Alla scoperta di gRPC

Codice: https://github.com/andreadottor/XE.Dottor.gRPC

Andrea Dottor

@dottor

Microsoft® MVP Most Valuable Professional

Da dove arriviamo

SOAP

- PRO
  - WSDL (permette la generazione di proxy)
  - Standard W3C
- CONTRO
  - Serializzazione XML
  - Formato messaggi troppo complesso
  - Non facilmente utilizzabile da browser

Da dove arriviamo

SOAP

- PRO
  - WSDL (permette la generazione di proxy)
  - Standard W3C
- CONTRO
  - Serializzazione XML
  - Formato messaggi troppo complesso
  - Non facilmente utilizzabile da browser

WCF

- PRO
  - Altamente configurabile
- CONTRO
  - Altamente configurabile
  - Non sempre interoperabile

Da dove arriviamo

SOAP

- PRO
  - WSDL (permette la generazione di proxy)
  - Standard W3C
- CONTRO
  - Serializzazione XML
  - Formato messaggi troppo complesso
  - Non facilmente utilizzabile da browser

WCF

- PRO
  - Altamente configurabile
- CONTRO
  - Altamente configurabile
  - Non sempre interoperabile

http API

- PRO
  - Interoperabile
  - Utilizzabile facilmente dal browser
  - Aderente al protocollo HTTP
- CONTRO
  - Non è possibile far generare un proxy *
  - Lo sviluppatore deve documentare DTO e chiamate
  - Non è uno Standard

Possiamo ancora migliorare?

...un po' di storia

RPC

In distributed computing a remote procedure call (RPC) is when a computer program causes a procedure (subroutine) to execute in **another address space** (commonly on another computer on a shared network), which is **coded as if it were a normal (local) procedure call**, without the programmer explicitly coding the details for the remote interaction.

That is, the programmer writes essentially **the same code** whether the subroutine is **local** to the executing program, or **remote**

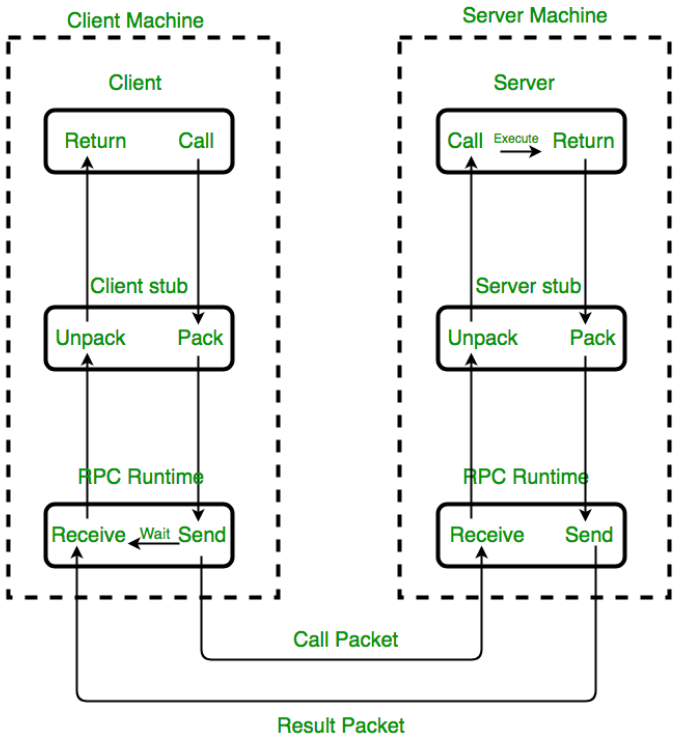
[fonte: https://en.wikipedia.org/wiki/Remote_procedure_call]

RPC

RPC non è una cosa nuova!

- Response–request protocols date to early distributed computing in the late 1960s, theoretical proposals of remote procedure calls as the model of network operations date to the 1970s, and practical implementations date to the early 1980s. Bruce Jay Nelson is generally credited with coining the term "remote procedure call" in 1981

[fonte: https://en.wikipedia.org/wiki/Remote_procedure_call]



Implementation of RPC mechanism


gRPC

Cos'è gRPC

A **high performance**, open-source universal RPC framework

gRPC is a modern open source **high performance** RPC framework that can run in **any environment**. It can efficiently **connect services** in and across data centers with pluggable support for load balancing, tracing, health checking and authentication. It is also applicable in last mile of distributed computing to **connect devices, mobile applications and browsers to backend services**.

