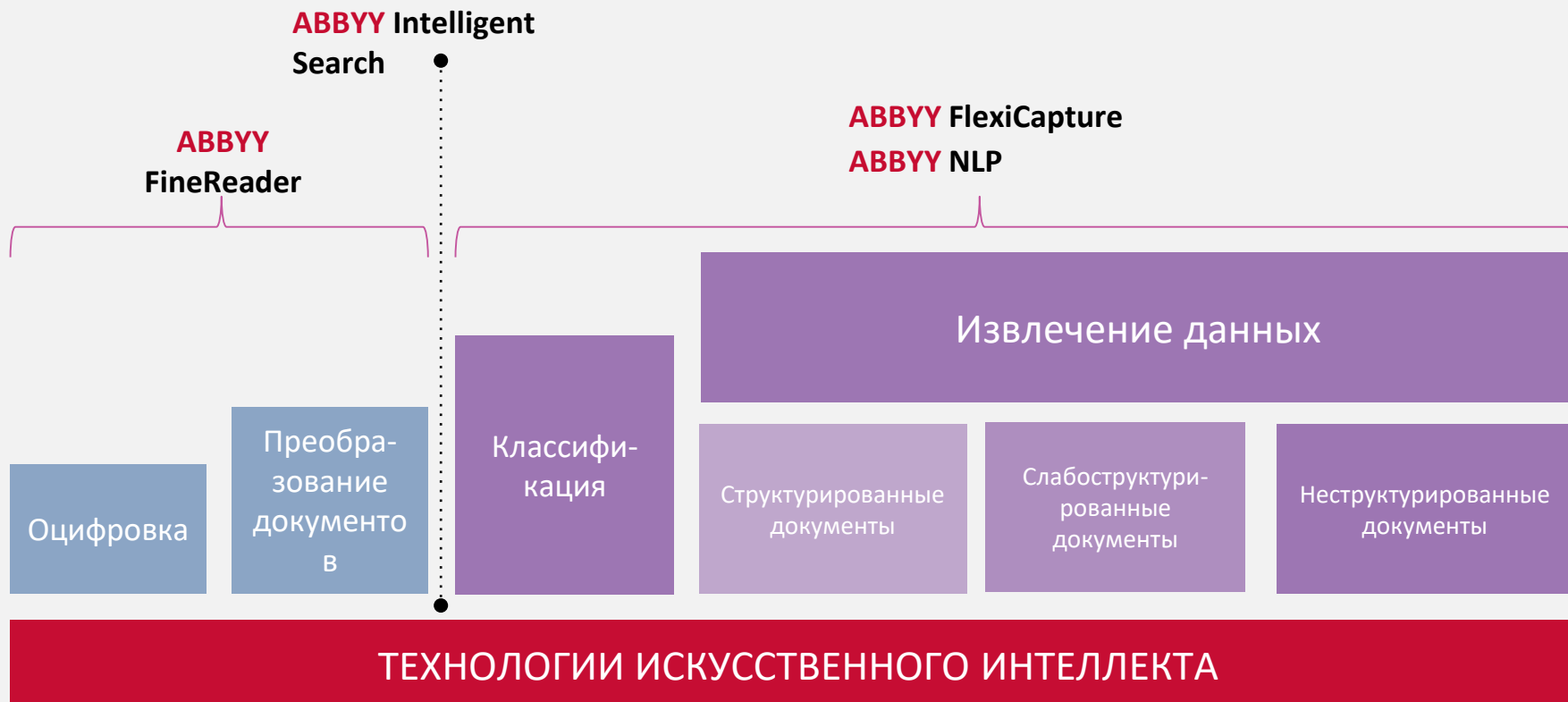


gRPC для .NET разработчика

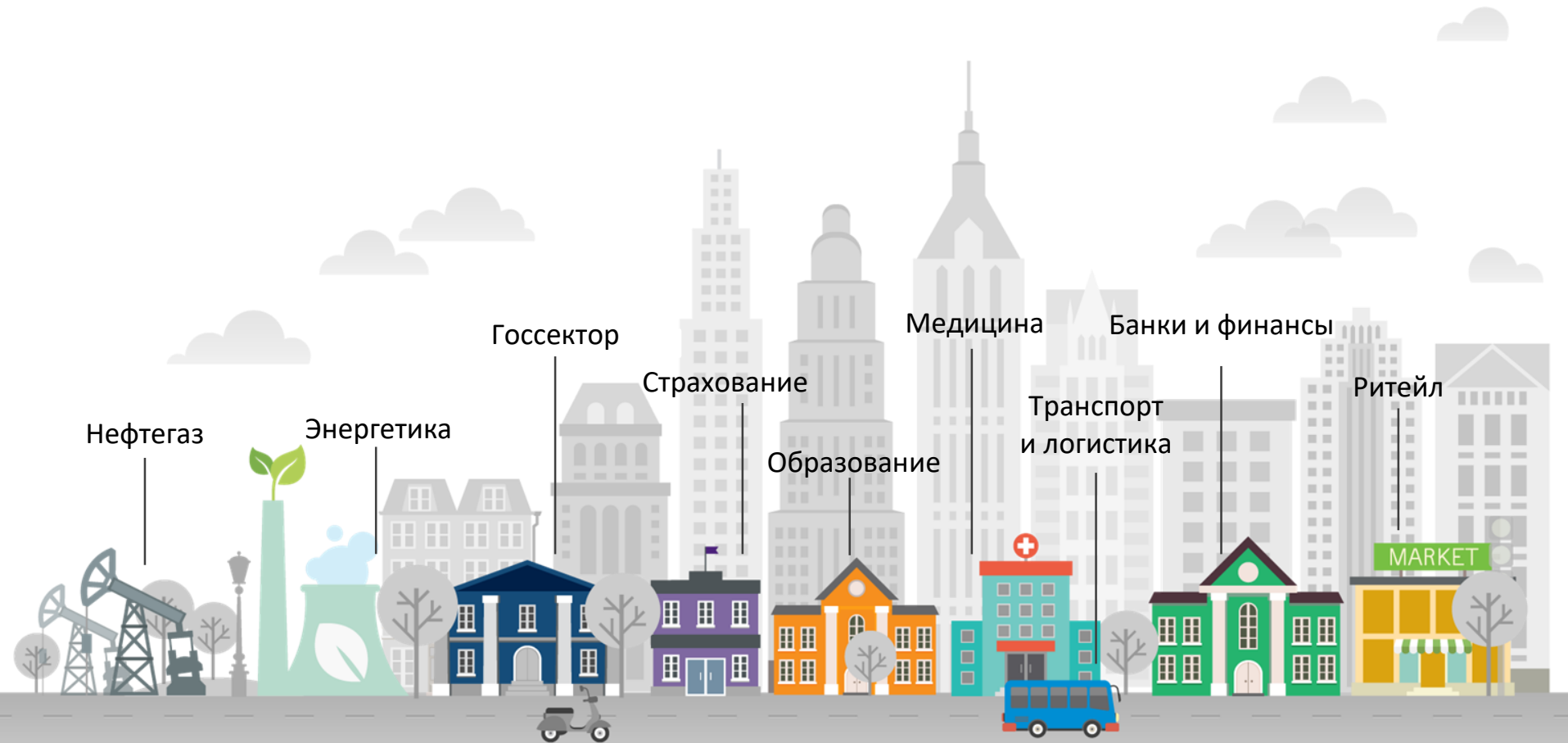
Обо мне

- Senior dotnet developer в АBBYY, Common Cloud
- Активист MskDotNet Community



Проекты ABBYY в разных отраслях

ABBYY®



Agenda

- Сравнение RPC и REST подходов
- Что такое gRPC
- Основные достоинства и недостатки
- gRPC в .NET
- Как внедрить gRPC
- И что нам это даст

RPC

REST

RPC

- ~ 1980-e

REST

RPC

- ~ 1980-е
- Вызов методов сервиса как локальных

REST

RPC

- ~ 1980-е
- Вызов методов сервиса как локальных
- Два компонента: сетевой протокол и технология сериализации

REST

RPC

- ~ 1980-е
- Вызов методов сервиса как локальных
- Два компонента: сетевой протокол и технология сериализации
- Не навязывает особых требований к архитектуре

REST

RPC

- ~ 1980-е
- Вызов методов сервиса как локальных
- Два компонента: сетевой протокол и технология сериализации
- Не навязывает особых требований к архитектуре
- Есть множество реализаций, часть привязаны к языку/платформе.

REST

RPC

- ~ 1980-е
- Вызов методов сервиса как локальных
- Два компонента: сетевой протокол и технология сериализации
- Не навязывает особых требований к архитектуре
- Есть множество реализаций, часть привязаны к языку/платформе.
- SOAP

REST

RPC

- ~ 1980-е
- Вызов методов сервиса как локальных
- Два компонента: сетевой протокол и технология сериализации
- Не навязывает особых требований к архитектуре
- Есть множество реализаций, часть привязаны к языку/платформе.
- SOAP
- WCF - .NET

REST

RPC

- ~ 1980-е
- Вызов методов сервиса как локальных
- Два компонента: сетевой протокол и технология сериализации
- Не навязывает особых требований к архитектуре
- Есть множество реализаций, часть привязаны к языку/платформе.
- SOAP
- WCF - .NET

REST

- ~ 2000 год

RPC

- ~ 1980-е
- Вызов методов сервиса как локальных
- Два компонента: сетевой протокол и технология сериализации
- Не навязывает особых требований к архитектуре
- Есть множество реализаций, часть привязаны к языку/платформе.
- SOAP
- WCF - .NET

REST

- ~ 2000 год
- Ресурсы в виде эндпоинтов

RPC

- ~ 1980-е
- Вызов методов сервиса как локальных
- Два компонента: сетевой протокол и технология сериализации
- Не навязывает особых требований к архитектуре
- Есть множество реализаций, часть привязаны к языку/платформе.
- SOAP
- WCF - .NET

REST

- ~ 2000 год
- Ресурсы в виде эндпоинтов
- HTTP методы для CRUD операций

RPC

- ~ 1980-е
- Вызов методов сервиса как локальных
- Два компонента: сетевой протокол и технология сериализации
- Не навязывает особых требований к архитектуре
- Есть множество реализаций, часть привязаны к языку/платформе.
- SOAP
- WCF - .NET

REST

- ~ 2000 год
- Ресурсы в виде эндпоинтов
- HTTP методы для CRUD операций
- Ряд требований к сервисам

RPC

- ~ 1980-е
- Вызов методов сервиса как локальных
- Два компонента: сетевой протокол и технология сериализации
- Не навязывает особых требований к архитектуре
- Есть множество реализаций, часть привязаны к языку/платформе.
- SOAP
- WCF - .NET

REST

- ~ 2000 год
- Ресурсы в виде эндпоинтов
- HTTP методы для CRUD операций
- Ряд требований к сервисам
- Достаточно предсказуем и унифицирован

RPC vs REST

Последние годы REST подход доминирует, но для микросервисных решений он не всегда подходит

