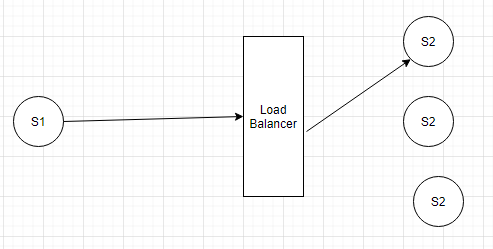
**Microservices**

Microservice can have many definitions.

* It is architectural style or an approach to developing a single application as a suit of small services, each running in its own process and communicating with light weight mechanisms, often HTTP resource API.
* Each service is independently deployable by fully automated deployment tools.
* Each can be written in different programming language. And different persistence can be used. Where as in monolith we have to stick on same language.
* If we compare microservice with monolith application it is easier to understand and developer can be more productive.
* Here we can accomplish parallel development. Where as in Monolith application it may create some issue.
* It can be easily scaled in terms of resources. Where as it is hard to scaled Monolith application.
* Each microservice can be deployable independently, it also improves fault isolation and conflicting resource can be resolved.

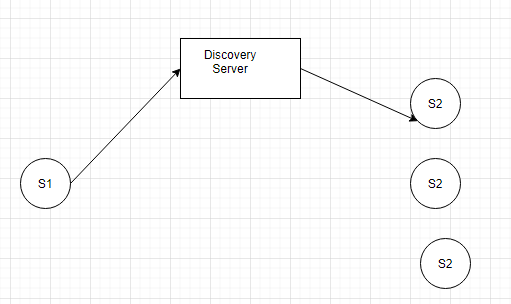
**Challenges and Solution of Microservices:**

1. In microservice generally one service calls another service take as example below.

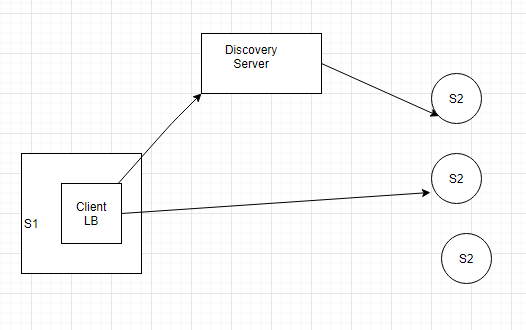


In above example we have two microservice i.e. S1 and S2, S1 is making a call to S2 and based on LB logic instance of S2 is assigned to that call. Here we can see we are having two remote calls. S1 to load balancer and then LB to S2. So, it will affect performance.

Now to over come this issue what microservice architect do that they have introduced another component call discovery server as shown in below diagram.



Discovery server will keep track of all the instance of S2. So, when call coming from S1 it will assign one instance of S2 to serve that request. But still our above problem is not resolve. So now what they do they have introduced another concept called client-side load balancing as shown in below diagram.



Here client side LB download all the registry of S2 instance from of discovery server and when S1 calls S2 based on some logic it will assign S2 instance to full fill the request. But here is another problem is that suppose if after sometime an instance of S2 is down then how client side LB know that S2 instance is down. Also, to download all the registry from discovery server it has to make a call to discovery server.

So, what Microservice architect has done during startup of S1 they have assigned one thread which actually get all the registry from discovery server and also after some seconds based on configuration it continuously synching the registry from discovery server. Similarly, Discovery server also call health check status of each S2 instance after some time interval and based on result it updates its registry and status.

For this purpose, Netflix have created Eureka (Discovery server) and Ribbon (Client side LB). what Netflix developer saw 30 secs is good time to synch and update registry as well as update the registry in Client-side LB.

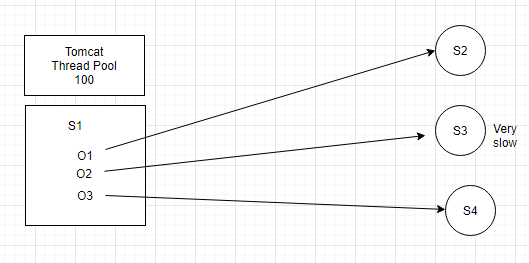
Now suppose during that 30 sec one of instance of S2 went down. Every instance of S2 send heart beat on regular interval (30 sec). Now suppose one of heart beat is missing then what will Eureka do. Should it remove it from registry. No, we can configure that if 3 or more heart beat consecutively missing then only Eureka will remove this server from registry.

