Tarantool, NewSql & .net.

Анатолий Попов (evote.com)

Тезисы

- Что такое NewSql? Куда делся NoSql?
- Что такое Tarantool?
- Как использовать Tarantool из .net?
- Производительность progaudi.tarantool

Обо мне

- Работаю с .net с 2006 года
- Активно в OSS с 2016 года

Тот, кто не помнит прошлого, обречён на его повторение.

Тот, кто не помнит прошлого, обречён на его повторение.

Джордж Сантаяна, Жизнь разума, 1905

RDBMS

- General purpose database
- Usually SQL
- Developed since 1990s or so

NoSql

- Strozzi NoSQL open-source relational database – 1999
- Open source distributed, non relational databases – 2009
- Types:
  - Column
  - Document
  - KV
  - Graph
  - etc

Цели создания

- Простота: дизайна и администрирования
- Высокая пропускная способность
- Более эффективное использование памяти
- Горизонтальное масштабирование

Shiny new code:

- RDBMS are 25 year old legacy code lines that should be retired in favor of a collection of “from scratch” specialized engines. The DBMS vendors (and the research community) should start with a clean sheet of paper and design systems for tomorrow’s requirements, not continue to push code lines and architectures designed for yesterday’s needs

“The End of an Architectural Era” Michael Stonebraker et al.

Результат


OrientDB
DynamoDB
CouchDB
Redis
Solr
MarkLogic
RavenDB
MongoDB
Cassandra
xDB
NoSQL
Voldemort
Riak
eXist-db
Elasticsearch
Berkeley
CouchBase
Hadoop
ObjectDB
Neo4J

Недостатки

- Eventual consistency
- Ad-hoc query, data export/import, reporting
- Шардинг всё ещё сложный
- MySQL is fast enough for 90% websites

NewSQL

- Matthew Aslett in a 2011
- Relations and SQL
- ACID
- Бонусы NoSQL

NewSQL: код около данных

- VoltDB: Java
- Sql Server: .net & sql native
- Tarantool: lua

Sql Server

- Columnstore - 2012
- Hekaton (In-Memory OLTP) - 2014
- Cluster: up to 9 nodes

Tarantool

- memtx – in-memory store
  - TREE
  - HASH
  - RTREE
  - BITSET
- vinyl – write-mostly store (LSM + BTREE)
  - TREE

Tarantool

- SQL – 2.0+
- ACID
- vshard

.net и tarantool

- https://github.com/progaudi/progaudi.tarantool
  4 month ago, 1.0.0 is on the way

- https://github.com/donmikel/tarantool-net
  2 years ago

- https://github.com/asukhodko/dotnet-tarantool-client
  1 year ago

progaudi.tarantool

- Поддерживает .netstandard2.0
- Поддерживает Windows, Linux и Mac OSX
- Поддерживает почти весь протокол
- Устанавливается через nuget

Фичи

- Полностью асинхронный
- Встроенное мультиплексирование
- Keep-alive
- Автоматическое обновление схемы [1.0+]

Интерфейс: было

```
var index = client.GetSchema()["a"]["b"];

await index.Insert((2, "Music", 0.0));
await index.Insert((3, "Music"));
await index.Insert((4, "Music", false, "4th"));
```

Интерфейс: было

```
async Task<IReadOnlyList<AnswerPublishRequest>> GetById(QuestionId questionId)
{
    var index = client.GetSchema()["a"]["b"];

    var answerResponse = await index
        .Select<(bool, QuestionId), AnswerPublishRequest>(
        (false, questionId),
        new SelectOptions
        {
            Iterator = Iterator.Req
        }
    );

    return answerResponse.Data;
}
```

Интерфейс: было

```
async Task<IReadOnlyList<AnswerPublishRequest>> GetById(QuestionId questionId)
{
    var index = client.GetSchema()["a"]["b"];

    var answerResponse = await index
        .Select<(bool, QuestionId), AnswerPublishRequest>(
        (false, questionId),
        new SelectOptions
        {
            Iterator = Iterator.Req
        }
    );

    return answerResponse.Data;
}
```

Интерфейс: было

```
async Task<IReadOnlyList<AnswerPublishRequest>> GetById(QuestionId questionId)
{
    var index = client.GetSchema()["a"]["b"];

    var answerResponse = await index
        .Select<(bool, QuestionId), AnswerPublishRequest>(
        (false, questionId),
        new SelectOptions
        {
            Iterator = Iterator.Req
        }
    );

    return answerResponse.Data;
}
```