Немного истории



Проект Stubby

Немного истории

Март, 2015



gRPC Remote Procedure Calls

Немного истории



Microsoft



Core OS

Немного истории



CLOUD NATIVE
COMPUTING FOUNDATION

gRPC

HTTP/2



protobuf
Protocol Buffers

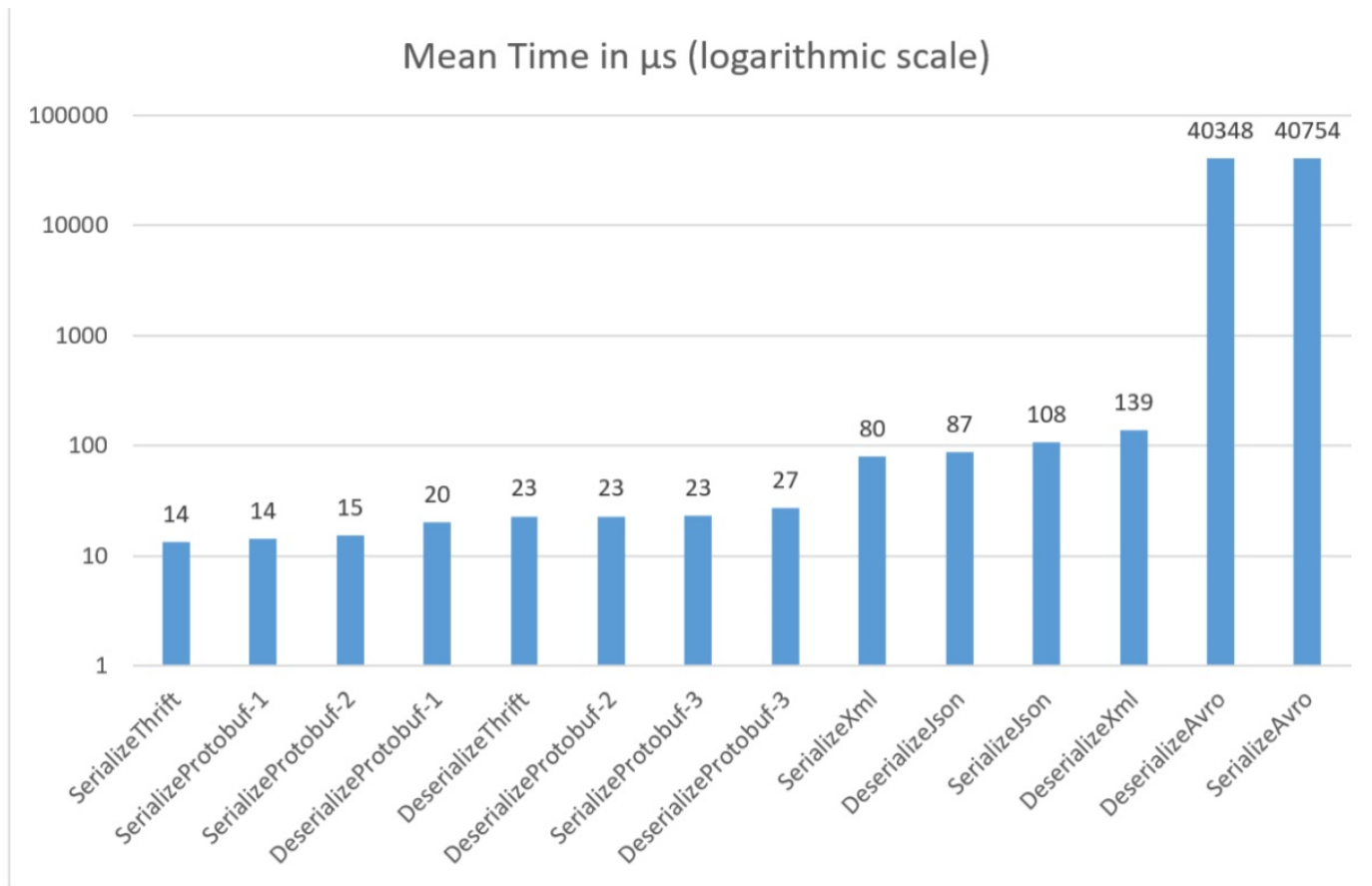
gRPC Плюсы

- Малый размер сообщений

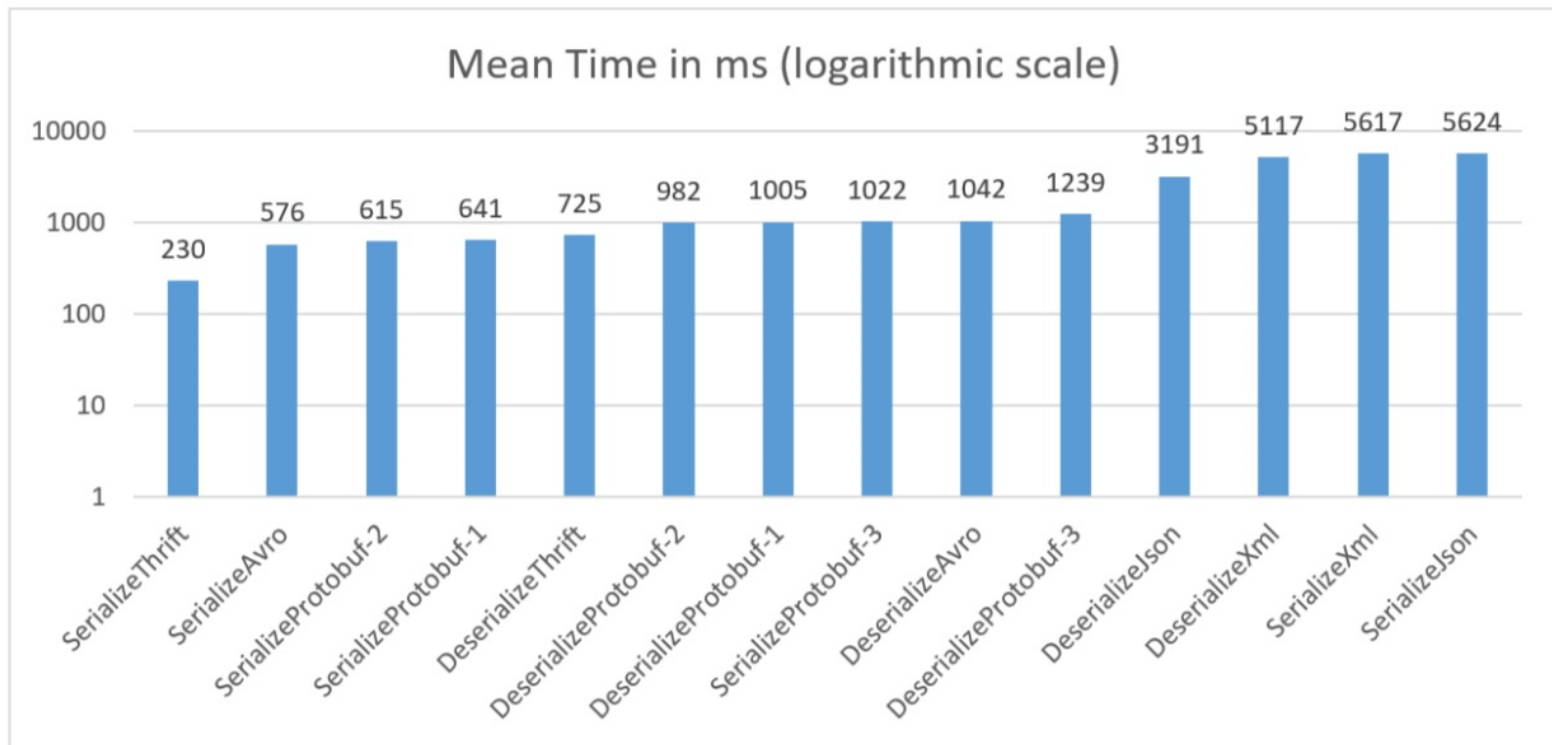
gRPC Плюсы

- Малый размер сообщений
- Эффективная сериализация/десериализация

Performance



Performance

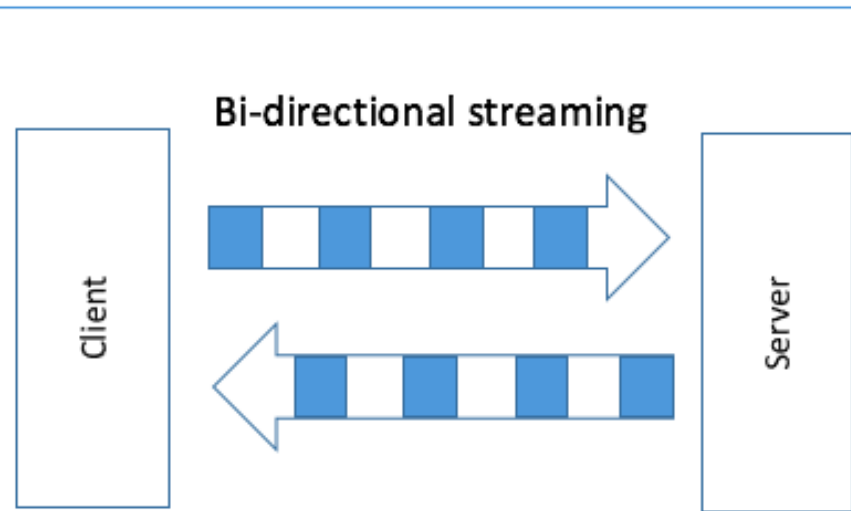
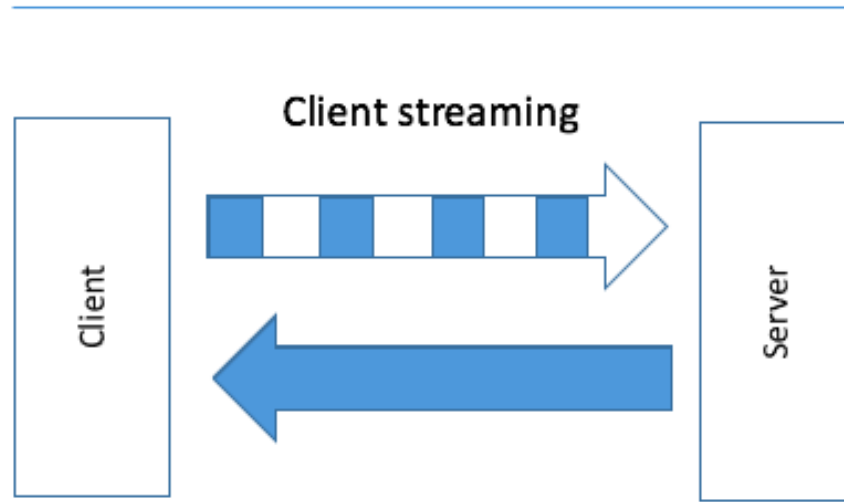
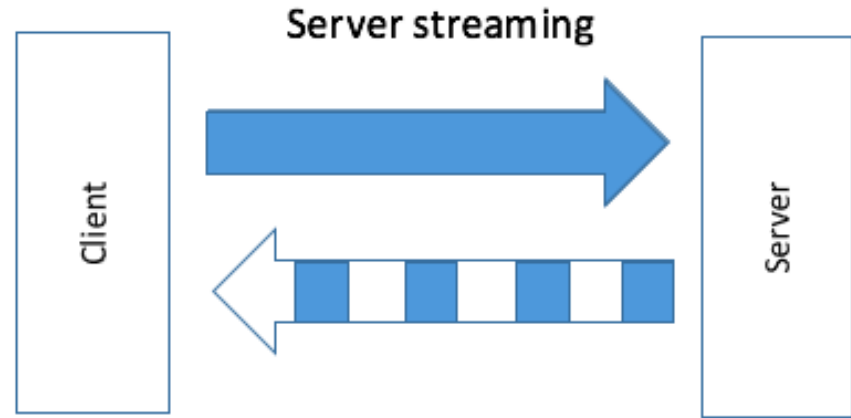
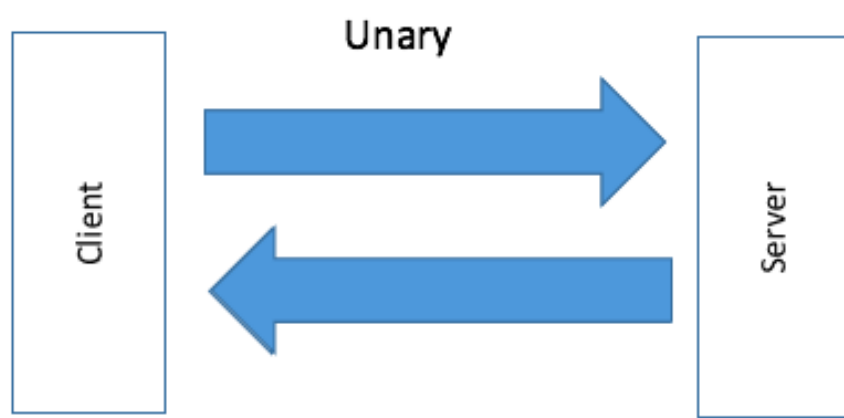


