Использование Tarantool в .NET-проектах

Анатолий Попов

Director of Engineering, Net2Phone

net2phone

Тезисы

- Что такое NewSql? Куда делся NoSql?
- 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.

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

net2phone

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

Sql Server

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

RDBMS: storage model

Id	Name	Age
1	Вася	20
2	Петя	40
3	Коля	30

Tarantool: storage model

Id	Name	Age		
1	Вася	20	123,09	Молоко
2	Петя	40	МГУ	
3	Коля	30	Женат	

net2phone

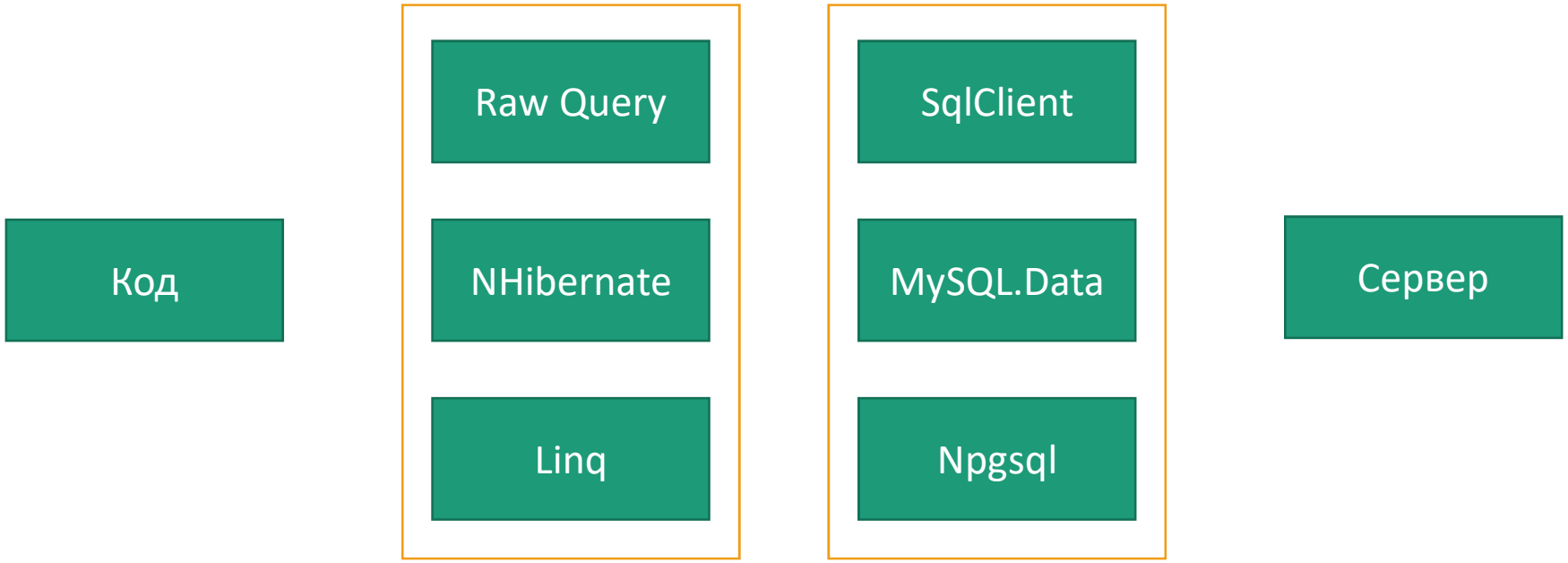
Tarantool: storage model

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

Tarantool: features

- SQL – 2.1 beta
- ACID
- vshard

Что такое коннектор/драйвер?



Что такое коннектор/драйвер?