https://grpc.io/

Un po' di storia di gRPC

gRPC è l'evoluzione di **Stubby**, una tecnologia sviluppata da Google per far comunicare i suoi microservizi (anche tra data-center differenti)

- "For the past 15 years, Google has solved these problems internally with Stubby, an RPC framework that consists of a core RPC layer that can handle internet-scale of tens of billions of requests per second (yes, billions!). Now, this technology is available for anyone as part of the open-source project called gRPC. It's intended to provide the same scalability, performance and functionality that we enjoy at Google to the community at large."
  [fonte: https://medium.com/@nakabonne/why-was-grpc-born-7e65bbca9633]

Progetto ora gestito da **Cloud Native Computing Foundation**

high performance

Perché?

HTTP/2 transport

Uso di HTTP/2 per la connessione

Vantaggi:

- binario (invece che testuale)
- utilizzo di una singola connessione tra client e server e abilitazione del multiplexing per il download parallelo delle risorse
- compressione degli header HTTP
- stessa semantica HTML di HTTP 1.1

[Test performance HTTP/1.1 vs HTTP/2: http://www.http2demo.io]

Protocol Buffers

Uso di Protocol Buffers per la definizione dei servizi e come formato di serializzazione dei dati

- "Protocol buffers are a **language-neutral**, **platform-neutral** extensible mechanism for **serializing structured data"**

Sviluppato da Google https://developers.google.com/protocol-buffers/

Specifiche del linguaggio: https://developers.google.com/protocol-buffers/docs/proto

PRO:

- Serializzazione dei dati in formato binario
- Da 3 a 10 volte più piccolo di un XML
- Da 20 a 100 volte più veloce di un serializzatore o parser XML

Protocol Buffers - proto file

Permette di definire:

- **metodi** esposti dai servizi gRPC
- struttura dei **messaggi** utilizzati nei servizi

```
message Person {
  string name = 1;
  int32 id = 2;  // Unique ID number for this person.
  string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    string number = 1;
    PhoneType type = 2;
  }

  repeated PhoneNumber phones = 4;

  google.protobuf.Timestamp last_updated = 5;
}

// Our address book file is just one of these.
message AddressBook {
  repeated Person people = 1;
}
```

Protocol Buffers - Language Guide

- Specifying Field Rules
  - singular -> valore di default quando non viene specificato nulla
  - repeated
- Scalar Value Types:
  - double, float, int32, int64, *int64, uint32, uint64, sint32, sint64, fixed32, fixed64, sfixed32, sfixed64, bool, string, bytes
- In proto3 non sono ammessi campi opzionali
  - I **null** nei Value Types vanno gestiti manualmente es: Google.Protobuf.WellKnownTypes

any environment

Perché?

Code Generation

Language	Platform	Compiler
C/C++	Linux/Mac	GCC 4.8+ Clang 3.3+
C/C++	Windows 7+	Visual Studio 2015+
C#	Linux/Mac	.NET Core, Mono 4+
C#	Windows 7+	.NET Core, .NET 4.5+
Dart	Windows/Linux/Mac	Dart 2.0+
Go	Windows/Linux/Mac	Go 1.6+
Java	Windows/Linux/Mac	JDK 8 recommended. Gingerbread+ for Android
Node.js	Windows/Linux/Mac	Node v4+
Objective-C	Mac OS X 10.11+/iOS 7.0+	Xcode 7.2+
PHP (Beta)	Linux/Mac	PHP 5.5+ and PHP 7.0+
Python	Windows/Linux/Mac	Python 2.7 and Python 3.4+
Ruby	Windows/Linux/Mac	Ruby 2.3+

C# Tooling support for .proto files

- Il package NuGet **Grpc.Tools** contiene il plugin protoc e protobuf in C# utile a generare il codice (server e client). A partire dalla versione 1.17 il package si integra con MSBuild e permette di generare automaticamente il codice C# a partire dai file .proto.