gRPC Плюсы

- Малый размер сообщений
- Эффективная сериализация/десериализация
- Мультиплексирование

gRPC Плюсы

- Малый размер сообщений
- Эффективная сериализация/десериализация
- Мультиплексирование
- Поддержка потоковой передачи данных



gRPC Плюсы

- Малый размер сообщений
- Эффективная сериализация/десериализация
- Мультиплексирование
- Поддержка потоковой передачи данных
- Contract first

gRPC Плюсы

- Малый размер сообщений
- Эффективная сериализация/десериализация
- Мультиплексирование
- Поддержка потоковой передачи данных
- Contract first
- Поддержка большого кол-ва языков

gRPC Минусы

- Нет поддержки со стороны браузеров(но есть официальная обертка)

gRPC Минусы

- Нет поддержки со стороны браузеров(но есть официальная обертка)
- Нельзя хостить в Azure App Services и IIS(пока)

gRPC Минусы

- Нет поддержки со стороны браузеров(но есть официальная обертка)
- Нельзя хостить в Azure App Services и IIS(пока)
- Плохо с тулингом

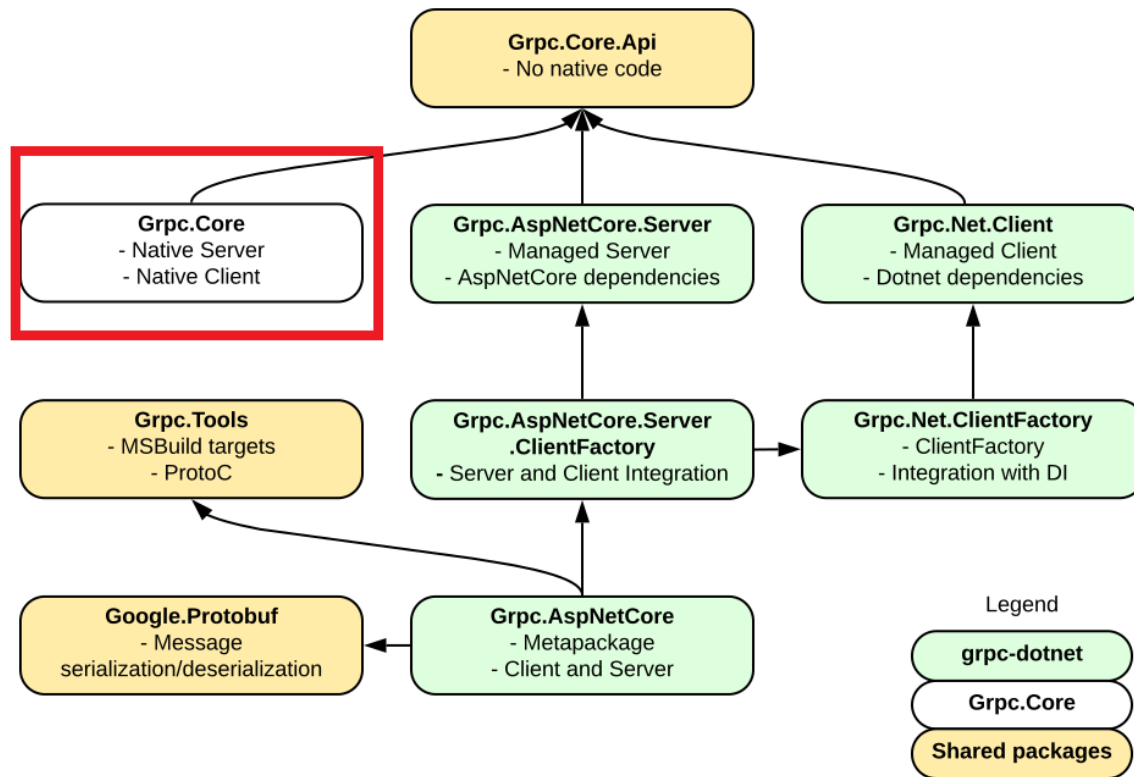
gRPC Минусы

- Нет поддержки со стороны браузеров(но есть официальная обертка)
- Нельзя хостить в Azure App Services и IIS(пока)
- Плохо с тулингом
- Некоторые вещи делаются сложнее чем в обычных рестовых сервисах(например кэширование, балансировка).

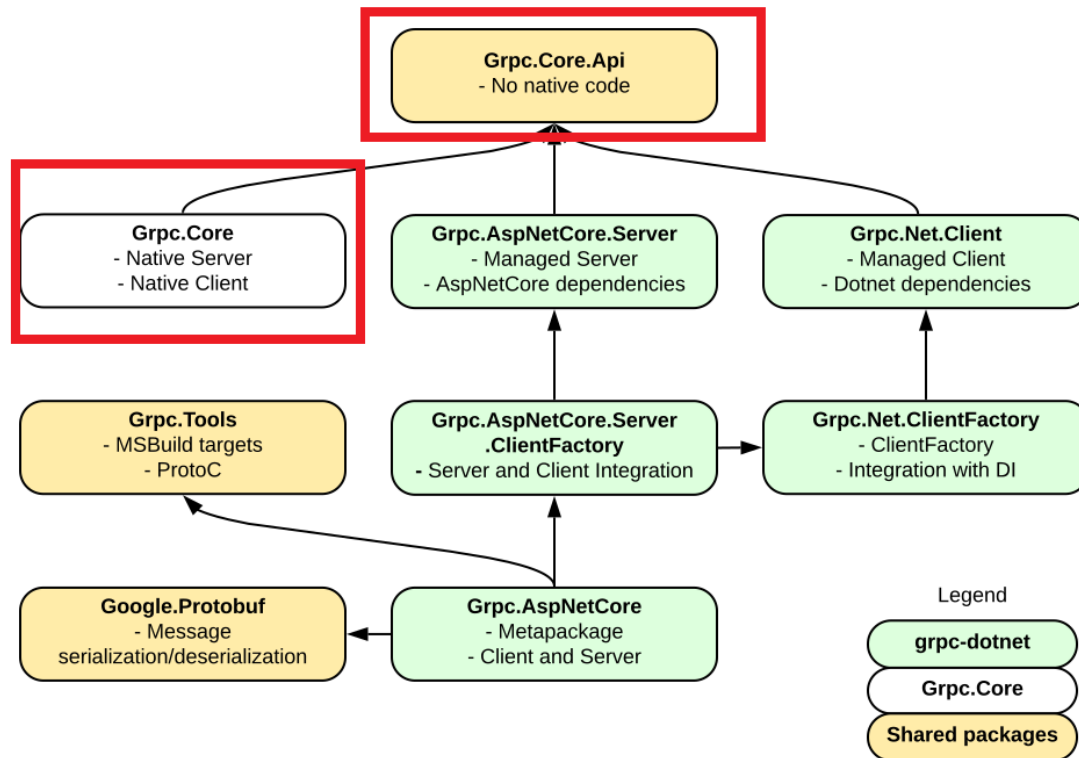
Что же в .Net?