Now suppose during this time S1 call that down instance of S2. It got fail so we can configure retry logic, (Ribbon is providing this retry logic). But what should we do if all retry is over then in that case we can implement some fallback logic.

Also take one example if S1 calling one of S2 instance and that instance is busy or taking some time to response. In that case, may S1 can assign weight to that instance and similarly to other instance of S2 and based on this it may use weight round robin to call S2 instance.

1. Bulk-head issue.

Let take an example we have following server S1, S2, S3, S4 etc. And S1 have operation O1, O2 and O3. And S3 service is very slow. We may deploy those service to tomcat or other server and it has some thread pool. Suppose S1 is deployed to tomcat server and its thread pool size is 100. So, what will happen if O2 calls S3 service, due to slowness it might be waiting for response. Similarly, there might be chance that all 100 threads are calling to S3 and due to slowness is waiting for response and our application consume all the thread pool and no thread is available for other operation.

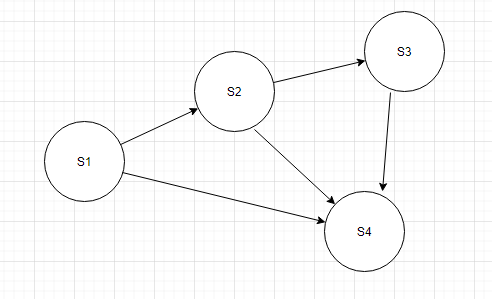


Now take example of bank having toll free no. with 100 limits. Suppose it has many departments like saving account, loan, credit card etc. When caller call the toll, free no. based on it’s choice call connected to different department. Suppose Loan department taking time to describe loan feature and if we don’t have any mechanism then all the 100 limits get used. So, what we can do we can assign some limit to each department like SA can have 30 call max, loan can have 40 calls max and CC can have 30 calls max. So, what will happen in this case if more than 40 callers try to connect to loan department either they will wait or call get disconnected after sometime automatically. By this here we can utilize limits. Means 60 call will be available for another department.

Similarly, in above case of microservice we can limit pool size specific to each service. This pattern is call Bulk head pattern. Means issue in one service could not make entire system down. This concept came from ship companies. We don’t need to implement bulk head pattern Netflix provide one more library called Hystrix which implement bulk head pattern. It also used in circuit breaker pattern.

1. Circuit breaker pattern.

Look below diagram, S1 calling S2, S2 calling S3 and S3 calling S4, S2 calling S4 and S1 calling S4. By the looks of this it looks like a circuit.

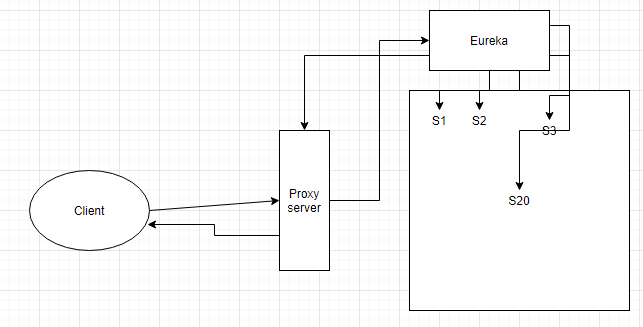


Here suppose due to some reason Service S4 fails, what will happen then after sometime S3 fails then S2 and then complete services.

One thing we can do we can write some fallback logic which will execute once S4 is down. Now suppose S4 is down and S3 is calling S4 1,2… 20 times it got failure and it execute fallback logic. Suppose if we have mechanism which will already identify that S4 is down without wasting time calling S4 it should execute fallback logic. In circuit breaker there are lots of logic which decide whether a circuit is open or close based. E.g. success count, failure count, time out, bucket with data all SC, FC, TC of 100sec etc.

1. Suppose we have 20 services like S1, S2 … S20 all register to Eureka. So, if internal call happens between these services then it will go to eureka and found the service registry and call those service.

Now suppose if any external client wants to use one of your service can you give it all the registry present in Eureka. So instead of giving all the registry present in Eureka we will provide a proxy server. So now client will call proxy server and proxy server will go to eureka get the registry and based on service call return data. Zuul is one of the libraries which is used as API gateway. It also implements hystrix. Suppose I want to secure my all service we will implement at zuul, similarly we can also configure process like pre-processing or post processing of any request. Also, we can write routing logic also if client wants to use only some of the services.



1. In our microservice there can be many services, so ideas are keeping all the configuration inside the central repo. And we should have config server. So, whenever a service starts then it calls config server and it will get from central repo. For this we can use spring cloud config server