Calling Services Using Client-side Load Balancing

Dustin Schultz
SOFTWARE ENGINEER

@schultzdustin http://dustin.schultz.io/ dustin@schultz.io

Outline

Load balancing

- Server-side
- Client-side

Netflix Ribbon

- With & without service discovery
  - @Loadbalanced
  - @RibbonClient
- Custom Ribbon configuration

What is load balancing?

Load Balancing

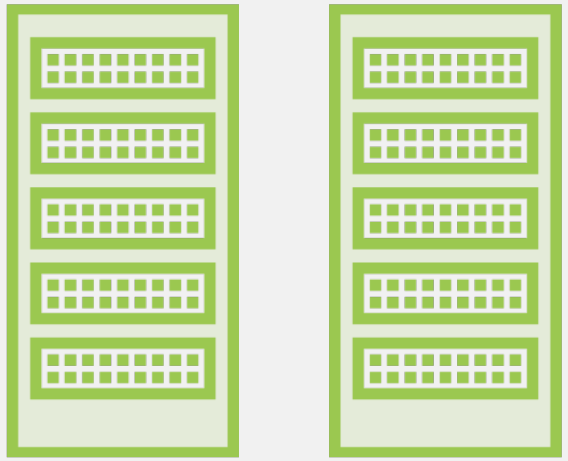
… improves the distribution of workloads across multiple computing resources …

- *Wikipedia*

What is the role of load balancing in a cloud-native architecture?

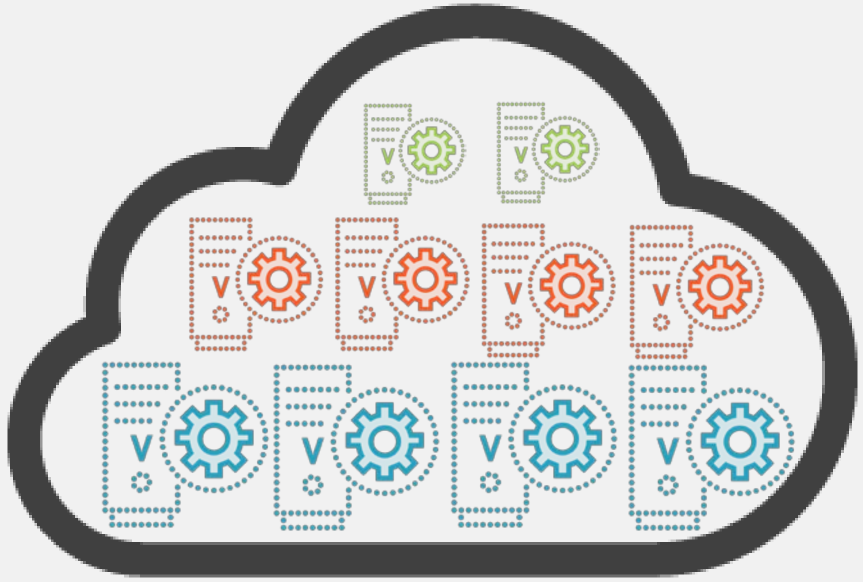
A very important one, actually!

