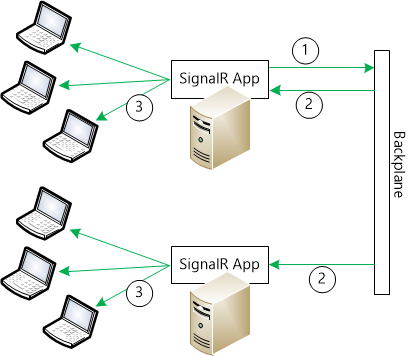
SignalR Scaleout

SignalR появился достаточно давно, года 3 назад были первые версии и нужен он для создания Real-time веб приложений. Во всех блогах показывали “hello word” с чатом, в одно серверной конфигурации, но никто не рассказывал о приложениях из реального мира.

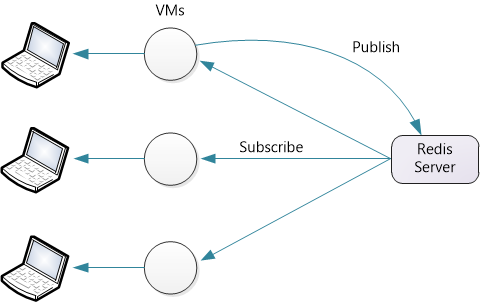
В реальном/взрослом мире веб сервер не работает в одиночку, всегда работает пусть и маленькая, но ферма веб серверов, стоящих за балансировщиком. Следовательно, SignalR должен быть распределенным и событие, возникающее на одном из серверов, приходило и клиентам подключенным к другому. Поскольку о том, что SignalR поддерживает такую возможность не заслуженно не писалось, я хочу исправить это.

В SignalR есть такая фича- backplane, и представляет она из себя сервер баз данных (либо Redis, либо MSSQL Server и для Azure есть ServiceBus. Других официальных нет). На одном из хостов SignalR происходит событие, оно отправляется в базу, а оттуда уже рассылается всем подключенным к нему остальным хостам SignalR, которые ретранслируют это сообщения своим клиентам.

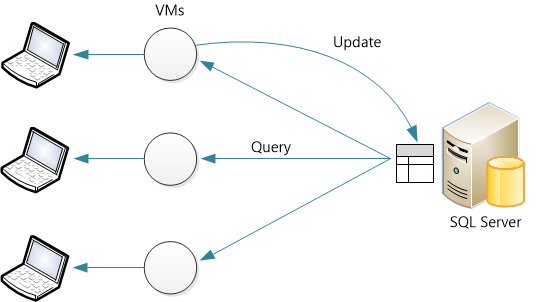


BackPlane в электронике- группа параллельных транзисторов.

В случаи использования Redis используется нативный для него механизм Pub/Sub <http://redis.io/commands/pubsub> <http://redis.io/topics/pubsub> , появившийся с версии Redis 2.8 (хотя и еще с версии 2.0 была возможность работать с каналами и публиковать в них сообщения).



В случаи SQL Server есть 2 варианта: с использованием Service Broker (это быстрый способ поддержки обмена сообщениями) и без его использования.



**Конфигурирование**

Конфигурирование во всех 3 случаях одинаковое и достаточно простое.

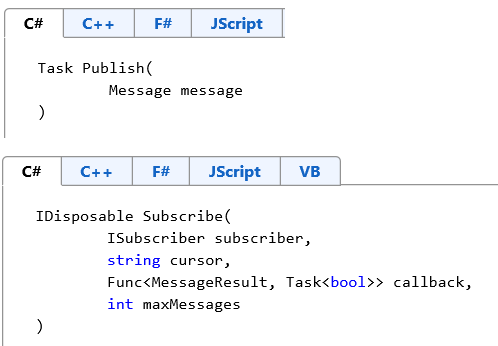


Именно поэтому я бы рекомендовал использовать это решение, а не тратить человеко-месяцы старших разработчиков на написание собственных велосипедов с не гарантированным результатом.

**Реализация**

Чтобы все работало, мы должны реализовать класс, который имплементирует интерфейс IMessageBus. <http://msdn.microsoft.com/en-us/library/microsoft.aspnet.signalr.messaging.IMessageBus(v=vs.100).aspx>

