IMPLIMENTATION OF WEIGHTED ROUND ROBIN LOAD BALANCING ALGORITHEM FOR GRPC

DR.CH. V. PHANI KRISHNA
Professor
Department of CSE
phanik16@gmail.com
Teegala Krishna Reddy Engineering College, Hyderabad

A. LAVANYA
Department of CSE
Anumula.lavanya1936@gmail.com
Teegala Krishna Reddy Engineering College, Hyderabad
AKASH DOOSA

R. RANJITH KUMAR
Department of CSE
ranjithnnani1299@gmail.com
Teegala Krishna Reddy Engineering College, Hyderabad

P. RAHULL
Department of CSE
p.rahull1706@gmail.com
Teegala Krishna Reddy Engineering College, Hyderabad

Abstract:

GRPC is a modern open-source high performance Remote Procedure Call (RPC) framework that can run in any environment. It can efficiently connect services in and across data canters 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. A load balancer is a device that acts as a reverse proxy and distributes network or application traffic across a number of servers. Load balancers are used to increase capacity (concurrent users) and reliability of applications. They improve the overall performance of applications by decreasing the burden on servers associated with managing and maintaining application and network sessions, as well as by performing application-specific tasks. In this project we intend to develop a simple load balancer based on java that works efficiently with GRPC using weighted round robin algorithm.

I. INTRODUCTION

GRPC is a modern open-source high performance Remote Procedure Call (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. A load balancer is a device that acts as a reverse proxy and distributes network or application traffic across a number of servers. Load balancers are used to increase capacity (concurrent users) and reliability of applications. They improve the overall performance of applications by decreasing the burden on servers associated with managing and maintaining application and network sessions, as well as by performing application-specific tasks. In this project we intend to develop a simple load balancer based on java that works efficiently with GRPC using weighted round robin algorithm.

A load balancer is a device that acts as a reverse proxy and distributes network or application traffic across a number of servers. Load balancers are used to increase capacity (concurrent users) and reliability of applications. They improve the overall performance of applications by decreasing the burden on servers associated with managing and maintaining application and network sessions, as well as by performing application-specific tasks.

Load balancers are generally grouped into two categories: Layer 4 and Layer 7. Layer 4 load balancers act upon data found in network and transport layer protocols (IP, TCP, FTP, UDP). Layer 7 load balancers distribute requests based upon data found in application layer protocols such as HTTP.

Requests are received by both types of load balancers and they are distributed to a particular server based on a configured algorithm. Some industry standard algorithms are:

- Round robin
- Weighted round robin
- Least connections
- Least response time

Layer 7 load balancers can further distribute requests based on application specific data such as HTTP headers, cookies, or data within the application message itself, such as the value of a specific parameter.

Load balancers ensure reliability and availability by monitoring the "health" of applications and only sending requests to servers and applications that can respond in a timely manner.

II Literature survey:

Java is a set of computer software and specifications developed by James Gosling at Sun Microsystems, which was later acquired by the Oracle Corporation, that provides a system for developing application software and deploying it in a cross-platform computing environment. Java is used in a wide variety of computing platforms from embedded devices and mobile phones to enterprise servers and supercomputers. Java applets, which are less common than standalone Java applications, were commonly run in secure, sandboxed environments to provide many features of native applications through being embedded in HTML pages.

Writing in the Java programming language is the primary way to produce code that will be deployed as byte code in a Java virtual machine (JVM); byte code compilers are also available for other languages, including Ada, JavaScript, Python, and Ruby. In addition, several languages have been designed to run natively on the JVM, including Clojure, Groovy, and Scala. Java syntax borrows heavily from C and C++, but object-oriented features are modeled after Smalltalk and Objective-C. Java eschews certain low-level constructs such as pointers and has a very simple memory model where objects are allocated on the heap (while some implementations e.g., all currently supported by Oracle, may use escape analysis optimization to allocate on the stack instead) and all variables of object types are references. Memory management is handled through integrated automatic garbage collection performed by the JVM.