Multiple Services & Multiple Instances



From multiple instances with a single load balancer

To …



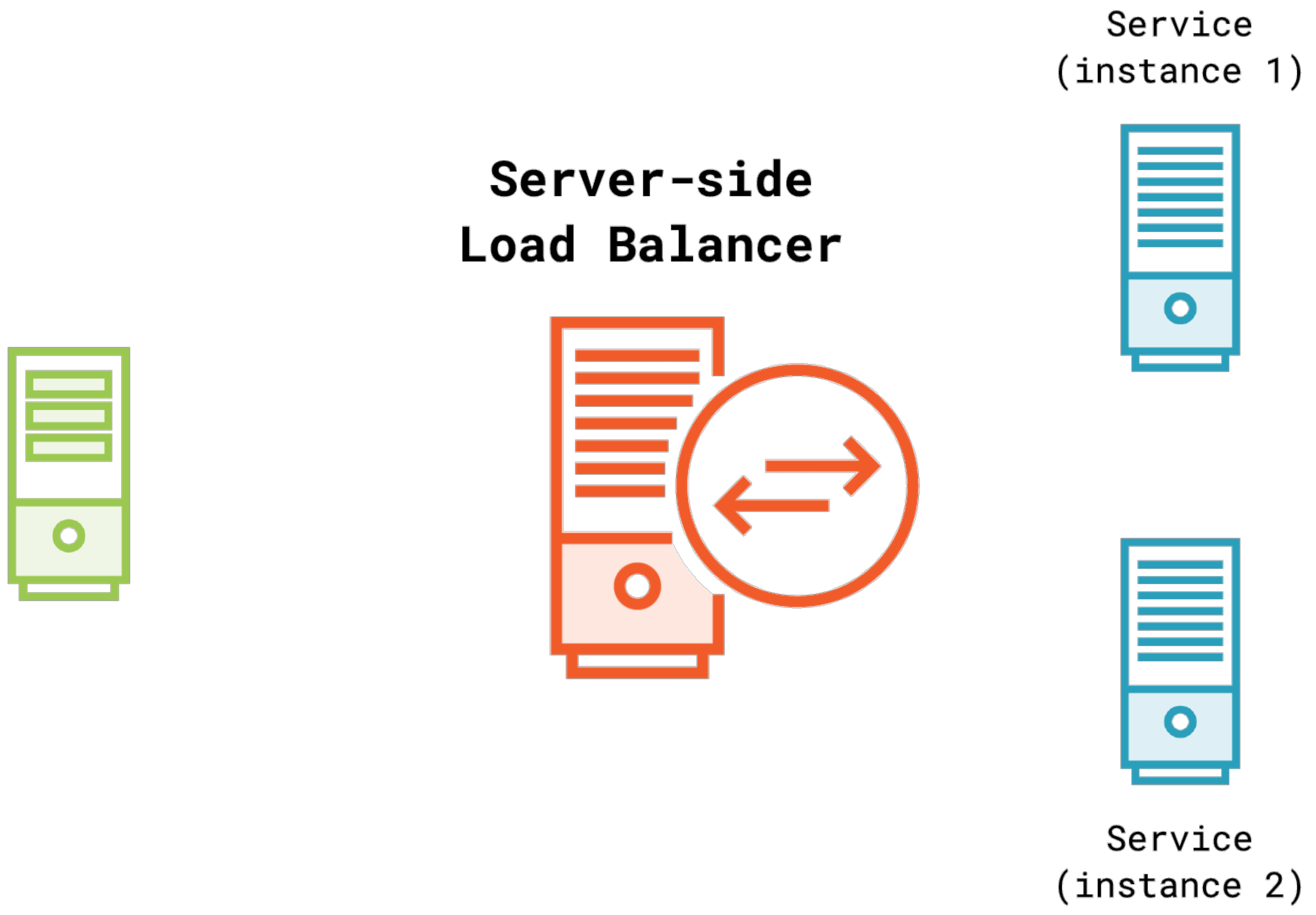
Multiple services & multiple instances with multiple load balancers

