это облачный сервис типа Backend as a Service (BaaS). Бэкенд как сервис (Backend as a Service) предоставляет готовую облачную серверную инфраструктуру для всех типов приложений. Это позволяет разработчикам, выигрывать время и деньги, отказавшись от разработки сервера, и сфокусироваться на функциональности приложений.

BaaS позволяет программистам не беспокоиться об обработке стандартных функций, так как логика обработки функций уже реализовано в сервисе BaaS. Единственное что необходимо программисту так это, написать уникальную клиентскую часть. Таким образом, выигрыш во времени в реализации программного обеспечения увеличивается в разы.

Mobile Backend as a service (MBaaS) – это технология BaaS в мобильных устройствах. Разница между BaaS и MBaaS лишь в том, что MBaaS создано исключительно для работы в мобильных приложениях. Таким образом, MBaaS является составной частью BaaS сервиса.

В данной статье рассмотрено создание алгоритма в среде mbaas сервиса для массовой отправки push-уведомлений google gcm с использованием многопоточности.

Объект, цель и задачи исследования

Объектом исследования является процесс отправки push-уведомлений на мобильные устройства, включая смартфоны и планшетные компьютеры (Android), используя параллельный алгоритм в облачном сер-

висе. Push-уведомления могут содержать в себе помимо текста, также кадры с камер наблюдений и разных датчиков с графической аннотацией, что повышает информативность отправляемого сообщения. Данная технология должна использовать эффективный механизм доставки push-уведомлений для работы в режиме реального времени при минимальной нагрузке на канал связи и минимальном расходе батареи, для быстрого реагирования на push-уведомления.

Цель исследования - разработка средства работы отправки push-уведомлений облачного приложения и получения выигрыша в скорости в доставке уведомлений. Для достижения цели, необходимо максимально оптимально использовать облачное приложение и его ресурсы.

Задача исследования - разработать и создать эффективный алгоритм при помощи которого удастся получить выигрыш в несколько раз при отправке push-уведомлений на мобильные устройства, включая смартфоны и планшетные компьютеры.

Push-технология

Push-уведомления – это краткие всплывающие уведомления, которые появляются на экране мобильного телефона и сообщают о важных событиях и обновлениях. Их также можно рассматривать как небольшой диалог между потребителем и продавцом. При эффективном использовании эти краткие информативные сообщения являются мощным маркетинговым инструментом.

Существует множество вариантов использования Push-уведомлений. Можно выделить четыре основных направления в применении этих уведомлений для улучшения бизнеса.

- Основная масса Push-уведомлений (41 %) используется для сообщения о выпуске новых приложений или о появлении новых версий уже существующих.
- Вторым, не менее важным, применением (24%) является рассылка специальных предложений или ссылок на определенные посадочные страницы.
- Для информирования пользователей о новом контенте, доступном в приложении задействовано 14 % Push-уведомлений

- 12 % — используются для рассылки реферальных или рекламных ссылок.
- Остальные 9 % используются для других разновидностей информирования

Концепция Push-уведомлений очень быстро проникла в сферу маркетинга. Но существует тонкая грань между сотрудничеством и чрезмерной навязчивостью. Push-уведомления должны предоставлять пользователю ценную информацию и способствовать длительному сотрудничеству. Поэтому, сообщения должны быть своевременными и содержать релевантный контент. Например, если разработчик детских приложений продаст место под Push-уведомления компании по производству кофе, то такие сообщения будут расцениваться как спам. Это не будет способствовать созданию взаимоотношений – приложение потеряет свою популярность.

Также, эффективным применением Push-уведомлений является информирование пользователя о действиях его друзей. Например, приложение твиттера отправляет уведомление о полученном сообщении и новых твитах. Некоторые маркетологи используют технологию GPS для отправки Push-уведомлений, когда пользователь находится недалеко от их торговых точек. Главное – это предоставлять информацию, которую пользователи хотят получать своевременно.

Стоит отметить, что Push-уведомлений не зависят от наличия Интернета в телефоне пользователя, в то же время их можно использовать совместно с мобильным продвижением в рамках одной маркетинговой стратегии, тем более, что проникновение мобильного Интернета растет [2].