On November 13, 2006, Sun Microsystems made the bulk of its implementation of Java available under the GNU General Public License (GPL).

The latest version is Java 18, released in March 2022 while Java 17, the latest long-term support (LTS), was released in September 2021. As an open-source platform, Java has many distributors, including Amazon, IBM, Azul Systems, and Adopt OpenJDK. Distributions include Amazon Cornetto, Zulu, Adopt OpenJDK, and Liberia. Regarding Oracle, it distributes Java 8, and also makes available e.g., Java 11, both also currently supported LTS versions. Oracle (and others) "highly recommend that you uninstall older versions of Java" than Java 8, because of serious risks due to unresolved security issues. Since Java 9 is no longer supported, Oracle advises its users to "immediately transition" to a supported version. Oracle released the last free-for-commercial-use public update for the legacy Java 8 LTS in January 2019, and will continue to support Java 8 with public updates for personal use indefinitely. Oracle extended support for Java 6 ended in December 2018.

Remote procedure calls

Remote Procedure Call is a software communication protocol that one program can use to request a service from a program located in another computer on a network without having to understand the network's details. RPC is used to call other processes on the remote systems like a local system. A procedure call is also sometimes known as a function call or a subroutine call.

RPC uses the client-server model. The requesting program is a client, and the service-providing program is the server. Like a local procedure call, an RPC is a synchronous operation requiring the requesting program to be suspended until the results of the remote procedure are returned. However, the use of lightweight processes or threads that share the same address space enables multiple RPCs to be performed concurrently.

The interface definition language (IDL) -- the specification language used to describe a software component's application programming interface (API) -- is commonly used in Remote Procedure Call software. In this case, IDL provides a bridge between the machines at either end of the link that may be using different operating systems (OSes) and computer languages.

When program statements that use the RPC framework are compiled into an executable program, a stub is included in the compiled code that acts as the representative of the remote procedure code. When the program is run and the procedure call is issued, the stub receives the request and forwards it to a client runtime program in the local computer. The first time the client stub is invoked, it contacts a name server to determine the transport address where the server resides.

The client runtime program has the knowledge of how to address the remote computer and server application and sends the message across the network that requests the remote procedure. Similarly, the server includes a runtime program and stub that interface with the remote procedure itself. Response-request protocols are returned the same way.

When a remote procedure call is invoked, the calling environment is suspended, the procedure parameters are transferred across the network to the environment where the procedure is to execute, and the procedure is then executed in that environment.

When the procedure finishes, the results are transferred back to the calling environment, where execution resumes as if returning from a regular procedure call.

During an RPC, the following steps take place:

- The client calls the client stub. The call is a local procedure call with parameters pushed onto the stack in the normal way.
- The client stub packs the procedure parameters into a message and makes a system call to send the message. The packing of the procedure parameters is called marshalling.
- The client's local OS sends the message from the client machine to the remote server machine.
- The server OS passes the incoming packets to the server stub.
- The server stub unpacks the parameters -- called unmarshalling -- from the message.

When the server procedure is finished, it returns to the server stub, which marshals the return values into a message. The server stub then hands the message to the transport layer.

The transport layer sends the resulting message back to the client transport layer, which hands the message back to the client stub.

The client stub unmarshalls the return parameters, and execution returns to the caller.

Types of RPC

There are several RPC models and distributed computing implementations. A popular model and implementation is the Open Software Foundation's (OSF) Distributed Computing Environment (DCE). The Institute of Electrical and Electronics Engineers (IEEE) defines RPC in its ISO Remote Procedure Call Specification, ISO/IEC CD 11578 N6561, ISO/IEC, November 1991.

Examples of RPC configurations include the following:

- The normal method of operation where the client makes a call and doesn't continue until the server returns the reply.
- The client makes a call and continues with its own processing. The server doesn't reply.
- A facility for sending several client nonblocking calls in one batch.
- RPC clients have a broadcast facility, i.e., they can send messages to many servers and then receive all the resulting replies.
- The client makes a nonblocking client/server call; the server signals the call is completed by calling a procedure associated with the client.