- La build **build** rigenera I seguenti files nella cartella **Greeter/obj/Debug/TARGET_FRAMEWORK**:
  - **Greet.cs** contiene tutto il codice di Procol Buffer utile a popolare, serializzare e tutte le classi di request e response
  - **GreetGrpc.cs** contiene il codice generato del client e/o server, ed include:
    - una classe astratta **Greeter.GreeterBase** da cui ereditare per implementare il servizio Greeter
    - una classe **Greeter.GreeterClient** utile per chiamare I metodi esposti dal servizio remoto Greeter

gRPC services with ASP.NET Core

- Installare il package nuget **Grpc.AspNetCore**
- Abilitare gRPC con **AddGrpc**
- Aggiungere ogni servizio gRPC alla pipeline di routing con **MapGrpcService**

```
public class Startup
{
    // This method gets called by the runtime. Use this method to add services to the container.
    // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddGrpc();
    }

    // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }

        app.UseRouting();

        app.UseEndpoints(endpoints =>
        {
            // Communication with gRPC endpoints must be made through a gRPC client.
            // To learn how to create a client, visit: https://go.microsoft.com/fwlink/?linkid=2086909
            endpoints.MapGrpcService<GreeterService>();
        });
    }
}
```

DEMO

gRPC - Simple Service

gRPC Service Definitions

- **Unary**: Quando il client invia una singola richiesta ed il server risponde, come fosse una normale chiamata ad una funzione
- **Server streaming**: Il client invia una richiesta al server e ottiene uno stream per leggere in sequenza i messaggi ritornati. Il client continua la lettura fino a che non ci son più messaggi.
- **Client streaming**: Il client invia una sequenza di messaggi al server utilizzando uno stream. Quando il client completa l'invio, attende che il server ritorni una risposta.
- **BiDi streaming**: Sia il server che il client inviano e leggono i messaggi utilizzando due stream indipendenti. L'ordine dei messaggi in ogni stream viene mantenuto.

DEMO

gRPC - Complex Service

Authentication and authorization

- Con ASP.NET Core sono suppertati i seguenti metodi di autenticazione:
  - Azure Active Directory
  - Client Certificate
  - IdentityServer
  - JWT Token
  - OAuth 2.0
  - OpenID Connect
  - WS-Federation
- Decorare il servizio con l'attributo [Authorize]
  - Valido sia a livello di classe che di singolo metodo

gRPC vs HTTP APIs (REST)

gRPC	**HTTP API**
Strong Typing	Serialization
Streaming	Request-Response
HTTP/2	HTTP/1.1
Protobuf	JSON

"gRPC largely follows HTTP semantics over HTTP/2 but we explicitly allow for full-duplex streaming. We diverge from typical REST conventions as we use static paths for performance reasons during call dispatch as parsing call parameters from paths, query parameters and payload body adds latency and complexity. We have also formalized a set of errors that we believe are more directly applicable to API use cases than the HTTP status codes."

gRPC vs WCF

In .NET core non sarà possible realizzare servizi Windows Communication Foundation (WCF), e non verrà fatto il porting nemmeno in .NET 5

- https://devblogs.microsoft.com/dotnet/supporting-the-community-with-wf-and-wcf-oss-projects/

In .NET Core è stato portato solamente la possibilità di consumare servizi WCF

- https://github.com/dotnet/wcf

gRPC vs SignalR

I servizi gRPC non possono essere sfruttati completamente da browser ** gRPC ad oggi non verrà utilizzato da SignalR

Esistono scenari per cui SingalR è da preferire

- ***Broadcast real-time communication*** *– gRPC supports real-time communication via streaming, but the concept of broadcasting a message out to registered connections doesn't exist. For example in a chat room scenario where new chat messages should be sent to all clients in the chat room, each gRPC call is required to individually stream new chat messages to the client. SignalR is a useful framework for this scenario. SignalR has the concept of persistent connections and built-in support for broadcasting messages.*