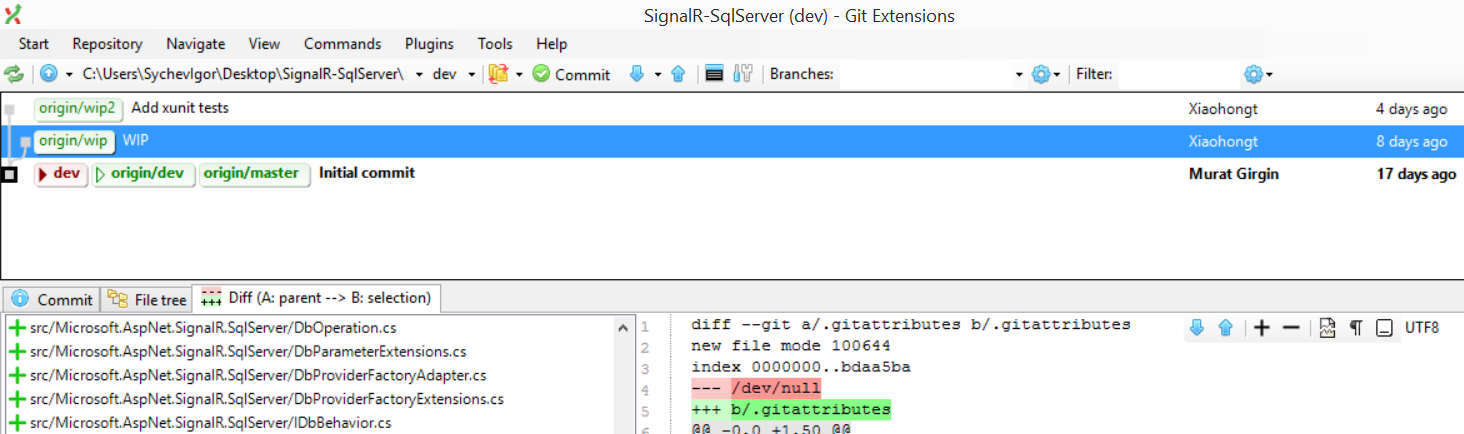
В нем всего 2 метода



Клиент к Redis/SQLServer реализует этот интерфейс через наследование от ScaleOutMessageBus, который в свою очередь наследуется от MessageBus, который уже имплементирует IMessageBus. Если захотим написать сами с нуля, то можно всю эту иерархию наследования пропустить. <http://msdn.microsoft.com/en-us/library/microsoft.aspnet.signalr.redis.redismessagebus(v=vs.100).aspx>

**SQL Server реализация.**

На github есть вот такой проект <https://github.com/aspnet/SignalR-SqlServer> , в котором есть код для backplane использующий SQL Server. Внутреннее устройство я расскажу по нему, хотя не до конца уверен, что это именно то, что есть на nuget.

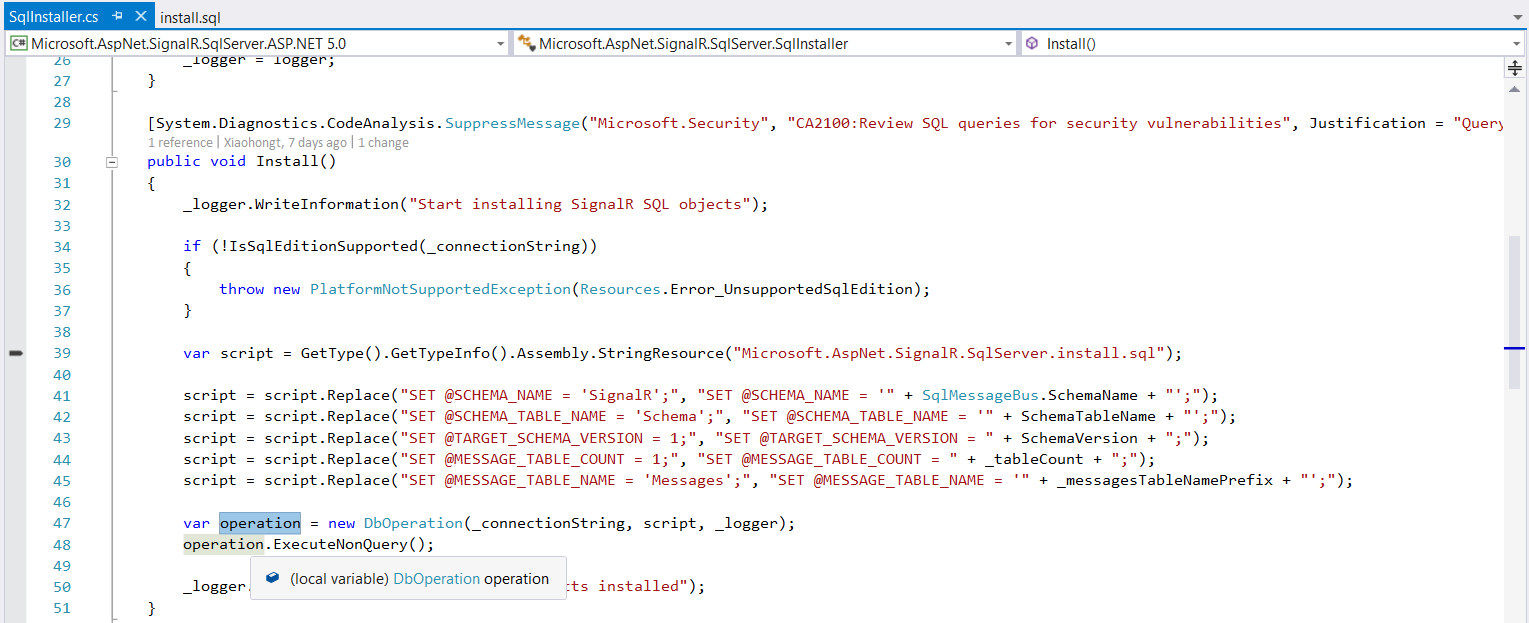
Проект был порядка месяца назад опубликован, а пакеты уже года 2 доступны. Плюс в проекте всего 3 коммита, и 3 ветки, которые слабо объясняют, что в них находится, а главное - в ветке master/dev вообще ничего нет.

