Latency in System Design

Imagine a pipe with water flowing through it. The speed at which water flows through the pipe may vary i.e. sometimes flowing very fast and sometimes flowing very slow. This concept is similar to latency in system design. Latency determines the speed at which data can be transferred from one end of a system to another.

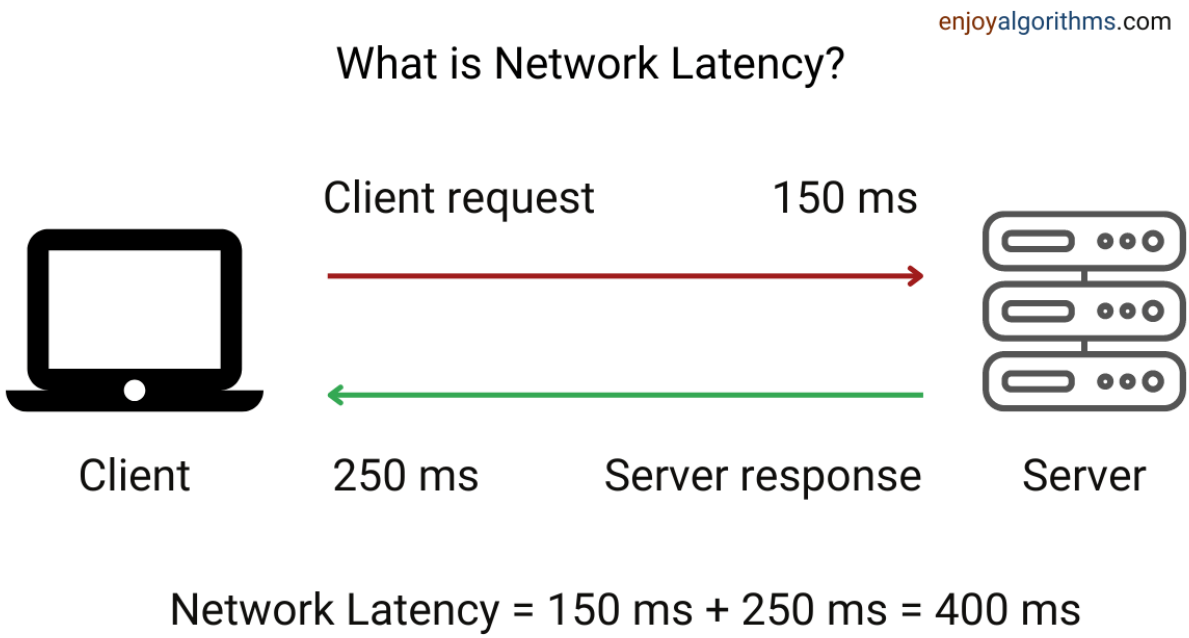
In this blog, we will focus on the concept of latency and how it affects the performance of a system. We will also discuss measures that can be taken to improve the latency of a system.

What is Latency?

Latency is a measure of how quickly data can be transferred between a client and a server. It is directly related to performance of the system. Lower latency indicating better performance.

Let's understand this from another perspective! Browser (client) sends a signal to the server whenever a request is made. Servers process the request, retrieve the information and send it back to the client. So, latency is the time interval between start of a request from the client end to delivering the result back to the client by the server i.e. the round trip time between client and server.

Our ultimate goal would be to develop a latency-free system, but various bottlenecks prevent us in developing such an ideal system. The core idea is simple: We can try to minimize it. Lower the system latency, the less time it takes to process our requests.



How does latency work?

Latency is nothing other than the estimated time the client has to wait after starting the request to receive the result. Let's take an example and look into how it works.

Suppose you interact with an e-commerce website, you liked something, and added it to the cart. Now when you press the "Add to Cart" button, following events will happen:

The instant "Add to Cart" button is pressed, the clock for latency starts and browser initiate a request to the server.

Server acknowledges the request and processes it.

Server response to the request, the response reaches your browser, and product gets added to your Cart.

You can start the stopwatch in the first step and stop the stopwatch in the last step. This difference in time would be the latency.

What causes Latency?

Latency in a network depends on various parameters. One of the major factors contributing to latency is outbound calls. In the previous example of adding cart exercise, when you click the button on browser, the request goes to some server in

the backend, which again calls multiple services internally for computation (in parallel or sequentially) and then waits for a response. All this adds to the latency of the call. However, It is mainly caused by following factors:

Transmission mediums: Transmission medium is the physical path between the start and endpoints. So system latency depends on the type of medium used to transmit requests. Transmission mediums like WAN and Fiber Optic Cables are widely used, but each medium has limitations.

Propagation: It refers to the amount of time required for a packet to travel from one source to another. So system latency highly depends on the distance between communicating nodes. Farther the nodes are located, more the latency.

Routers: Routers are an essential component in communication and take some time to analyze the header information of a packet. So latency depends on how efficiently router processes the request. Router to router hop increases the system latency.

Storage delays: System latency also depends on the type of storage system used, as it may take some time to process and return data. So accessing stored data can increase latency of the system.

How to measure Latency?

There are many methods used to quantify latency. Three most common methods are:

Ping: Ping is the most common utility used to measure latency. It sends packets to an address and sees how fast response is coming. It measures how long it takes for the data to travel from source to destination and back to the source. A faster ping corresponds to a more responsive connection.

Traceroute: Traceroute is another utility used to test latency. It also uses packets to calculate the time taken for each hop when routed to the destination.

MTR: MTR is a combination of both ping and Traceroute. MTR gives a list of reports on how each hop in a network is required for a packet to travel from one end to the other. The report generally includes details such as percentage Loss, Average Latency, etc.