Push-технология - это один из способов уведомления клиентов, который предполагает наличие мобильного устройства с установленным приложением для осуществления подписки к службам на сервере приложений. Идея подхода состоит в получении информации непосредственно мобильным приложением по совершению определенного события. В данном случае событие является триггером для сервера приложений, который срабатывает при определен-

ных условиях, например, в соответствии с установленным расписанием или завершением определенной работы. Сервер приложений и осуществляет рассылку требуемой информации клиенту. Реализация данной технологии подразумевает использование клиент-серверной архитектуры. В настоящее время эта технология является одним из вариантов предоставления контента пользователям мобильных устройств, например, стоимости акций на биржах, прогноза погоды, либо любого другого информационного сообщения. Большим достоинством является то, что в данном случае от получателя не требуется каких-либо действий в отношении отправителя. Данная технология начинает активно использоваться именно в силу минимальных затрат, необходимых для ее запуска. Идея развития push-технологии в мобильных устройствах предполагает, что основное приложение находится в неактивном режиме, и призвана прежде всего преодолеть ряд ограничений, с которыми сталкиваются пользователи и разработчики:

- ограниченный емкостной ресурс системы питания
- относительно высокий уровень оплаты услуг за контент телекоммуникационным провайдерам (SMS, MMS)
- высокий уровень конкуренции на рынке мобильных операционных систем и как следствие, развитие дополнительных бесплатных сервисов
- сложность реализации взаимодействия с телекоммуникационной инфраструктурой

Данные факторы создают основу для популярности мобильных приложений, которые способствуют развитию инфраструктуры для передачи push - сообщений [3].

BAAS – платформа бэкенд как сервис (Backend as a Service), которая предоставляет готовую облачную серверную инфраструктуру для всех типов приложений. Это позволяет разработчикам, стартапам и крупным компаниям выигрывать время и деньги, отказавшись от разработки своего сервера, и сфокусироваться на функциональности приложений, их продвижении и пользователях. API платформы доступны для следующих клиентских окружений: JavaScript, Android, iOS, Windows Phone, Flex/AIR.

Основная идея, заключается в то что имеются готовые серверные сервисы, наборы которых вместе складывают необходимый универсальный кроссплатформенный бэкенд для любого проекта. И это позволяет выиграть большое количество времени, ведь нет необходимости написания и поддержки своего серверного бэкенда [4].

Сервис Google Cloud Messaging (GCM) нужен для того, чтобы приложение всегда показывало актуальные данные пользователю. Схема работы сервиса включает в себя три компонента. Непосредственно сервер GCM, пуш-сервер и устройство с установленным приложением. Алгоритм работы простой: устройство регистрируется в GCM, получает registration ID – некий токен, который используется в дальнейшем, – сохраняет его у себя локально и передает серверу. Далее пуш-сервер использует этот registration ID для отправки сообщений приложению на устройстве [5], где процесс принятия push-уведомления мобильным приложением отображен на рис.1.

Описание рис.1 :

А. Мобильное приложение регистрирует Android-устройство, на котором запущено приложение, в сервисе GCM.

В. Мобильное приложение принимает регистрационный ID, отправленный на устройство сервисом GCM.

С. Мобильное приложение отправляет регистрационный ID на хост, где он хранится для дальнейшего использования.

Д. Хост отправляет сообщение в GCM, указав регистрационный ID устройству в качестве получателя.

Е. GCM принимает сообщение от хоста и отправляет его тем устройствам, ID которых было указано на предыдущем шаге.

Ф. Мобильное устройство, принимает сообщение, перенаправляет сообщение в приложение, которое отображает уведомление.

Регистрационный ID идентифицирует устройство, на котором запущено приложение, и не более того. В процессе получения уведомления нет никакого соединения с какой-либо учетной записью. Система оповещения пользователей добавляет возможность создавать отношение один-ко-многим. Таким образом, вместо того, чтобы отпра-

влять уведомления пользователям, указав список устройств на хосте, предоставляете один токен, называющийся “ключом уведомлений” (notificationkey). Этот ключ предоставляет из себя длинную строку, сгенерированную по запросу сервисом GCM. Для того, чтобы иметь ключ уведомления для конкретного пользователя, необходимо найти из списка идентификаторов регистрационный ID того устройства, который привязан к этому пользователю. Затем предоставить локально-уникальную строку этого ID, и, наконец, отправить эти данные с помощью HTTP запроса POST в сервис GCM. В ответ приходит новый ключ уведомления, привязанный к конкретному устройству. Когда пользователь добавляет или удаляет устройство, можно отправлять не всю информацию, а только о внесенных изменениях.