Операция Send выполнена как просто вставка в базу данных через SQLCommand.

Получение обновлений проходит через класс SQLStream (Кто знаком с концепциями реактивного программирования узнают код.), где SQLReciever подписывается на обновления в базе.

Для получения обновлений используется SqlDependency <http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldependency(v=vs.110).aspx> . Текст sql команд жестко забит в код и как только происходит изменения таблицы в бд, signalr получит уведомление об этом.

В этом же проекте можно посмотреть, как происходит инициализация базы… Происходит оно через класс SqlInstaller.cs, который читает файл install.sql, через string.replace подменяет значения параметров (имя схемы и еще несколько параметров), затем выполняя скрипт через обычный ado.net sqlcommand.



Одним словом- ракетными технологиями и не пахло.

**Реализация Redis**

В случаи Redis, клиент подписывается на канал и ожидает ответа. Исходного кода nuget пакета на github я не нашел, а декомпилировать не вижу смысла и сейчас объясню почему.

Есть зависимость на nuget, и без декомпиляции можно сказать, что для работы с Redis используется библиотека <http://www.nuget.org/packages/BookSleeve/> Эта библиотека устарела, и надо использовать StackExchange.Redis как и в остальных проектах команды Asp.net (EntityFramework, кэширование, сессии). По этому поводу даже задачка у команды SignalR есть <https://github.com/SignalR/SignalR/issues/2961> и в версии 2.2. команда SignalR обещает переписать. Если будут переписывать, то и смотреть сейчас смысла особого нет.

Текущая версия 2.1.2 от сентября 2014 года.

**Производительность**

Я думаю ни для кого не станет открытием, что использование BackPlane снизит производительность на каждую конкретную ноду кластера (т.к. идет отправка данных на базу данных), и увеличит трафик между серверами т.к. сообщения полетят от базы данных ко всем нодами. Но это та цена, которую надо заплатить для если мы хотим получить горизонтальное масштабирование. Точный процент ни в статьях не указан, ни у меня посчитать не получилось. Все зависит от вашей конфигурации железа, топологии и т.п.

Я провел эксперимент. В моем эксперименте я использовал свой ноуткбук с 2 инстансами приложения типа чата, на asp.net5 + Redis. В своих замерах я ничего интересного не смог показать, т.к. SignalR собирает несколько сообщений в пакет и после какой-то частоты замерить время отправки сообщения уже невозможно. Ну и сам net не обеспечивает сильно высокую точность замера времени, чтобы что-то меньше чем 50мс замерить.

Сами Microsoft говорят о 2 потенциальных сценариях, когда backplane может быть не очень хорошо.

1. Когда очень **много серверов связанные через backplane**. Т.к. слишком много получается трафика между базой и серверами, с каждым новым сервером объем трафика растет квадратично…
2. Для **приложений**, которым нужна **минимальная задержка** прихода сообщений backplane внесет дополнительную задержку (latency растет).

В остальных случаях, BackPlane использовать можно, и он дает возможность использовать кластер веб серверов с SignalR.

**Ссылки**

* <http://www.asp.net/signalr/overview/performance/scaleout-in-signalr>
* <http://www.asp.net/signalr/overview/performance/scaleout-with-sql-server>
* <http://www.asp.net/signalr/overview/performance/scaleout-with-redis>
* <https://www.nuget.org/packages/Microsoft.AspNet.SignalR.Redis>
* <https://www.nuget.org/packages/Microsoft.AspNet.SignalR.SqlServer>
* <https://github.com/aspnet/SignalR-SqlServer>