Latency optimization

Latency restricts performance of the system, so it is necessary to optimize it. We can reduce latency by adopting following measures:

HTTP/2: We can reduce latency by the use of HTTP/2. It allows parallelized transfers and minimizes round trips from the sender to the receiver.

Less external HTTP requests: Latency increases because of third-party services. By reducing number of external HTTP requests, system latency gets optimized as third-party services affect speed of the application.

CDN: CDN stores resources in multiple locations worldwide and reduces the request and response travel time. So instead of going back to the origin server, request can be fetched using the cached resources closer to the clients.

Browser Caching: Browser caching can also help to reduce the latency by caching specific resources locally to decrease the number of requests made to the server.

Disk I/O: Instead of often writing to disk, we can use write-through caches or in-memory databases or combine writes where possible or use fast storage systems, such as SSDs.

Latency can also be optimized by making smarter choices regarding storage layer, data modelling, outbound call fanout, etc. Here are some ways to optimize it at an application level:

- Inefficient algorithms are the most apparent sources of latency in code. It is necessary to avoid unnecessary loops or nested expensive operations.
- Use design patterns that avoid locking as multithreaded locks introduce latency.
- Use an asynchronous programming model to utilize better hardware resources as blocking operations cause long wait times.
- Limiting the unbounded queue depths and providing back pressure typically lead to less wait time in the code resulting in more predictable latencies.

Conclusion

It is a vital concept associated with the design of every system. One can never make a fully latency-free system, but one can easily optimize it. With modern

hardware and heavy computationally efficient machines, latency is no longer a bottleneck of the system.

Thanks to Chiranjeev and Suyash for his contribution in creating the first draft of this content. If you have any queries/doubts/feedback, please write us at contact@enjoyalgorithms.com. Enjoy learning, Enjoy system design, Enjoy algorithms!

Author: Shubham Gautam

Reviewer: Keshav Nandan

Machine Learning

ML Course

ML Blogs

System Design

SD Course

SD Blogs

OOP Concepts

OOP Course

OOP Blogs

EnjoyAlgorithms Newsletter

Subscribe to get well designed content on data structure and algorithms, machine learning, system design, object orientd programming and math.

Email address

Subscribe

Explore More Content

Availability in System Design

Availability of a system is measured as the percentage of the system's uptime in a given time period so that a system is available to perform tasks and functions under normal conditions. One way to look at system availability is how resistant a system is to failures. High availability systems have tradeoffs, such as higher latency or lower throughput.

Read More

Process Management in Operating System (OS)

In operating system, Process management involves executing various tasks such as creating processes, scheduling processes, managing deadlock, and termination of processes. It is the responsibility of the OS to manage all running processes of the system. In this blog, we will learn about process management in OS and its various related algorithms.

Read More

Least Frequently Used (LFU) Cache Implementation

The least frequently used (LFU) is a cache algorithm used to manage memory within a computer. In this method, the system keeps track of the number of times a block is referenced in memory, and when the

cache is full, our system removes the item with the lowest reference frequency. LFU cache get and put operation works in O(1) average time complexity.

Read More

Notification Service System Design

Notification services are widely used in almost every product. They are helpful if we want to be alerted of a price change or availability of a new product feature or if we want to be updated if a new job specification for a job search becomes available. This blog will focus on system design of notification service and discussion around various components.

Read More

Data Partitioning (Sharding) in System Design

Data partitioning or sharding is a technique of dividing data into independent components. It is a way of splitting data into smaller pieces so that data can be efficiently accessed and managed. Database partitioning is the backbone of modern system design, which helps to improve scalability, manageability, and availability.

Read More

Throttling and Rate Limiting in System Design

At its most basic level, a rate limiter restricts the number of events a certain object (person, device, IP, etc.) can do in a given time range. In general, a rate limiter caps how many requests a sender can issue in a specific time window. Rate Limiter then blocks requests once the cap is reached.

Read More

Typeahead (Autocomplete) System Design

As users type their search query, the Typeahead search feature guesses the rest of a word and offers the top suggestions that begin with whatever they have written. Here frequency and recentness of a query are used to sort suggestions. Search autocomplete system is used by many search platforms like Facebook, Google, Instagram, etc.

Read More

Whatsapp System Design

Whatsapp is a social messenger platform, which allows users to send messages to each other. It is a messaging system that is widely used throughout the globe. Here in this blog, we'll be discussing WhatsApp's generic architecture and which could also be used as a base for designing any such chat application. So let's get started by discussing the key requirements of our service.

Read More

Comparison of SQL and NoSQL Databases

Whether NoSQL or SQL databases are more appropriate for a given use case depends on a variety of factors, including the nature of the data, the application requirements, and the available resources and expertise. In this blog, we will discuss the key differences between SQL and NoSQL databases that software developers must consider when making a decision.

Read More

Introduction to Hadoop Framework for Big Data

Hadoop is an open-source framework that addresses the analytical and operational needs of Big Data by overcoming the limitations of traditional data analysis methods. With support for highly scalable and fault-tolerant distributed file systems, it allows for parallel processing. It comprises four main components - HDFS, YARN, MapReduce, and Hadoop Common.

Read More

Network Protocols Concept

In computer networking, a network is a group of devices connected in some way to exchange data. Similarly, network protocols define a common format and set of rules for exchanging messages between devices. Some common networking protocols are Hypertext Transfer Protocol (HTTP), Transmission Control Protocol (TCP), and Internet Protocol (IP).

Read More

Google Docs System Design

Google docs is an online word processor that is part of Google's free, web-based Google docs editors package. It is a massive system with tons of features. If we think about how google docs works, we can realize that it is much more complex than it seems to be. This blog will focus on system design of google docs and discussion around various components.

Read More