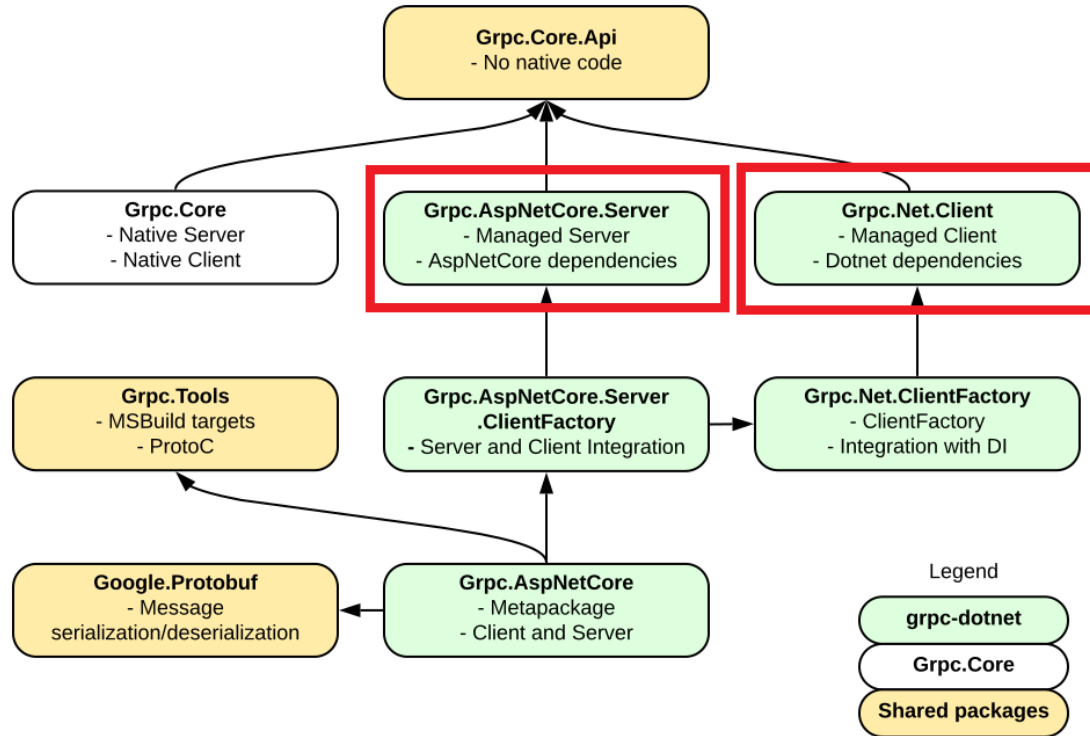
gRPC- .NET общее



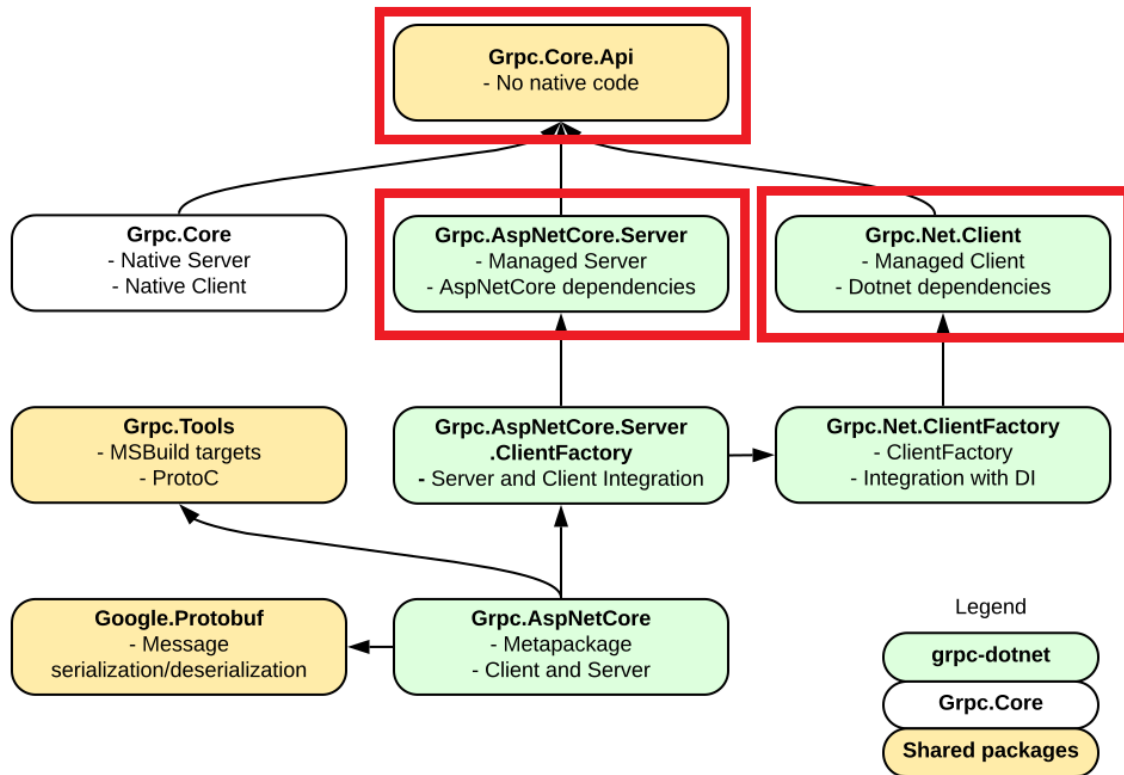
gRPC- .NET общее



gRPC- .NET общее



gRPC- .NET общее

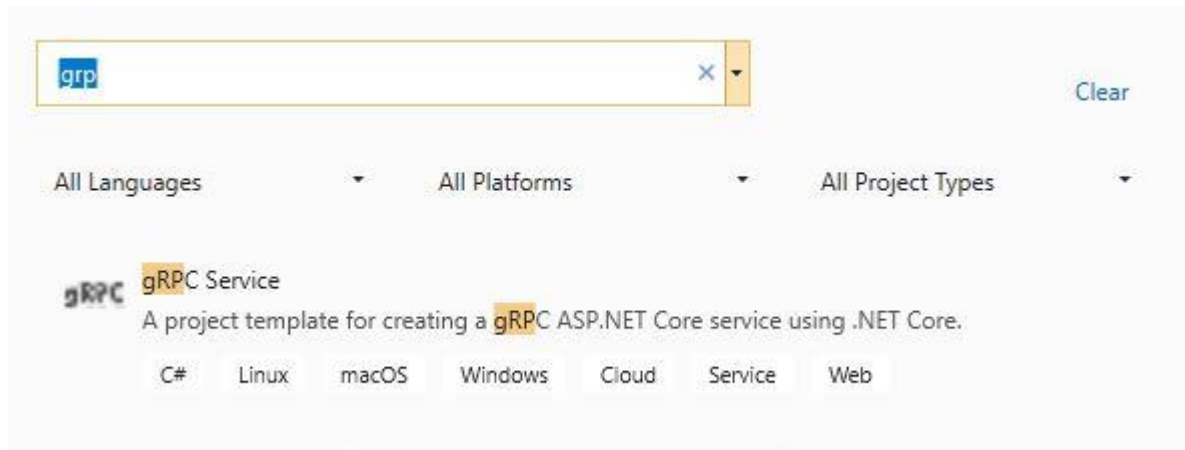


Внедряем gRPC

Дано:

- Простейший файловый сервис
- Есть вариант на Asp.NET Core 3 для сравнения
- Три метода: взять файл, загрузить файл и получить метаданные файлов

gRPC- .NET сервер



gRPC- .NET сервер

```
<ItemGroup>  
  <PackageReference Include="Grpc.AspNetCore" Version="2.23.1" />  
</ItemGroup>
```

gRPC- .NET сервер

```
-- "Kestrel": -- {  
--   -- "EndpointDefaults": -- {  
--     -- "Protocols": -- "Http2"  
--   -- }  
-- }
```

gRPC- .NET сервер

```
→ public void ConfigureServices(IServiceCollection services)
→ {
→     services.AddGrpc(options=>
→     {
→         options.MaxReceiveMessageSize = 200 * 1024 * 1024; // 200 Mb
→     });
→ }
```

gRPC- .NET сервер

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseRouting();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapGrpcService<Services.FileService>();
    });
}
```


gRPC- IDL

```
syntax = "proto3";  
  
option csharp_namespace = "FileServerGrpc.Protos";  
□
```

gRPC- IDL

```
syntax = "proto3";
```

```
option csharp_namespace = "FileServiceGrpc";
```

```
package FileService;
```

gRPC- IDL

```
//The file service definition.
service FileService{
  □//Get the requested file by its name
  |  ···rpc GetFile (RequestedFile) returns (FileResponse);
  □//Upload file to file storage
  |  ···rpc UploadFile (stream UploadedFile) returns (UploadStatus);
  □//Get metadata for all files in the storage
  |  ···rpc GetFilesMetadata (FileStorageMetadataRequest) returns (FilesMetadata);
  }
}
```

gRPC- IDL

```
// Request for file from storage
message RequestedFile {
    string FileName = 1;
}

// Actual requested file from storage
message FileResponse {
    bytes Content = 1;
}
```

gRPC- IDL

```
// Request message for metadata of all files, it is empty
message FileStorageMetadataRequest {
}

// All files metadata
message FilesMetadata {
  → repeated StoredFileMetadata Metadata = 1;
  → message StoredFileMetadata {
  →   → string FileName = 1;
  →   → google.protobuf.Timestamp CreatedTime = 2;
  →   → bool IsReadOnly = 3;
  →   → int64 Size = 4;
  → }
}
```

gRPC- IDL

```
// Uploading file
message UploadedFile {
  oneof Chunk {
    FileMetadata Metadata = 1;
    bytes Content = 2;
  }
  message FileMetadata {
    string FileName = 1;
    int32 UserId = 2;
  }
}

// Status of the upload
message UploadStatus {
  Result Result = 1;
}

// Possible results of file upload
enum Result {
  NOT_DEFINED = 0;
  SUCCESS = 1;
  FAILED = 2;
}
```

gRPC- IDL

```
//-Request-for-file-from-storage
message RequestedFile {
  → string FileName = 1;
}
//-Actual-requested-file-from-storage
message FileResponse {
  → bytes Content = 1;
}
```

- 1 - тип данных
- 2 - название поля
- 3- тег поля

gRPC - .Net protobuf

.proto type	C# type
double	double
float	float
int32	int
int64	long
uint32	uint
uint64	ulong
sint32	int
sint64	long

.proto type	C# type
fixed32	uint
fixed64	ulong
sfixed32	int
sfixed64	long
bool	bool
string	string
bytes	ByteString

gRPC- IDL

```
// Uploading file
message UploadedFile {
  1 oneof Chunk {
    → FileMetadata Metadata = 1;
    → bytes Content = 2;
  }
  message FileMetadata {
    → string FileName = 1;
    → int32 UserId = 2;
  }
}

// Status of the upload
message UploadStatus {
  → Result Result = 1;
}

// Possible results of file upload
2 enum Result {
  → NOT_DEFINED = 0;
  → SUCCESS = 1;
  → FAILED = 2;
}
```

gRPC- IDL

```
// Request message for metadata of all files, it is empty
message FileStorageMetadataRequest {
}

// All files metadata
message FilesMetadata {
  1 → repeated StoredFileMetadata Metadata = 1;
  2 → message StoredFileMetadata {
    → string FileName = 1;
    → google.protobuf.Timestamp CreatedTime = 2;
    → bool IsReadOnly = 3;
    → int64 Size = 4;
  }
}
```

gRPC- IDL

```
import "google/protobuf/timestamp.proto";
```

gRPC- .NET сервер

```
-<ItemGroup>  
---<Protobuf Include="Protos\fileservice.proto" GrpcServices="Server" -/>  
-</ItemGroup>
```

gRPC- .NET сервер

```
4 references
public class FileService : FileServerGrpc.FileService, FileServiceBase
{
    → private ILogger<FileService> _logger;
    → private FileStorageConfig _config;

    0 references
    → public FileService(ILogger<FileService> logger, IOptions<FileStorageConfig> config) {...}

    3 references
    → public override async Task<FileResponse> GetFile(RequestedFile request, ServerCallContext context) {...}

    3 references
    → public override async Task<UploadStatus> UploadFile(IAsyncStreamReader<UploadedFile> requestStream, ServerCallContext context) {...}

    3 references
    → public override async Task<FilesMetadata> GetFilesMetadata(FileStorageMetadataRequest request, ServerCallContext context) {...}
}
```

gRPC- .NET сервер

```
//<auto-generated>
//-----Generated by the protocol buffer compiler, DO NOT EDIT!
//-----source: Protos/fileservice.proto
//</auto-generated>
#pragma warning disable 0414, 1591
#region Designer-generated code

using grpc = global::Grpc.Core;

namespace FileServerGrpc {
    ///<summary>
    ///The file service definition.
    ///</summary>
    [global::System.Runtime.InteropServices.ComVisible(false)]
    public static partial class FileService
    {
        static readonly string __ServiceName = "Files.FileService";