RPC spans the transport layer and the application layer in the Open Systems Interconnection (OSI) model of network communication. RPC makes it easier to develop an application that includes multiple programs distributed in a network. Alternative methods for client-server communication include message queueing and IBM's Advanced Program-to-Program Communication (APPC).

- Though it boasts a wide range of benefits, there are certainly a share of pitfalls that those who use RPC should be aware of.

- Here are some of the advantages RPC provides for developers and application managers:

- Helps clients communicate with servers via the traditional use of procedure calls in high-level languages.
- Can be used in a distributed environment, as well as the local environment.
- Supports process-oriented and thread-oriented models.
- Hides the internal message-passing mechanism from the user.
- Requires only minimal effort to rewrite and redevelop the code.
- Provides abstraction, i.e., the message-passing nature of network communication is hidden from the user.
- Omits many of the protocol layers to improve performance.

On the other hand, some of the disadvantages of RPC include the following:

- The client and server use different execution environments for their respective routines, and the use of resources (e.g., files) is also more complex. Consequently, RPC systems aren't always suited for transferring large amounts of data.
- RPC is highly vulnerable to failure because it involves a communication system, another machine and another process.
- There is no uniform standard for RPC; it can be implemented in a variety of ways.
- RPC is only interaction-based, and as such, it doesn't offer any flexibility when it comes to hardware architecture.

GRPC

GRPC is a modern open-source high performance Remote Procedure Call (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.

GRPC is a high performance, open-source framework developed by Google to handle remote procedure calls (RPCs). GRPC is Google's approach to a client-server application. It lets client and server applications communicate transparently, simplifying the process for developers to build connected systems. Released in August 2016, GRPC has been adopted by enterprises, startups and open-source projects worldwide.

GRPC runs in any environment, connecting services in and across data centers with pluggable support for tracing, health checking, load balancing and authentication. GRPC can also be used to connect mobile devices, Mobile apps and browsers to backend services.

Developers use GRPC in the last mile of computing in mobile and web clients because it can generate libraries for Android and iOS. Additionally, it uses standards-based HTTP/2 as transport so it can cross firewalls and proxies easily.

GRPC clients and servers can run and communicate with each other in various environments, including a user's desktop and servers inside Google. Furthermore, GRPC clients can be written in any of GRPC's supported languages, including:

- Java (with support for Android)
- Objective-C (for iOS)
- C++
- Ruby
- JSON
- Python
- Go
- C#

For instance, a developer can easily create a GRPC server in Java with clients in Python, Ruby or Go. Additionally, since the latest Google application program interfaces (APIs) have GRPC versions of their interfaces, developers can build Google functionality into their applications.

Benefits of using GRPC

GRPC, like other RPC systems, revolves around the idea of defining a service, such as identifying the methods that can be remotely called with their parameters and return types. However, GRPC lets developers use more sophisticated technologies that are efficient and scalable, such as HTTP/2 and streams. Since it is technology-agnostic, it can be used by and interact with server and clients from several different programming languages.

GRPC is also built upon protocol buffers, also known as protosuns. Protosuns are Google's tool for sequencing structured data, which allows for communication and data storage that can be predicted and analyzed.

Types of GRPC

GRPC lets developers define four types of service methods:

Unary RPC – The client sends one request to the server and gets one response back, the same as with a normal function call.

Server streaming – The client sends a request to the server and receives a stream of messages back. The client reads from the returned stream until there are no messages left. Here, GRPC guarantees message ordering within an individual RPC call.

Client streaming – The opposite of server streaming, the client writes a sequence of messages and sends them to the server, using a provided stream. Once the client has finished writing the messages, it waits for the server to read them and return its responses. Once again, GRPC guarantees message ordering within an individual RPC call.