.net и tarantool

- https://github.com/progaudi/progaudi.tarantool
  2 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: вставка

```
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: выборка

```
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: выборка

```
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: выборка

```
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: выборка

```
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: выборка

```
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: выборка

```
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?

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

Интерфейс: выборка

```
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: будет

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

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

```
public interface IIndex
{
      Task<DataResponse<TTuple[]>> Select<TKey, TTuple>(TKey key);

      Task<DataResponse<TTuple[]>> Insert<TTuple>(TTuple tuple);
}
```

```
public interface IIndex<TTuple>
{
      Task<DataResponse<TTuple[]>> Select<TKey>(TKey key);

      Task<DataResponse<TTuple[]>> Insert(TTuple tuple);
}
```

Интерфейс: mini-ORM

```
[MsgPackArray]
public class ImageInfo : IImageInfo
{
    [MsgPackArrayElement(0)]
    public int Width { get; set; }

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

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

    [MsgPackArrayElement(4)]
    public string Credits { get; set; }

    public string NotSerializedProperty { get; set; }
}
```

net2phone

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

- tarantool/queue
- 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

Подводные камни

.net core 1.1 DnsEndPoint, linux

```
[Fact]
public void DnsTest()
{
    var socket = new Socket(SocketType.Stream, ProtocolType.Tcp)
    {
        NoDelay = true
    };
    socket.Connect(new DnsEndPoint("www.google.com", 443));
}
```

- Exception: .NET Core not supporting DnsEndPoint in Socket.Connect.

.net core 1.1 DnsEndPoint, linux

```
public async Task DnsTest() {
   var socket = new Socket(SocketType.Stream, ProtocolType.Tcp);
   var resolved = await Dns.GetHostAddressesAsync("…");
   foreach (var addr in resolved) {
      try {
         await socket.ConnectAsync(addr, 443);
         return;
      }
      catch {} continue;
   }
   throw new Exception("Can't connect");
}
```

net2phone

StackExchange.Redis: фичи

- Асинхронность
- Конвейерная обработка
- Мультиплексирование

progaudi.tarantool: фичи

- Асинхронность
- Конвейерная обработка
- Мультиплексирование

Асинхронность

- TaskCompletionSource<T>

- Custom awaiter

Мультиплексирование

- Несколько запросов могут использовать одно соединение
- Sql Server + MARS, HTTP/1.1, HTTP/2, HTTP/3, Redis,…
- RequestId + ConcurrentDictionary<,>

Конвейерная обработка

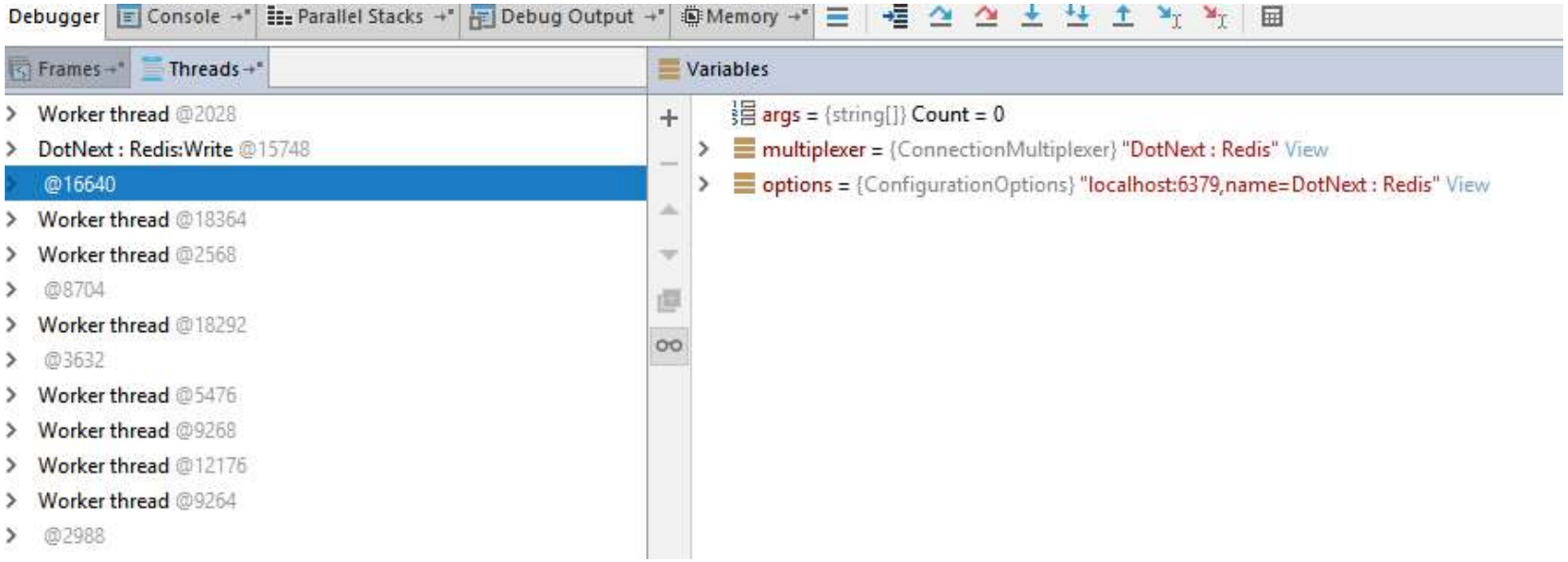
- Чтения и записи из сети разделены
- Запросы отправляются по мере запросов от юзера
- Ответы читаются по мере прихода от сервера