        static readonly grpc::Marshaller<global::FileServerGrpc.RequestedFile> __Marshaller_Files_RequestedFile = grpc::Marshallers.Create(serializer: (arg) => global::Google.Protobuf.FileServerGrpc.RequestedFile.FromByteArray(global::Google.Protobuf.Collections.ByteArrayExtensions.ToByteArray(arg)), deserializer: (arg) => global::Google.Protobuf.FileServerGrpc.RequestedFile.FromByteArray(global::Google.Protobuf.Collections.ByteArrayExtensions.ToByteArray(arg)));
        static readonly grpc::Marshaller<global::FileServerGrpc.FileResponse> __Marshaller_Files_FileResponse = grpc::Marshallers.Create(serializer: (arg) => global::Google.Protobuf.FileServerGrpc.FileResponse.FromByteArray(global::Google.Protobuf.Collections.ByteArrayExtensions.ToByteArray(arg)), deserializer: (arg) => global::Google.Protobuf.FileServerGrpc.FileResponse.FromByteArray(global::Google.Protobuf.Collections.ByteArrayExtensions.ToByteArray(arg)));
        static readonly grpc::Marshaller<global::FileServerGrpc.UploadedFile> __Marshaller_Files_UploadedFile = grpc::Marshallers.Create(serializer: (arg) => global::Google.Protobuf.FileServerGrpc.UploadedFile.FromByteArray(global::Google.Protobuf.Collections.ByteArrayExtensions.ToByteArray(arg)), deserializer: (arg) => global::Google.Protobuf.FileServerGrpc.UploadedFile.FromByteArray(global::Google.Protobuf.Collections.ByteArrayExtensions.ToByteArray(arg)));
        static readonly grpc::Marshaller<global::FileServerGrpc.UploadStatus> __Marshaller_Files_UploadStatus = grpc::Marshallers.Create(serializer: (arg) => global::Google.Protobuf.FileServerGrpc.UploadStatus.FromByteArray(global::Google.Protobuf.Collections.ByteArrayExtensions.ToByteArray(arg)), deserializer: (arg) => global::Google.Protobuf.FileServerGrpc.UploadStatus.FromByteArray(global::Google.Protobuf.Collections.ByteArrayExtensions.ToByteArray(arg)));
        static readonly grpc::Marshaller<global::FileServerGrpc.FileStorageMetadataRequest> __Marshaller_Files_FileStorageMetadataRequest = grpc::Marshallers.Create(serializer: (arg) => global::Google.Protobuf.FileServerGrpc.FileStorageMetadataRequest.FromByteArray(global::Google.Protobuf.Collections.ByteArrayExtensions.ToByteArray(arg)), deserializer: (arg) => global::Google.Protobuf.FileServerGrpc.FileStorageMetadataRequest.FromByteArray(global::Google.Protobuf.Collections.ByteArrayExtensions.ToByteArray(arg)));
        static readonly grpc::Marshaller<global::FileServerGrpc.FilesMetadata> __Marshaller_Files_FilesMetadata = grpc::Marshallers.Create(serializer: (arg) => global::Google.Protobuf.FileServerGrpc.FilesMetadata.FromByteArray(global::Google.Protobuf.Collections.ByteArrayExtensions.ToByteArray(arg)), deserializer: (arg) => global::Google.Protobuf.FileServerGrpc.FilesMetadata.FromByteArray(global::Google.Protobuf.Collections.ByteArrayExtensions.ToByteArray(arg)));

        static readonly grpc::Method<global::FileServerGrpc.RequestedFile, global::FileServerGrpc.FileResponse> __Method_GetFile = new grpc::Method<global::FileServerGrpc.RequestedFile, global::FileServerGrpc.FileResponse>(
            grpc::MethodType.Unary,
            __ServiceName,
            name: "GetFile",
            requestMarshaller: __Marshaller_Files_RequestedFile,
            responseMarshaller: __Marshaller_Files_FileResponse);

        static readonly grpc::Method<global::FileServerGrpc.UploadedFile, global::FileServerGrpc.UploadStatus> __Method_UploadFile = new grpc::Method<global::FileServerGrpc.UploadedFile, global::FileServerGrpc.UploadStatus>(
            grpc::MethodType.ClientStreaming,
            __ServiceName,
            name: "UploadFile",
            requestMarshaller: __Marshaller_Files_UploadedFile,
            responseMarshaller: __Marshaller_Files_UploadStatus);

        static readonly grpc::Method<global::FileServerGrpc.FileStorageMetadataRequest, global::FileServerGrpc.FilesMetadata> __Method_GetFilesMetadata = new grpc::Method<global::FileServerGrpc.FileStorageMetadataRequest, global::FileServerGrpc.FilesMetadata>(
            grpc::MethodType.Unary,
            __ServiceName,
            name: "GetFilesMetadata",
            requestMarshaller: __Marshaller_Files_FileStorageMetadataRequest,
            responseMarshaller: __Marshaller_Files_FilesMetadata);
    }
}
```

gRPC- .NET сервер

4 references

```
public class FileService : FileServerGrpc.FileService, FileServiceBase
```

```
{
```

```
→ private ILogger<FileService> _logger;
```

```
→ private FileStorageConfig _config;
```

0 references

```
→ public FileService(ILogger<FileService> logger, IOptions<FileStorageConfig> config)...
```

3 references

```
→ public override async Task<FileResponse> GetFile(RequestedFile request, ServerCallContext context)...
```

3 references

```
→ public override async Task<UploadStatus> UploadFile(IAsyncStreamReader<UploadedFile> requestStream, ServerCallContext context)...
```

3 references

```
→ public override async Task<FilesMetadata> GetFilesMetadata(FileStorageMetadataRequest request, ServerCallContext context)...
```

```
}
```

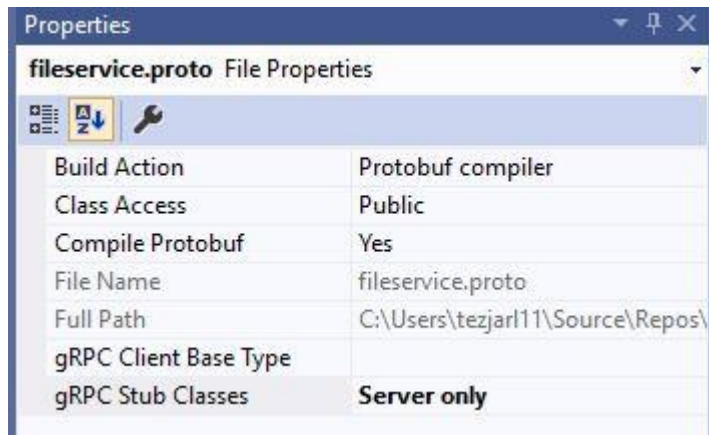
gRPC- .NET клиент

```
-<ItemGroup>
  ---<PackageReference Include="Google.Protobuf" Version="3.11.1" />
  ---<PackageReference Include="Grpc.Net.Client" Version="2.25.0" />
  ---<PackageReference Include="Grpc.Tools" Version="2.25.0">
    ----<PrivateAssets>all</PrivateAssets>
    ----<IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
  ---</PackageReference>
-</ItemGroup>
```


gRPC- .NET клиент

```
-<ItemGroup>  
-  <Protobuf Include="Protos\fileservice.proto" GrpcServices="Client"/>  
-</ItemGroup>
```

gRPC- .NET клиент



gRPC- .NET клиент

```
using (var channel = GrpcChannel.ForAddress(_grpcServerUrl, new GrpcChannelOptions() { MaxSendMessageSize = 200 * 1024 * 1024 }))
{
    → var client = new FileService.FileServiceClient(channel);
    →
    → var request = new RequestedFile()
    → {
    →     FileName = "test1.txt"
    → };
    → var responseSync = client.GetFile(request, new CallOptions(deadline: DateTime.UtcNow.AddMinutes(10)));
    → var responseAsync = await client.GetFileAsync(request, new Metadata(), new CallOptions(deadline: DateTime.UtcNow.AddMinutes(10)));
}
```

gRPC- .NET клиент

```
→ var stream = client.UploadFile();
→ var fileMetadata = new UploadedFile()
→ {
→     Metadata = new UploadedFile.Types.FileMetadata()
→     {
→         FileName = "test1.txt",
→         UserId = 12
→     }
→ };
→ await stream.RequestStream.WriteAsync(fileMetadata);
→ await using (var fileStream = new FileStream(_filePath, FileMode.Open, FileAccess.Read))
→ {
→     var fileContent = new UploadedFile()
→     {
→         Content = ByteString.FromStream(fileStream)
→     };
→     await stream.RequestStream.WriteAsync(fileContent);
→ }

→ await stream.RequestStream.CompleteAsync();
→ var response = await stream.ResponseAsync;
```

Таймауты

Таймауты

```
using (var channel = GrpcChannel.ForAddress(_grpcServerUrl, new GrpcChannelOptions() { MaxSendMessageSize = 200 * 1024 * 1024 }))
{
    var client = new FileService.FileServiceClient(channel);
    var request = new RequestedFile()
    {
        FileName = "test1.txt"
    };
    var responseSync = client.GetFile(request, new CallOptions(deadline: DateTime.UtcNow.AddMinutes(10)));
    var responseAsync = await client.GetFileAsync(request, new Metadata(), deadline: DateTime.UtcNow.AddMinutes(10));
}
```



Таймауты

```
⇒ if (DateTime.UtcNow > context.Deadline)
⇒     throw new RpcException(new Status(StatusCode.DeadlineExceeded, detail: "Deadline exceeded"));
```

Немного про ошибки.

Ошибки

GRPC_STATUS_CANCELLED

GRPC_STATUS_DEADLINE_EXCEEDED

GRPC_STATUS_UNIMPLEMENTED

GRPC_STATUS_UNAVAILABLE

GRPC_STATUS_UNKNOWN

GRPC_STATUS_INTERNAL

GRPC_STATUS_RESOURCE_EXHAUSTED

GRPC_STATUS_UNAUTHENTICATED

Ошибки

Aborted

AlreadyExists

DataLoss

FailedPrecondition

InvalidArgument

NotFound

OutOfRange

PermissionDenied

Версионирование

Версионирование

Сервисы

Сообщения

Версионирование сообщений

Версионирование.Добавление поля

```
// Actual requested file from storage  
message FileResponse {  
    bytes Content = 1;  
}
```



```
// Actual requested file from storage  
message FileResponse {  
    bytes Content = 1;  
    string HashSum = 2;  
}
```

Версионирование.Добавление поля

responseAsync	{{ "Content": "0e7n5ODt6OUNCt/n++oq4u7n7ejqlOlq7eD34OvIIDE5OD..." }}	FileServerGrpc.FileRes...
Content	{Google.Protobuf.ByteString}	Google.Protobuf.Byte...
_unknownFields	{Google.Protobuf.UnknownFieldSet}	Google.Protobuf.Unk...
content_	{Google.Protobuf.ByteString}	Google.Protobuf.Byte...
Google.Protobuf.IMessage.Descriptor	{Google.Protobuf.Reflection.MessageDescriptor}	Google.Protobuf.Refle...
Static members		

Версионирование. Удаление поля

```
// -All-files-metadata
message FilesMetadata {
  repeated StoredFileMetadata Metadata = 1;
  message StoredFileMetadata {
    string FileName = 1;
    google.protobuf.Timestamp CreatedTime = 2;
    bool IsReadOnly = 3;
    int64 Size = 4;
  }
}
```



```
// -All-files-metadata
message FilesMetadata {
  repeated StoredFileMetadata Metadata = 1;
  message StoredFileMetadata {
    reserved 3;
    string FileName = 1;
    google.protobuf.Timestamp CreatedTime = 2;
    int64 Size = 4;
  }
}
```


Версионирование. Удаление поля