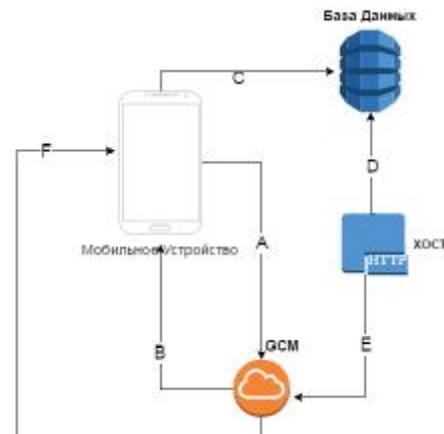


Рис.1. Схема принятия уведомлений мобильными приложениями

Описания проблемы

Облачное приложение может обрабатывать данные и отправлять Push-уведомления в асинхронном режиме. В такой модели на очереди событий собираются все сообщения необходимые для отправки. При возникновении события отправки Push-уведомления, поток, который обрабатывает эту очередь, берет событие с начала очереди, и выполняет связанный с этим событием код. Пока очередь не пуста процессор будет занят работой. Таким образом, может быть единственный поток, обрабатывающий очередь событий, который будет использовать всего один процессор облачного приложения. В результате получается проигрыш в

скорости отправки Push-уведомлений.

Решения проблемы

Для решения выше описанной проблемы, в облачном приложении создаем некоторое количество потоков (пул). Пул потоков — это коллекция потоков, которые могут использоваться для выполнения нескольких задач в фоновом режиме. Каждому пулу потоков передаем задачу и данные для обработки. Задачи выполняются параллельно. Эти потоки не имеют общих данных, следовательно нет накладных расходов на синхронизацию, что делает работу достаточно быстрой. После завершения работы поток не убивается, а лежит в пуле, ожидая следующей задачи. Это убирает накладные расходы на создание и удаление потоков.

Алгоритм параллельной отправки сообщений

Шаг 1. Облачный сервер собирает все регистрационные коды, кому необходимо отправить сообщения. Для этого собирается обычный массив, где в качестве параметра используется \$regId - регистрационный код телефона:

```
$registration_id = array($regId);
```

Шаг 2. Облачный сервер заранее знает число своих процессоров. Обозначим их как P=P1,P2, .. ,Pn где n число процессоров.

Шаг 3. Формируется одинаковое сообщение для всех получателей

```
$message = array("msg" => $message_text), где $message_text - текст сообщения
```

Шаг 4. Создается Pi пулы потоков, где i меньше количество процессоров облачного приложения

Шаг 5. Каждый входящий запрос назначается потоку Pi из пула, таким образом, запрос обрабатывается без задействования главного потока и задержки обработки последующих запросов.

Шаг 6. Каждый Pi поток отправляет отдельную задачу по отправке запроса к GCM к определенному процессору \$Proc_i, где i номер процессора. Таким образом для каждого сообщения из массива \$registration_id (массива шага 1) на n процессоров будет оправлен следующая зада-

```
ча $result = $gcm->send_notification($registatoin_ids, $message,$Proc_i); где $gcm - объект из библиотеки google, $Proc_i i-ый процессор
```

Шаг 7. В результате мобильное приложение должно получить и правильно отобразить сообщение, полученное от GCM.

Практическая реализация алгоритма

В облачном приложении было создано два способа отправки сообщений. Первый способ назовем его обычной отправкой без использования пулов потоков. Вторым способом многопоточный с использованием пулов и нескольких процессоров.

В эксперименте 1 было отправлено 500 одинаковых уведомлений различным приложениям с использованием многопоточности, так и без него. С использованием многопоточности за 5 секунд отправилось 500 уведомлений, но обычным способом за 15 секунд, получили ускорение равное трем, то есть при многопоточном отправлении выигрыш по времени составил три раза быстрее чем при отправке обычным способом. В эксперименте 2 при отправке многопоточным способом 1000 уведомлений отправилось за 9 секунд, когда при обычной отправке за 26 секунд. Ускорение в эксперименте 2 составило 2.88. В эксперименте 3 1500 уведомлений предложенным алгоритмом отправились за 13 секунд, когда обычным способом за 34 секунды, ускорение составило 2.61. В эксперименте 4 при отправке 2000 уведомлений многопоточным способом было отправлено за 16 секунд, когда обычным способом за 45 секунд. Ускорение эксперименте 4 составило 2.81. В эксперименте 5 ускорение составило 2.94. В эксперименте 6 это на 500 уведомлений больше чем в эксперименте 5, ускорение составило 3.04. В эксперименте 7 при отправке 3500 уведомлений ускорение составило 2.96. В эксперименте 8 ускорение составило 2.93. В экспериментах 9 и 10 ускорение составило 2.97 и 2.92 соответственно.