Интерфейс: было

```
async Task<IReadOnlyList<AnswerPublishRequest>> GetById(QuestionId questionId)
{
    var index = client.GetSchema()["a"]["b"];

    var answerResponse = await index
        .Select<(bool, QuestionId), AnswerPublishRequest>(
        (false, questionId),
        new SelectOptions
        {
            Iterator = Iterator.Req
        }
    );

    return answerResponse.Data;
}
```

Интерфейс: было

```
async Task<IReadOnlyList<AnswerPublishRequest>> GetById(QuestionId questionId)
{
    var index = client.GetSchema()["a"]["b"];

    var answerResponse = await index
        .Select<(bool, QuestionId), AnswerPublishRequest>(
        (false, questionId),
        new SelectOptions
        {
            Iterator = Iterator.Req
        }
    );

    return answerResponse.Data;
}
```

Интерфейс: было

```
async Task<IReadOnlyList<AnswerPublishRequest>> GetById(QuestionId questionId)
{
    var index = client.GetSchema()["a"]["b"];

    var answerResponse = await index
        .Select<(bool, QuestionId), AnswerPublishRequest>(
        (false, questionId),
        new SelectOptions
        {
            Iterator = Iterator.Req
        }
    );

    return answerResponse.Data;
}
```

Почему?

- Люди пойдут и почитают доку к Tarantool
- Не надо писать доку
- Полная мимика Iproto
- Гибкость

Интерфейс: было

```
async Task<IReadOnlyList<AnswerPublishRequest>> GetById(QuestionId questionId)
{
    var index = client.GetSchema()["a"]["b"];

    var answerResponse = await index
        .Select<(bool, QuestionId), AnswerPublishRequest>(
        (false, questionId),
        new SelectOptions
        {
            Iterator = Iterator.Req
        }
    );

    return answerResponse.Data;
}
```

Интерфейс: стало

```
Task<IReadOnlyList<AnswerPublishRequest>> GetById(QuestionId questionId)
{
    var index = client.GetSchema().GetSpace<AnswerPublishRequest>["a"]["b"];

    return index.Select((false, questionId), Iterator.Req));
}
```

Интерфейс: mini-ORM

```
[MsgPackArray]
public class SpaceMeta : IEquatable<SpaceMeta>
{
	[MsgPackArrayElement(0)]
	public uint Id { get; set; }

	[MsgPackArrayElement(1)]
	public int OwnerId { get; set; }

	[MsgPackArrayElement(2)]
	public string Name { get; set; }

	[MsgPackArrayElement(3)]
	public StorageEngine Engine { get; set; }

	[MsgPackArrayElement(4)]
	public uint FieldCount { get; set; }

	[MsgPackArrayElement(5)]
	public SpaceOptions Options { get; set; }

	[MsgPackArrayElement(6)]
	public SpaceField[] Fields { get; set; }
```

Интерфейс: MsgPackToken

```
public class MsgPackToken
{
    public static explicit operator MsgPackToken(uint value)
    {
        return new MsgPackToken(value);
    }

    public static explicit operator uint(MsgPackToken token)
    {
        return token.CastTokenToValue<uint>();
    }
```

Tarantool: сервер приложений

- Мы используем только tarantool/queue
- Также у нас есть своя логика на lua

tarantool/queue

- Сейчас:

```
await this.box.Call<((string, string), QueueOptions), MsgPackToken>(
    "queue.queue.tube.queue_name:put",
    ((token.Token, payload), QueueOptions.WithDelay(TimeSpan.Zero)));
```

- Хочется:

```
var queue = this.box.GetQueue<(string, string)>("queue_name");
await queue.Put((token.Token, payload), TimeSpan.Zero);
```

- Работы начну после 1.0 релиза

А что со скоростью?

- Бенчмарк: 1M вставок в пустой temporary space

- Go: ≈215k RPS
  Insert took 4.651657s

- .net: ≈27k RPS
  30-45 sec per 1M inserts

А что мы тестируем?

```
var lst = new Task[1000000];
for (int i = 0; i < lst.Length; i++)
{
      lst[i] = space.Insert((i, new[] { i, i }, i));
}

Task.WaitAll(lst);
```

Уменьшим размер пачки

```
var lst = new Task[1000]; // 30 секунд, 30к RPS, стабильно
for (var i = 0; i < 1_000_000; i++)
{
    lst[i % 1000] = space.Insert((i, (i, i), i));

    if (i % 1000 == 999)
    {
            Task.WaitAll(lst);
    }
}
```

MsgPack.Light медленный!