StackExchange.Redis

```
var options = new ConfigurationOptions();
options.EndPoints.Add("localhost", 6379);
options.ClientName = "DotNext : Redis";
var multiplexer = ConnectionMulteplexer.Connect(options);
```

StackExchange.Redis, 1.2.6

- Отдельный поток на запись



Отправка сообщений, простой вариант

```
// получение таски
public Task<MemoryStream> GetTaskNaive(RequestId requestId) {
    var tcs = _concurrentCache[requestId] = CreateTcs();
    return tcs.Task;
}

// запись
lock (_physicalConnection) {
   _logWriter?.WriteLine($"Begin sending request");
   _physicalConnection.Write(header);
   _physicalConnection.Write(body);
}
```

Отправка сообщений, вариант посложнее

```
try {
   _physicalConnectionLock.EnterWriteLock();
   _logWriter?.WriteLine($"Begin sending request");
   _physicalConnection.Write(header);
   _physicalConnection.Write(body);
}
finally {
   _physicalConnectionLock.ExitWriteLock();
}
```

Отправка сообщений, вариант посложнее

```
try {
      _pendingRequestsLock.EnterWriteLock();
      var tcs = _cache[requestId] = CreateTcs();
      return tcs.Task;
}
finally {
      _pendingRequestsLock.ExitWriteLock();
}
```

Отправка сообщений, правильно

```
public void Write(ArraySegment<byte> header, ArraySegment<byte> body)
{
	if (_disposed) throw new ODE(nameof(ResponseReader));

	_clientOptions?.LogWriter?.WriteLine($"Enqueuing request.");
	bool shouldSignal;
	lock (_lock) {
		_buffer.Enqueue(Tuple.Create(header, body));
		shouldSignal = _buffer.Count == 1;
	}

	if (shouldSignal) _newRequestsAvailable.Set();
}
```

Отправка сообщений, правильно

```
private void WriteFunction() {
   var handles = new[]{ _exitEvent, _newRequestsAvailable};

   while (true) {
      switch (WaitHandle.WaitAny(handles)) {
         case 0: return;
         case 1: WriteRequests(200); break;
         default:
         throw new ArgumentOutOfRangeException();
      }
   }
}
```

Отправка сообщений, правильно

```
Tuple<ArraySegment<byte>, ArraySegment<byte>> request;
var count = 0;
while ((request = GetRequest()) != null) {
      WriteBuffer(request.Item1);
      WriteBuffer(request.Item2);

      count++;
      if (limit > 0 && count > limit) break;
}

lock (_lock)
      if (_buffer.Count == 0)
            _newRequestsAvailable.Reset();

_physicalConnection.Flush();
```

Keep-alive

- Options.ConnectionOptions.PingCheckTimeout
- Служебный пакет Ping
- Переподключение почти прозрачно для юзера
- Перепосылку запроса мы пока не делаем

Чтение

- Читаем пачку байт
- Парсим всю пачку перед тем, как читать следующую
- Если остался хвост, переносим его в начало пачки
- Первые пять байт пакета – это длина, очень удобно

