Distributed System - Service Discovery & Load Balancing

Anushree Desai and Divyesh Chitroda

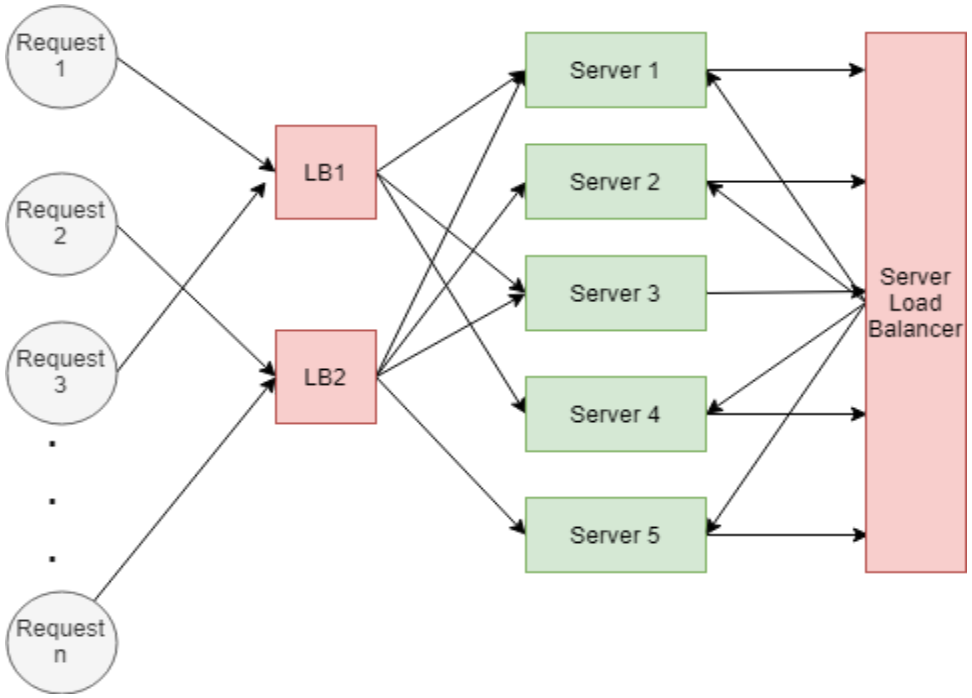
1. Overview

Implement a distributed system to communicate with web services. There is a many-to-many relationship between collection of **web services** and **sites** hosting those services. We have to build an architecture to efficiently locate and execute a web service majorly using two concepts: service discovery and load-balancing.

2. Abbreviations and notations

LB - Load Balancer
SLB - Server Load Balancer
S# - Server #
"|" is a API call chaining and not OR operator

3. System Architecture and Design



3.1. **LB1** and **LB2** are load balancers used to handle multiple requests from client to web services. They are hosted on domain name as services.maths.com on different machines and will have different IP address.

3.2. Each web service may reside on one or more than one servers. Thus, we need a load to pick the most efficient web service through LB1 and LB2.

3.3. When a service request is made using API endpoint services.maths.com/math/add?x=5&y=7, DNS resolves the host to one of the LBs in a round-robin manner.

3.4. Now the LBs will forward the *add* request to the server that hosts *add* method. The *add* web service is hosted on multiple machines. The machine to which the *add* request is forwarded is chosen by LB on the basis that the machine which is the least loaded site and is below an average threshold of the load limit.

3.5. For multiplication operation, users call services.maths.com/math/mul?x=5&y=7, which in turn calls *add* and aggregates the result and sends it with the response.

3.6. **Service Load Balancer (SLB)** is used to handle multiple requests from one service to another (basically for request forwarding)

3.7. Note that *add* is also called by the other clients and not just by other services, which makes the services add load heavily if we always forward the the requests made by servers to one static site hosting *add*. To overcome this, we have added another load balancer SLB, that balances the requests for sites that are requested by other servers.

4. Functional Requirements

4.1. Web service interface

4.1.1. Arithmetic Operations

4.1.1.1. Addition

Input: 2 integers (x & y)
Output: ans
Endpoint: arith/add?x=6&y=-7
Processing: ans = x + y

4.1.1.2. Subtraction

Input: 2 integers (x & y)
Output: ans
Endpoint: arith/sub?x=7&y=3
Processing: ans = x - y

4.1.1.3. Multiplication

Input: 2 integers (x & y)
Output: ans
Endpoint: arith/mul?x=7&y=2
Processing: ans = arith/add?x=7&y=1 | arith/add?x=7&y=1 | ... (y times)

4.1.1.4. Division

Input: 2 integers (x & y)
Output: ans
Endpoint: arith/div?x=5&y=2
Processing: ans = arith/sub?x=5&y=1 | arith/sub?x=5&y=1 | ... (y times)

4.2. Load balancer

4.2.1. On each load balancer machine, we maintain a min-heap for each service to store its endpoint and its current load as the key value.

4.2.2. For all services k_i, we have Q_i, where $k \in S_i$. Q is the min-heap queue, S is the server and k is the web service.

4.2.3. Each item in Q is a tuple (c,s), where c is the current load/cost on server s.

4.2.4. When we receive a request, we lookup the service request to access its list of servers available. Then we do a Extract-Min operation on the Q to get the least loaded site and forward the request to that server's endpoint.

4.2.5. We reinsert that server's endpoint into the Q by appending the new value of the load to its key value.

4.2.6. When the server responds with the reply for the request placed, we call a Decrease-Key on Q for that server and decrease its key value to reflect the current load which is now have been reduced.

5. Assumptions

- Load-Balancing is done only for web-services
- All client applications are hosted locally
- All web services have equal priorities and all servers have equal load capacities
- For phase1, we assume that we don't face the problem of discovering the services
- Notional load is used for each service instead of actual load

6. Timeline

6.1. Initial Report - **11/08**

6.2. Server Applications - 11/14

6.3. Load Balancing, Report 2- **11/21**

6.4. Service Discovery - 12/28

6.5. Load Testing & Simulations - 12/05

6.6. Final Report - **12/12**