```
// All files metadata
message FilesMetadata {
  repeated StoredFileMetadata Metadata = 1;
  message StoredFileMetadata {
    reserved 3;
    string FileName = 1;
    google.protobuf.Timestamp CreatedTime = 2;
    int64 Size = 4;
    uint32 AnotherField = 3;
  }
}
```

Версионирование. Удаление поля

responseSync	{{ "Metadata": [{ "FileName": "gRPC-final-01.pptx", "CreatedTime": "201...	FileServerGrpc.FilesM...
Metadata	[{ "FileName": "gRPC-final-01.pptx", "CreatedTime": "2019-12-09T00:2...	Google.Protobuf.Coll...
Capacity	8	int
Count	4	int
IsReadOnly	false	bool
Static members		
Non-Public members		
Results View	Expanding the Results View will enumerate the IEnumerable	
[0]	[{ "FileName": "gRPC-final-01.pptx", "CreatedTime": "2019-12-09T00:21:23.352273300Z" }	FileServerGrpc.FilesM...
CreatedTime	{ "2019-12-09T00:21:23.352273300Z" }	Google.Protobuf.Well...
FileName	"gRPC-final-01.pptx"	string
IsReadOnly	false	bool
Size	2820144	long
_unknownFields	null	Google.Protobuf.Unk...
createdTime_	{ "2019-12-09T00:21:23.352273300Z" }	Google.Protobuf.Well...
fileName_	"gRPC-final-01.pptx"	string
isReadOnly_	false	bool
Google.Protobuf.IMessage....	{ Google.Protobuf.Reflection.MessageDescriptor }	Google.Protobuf.Refle...
size_	2820144	long
Static members		
[1]	[{ "FileName": "in.txt", "CreatedTime": "2016-06-20T20:02:39.661429Z", "...	FileServerGrpc.FilesM...
[2]	[{ "FileName": "out.txt", "CreatedTime": "2016-06-20T20:02:39.661429Z", "...	FileServerGrpc.FilesM...
[3]	[{ "FileName": "test1.txt", "CreatedTime": "2019-12-09T21:17:59.3026584...", "...	FileServerGrpc.FilesM...

Версионирование.Смена типа поля

Версионирование

- Теги всегда должны быть уникальны и неизменны
- Удаленные поля рекомендуется помечать ключевым словом reserved
- Смена типа возможна, но ограничена

Версионирование сервисов

Версионирование. Добавление метода

```
// - The file service definition.
service FileService {
  // - Get the requested file by its name
  rpc GetFile (RequestedFile) returns (FileResponse);
  // - Upload file to file storage
  rpc UploadFile (stream UploadedFile) returns (UploadStatus);
  // - Get metadata for all files in the storage
  rpc GetFilesMetadata (FileStorageMetadataRequest) returns (FilesMetadata);
}
```

Версионирование. Добавление метода

```
//The file service definition.
service FileService {
  //Get the requested file by its name
  [+rpc-GetFile (RequestedFile) returns (FileResponse);
  //Upload file to file storage
  [+rpc-UploadFile (stream UploadedFile) returns (UploadStatus);
  //Get metadata for all files in the storage
  [+rpc-GetFilesMetadata (FileStorageMetadataRequest) returns (FilesMetadata);
  //Delete file
  [+rpc-DeleteFile (RequestedFile) returns (DeletedStatus);
}
```

Версионирование. Добавление метода











```
var responseAsync = await client.GetFileAsync(request, new Metadata(), DateTime.UtcNow.A
```

≡

```
async Task GrpcClientGetFilesM
```

≡

```
async Task GrpcClientUploadFil
```

 GetFile	FileResponse
 GetFileAsync	AsyncUnaryCall<FileResponse>
 GetFilesMetadata	FilesMetadata
 GetFilesMetadataAsync	AsyncUnaryCall<FilesMetadata>
 UploadFile	AsyncClientStreamingCall<UploadedFile, UploadStatus>
 WithHost	FileServiceClient
 Equals	bool
 GetHashCode	int
 GetType	Type
 ToString	string

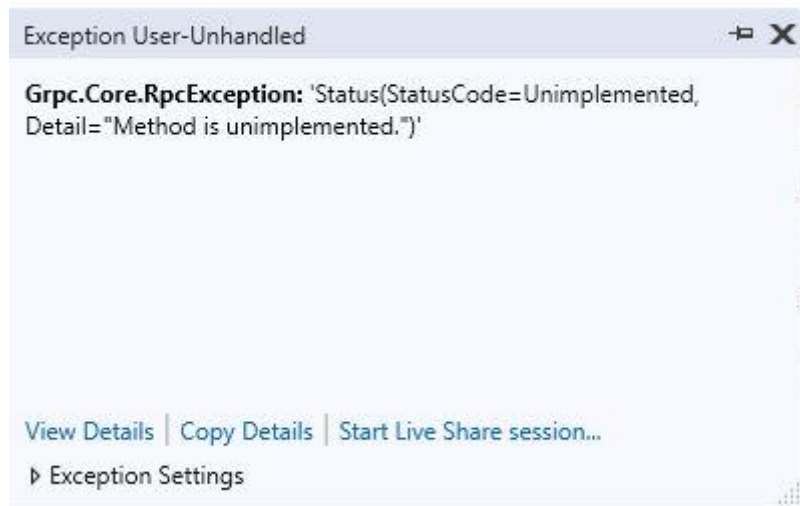
Версионирование. Удаление метода

```
// - The file service definition.  
service FileService {  
  // - Get the requested file by its name  
  rpc GetFile (RequestedFile) returns (FileResponse);  
  // - Upload file to file storage  
  rpc UploadFile (stream UploadedFile) returns (UploadStatus);  
  // - Get metadata for all files in the storage  
  rpc GetFilesMetadata (FileStorageMetadataRequest) returns (FilesMetadata);  
}
```

Версионирование. Удаление метода

```
//The file service definition.  
service FileService {  
  //Get the requested file by its name  
  | -rpc GetFile (RequestedFile) returns (FileResponse);  
  //Upload file to file storage  
  | -rpc UploadFile (stream UploadedFile) returns (UploadStatus);  
  //Get metadata for all files in the storage  
  | -rpc GetFilesMetadata (FileStorageMetadataRequest) returns (FilesMetadata);  
}
```

Версионирование. Удаление метода



Версионирование

Ключевая проблема - proto файлы не имеют версии.

Авторизация и Аутентификация

Авторизация и Аутентификация

Channel credentials

Call credentials

Channel creds

```
* var clientCrt = File.ReadAllText(path: "ssl/client.crt");
* var clientKey = File.ReadAllText(path: "ssl/client.key");
* var caCrt = File.ReadAllText(path: "ssl/ca.crt");

* var channelCredentials = new SslCredentials(caCrt, new KeyCertificatePair(certificateChain: clientCrt, clientKey));
* using (var channel = GrpcChannel.ForAddress(_grpcServerUrl, new GrpcChannelOptions() {
*     → MaxSendMessageSize = 200 * 1024 * 1024,
*     → Credentials = channelCredentials
*     → })))
```

Channel creds



Microsoft.AspNetCore.Authentication.Certificate by Microsoft, 15,6K downloads

ASP.NET Core middleware that enables an application to support certificate authentication.

v3.1.0

Channel creds

```
|  
→ services.AddAuthentication(CertificateAuthenticationDefaults.AuthenticationScheme)  
→ → .AddCertificate();
```

Channel creds

0 references

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    → app.UseRouting();

    → app.UseAuthentication();
    → app.UseAuthorization();

    |
    → app.UseEndpoints(endpoints =>
    → {
    →     endpoints.MapGrpcService<Services.FileService>();
    → });
}
```

Channel creds

```
Host.CreateDefaultBuilder(args)
→ .ConfigureWebHostDefaults(webBuilder =>
→ {
→     webBuilder.UseStartup<Startup>();
→     webBuilder.ConfigureKestrel(options =>
→     {
→         options.ConfigureHttpsDefaults(configureOptions: o =>
→         {
→             o.ClientCertificateMode = ClientCertificateMode.RequireCertificate;
→         });
→     });
→ });
```

Call creds

```
|  
var credentials = CallCredentials.FromInterceptor((context, metadata) =>  
{  
    metadata.Add("Authorization", $"Bearer {_token}");  
    return Task.CompletedTask;  
});  
using (var channel = GrpcChannel.ForAddress(_grpcServerUrl, new GrpcChannelOptions()  
{  
    MaxSendMessageSize = 200 * 1024 * 1024,  
    Credentials = ChannelCredentials.Create(new SslCredentials(), credentials)  
}))  
{  
    r
```

Call creds

```
var client = new FileService.FileServiceClient(channel);
var headers = new Metadata();
headers.Add("Authorization", $"Bearer {_token}");

var request = new RequestedFile()
{
    → FileName = "test1.txt"
};
var responseSync = client.GetFile(request, new CallOptions(deadline: DateTime.Now.AddMinutes(10), headers: headers));
var responseAsync = await client.GetFileAsync(request, headers, deadline: DateTime.Now.AddMinutes(10));
```

Call creds

0 references

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    → app.UseRouting();

    → app.UseAuthentication();
    → app.UseAuthorization();

    |
    → app.UseEndpoints(endpoints =>
    → {
    →     endpoints.MapGrpcService<Services.FileService>();
    → });
}
```

Call creds

[Authorize]

4 references

```
public class FileService : FileServerGrpc.FileService, FileServiceBase
```

```
{
```

```
→ private ILogger<FileService> _logger;
```

```
→ private FileStorageConfig _config;
```

0 references

```
→ public FileService(ILogger<FileService> logger, IOptions<FileStorageConfig> config) [...]
```

3 references

```
→ public override async Task<FileResponse> GetFile(RequestedFile request, ServerCallContext context) [...]
```

3 references

```
→ public override async Task<UploadStatus> UploadFile(IAsyncStreamReader<UploadedFile> requestStream, ServerCallContext context) [...]
```

3 references

```
→ public override async Task<FilesMetadata> GetFilesMetadata(FileStorageMetadataRequest request, ServerCallContext context) [...]
```