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

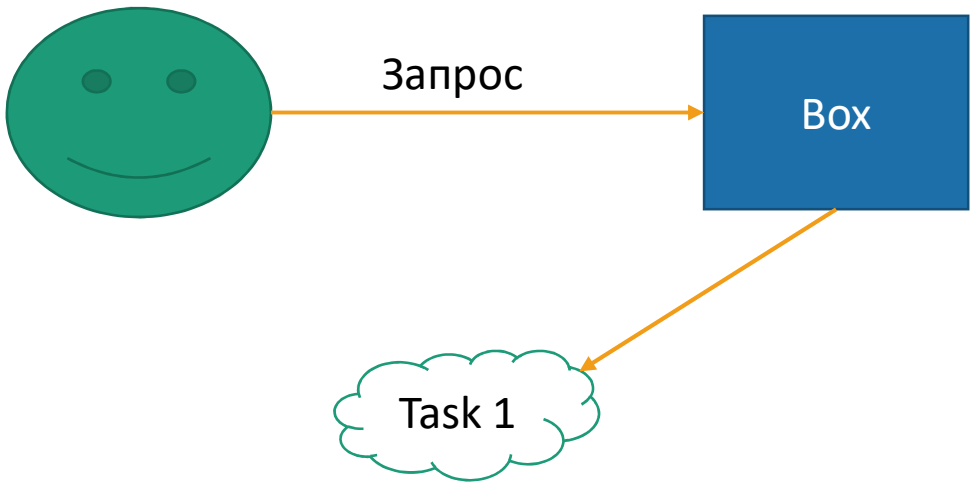
А что мы тестируем?

```
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

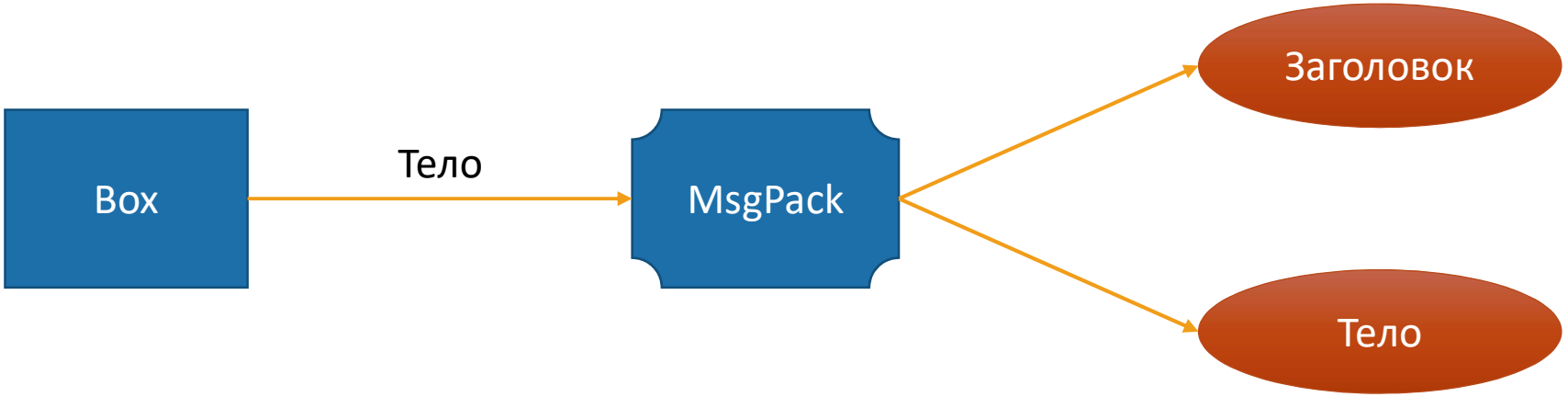
```
// 30 секунд, 30к RPS, стабильно
var lst = new Task[1000];
for (var i = 0; i < 1_000_000; i++) {
    lst[i % 1000] = space.Insert((i, (i, i), i));
    if (i % 1000 == 999)
        Task.WaitAll(lst);
}
```