Bidirectional streaming – Both sides send a sequence of messages via a read-write stream. The two streams work independently of each other and, as such, the clients and servers can read and write in any order. For instance, the server reads a message then writes a response. Or the server waits to receive all the client messages before writing its responses. GRPC preserves the order of messages in each stream.

Load balancer

Load balancing refers to efficiently distributing incoming network traffic across a group of backend servers, also known as a server farm or server pool.

Modern high-traffic websites must serve hundreds of thousands, if not millions, of concurrent requests from users or clients and return the correct text, images, video, or application data, all in a fast and reliable manner. To cost-effectively scale to meet these high volumes, modern computing best practice generally requires adding more servers.

A load balancer acts as the "traffic cop" sitting in front of your servers and routing client requests across all servers capable of fulfilling those requests in a manner that maximizes speed and capacity utilization and ensures that no one server is overworked, which could degrade performance. If a single server goes down, the load balancer redirects traffic to the remaining online servers. When a new server is added to the server group, the load balancer automatically starts to send requests to it.

In this manner, a load balancer performs the following functions:

- Distributes client requests or network load efficiently across multiple servers
- Ensures high availability and reliability by sending requests only to servers that are online
- Provides the flexibility to add or subtract servers as demand dictates

Different load balancing algorithms provide different benefits; the choice of load balancing method depends on your needs:

Round Robin – Requests are distributed across the group of servers sequentially.
Least Connections – A new request is sent to the server with the fewest current connections to clients. The relative computing capacity of each server is factored into determining which one has the least connections.
Least Time – Sends requests to the server selected by a formula that combines the
fastest response time and fewest active connections. Exclusive to NGINX Plus.
Hash – Distributes requests based on a key you define, such as the client IP address or the request URL. NGINX Plus can optionally apply a consistent hash to minimize redistribution of loads if the set of upstream servers' changes.
IP Hash – The IP address of the client is used to determine which server receives the request.
Random with Two Choices – Picks two servers at random and sends the request to the one that is selected by then applying the Least Connections algorithm (or for NGINX Plus the Least Time algorithm, if so configured).

- Benefits of Load Balancing:
- Reduced downtime
- Scalable
- Redundancy
- Flexibility
- Efficiency

III. EXISTING SYSTEM

Dedicated software is required for load balancing. Complex setups need to done for implementation. And at this moment there are very minimal load balancing tools available for GRPC which makes a new load balancer necessary introduction to the applications running on GRPC.
Disadvantages:
This might eat up more space from machine.
Also, may utilize fair share of CPU which could cause an overall impact on application performance.

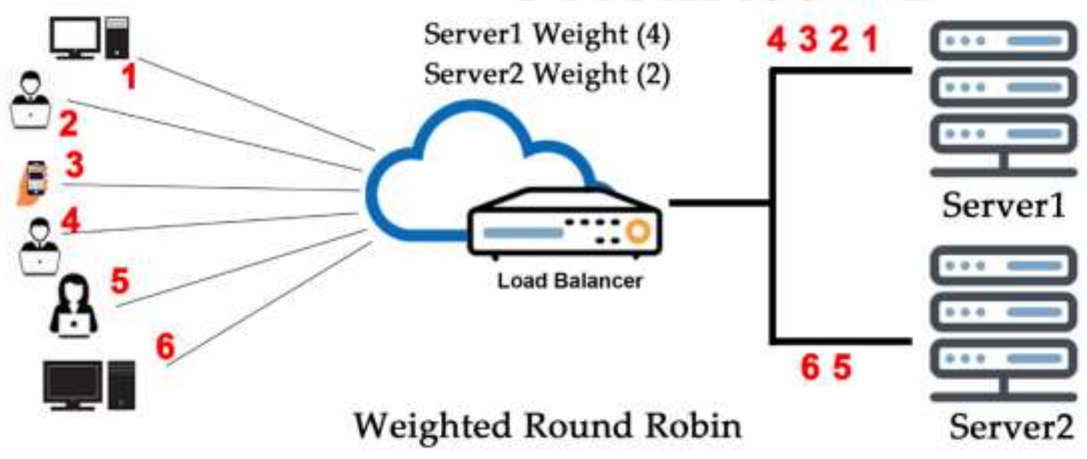
IV. PROPOSED SYSTEM:

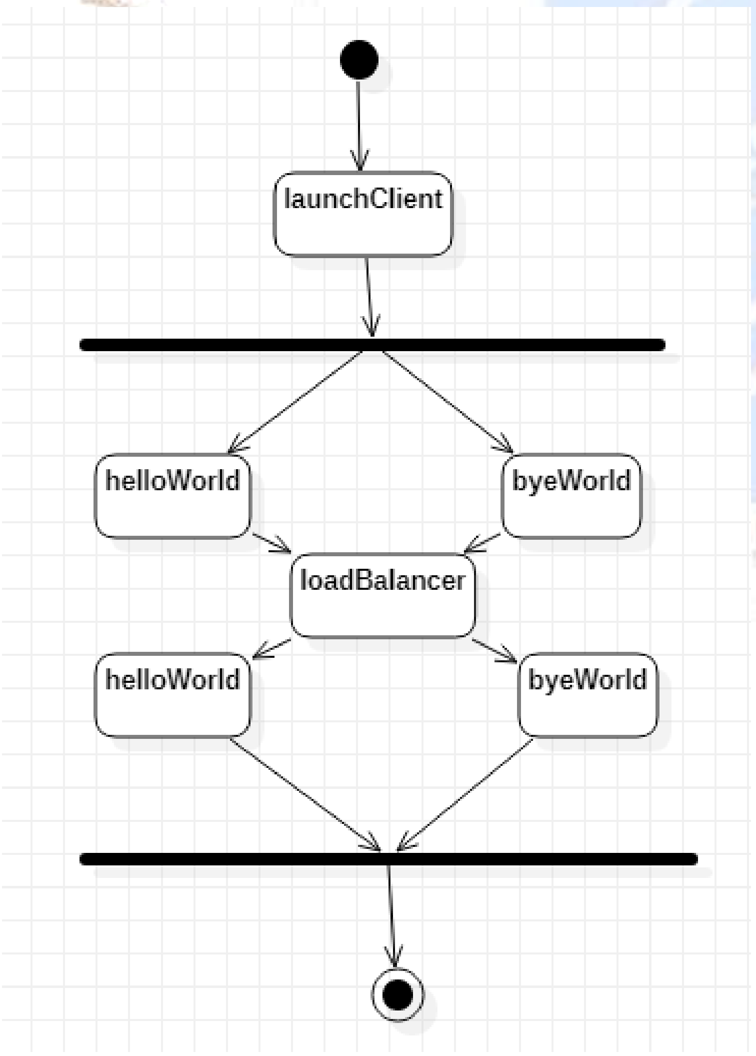
We suggest a java-based load balancing set up that runs on the client's end with no additional set up or software's required. We develop a system that would take the list of servers available and evaluate based on capacities and implement weighted round robin for choosing the server for routing.

Advantages:

1. Impact on performance is low.
2. Free of cost.
3. Less dependency on CPU

V.SYSTEM ACHITECTURE





ACTIVITY DIAGRAM

VI. CONCLUSION:

Here is our project that address problems with existing system and solves them effectively. In the end, we have achieved a process that load balances the requests equally between two servers.

VII. FUTURE ENHANCENMENT:

Updates are best to continue the legacy of any applications. For this we propose to integrate micro businesses into the research and try to make the accuracy maximum.

VIII. REFERENCES:

[1] Al Nuaimi, K., Mohamed, N., Al Nuaimi, M., & Al-Jarod, J. (2012, December). A survey of load balancing in cloud computing: Challenges and algorithms. In Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on (pp.137-142). IEEE.