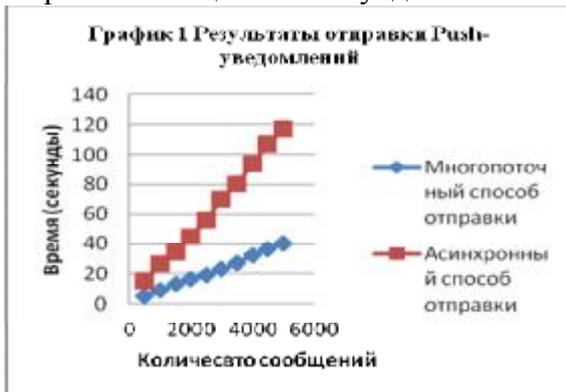
В таблице 1 приведены результаты эксперимента, демонстрирующие эффективность данного алгоритма. При этом количество использованных процессоров во

всех экспериментах была равна восьми.

Таблица 1. Результаты экспериментов

Экс-шт №	Кол-во сообщений	Время при многопоточном способе отправки (секунды)	Время при асинхронном способе отправки (секунды)
1	500	5	15
2	1000	9	26
3	1500	13	34
4	2000	16	45
5	2500	19	56
6	3000	23	70
7	3500	27	80
8	4000	32	94
9	4500	36	107
10	5000	40	117

Результаты отправки Push-уведомлений показаны также в виде графика 1, где по оси X это количество отправленных сообщений, а по оси Y время отправки сообщений в секундах.



Выводы

Таким образом, в результате проведенных исследований было определено, что для разработки средства отправки push-уведомлений из облачного приложения с использованием многопоточности дает выигрыш в скорости в доставки уведомлений в несколько раз. При этом существует требования к облачному приложению - это наличие нескольких процессоров для настройки параллельности.

На основе цели статьи был разработан алгоритм работы облачного приложения, в несколько раз увеличивающий скорость отправки Push-уведомлений. Это было выявлено путем проведения экспе-

риментов и заведения результатов тестов скорости отправки в таблицу и график.

Проведенные эксперименты показали, что в независимости от количество отправляемых Push-уведомлений фиксируется ускорение. Использование указанного алгоритма дает ускорение примерно в несколько раз, в отличие от асинхронного способа отправки Push-уведомлений.

Список литературы

1. “Приключения в Android: уведомления пользователей”, февраль 2015, [Electronic resource]. – Available at: [\www/URL:http://tproger.ru/translations/adventures-in-android-user-notifications/](http://tproger.ru/translations/adventures-in-android-user-notifications/)

2. А.М. Сальников, Е.А. Ярошенко, О.С. Гребенник, С.В. Спиридонов, “Введение в параллельные вычисления. основы программирования на языке си с использованием интерфейса mpi”, Москва 2009, 129 с.

3. Что такое Push-уведомления и как их правильно использовать [Electronic resource]. – Available at: [\www/URL: http://marketingbuzz.info/что-такое-push-уведомления-и-как-их-правильно-использовать.html](http://marketingbuzz.info/что-такое-push-уведомления-и-как-их-правильно-использовать.html)

4. Свой облачный бэкенд в одну строчку кода. Обзор BaaS платформы «Backendless»

[Electronic resource]. – Available at: [\www/URL: https://habrahabr.ru/company/backendless/blog/180367/](https://habrahabr.ru/company/backendless/blog/180367/)

5. Ravi Tamada, “Android Push Notifications using Google Cloud Messaging (GCM), PHP and MySQL”, [Electronic resource]. – Available at: [/http://www.androidhive.info/2014/01/how-to-create-rest-api-for-android-app-using-php-slim-and-mysql-day-2](http://www.androidhive.info/2014/01/how-to-create-rest-api-for-android-app-using-php-slim-and-mysql-day-2)

Статью представлено к печати 20.09.2016