Different Types of Load Balancing

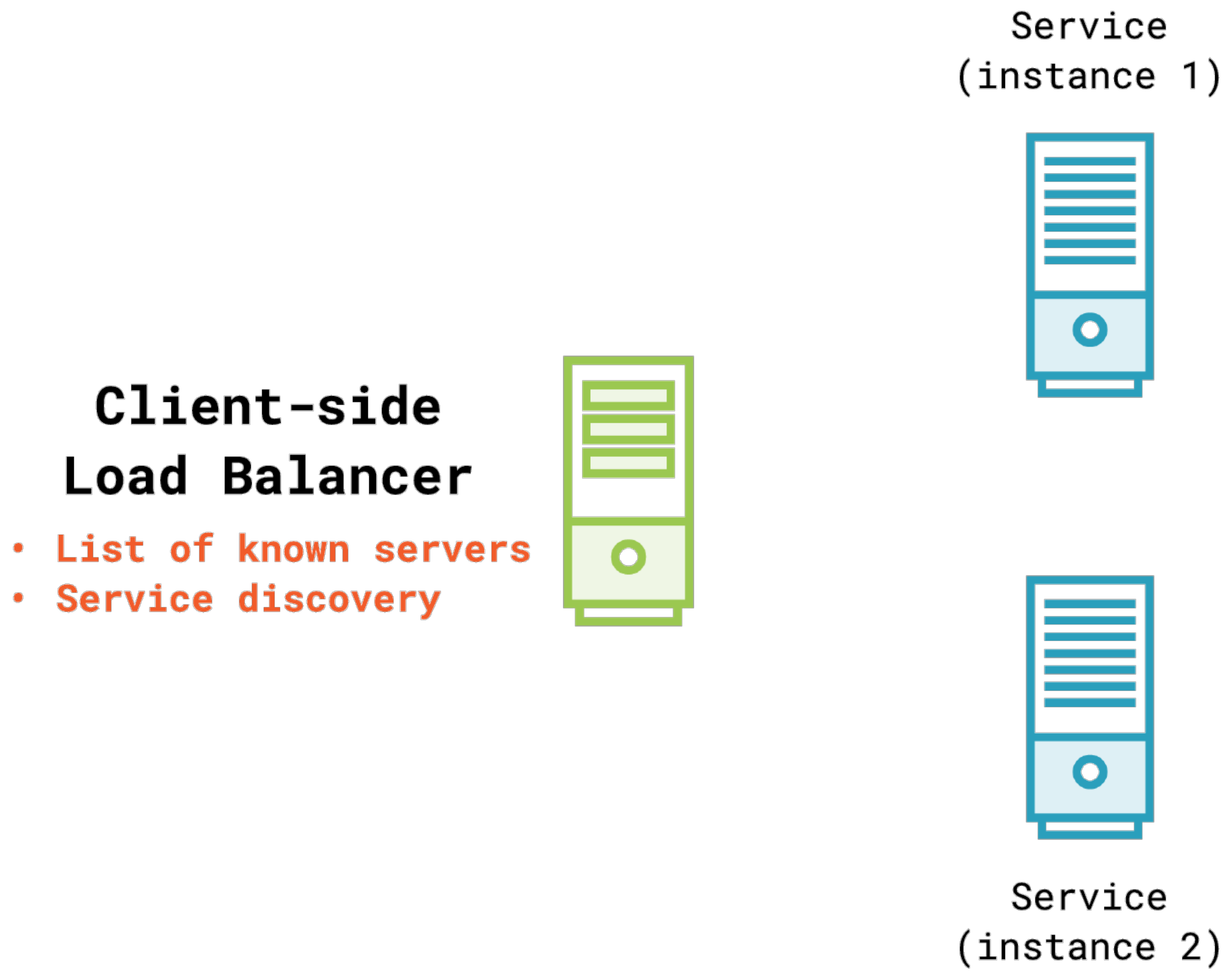
Server-side

Client-side (caller)

Server-side Load Balancing



Client-side Load Balancing



- List of known servers
- Service discovery

Server-side vs Client-side

Server-side	Client-side
Server distributes requests	Client distributes request
Hardware or software based	Software based
Extra hop	No extra hops
Various balancing algorithms support	Various balancing algorithms support
Occurs outside of the request process	Occurs within the request process
Centralized or distributed	Typically distributed

Client-side load balancing is a *natural fit* for cloud native architectures.