[2] Ali, A. D., &Belal, M. A. (2007). Multiple ant colonies optimization for load balancing in distributed systems. Proceedings of ICTA, 2007.

3] Asha, M. L. & Neethu Myhrer. (2014), Performance Evaluation of Round Robin Algorithm in Cloud Environment. International Journal of Computer Applications (pp.12-16). ICICT-2014.

[4] Aslam, S., & Shah, M. A. (2015, December). Load balancing algorithm in cloud computing, A survey of modern techniques. In 2015 National Software Engineering Conference (NSEC) (pp. 30-35). IEEE.

[5] Behalf, V., & Kumar, A. (2014). Comparative Study of Load Balancing Algorithms in Cloud Environment using Cloud Analyst. International Journal of Computer Applications, 97(1).

[6] Dimi's. C. (2015). Various Load Balancing Algorithms in Cloud Environment.

[7] Domanial, S. G., & Reddy, G. R. M. (2014, January). Optimal load balancing in cloud computing by efficient utilization of virtual ma-chines. In 2014 Sixth International Conference on Communication System and Networks (COMSNETS) (pp. 1-4). IEEE.

[8] Ezaki, O.M., Rashad, M.Z. & Elwood, M.A. (2012), Improved Max-Min Algorithm in Cloud Computing. Vol. 50-No 12, July 2012, pp.22-27.

[9] Gang Liu, Jing Li & Juancho Xu, An Improved Min-Min Algorithm in Cloud Computing. Proceedings of the 2012 International Conference of Modern Computer Science and Applications (pp.47-50), Springer Berlin Heidelberg.

[10] Gopinath, P. G., & Vasudevan, S. K. (2015). An in-depth analysis and study of Load balancing techniques in the cloud computing environment. Procedia Computer Science, 50, 427-432.

[11] Gopinath, P.G., & Vasudevan's. K. (2015). An in -depth analysis and study of Load balancing techniques in the cloud computing environment. Procedia Computer Science, 50, 427-432.

[12] Haidari, R.A., Kati, C.P. and Saxena, P.C., 2014, July. A load balancing strategy for Cloud Computing environment. In Signal Propagation and Computer Technology (ICSPCT), 2014 International Conference on (pp. 636-641). IEEE.

[13] Haryana, N., & Jangle, D. (2014). Dynamic Method for Load Balancing in Cloud Computing. IOSR Journal of Computer Engineering (IOSR-JCE), 16(4), 23-28.

[14] Kashyap., & Vira Diya, J. (2014). A survey of various load balancing algorithms in cloud computing. International Journal of Scientific and Technology Research, 3 (11), 115-19.

[15] Katyal, M., &Mishra, A. (2014). A comparative study of load balancing algorithms in cloud computing environment. axis preprint arXiv:1403.6918.

[16] Kaur, R., & Luthra, P. (2012, December). Load Balancing in Cloud Computing. In Proceedings of International Conference on Recent Trends in Information, Telecommunication and Computing, ITC.

[17] Khorana, F. F, & Vania, J (2014). Load Balancing in cloud computing.

[18] Mao, Y., Chen, X. and Li, X., 2014. Max–min task scheduling algorithm for load balance in cloud computing. In Proceedings of International Conference on Computer Science and Information Technology (pp. 457-465). Springer India.

[19] Mishra, R., & Jaiswal, A. (2012). Ant colony optimization: Absolution offload balancing in cloud. International Journal of Web & Semantic Technology,3(2), 33.

[20] More, M. S. D., & Mohapatra, A. Load Balancing Strategy Based on Cloud Partitioning Concept.

[21] Nadcap, A., & Marla, V. Cloud Computing–Partitioning Algorithm and Load Balancing Algorithm.

[22] Natyam., Shivaism., &Raj,M.G.(2012). Comparative analysis of load balancing algorithms in cloud computing. International Journal of Advanced Research in Computer Engineering & Technology (IJAR-CET), 1(3), pp-120.