Запрос в Тарантул

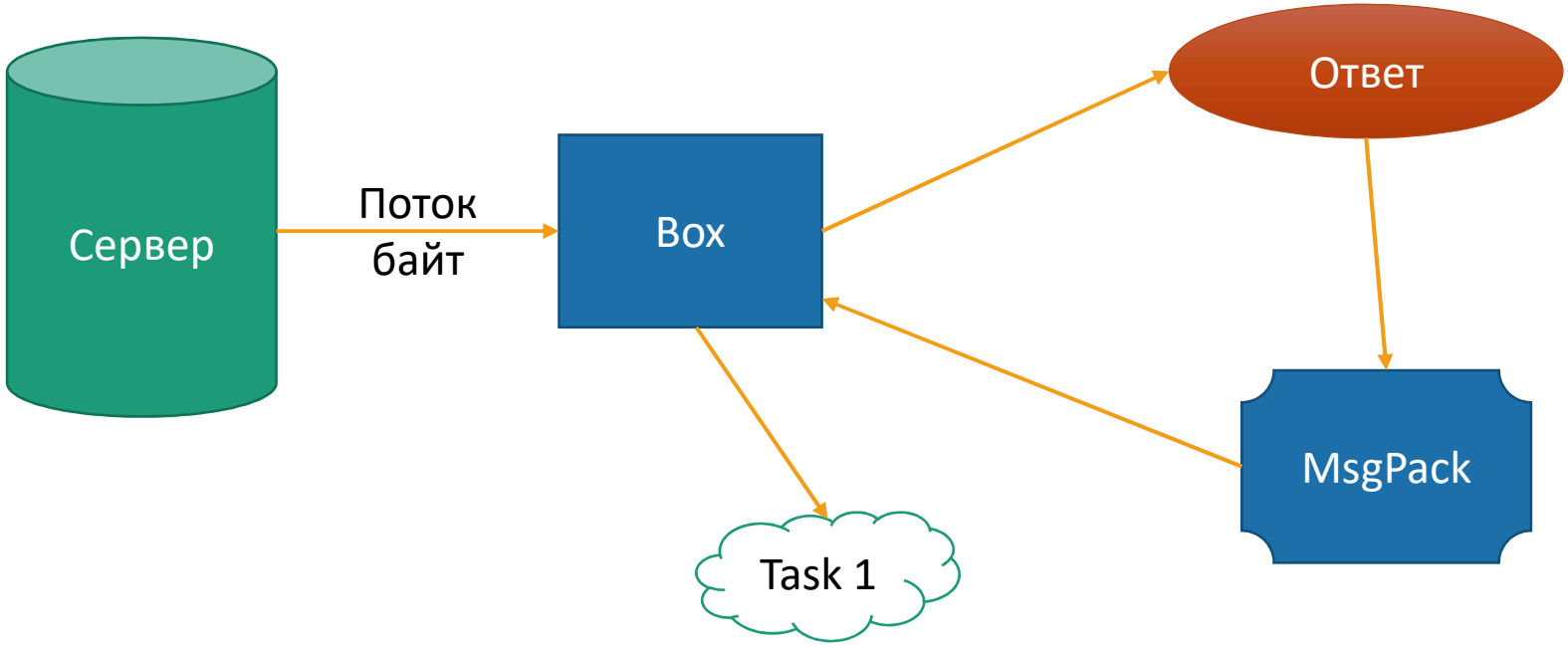


Запрос в Тарантул

Запрос в Тарантул

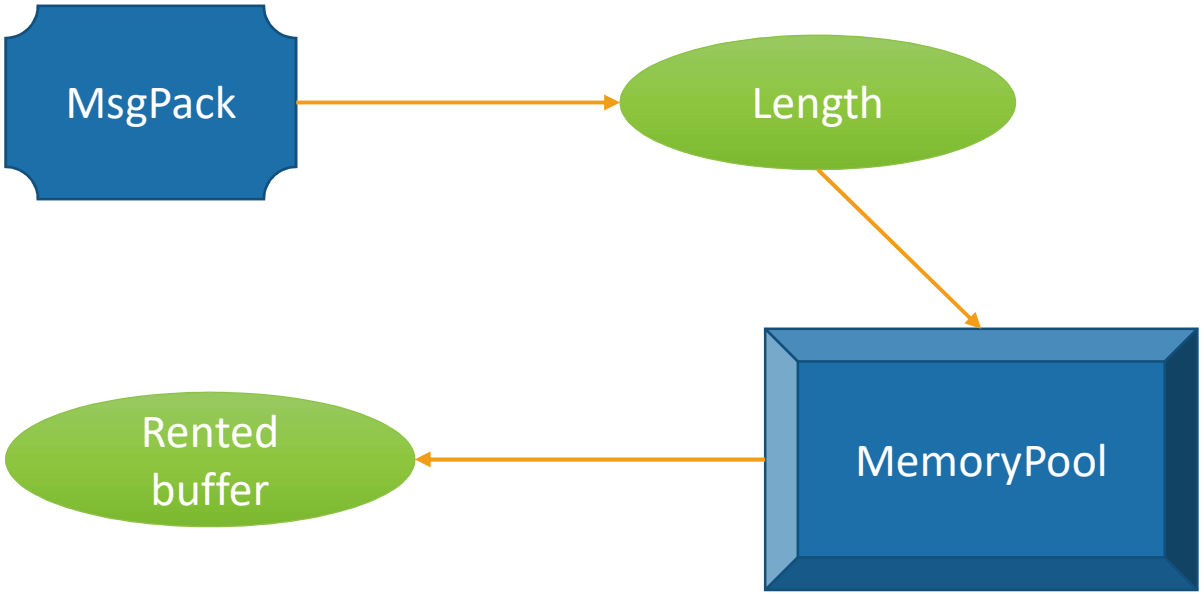


Ответ от Тарантула



Сериализация: было

Сериализация: стало



Результаты

- 60k RPS, 16 sec
- Если вставлять сразу 1M, то 18 sec, 55K RPS
- Не до конца проверенные: ~12 sec, 80k RPS

Новые абстракции

- Span<T> - ref stackonly struct
- Memory<T> - struct
- IMemoryOwner<T>

Span<T>: цена, C#