- Neuecc/Messagepack быстрее в 2-4 раза

- Давайте заменим

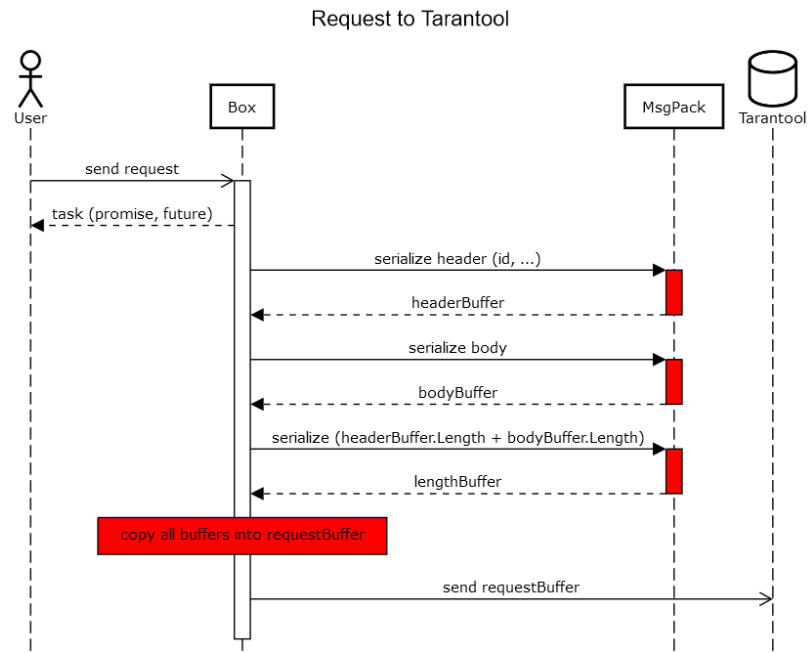

ПОТРАТИЛ ТРИ ДНЯ
ЗАМЕНИЛ СЕРИАЛИЗАТОР НА БОЛЕЕ БЫСТРЫЙ
ВЕСЬ КОД СТАЛ РАБОТАТЬ МЕДЛЕННЕЕ
ВЕСЬ КОД СТАЛ РАБОТАТЬ МЕДЛЕННЕЕ
imgflip.com