Client-side Load Balancing with Spring Cloud

Netflix Ribbon

Ribbon is a Inter Process Communication (remote procedure calls) library with *built in software load balancers.*

-Netflix Ribbon Project page

Spring Cloud
+
Netflix Ribbon

Full integration with Spring's `RestTemplate`

Customize configuration for different

- Balancing algorithms
- Availability checks

Using Spring Cloud & Netflix Ribbon

pom.xml

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>Camden.SR2</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
```

Using Spring Cloud & Netflix Ribbon

pom.xml

```xml
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-ribbon</artifactId>
</dependency>
```

Two New Annotations

`@LoadBalanced`

Marks a `RestTemplate` to support load balancing

`@RibbonClient`

Used for custom configuration and when Service Discovery is absent

Creating a Load Balanced **RestTemplate**

MyConfiguration.java

```java
@Configuration
public class MyConfiguration {

    @Bean
    @LoadBalanced
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
}
```

Using a Load Balanced `RestTemplate` With Service Discovery

Suppose …

`my-service` **is the name of a service running on port 9000 at mycompany.com and is discoverable via Service Discovery. There are 2 instances running.**

Instead of …

```
restTemplate.getForEntity("http://mycompany.com:9000/u/1", ...)
```

or

```
restTemplate.getForEntity("http://128.168.10.10:9000/u/1", ...)
```

Use `RestTemplate` **like this instead …**

```
restTemplate.getForEntity("http://my-service/u/1", ...)
```

Demo

Using Ribbon with @LoadBalanced and Service Discovery

Two New Annotations

`@LoadBalanced`

Marks a `RestTemplate` to support load balancing

`@RibbonClient`

Used for custom configuration and when Service Discovery is absent

Using a **@LoadBalanced RestTemplate** Without Service Discovery

MyConfiguration.java

```
@Configuration
@RibbonClient(name = "someservice")
public class MyConfiguration {
    ...
}
```

Using a `@LoadBalanced RestTemplate` Without Service Discovery

application.properties

```
<ribbon_client_name>.ribbon.eureka.enabled=false
<ribbon_client_name>.ribbon.listOfServers=http://host:9000, http://host:9001
```

OR

application.yml

```
<ribbon_client_name>:
  ribbon:
    eureka:
      enabled: false
    listOfServers=http://host:9000, http://host:9001
```

* ***Replace*** *`<ribbon_client_name>`* ***with the*** *`name`* ***field value of*** *`@RibbonClient`*

```
restTemplate
   .getForEntity("http://someservice/", ...)
```

Using a `@LoadBalanced RestTemplate` Without Service Discovery

Demo

Using Ribbon without service discovery

Custom `RibbonClient` Configuration

Custom Configuration of Ribbon Clients

MyConfiguration.java

```java
package io.schultz.dustin;

@Configuration
@RibbonClient(
    name = "otherservice",
    configuration = OtherServiceConfig.class)
public class MyConfiguration {
   ...
}
```

```java
package io.schultz.config.dustin;


@Configuration
public class OtherServiceConfig {

    ...

}
```

- Different package so it is not picked up by `@ComponentScan`
- Standard @Configuration class
- Define `@Beans` for customization

```
@Configuration
public class OtherServiceConfig {

    @Bean
    public <bean_type> <method_name>() {
        ...
    }

}
```

Default Ribbon Client @Beans

Replace `<bean_type>` **and** `<method_name>` **with values to override:**
http://cloud.spring.io/spring-cloud-static/Camden.SR6/#_customizing_the_ribbon_client

Most likely to be customized: `IRule` **&** `IPing`

The `IRule` Bean

IRule Implementations

RoundRobinRule

ResponseTime
WeightedRule

RandomRule

ZoneAvoidance
Rule

IRule: Load Balancing Strategy

```
@Configuration
public class OtherServiceConfig {

    @Bean
    public IRule ribbonRule() {
        return new RoundRobinRule();
    }

}
```

The `IPing` Bean

`IPing` Implementations

`DummyPing`

`PingUrl`

`NIWSDiscovery`
`Ping`

IPing: Liveliness Check

```
@Configuration
public class OtherServiceConfig {

    @Bean
    public IPing ribbonPing() {
        PingUrl pingUrl = new PingUrl();
    pingUrl.setExpectedContent("true");
    return pingUrl;
    }
}
```

https://github.com/dustinschultz/scf-discovery-server

Demo

Customizing a RibbonClient

Summary

Differences between client-side & server-side load balancing

Netflix Ribbon

- @LoadBalanced & @RibbonClient
  - With & without service discovery

Custom Ribbon client configuration