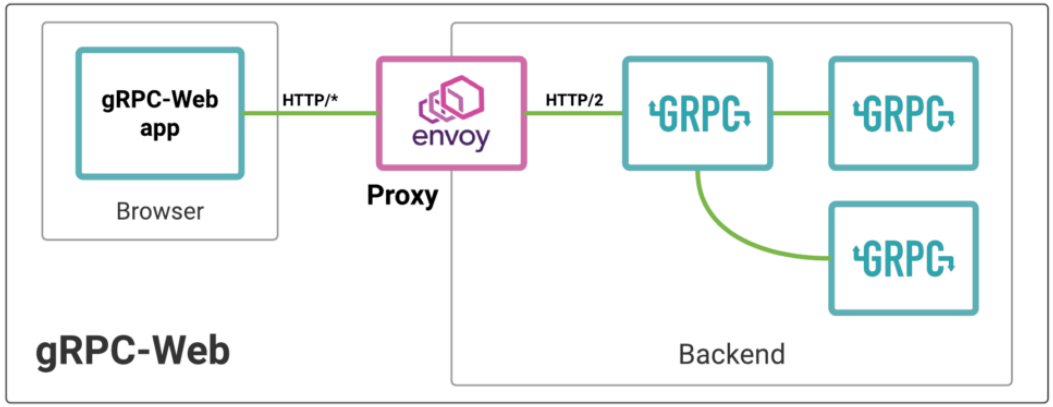
gRPC-Web

gRPC-Web è una libreria JavaScript che permette l'interazione con servizi gRPC

https://github.com/grpc/grpc-web

Richiede l'uso di un proxy che adatti le chiamate al servizio

https://www.envoyproxy.io/



Quali sono i casi d'uso ideali?

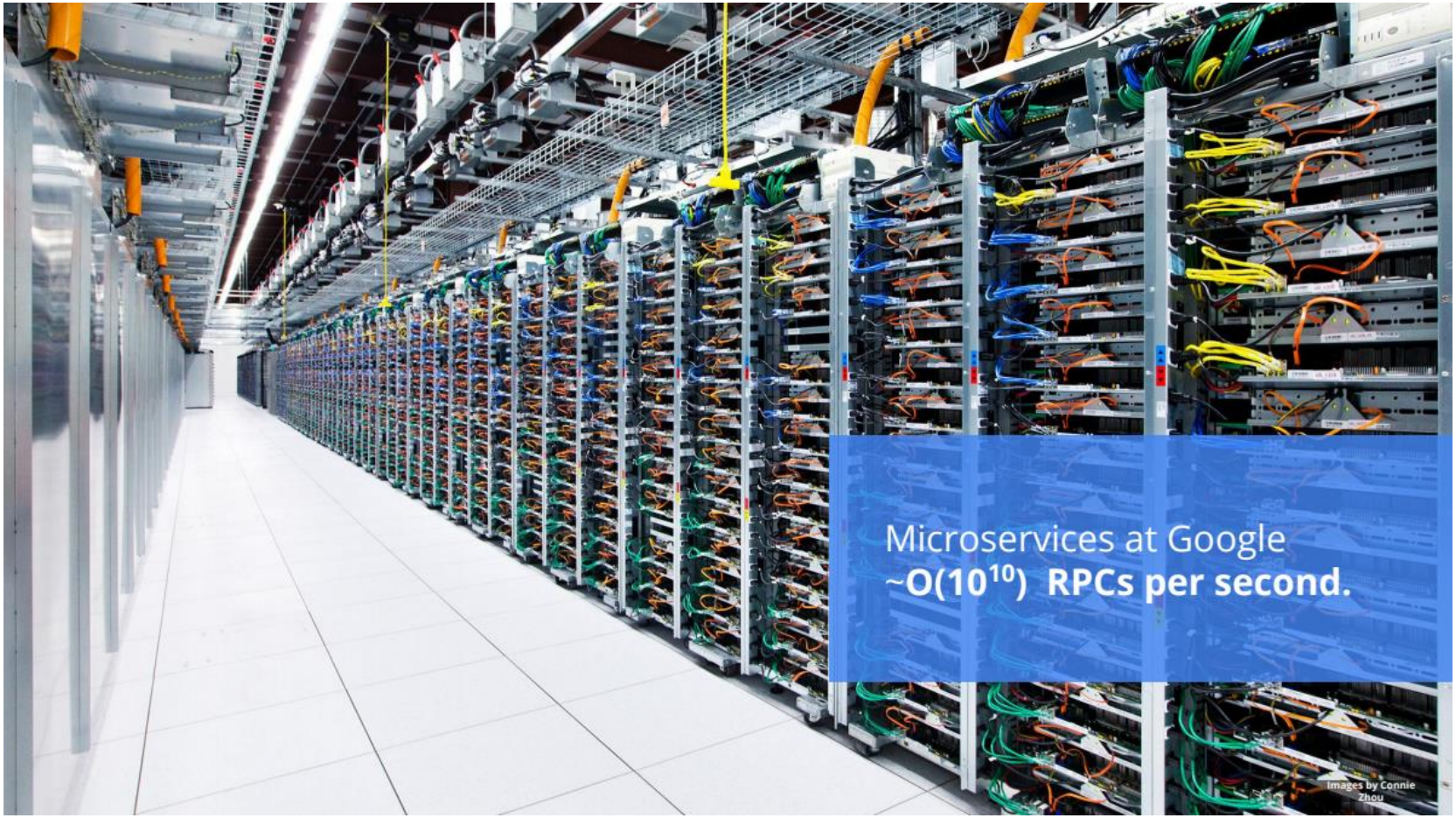
Lightweight **microservices** where efficiency is critical.