```
private const uint _length = 100;
private uint _baseInt = 99000;
private readonly byte[] _buffer = ArrayPool<byte>.Rent(short.MaxValue);

[Benchmark]
public int Span() {
    var buffer = _buffer.AsSpan();
    var i = 0;
    for (; i < _length; i++) {
        _baseInt -= 1000u;
        MsgPackSpecSpan.WriteUInt32(buffer.Slice(5 * i), _baseInt);
    }

    return i;
}
```

Span<T>: цена, C

```
char buf[65535];

extern uint32_t serializeInts(uint32_t size) {
   char *w = buf;
   uint32_t base = 99000, idx = 0;
   for (; idx < size; ++idx) {
      base -= 1000;
      w = mp_encode_uint(w, base);
   }

   return idx;
}
```

net2phone

Span<T>: цена, C++

```
uint32_t base = 99000, idx = 0;
sbuffer buffer;
packer<sbuffer> pk(&buffer);

for (; idx < size; ++idx) {
     base -= 1000;
     pk.pack(base);
}

return idx;
```

Span<T>: цена, C# pointers

```
public unsafe int Pointers()
{
    fixed (byte* pointer = &_buffer[0]) {
        var i = 0;
        for (; i < _length; i++) {
            _baseInt -= 1000;
            MsgPackSpecPointer.WriteUInt32(pointer, 5 * i, _baseInt);
        }
        return i;
    }
}
```

Span<T>: результаты

Method	Mean	Error	StdDev	Q3	Scaled	ScaledSD	Allocated
Span	378.8 ns	6.778 ns	6.340 ns	385.2 ns	2.01	0.04	0 B
SpanConst	198.7 ns	1.255 ns	1.174 ns	199.5 ns	1.06	0.02	0 B
Pointer	235.5 ns	4.583 ns	4.501 ns	237.6 ns	1.25	0.03	0 B
C	188.1 ns	2.714 ns	2.538 ns	190.3 ns	1.00	0.00	0 B
Cpp	189.5 ns	2.896 ns	2.709 ns	191.5 ns	1.01	0.02	0 B

Span<T>: цена, SpanConst

```
private const uint _length = 100;
private uint _baseInt = 99000;
private readonly byte[] _buffer = ArrayPool<byte>.Rent(short.MaxValue);