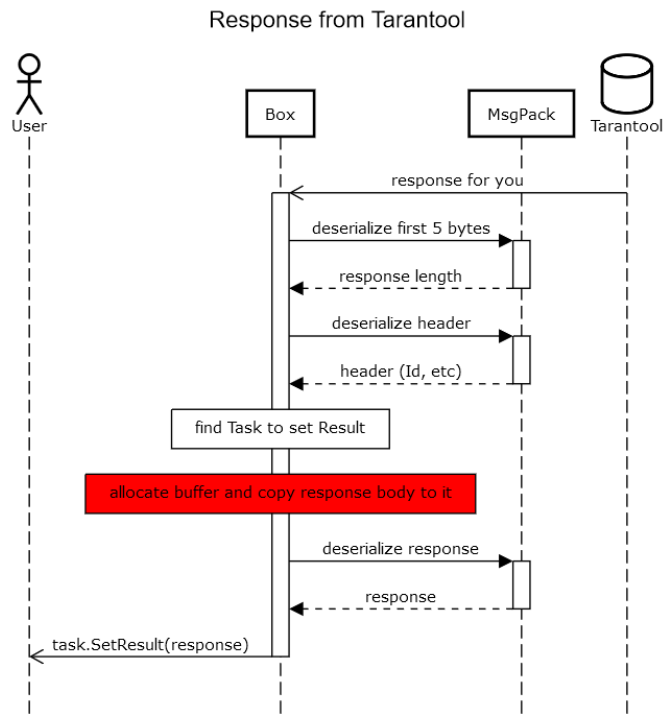
20k RPS, 50 sec

- Откатываем

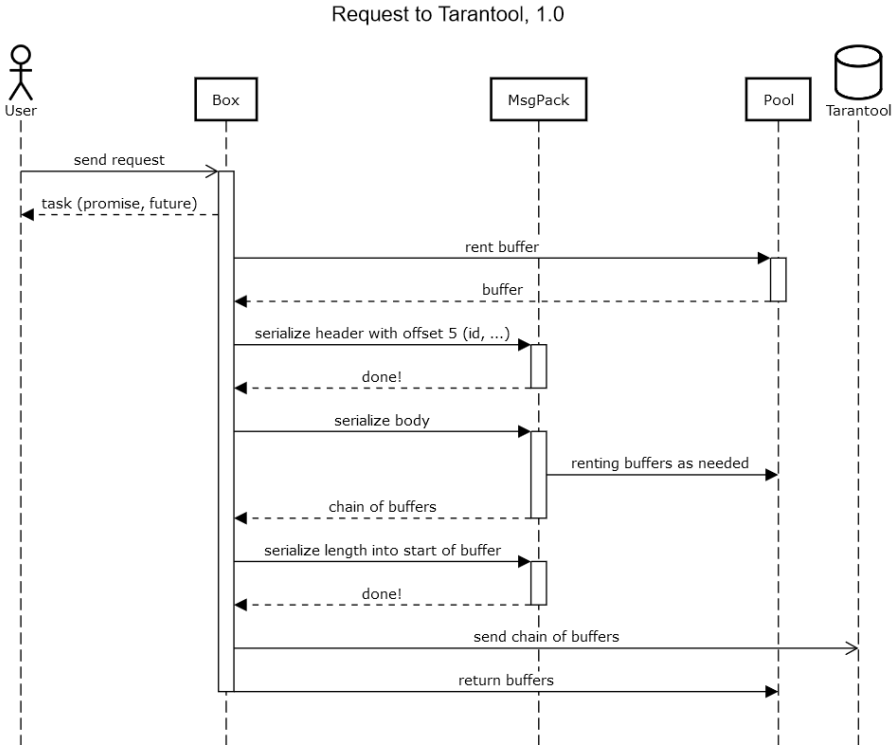
- Смотрим на схему работы коннектора

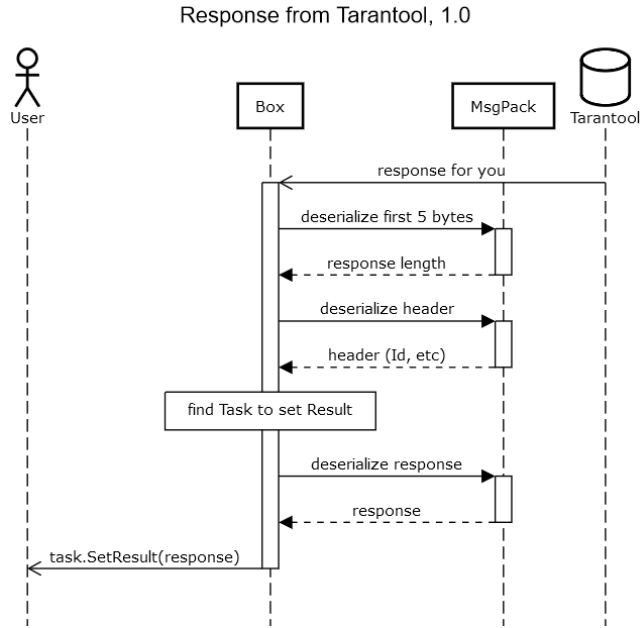
Запрос в Тарантул сейчас





Запрос в Тарантул в 1.0





Результаты

- Renting buffer as needed – не готово
- 60k RPS, 16 sec
- Если вставлять сразу 1M, то 18 sec, 55K RPS

Выводы

- Коннектор кроссплатформенный
- Активно развивается
- Работы над производительностью ведутся

Разработка под Windows

- VM
- Windows subsystem for Linux
- Docker

Windows subsystem for Linux

- Ubuntu [default]
- OpenSuse
- Kali Linux

Docker-образы

- Официальный: https://hub.docker.com/r/tarantool/tarantool/

- Неофициальный: https://hub.docker.com/r/progaudi/tarantool/

Официальный образ

- Стабилен
- При обновлении билда могут не обновить тег:
  - До: 1.9 -> 1.9.0
  - После: 1.9 -> 1.9.123
- Редко обновляется

Неофициальный образ

- «Нестабилен»
- Билд на каждый коммит
- Дополнительные модули

Зачем билд на каждый коммит?

1.9.1-36-gf41aac61a

- 1.9 – мажор
- 1.9.1 – минор
- 1.9.1-36-gf41aac61a - билд

Дополнительные модули

- lua-utf8 – не нужен с 1.10+
- spacer для миграции схемы
- Планируется добавление других

Lua и sql

- Lua – специфический:
  - box.NULL вместо nil
  - массивы с 1

- Sql – 2.0

IDE?

- VS Code + print for debug

- IDEA: https://tarantool.io/en/doc/1.9/book/app_server/using_ide.html

Мониторинг и логирование

- box.cfg.log_format = 'json'

```
local log = require('log')
log.info{space=space.name, status='created'}
```

- Fluentd docker logging driver

- tarantool/prometheus

Репликация

- Асинхронная
- Master-master
- Не все модули совместимы (queue!)

Выводы

- tarantool – СУБД общего назначения и сервер приложений
- Разработка может быть непривычна
- Эксплуатация довольна проста

Вопросы

Анатолий Попов
evote.com, server team lead
me@aensidhe.ru
https://github.com/aensidhe

Список литературы

- http://www.christof-strauch.de/nosqldbs.pdf

- https://habr.com/company/oleg-bunin/blog/413557/