```
}
```

Аутентификация - опции

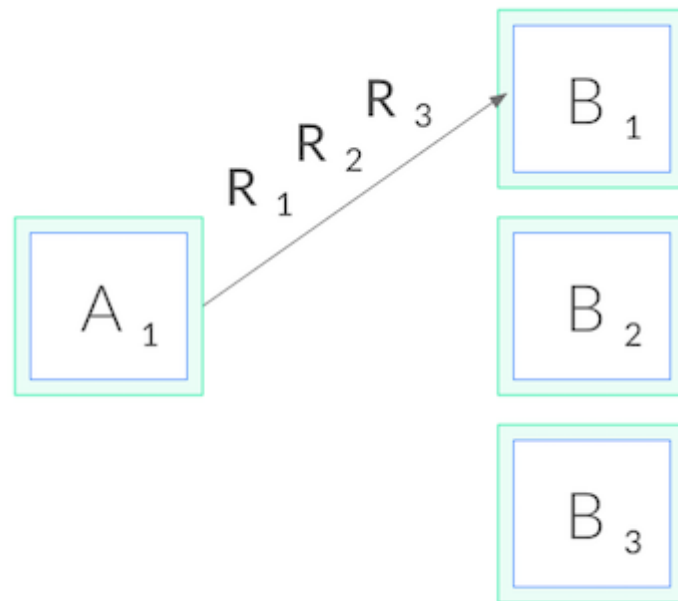
Поддерживаются:

- Azure AD
- OAuth 2
- IdentityServer
- OpenID

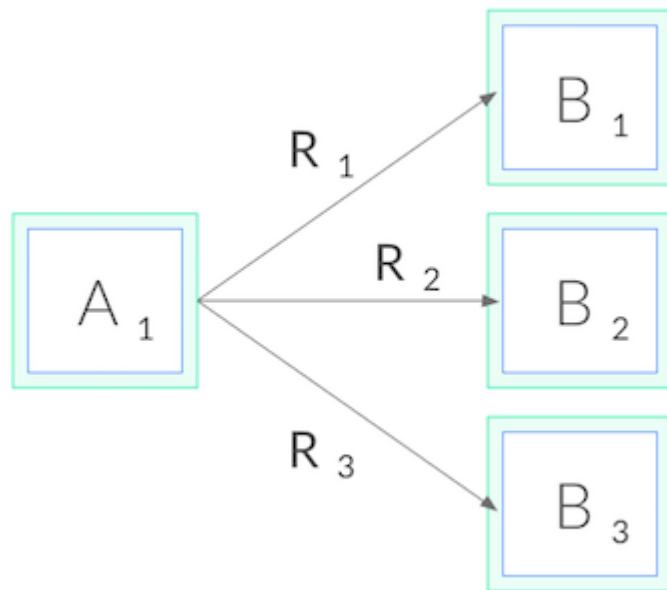
НЕ работает - windows аутентификация(NTLM/Kerberos)

Балансировка

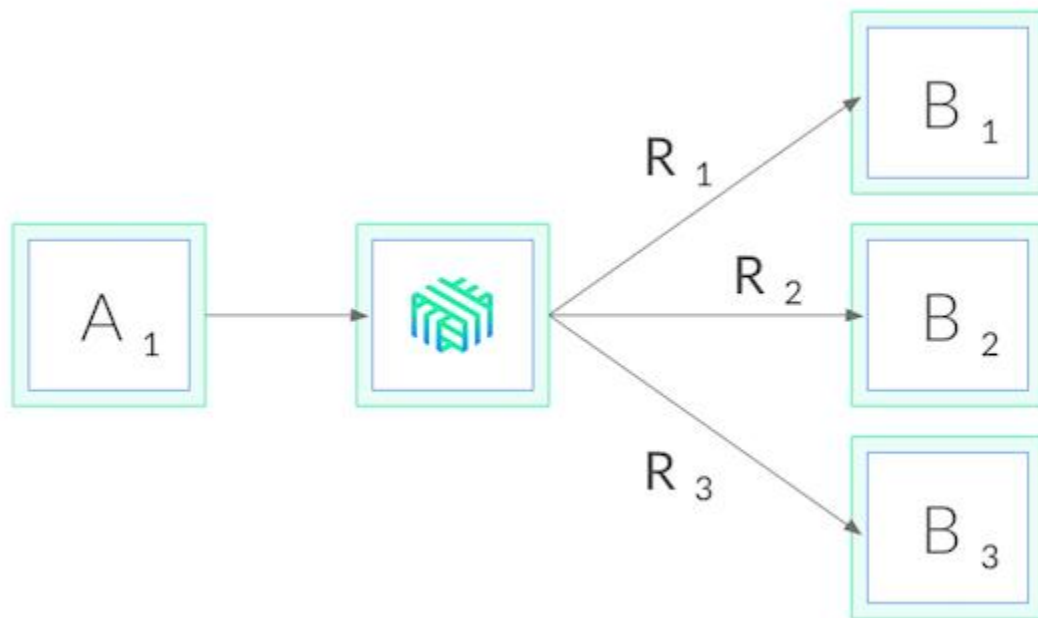
Балансировка



Балансировка



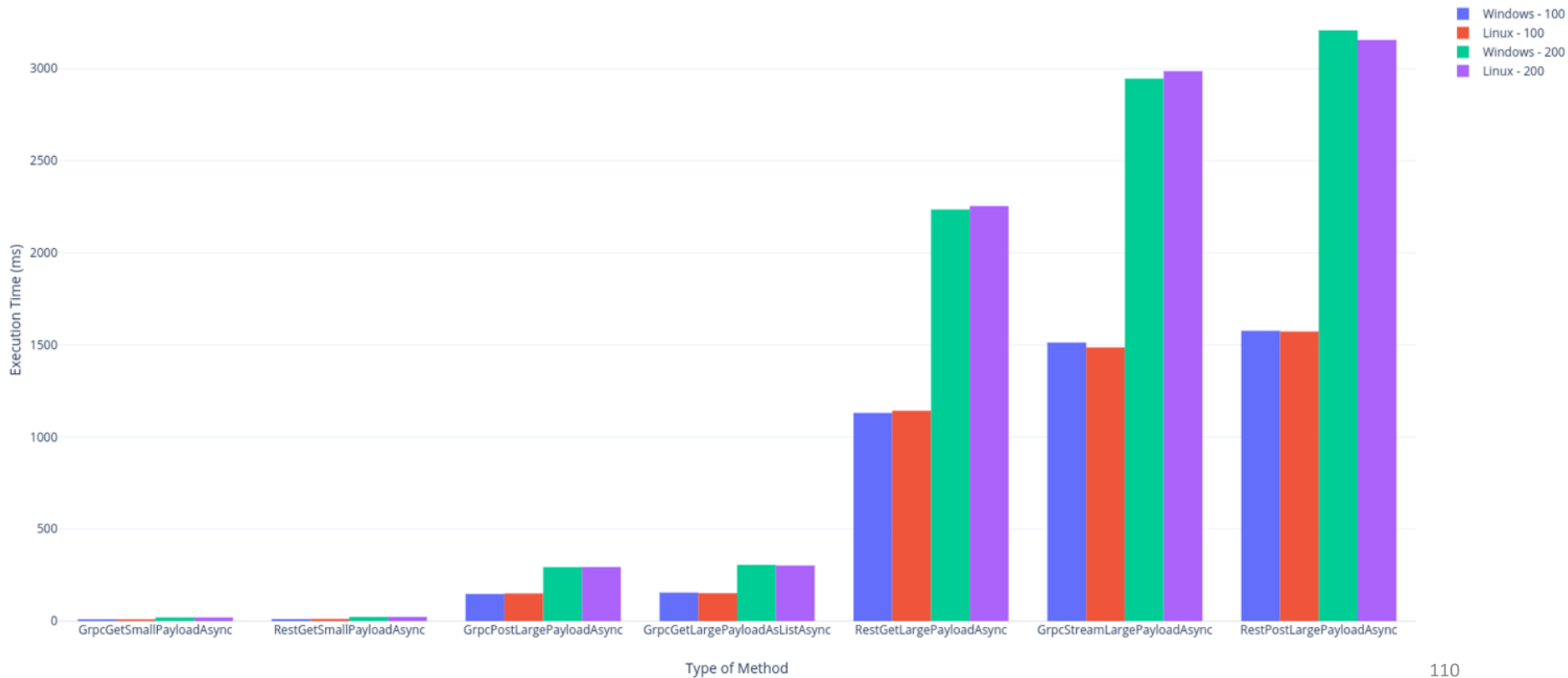
Балансировка



Настало время
бенчмаркать!

Бенчмарк. Эталон

REST vs GRPC Performance



Бенчмарк. Замер на локальной машине

Дано:

- Grpc сервис
- Asp.NET Core 3 сервис
- BenchmarkDotNet
- Текстовый файл на ~100Кб
- Файловое хранилище с парой тысяч файлов

Бенчмарк. Замер на локальной машине