[Benchmark]
public int Span() {
    var buffer = _buffer.AsSpan();
    var i = 0;
    for (; i < _length; i++) {
        _baseInt -= 1000u;
        MsgPackSpecSpan.WriteUInt32(buffer.Slice(5 * i), _length);
    }

    return i;
}
```

net2phone

Span<T>: цена, C#

```
private const uint _length = 100;
private uint _baseInt = 99000;
private readonly byte[] _buffer = ArrayPool<byte>.Rent(short.MaxValue);

[Benchmark]
public int Span() {
    var buffer = _buffer.AsSpan();
    var i = 0;
    for (; i < _length; i++) {
        _baseInt -= 1000u;
        MsgPackSpecSpan.WriteUInt32(buffer.Slice(5 * i), _baseInt);
    }

    return i;
}
```

Span<T>: цена, SpanConst

```
private const uint _length = 100;
private uint _baseInt = 99000;
private readonly byte[] _buffer = ArrayPool<byte>.Rent(short.MaxValue);

[Benchmark]
public int Span() {
    var buffer = _buffer.AsSpan();
    var i = 0;
    for (; i < _length; i++) {
        _baseInt -= 1000u;
        MsgPackSpecSpan.WriteUInt32(buffer.Slice(5 * i), _length);
    }

    return i;
}
```

Span<T>: цена, WriteUInt32

```
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static int WriteUInt32(Span<byte> buffer, uint value)
{
      if (value <= DataCodes.FixPositiveMax)
            return WritePositiveFixInt(buffer, (byte) value);
      if (value <= byte.MaxValue)
            return WriteFixUInt8(buffer, (byte) value);
      if (value <= ushort.MaxValue)
            return WriteFixUInt16(buffer, (ushort) value);
      return WriteFixUInt32(buffer, value);
}
```

Span<T>: цена, WriteUInt32

```
[MethodImpl(MethodImplOptions.AggressiveInlining)]
public static int WriteUInt32(Span<byte> buffer, uint value)
{
      return WriteFixUInt8(buffer, (byte) value);
}
```

IMemoryOwner<T>

- var buffer = MemoryPool<T>.Shared.Rent(100)

- buffer.Length = ????

FixedMemoryOwner<T>

- Default:

```
return new Memory<T>(array);
```

- Fixed:

```
return new Memory<T>(array, 0, _size);
```

net2phone

https://www.nuget.org/packages/progaudi.buffers/

Bounds check

```
[Benchmark(Baseline = true)]
public int Classic() {
   Span<byte> span = _buffer;
   span[0] = 0;
   span[1] = 128;
   return 2;
}
```

```
[Benchmark]
public int EgorBogatov() {
   Span<byte> span = _buffer;
   span[1] = 128;
   span[0] = 0;
   return 2;
}
```

Bounds check: 2 elements

```
// * Summary *

BenchmarkDotNet=v0.11.1, OS=Windows 10.0.17134.228 (1803/April2018Update/Redstone4)
Intel Core i7-8550U CPU 1.80GHz (Max: 1.79GHz) (Kaby Lake R), 1 CPU, 8 logical and 4 physical cores
.NET Core SDK=2.1.401
  [Host]     : .NET Core 2.1.3 (CoreCLR 4.6.26725.06, CoreFX 4.6.26725.05), 64bit RyuJIT
  DefaultJob : .NET Core 2.1.3 (CoreCLR 4.6.26725.06, CoreFX 4.6.26725.05), 64bit RyuJIT


    Method |      Mean |     Error |    StdDev | Scaled | ScaledSD | Allocated |
---------- |----------:|----------:|----------:|-------:|---------:|----------:|
   Classic | 1.0056 ns | 0.0844 ns | 0.2338 ns |   1.00 |     0.00 |       0 B |
 CoolHacker | 0.7648 ns | 0.0550 ns | 0.0514 ns |   0.79 |     0.16 |       0 B |
```

Bounds check: asm

Classic .NET Core 2.1.3 (CoreCLR 4.6.26725.06, CoreFX 4.6.26725.05), 64bit RyuJIT