Polyglot systems where **multiple languages** are required for development.

Point-to-point real-time services that need to handle **streaming** requests or responses.

Mobile clients which are communicating to a cloud server.

Numeri?



Building Scalable Microservices using Kubernetes, gRPC, and Containers
https://www.youtube.com/watch?v=UOIJNygDNlE

Alcune aziende che hanno migrato i loro servizi

Square

- gRPC — cross-platform open source RPC over HTTP/2
  https://medium.com/square-corner-blog/grpc-cross-platform-open-source-rpc-over-http-2-56c03b5a0173

Netflix

- Increasing Developer Productivity and Defeating the Thundering Herd with gRPC
  https://www.cncf.io/netflix-case-study/

Spotify

- The Story of Why We Migrate to gRPC and How We Go About It
  https://www.youtube.com/watch?v=fMq3IpPE3TU

Twilio

- Intro to gRPC: A Modern Toolkit for Microservice Communication
  https://www.youtube.com/watch?v=RoXT_Rkg8LA

WePay

- Migrating APIs from REST to gRPC at WePay
  https://wecode.wepay.com/posts/migrating-apis-from-rest-to-grpc-at-wepay

Docker

- Containerd: a daemon to control runC
  https://blog.docker.com/2015/12/containerd-daemon-to-control-runc/

Cisco

- Getting Started With gRPC in Cisco IOS XR
  https://github.com/CiscoDevNet/grpc-getting-started

…

links

- gRPC and why it can save you development time
  https://medium.com/red-crane/grpc-and-why-it-can-save-you-development-time-436168fd0cbc
- ASP.NET Community Standup - June 4th, 2019 - gRPC!
  https://www.youtube.com/watch?v=Pa6qtu1wIs8
- Calling gRPC Services With Server-side Blazor
  https://chrissainty.com/calling-grpc-services-with-server-side-blazor/
- gRPC for ASP.NET Core 3.0
  https://mustak.im/grpc-for-asp-net-core-3-0/
- From ’00s to ’20s: migrating from RESTful to gRPC
  https://medium.com/@giefferre/from-00s-to-20s-migrating-from-restful-to-grpc-5a5aa0cf27ba
- Envoy and gRPC-Web: a fresh new alternative to REST
  https://blog.envoyproxy.io/envoy-and-grpc-web-a-fresh-new-alternative-to-rest-6504ce7eb880

Contatti

Andrea Dottor

Microsoft MVP Developer Technologies

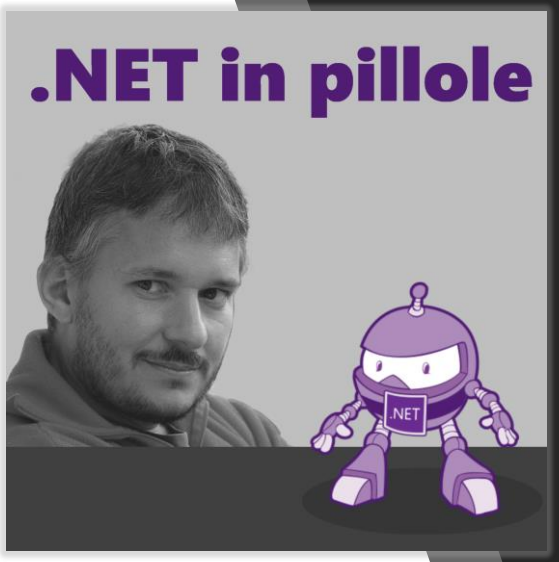
www.dottor.net

andrea@dottor.net

@dottor

Microsoft® Most Valuable Professional

Podcast ".NET in Pillole"



- Spotify:
  https://open.spotify.com/show/7jyoG6BBmzvScNOqSpVvQQ
- Spreaker:
  https://www.spreaker.com/show/net-in-pillole
- Apple podcast:
  https://podcasts.apple.com/it/podcast/net-in-pillole/id1478648398
- Google podcast:
  https://podcasts.google.com/?feed=aHR0cHM6Ly93d3cuc3ByZWFrZXIuY29tL3Nob3cvMzY4NTM0NC9lcGlzb2Rlcy9mZWVk
- YouTube:
  https://www.youtube.com/user/andreadottor/