```
BenchmarkDotNet=v0.12.0, OS=Windows 10.0.18363
Intel Core i5-6500 CPU 3.20GHz (Skylake), 1 CPU, 4 logical and 4 physical cores
.NET Core SDK=3.0.100
[Host] : .NET Core 3.0.0 (CoreCLR 4.700.19.46205, CoreFX 4.700.19.46214), X64 RyuJIT
DefaultJob : .NET Core 3.0.0 (CoreCLR 4.700.19.46205, CoreFX 4.700.19.46214), X64 RyuJIT
```

Method	Mean	Error	StdDev	StdErr	Median	Min	Q1	Q3	Max	Op/s	Gen 0	Gen 1	Gen 2	Allocated
'Grpc get file'	20.07 ms	0.442 ms	1.267 ms	0.130 ms	19.79 ms	17.88 ms	19.18 ms	20.99 ms	23.76 ms	49.83	187.5000	187.5000	187.5000	838.35 KB
'Grpc get all files metadata'	39.76 ms	1.129 ms	3.148 ms	0.332 ms	38.80 ms	35.04 ms	37.69 ms	40.90 ms	49.30 ms	25.15	500.0000	500.0000	300.0000	3428.27 KB
'Grpc upload a file'	25.72 ms	2.350 ms	6.353 ms	0.689 ms	24.13 ms	18.58 ms	20.75 ms	29.10 ms	48.18 ms	38.87	214.2857	214.2857	214.2857	938.56 KB
'AspNet Core get file'	26.18 ms	2.143 ms	5.939 ms	0.630 ms	23.82 ms	20.91 ms	22.31 ms	29.06 ms	44.05 ms	38.19	-	-	-	271.97 KB
'AspNet Core get all files metadata'	53.69 ms	2.011 ms	5.835 ms	0.592 ms	53.03 ms	44.87 ms	49.20 ms	56.43 ms	68.55 ms	18.62	888.8889	888.8889	888.8889	4045.46 KB
'AspNet Core upload file'	31.99 ms	1.932 ms	5.449 ms	0.568 ms	31.77 ms	23.93 ms	26.77 ms	35.67 ms	47.53 ms	31.26	62.5000	62.5000	62.5000	275.19 KB

Бенчмарк. Замер на локальной машине

Method	Mean	Error	StdDev	StdErr	Median	Min	Q1	Q3	Max	Op/s
'Grpc get file'	20.07 ms	0.442 ms	1.267 ms	0.130 ms	19.79 ms	17.88 ms	19.18 ms	20.99 ms	23.76 ms	49.83
'Grpc get all files metadata'	39.76 ms	1.129 ms	3.148 ms	0.332 ms	38.80 ms	35.04 ms	37.69 ms	40.90 ms	49.30 ms	25.15
'Grpc upload a file'	25.72 ms	2.350 ms	6.353 ms	0.689 ms	24.13 ms	18.58 ms	20.75 ms	29.10 ms	48.18 ms	38.87
'AspNet Core get file'	26.18 ms	2.143 ms	5.939 ms	0.630 ms	23.82 ms	20.91 ms	22.31 ms	29.06 ms	44.05 ms	38.19
'AspNet Core get all files metadata'	53.69 ms	2.011 ms	5.835 ms	0.592 ms	53.03 ms	44.87 ms	49.20 ms	56.43 ms	68.55 ms	18.62
'AspNet Core upload file'	31.99 ms	1.932 ms	5.449 ms	0.568 ms	31.77 ms	23.93 ms	26.77 ms	35.67 ms	47.53 ms	31.26

Бенчмарк. Замер на локальной машине

Method	Mean	Error	StdDev	StdErr	Median	Min	Q1	Q3	Max	Op/s
'Grpc get file'	20.07 ms	0.442 ms	1.267 ms	0.130 ms	19.79 ms	17.88 ms	19.18 ms	20.99 ms	23.76 ms	49.83
'Grpc get all files metadata'	39.76 ms	1.129 ms	3.148 ms	0.332 ms	38.80 ms	35.04 ms	37.69 ms	40.90 ms	49.30 ms	25.15
'Grpc upload a file'	25.72 ms	2.350 ms	6.353 ms	0.689 ms	24.13 ms	18.58 ms	20.75 ms	29.10 ms	48.18 ms	38.87
'AspNet Core get file'	26.18 ms	2.143 ms	5.939 ms	0.630 ms	23.82 ms	20.91 ms	22.31 ms	29.06 ms	44.05 ms	38.19
'AspNet Core get all files metadata'	53.69 ms	2.011 ms	5.835 ms	0.592 ms	53.03 ms	44.87 ms	49.20 ms	56.43 ms	68.55 ms	18.62
'AspNet Core upload file'	31.99 ms	1.932 ms	5.449 ms	0.568 ms	31.77 ms	23.93 ms	26.77 ms	35.67 ms	47.53 ms	31.26

Бенчмарк. Замер на локальной машине

Method	Mean	Error	StdDev	StdErr	Median	Min	Q1	Q3	Max	Op/s
'Grpc get file'	20.07 ms	0.442 ms	1.267 ms	0.130 ms	19.79 ms	17.88 ms	19.18 ms	20.99 ms	23.76 ms	49.83
'Grpc get all files metadata'	39.76 ms	1.129 ms	3.148 ms	0.332 ms	38.80 ms	35.04 ms	37.69 ms	40.90 ms	49.30 ms	25.15
'Grpc upload a file'	25.72 ms	2.350 ms	6.353 ms	0.689 ms	24.13 ms	18.58 ms	20.75 ms	29.10 ms	48.18 ms	38.87
'AspNet Core get file'	26.18 ms	2.143 ms	5.939 ms	0.630 ms	23.82 ms	20.91 ms	22.31 ms	29.06 ms	44.05 ms	38.19
'AspNet Core get all files metadata'	53.69 ms	2.011 ms	5.835 ms	0.592 ms	53.03 ms	44.87 ms	49.20 ms	56.43 ms	68.55 ms	18.62
'AspNet Core upload file'	31.99 ms	1.932 ms	5.449 ms	0.568 ms	31.77 ms	23.93 ms	26.77 ms	35.67 ms	47.53 ms	31.26

Бенчмарк. Замер на локальной машине

Method
'Grpc get file'
'Grpc get all files metadata'
'Grpc upload a file'
'AspNET Core get file'
'AspNET Core get all files metadata'
'AspNET Core upload file'

Gen 0	Gen 1	Gen 2	Allocated
187.5000	187.5000	187.5000	838.35 KB
500.0000	500.0000	300.0000	3428.27 KB
214.2857	214.2857	214.2857	938.56 KB
-	-	-	271.97 KB
888.8889	888.8889	888.8889	4045.46 KB
62.5000	62.5000	62.5000	275.19 KB

Бенчмарк. Замер в облаке

- Azure VM, standard
- North Europe регион
- Docker

Бенчмарк. Замер в облаке

BenchmarkDotNet=v0.12.0, OS=Windows 10.0.18363

Intel Core i5-6500 CPU 3.20GHz (Skylake), 1 CPU, 4 logical and 4 physical cores

.NET Core SDK=3.0.100

[Host] : .NET Core 3.0.0 (CoreCLR 4.700.19.46205, CoreFX 4.700.19.46214), X64 RyuJIT

DefaultJob : .NET Core 3.0.0 (CoreCLR 4.700.19.46205, CoreFX 4.700.19.46214), X64 RyuJIT

Method	Mean	Error	StdDev	StdErr	Min	Q1	Median	Q3	Max	Op/s	Gen 0	Gen 1	Gen 2	Allocated	Completed Work Items	Lock Contentions
'Grpc get file'	185.7 ms	2.86 ms	2.68 ms	0.69 ms	180.0 ms	183.5 ms	185.9 ms	186.9 ms	191.0 ms	5.385	-	-	-	231.3 KB	12.0000	-
'Grpc get all files metadata'	119.5 ms	0.90 ms	0.84 ms	0.22 ms	117.9 ms	119.0 ms	119.5 ms	120.2 ms	121.2 ms	8.367	-	-	-	98.68 KB	7.4000	-
'Grpc upload a file'	438.1 ms	8.33 ms	6.96 ms	1.93 ms	426.5 ms	430.4 ms	441.5 ms	443.6 ms	446.4 ms	2.283	-	-	-	974.01 KB	25.0000	-
'AspNet Core get file'	177.3 ms	2.12 ms	1.88 ms	0.50 ms	174.3 ms	176.0 ms	177.4 ms	178.2 ms	180.4 ms	5.640	-	-	-	60.03 KB	4.0000	-
'AspNet Core get all files metadata'	176.5 ms	2.05 ms	1.82 ms	0.49 ms	173.9 ms	174.9 ms	176.2 ms	178.2 ms	179.5 ms	5.667	-	-	-	78.96 KB	5.3333	-
'AspNet Core upload file'	635.4 ms	12.32 ms	15.13 ms	3.22 ms	581.1 ms	627.4 ms	639.9 ms	643.2 ms	651.6 ms	1.574	-	-	-	275.73 KB	6.0000	-

Бенчмарк. Замер в облаке

Method	Mean	Error	StdDev	StdErr	Min	Q1	Median	Q3	Max	Op/s
'Grpc get file'	185.7 ms	2.86 ms	2.68 ms	0.69 ms	180.0 ms	183.5 ms	185.9 ms	186.9 ms	191.0 ms	5.385
'Grpc get all files metadata'	119.5 ms	0.90 ms	0.84 ms	0.22 ms	117.9 ms	119.0 ms	119.5 ms	120.2 ms	121.2 ms	8.367
'Grpc upload a file'	438.1 ms	8.33 ms	6.96 ms	1.93 ms	426.5 ms	430.4 ms	441.5 ms	443.6 ms	446.4 ms	2.283
'AspNet Core get file'	177.3 ms	2.12 ms	1.88 ms	0.50 ms	174.3 ms	176.0 ms	177.4 ms	178.2 ms	180.4 ms	5.640
'AspNet Core get all files metadata'	176.5 ms	2.05 ms	1.82 ms	0.49 ms	173.9 ms	174.9 ms	176.2 ms	178.2 ms	179.5 ms	5.667
'AspNet Core upload file'	635.4 ms	12.32 ms	15.13 ms	3.22 ms	581.1 ms	627.4 ms	639.9 ms	643.2 ms	651.6 ms	1.574

Бенчмарк. Замер в облаке

Method	Mean	Error	StdDev	StdErr	Min	Q1	Median	Q3	Max	Op/s
'Grpc get file'	185.7 ms	2.86 ms	2.68 ms	0.69 ms	180.0 ms	183.5 ms	185.9 ms	186.9 ms	191.0 ms	5.385
'Grpc get all files metadata'	119.5 ms	0.90 ms	0.84 ms	0.22 ms	117.9 ms	119.0 ms	119.5 ms	120.2 ms	121.2 ms	8.367
'Grpc upload a file'	<u>438.1 ms</u>	8.33 ms	6.96 ms	1.93 ms	426.5 ms	430.4 ms	<u>441.5 ms</u>	443.6 ms	446.4 ms	2.283
'AspNet Core get file'	177.3 ms	2.12 ms	1.88 ms	0.50 ms	174.3 ms	176.0 ms	177.4 ms	178.2 ms	180.4 ms	5.640
'AspNet Core get all files metadata'	176.5 ms	2.05 ms	1.82 ms	0.49 ms	173.9 ms	174.9 ms	176.2 ms	178.2 ms	179.5 ms	5.667
'AspNet Core upload file'	<u>635.4 ms</u>	12.32 ms	15.13 ms	3.22 ms	581.1 ms	627.4 ms	<u>639.9 ms</u>	643.2 ms	651.6 ms	1.574

Бенчмарк. Замер в облаке

Method	Mean	Error	StdDev	StdErr	Min	Q1	Median	Q3	Max	Op/s
'Grpc get file'	185.7 ms	2.86 ms	2.68 ms	0.69 ms	180.0 ms	183.5 ms	185.9 ms	186.9 ms	191.0 ms	5.385
'Grpc get all files metadata'	119.5 ms	0.90 ms	0.84 ms	0.22 ms	117.9 ms	119.0 ms	119.5 ms	120.2 ms	121.2 ms	8.367
'Grpc upload a file'	438.1 ms	8.33 ms	6.96 ms	1.93 ms	426.5 ms	430.4 ms	441.5 ms	443.6 ms	446.4 ms	2.283
'AspNET Core get file'	177.3 ms	2.12 ms	1.88 ms	0.50 ms	174.3 ms	176.0 ms	177.4 ms	178.2 ms	180.4 ms	5.640
'AspNET Core get all files metadata'	176.5 ms	2.05 ms	1.82 ms	0.49 ms	173.9 ms	174.9 ms	176.2 ms	178.2 ms	179.5 ms	5.667
'AspNET Core upload file'	635.4 ms	12.32 ms	15.13 ms	3.22 ms	581.1 ms	627.4 ms	639.9 ms	643.2 ms	651.6 ms	1.574

Бенчмарк. Замер в облаке

Method	Mean	Error	StdDev	StdErr	Min	Q1	Median	Q3	Max	Op/s
'Grpc get file'	185.7 ms	2.86 ms	2.68 ms	0.69 ms	180.0 ms	183.5 ms	185.9 ms	186.9 ms	191.0 ms	5.385
'Grpc get all files metadata'	119.5 ms	0.90 ms	0.84 ms	0.22 ms	117.9 ms	119.0 ms	119.5 ms	120.2 ms	121.2 ms	8.367
'Grpc upload a file'	438.1 ms	8.33 ms	6.96 ms	1.93 ms	426.5 ms	430.4 ms	441.5 ms	443.6 ms	446.4 ms	2.283
'AspNET Core get file'	177.3 ms	2.12 ms	1.88 ms	0.50 ms	174.3 ms	176.0 ms	177.4 ms	178.2 ms	180.4 ms	5.640
'AspNET Core get all files metadata'	176.5 ms	2.05 ms	1.82 ms	0.49 ms	173.9 ms	174.9 ms	176.2 ms	178.2 ms	179.5 ms	5.667
'AspNET Core upload file'	635.4 ms	12.32 ms	15.13 ms	3.22 ms	581.1 ms	627.4 ms	639.9 ms	643.2 ms	651.6 ms	1.574

Бенчмарк. Замер в облаке

Method
'Grpc get file'
'Grpc get all files metadata'
'Grpc upload a file'
'AspNET Core get file'
'AspNET Core get all files metadata'
'AspNET Core upload file'

Gen 0	Gen 1	Gen 2	Allocated
-	-	-	231.3 KB
-	-	-	98.68 KB
-	-	-	974.01 KB
-	-	-	60.03 KB
-	-	-	78.96 KB
-	-	-	275.73 KB

Выводы

Плюсы:

- В целом хороший перфоманс, но надо тестить для своих задач.
- Поточковая передача данных
- Contract first подход- штука интересная, но есть альтернативы.

Выводы

Минусы:

- Плохой вариант если надо дергать апи из браузера
- Плохой вариант если хостится в IIS(пока по крайней мере)
- Не самый лучший DX при отладке
- Ряд вещей делается сложнее чем в REST.

Ссылки

1. <https://grpc.io/>
2. <https://performance-dot-grpc-testing.appspot.com/explore?dashboard=5636470266134528>
3. <https://visualrecode.com/blog/wcf-vs-grpc-round-2>
4. <https://github.com/grpc/grpc-dotnet>
5. <https://github.com/grpc-ecosystem/awesome-grpc>
6. <https://kubernetes.io/blog/2018/11/07/grpc-load-balancing-on-kubernetes-without-tears/>
7. <https://dev.to/thangchung/performance-benchmark-grpc-vs-rest-in-net-core-3-preview-8-45ak>
8. <https://medium.com/@EmperorRXF/evaluating-performance-of-rest-vs-grpc-1b8bdf0b22da>
9. <https://labs.criteo.com/2017/05/serialization/>
10. <https://github.com/grpc/grpc/issues/13586>

Контакты

telegram @tezjarl

facebook <https://www.facebook.com/tezjarl>

Ссылки и контакты