```
00007ffd`65b01ca0 BoundsCheck.BoundsBenchmark.Classic()
IL_0000: ldarg.0
IL_0001: ldfld System.Byte[] BoundsCheck.BoundsBenchmark::_buffer
IL_0006: call System.Span`1 System.Span`1::op_Implicit(!0[])
IL_000b: stloc.0
00007ffd`65b01ca4 488b4108        mov     rax,qword ptr [rcx+8]
00007ffd`65b01ca8 4885c0          test    rax,rax
00007ffd`65b01cab 7506            jne     00007ffd`65b01cb3
00007ffd`65b01cad 33d2            xor     edx,edx
00007ffd`65b01caf 33c9            xor     ecx,ecx
00007ffd`65b01cb1 eb07            jmp     00007ffd`65b01cba
00007ffd`65b01cb3 488d5010        lea     rdx,[rax+10h]
00007ffd`65b01cb7 8b4808          mov     ecx,dword ptr [rax+8]
IL_000c: ldloca.s V_0
IL_000e: ldc.i4.0
IL_000f: call !0& System.Span`1::get_Item(System.Int32)
IL_0014: ldc.i4 204
IL_0019: stind.i1
00007ffd`65b01cba 83f900          cmp     ecx,0
00007ffd`65b01cbd 7616            jbe     00007ffd`65b01cd5
00007ffd`65b01cbf c602cc          mov     byte ptr [rdx],0CCh
IL_001a: ldloca.s V_0
IL_001c: ldc.i4.1
IL_001d: call !0& System.Span`1::get_Item(System.Int32)
IL_0022: ldc.i4.s 120
IL_0024: stind.i1
00007ffd`65b01cc2 83f901          cmp     ecx,1
00007ffd`65b01cc5 760e            jbe     00007ffd`65b01cd5
00007ffd`65b01cc7 c6420178        mov     byte ptr [rdx+1],78h
IL_0025: ldc.i4.2
IL_0026: ret
00007ffd`65b01ccb b802000000      mov     eax,2
```

CoolHacker .NET Core 2.1.3 (CoreCLR 4.6.26725.06, CoreFX 4.6.26725.05), 64bit RyuJIT

```
00007ffd`65ae1ca0 BoundsCheck.BoundsBenchmark.CoolHacker()
IL_0000: ldarg.0
IL_0001: ldfld System.Byte[] BoundsCheck.BoundsBenchmark::_buffer
IL_0006: call System.Span`1 System.Span`1::op_Implicit(!0[])
IL_000b: stloc.0
00007ffd`65ae1ca4 488b4108        mov     rax,qword ptr [rcx+8]
00007ffd`65ae1ca8 4885c0          test    rax,rax
00007ffd`65ae1cab 7506            jne     00007ffd`65ae1cb3
00007ffd`65ae1cad 33d2            xor     edx,edx
00007ffd`65ae1caf 33c9            xor     ecx,ecx
00007ffd`65ae1cb1 eb07            jmp     00007ffd`65ae1cba
00007ffd`65ae1cb3 488d5010        lea     rdx,[rax+10h]
00007ffd`65ae1cb7 8b4808          mov     ecx,dword ptr [rax+8]
IL_000c: ldloca.s V_0
IL_000e: ldc.i4.1
IL_000f: call !0& System.Span`1::get_Item(System.Int32)
IL_0014: ldc.i4.s 120
IL_0016: stind.i1
00007ffd`65ae1cba 83f901          cmp     ecx,1
00007ffd`65ae1cbd 7611            jbe     00007ffd`65ae1cd0
00007ffd`65ae1cbf c6420178        mov     byte ptr [rdx+1],78h
IL_0017: ldloca.s V_0
IL_0019: ldc.i4.0
IL_001a: call !0& System.Span`1::get_Item(System.Int32)
IL_001f: ldc.i4 204
IL_0024: stind.i1
00007ffd`65ae1cc3 c602cc          mov     byte ptr [rdx],0CCh
IL_0025: ldc.i4.2
IL_0026: ret
00007ffd`65ae1cc6 b802000000      mov     eax,2
```

net2phone

Выводы

- Производительность - фича
- Знайте ваш рантайм
- Проектируйте тесты правильно

Анатолий Попов
Net2Phone, Director of Engineering
me@aensidhe.ru
https://github.com/aensidhe

net2phone

Список литературы про NewSql

- http://www.christof-strauch.de/nosqldbs.pdf
- https://habr.com/company/oleg-bunin/blog/413557/