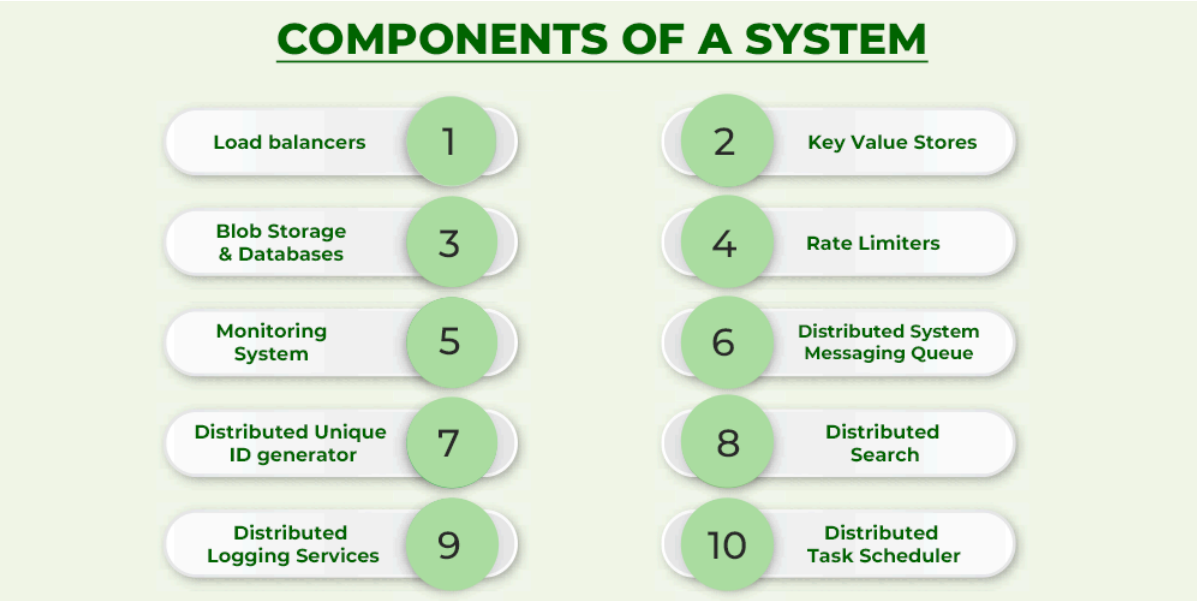
What are the components of System Design?

> System design is the process of defining the architecture, components, modules, interfaces, and data for a computer system. It involves analyzing the requirements of the system, identifying the constraints and assumptions, and defining the high-level structure and components of the system. The goal of system design is to create a blueprint for the development and implementation of a computer system that meets the needs of the users and stakeholders.

Components of System Design

- Load Balancer
- Key-value stores
- Blob storage & Databases
- Rate limiters
- Monitoring System
- Distributes system messaging queue
- Distributed unique id generator
- Distributes search
- Distributed logging services
- Distributes task scheduler

The components of system design refer to the different elements that are involved in the design of a computer system as follows :



Components of System Design

1. Load Balancer

A load balancer is a system design component that is used to distribute incoming requests or workloads across a number of different resources or servers. This can be useful in a number of different scenarios, such as when a system receives a large number of requests and needs to distribute them among multiple servers to avoid overloading any one server, or when a system has multiple servers and needs to distribute requests evenly among them to ensure that all servers are utilized efficiently.

There are many different types of load balancers, and the specific type that is used in a given system will depend on the specific requirements of the system. Some common types of load balancers include:

- **Layer 4 load balancers** operate at the network layer of the OSI model and distribute requests based on the source and destination IP addresses and port numbers of the requests.
- **Layer 7 load balancers** operate at the application layer of the OSI model and distribute requests based on the content of the requests, such as the URL or the type of HTTP method used.
- **Global load balancers** are used in distributed systems to distribute requests among multiple servers located in different geographic regions.
- **Application load balancers** are specialized load balancers that are designed to work with specific types of applications or protocols, such as HTTP or HTTPS.

In general, a load balancer is a key component of many system designs and can play a critical role in ensuring that a system can handle a large number of requests efficiently and without overloading any of its resources.

2. Key-value stores

A key-value store is a type of NoSQL database that is designed to store data as a set of key-value pairs. In a key-value store, each piece of data is stored under a unique key, and the value is the data itself. Key-value stores are often used to store data that is accessed frequently, as they can provide fast access to data by key.

- There are several different types of key-value stores, including in-memory key-value stores, which store data in memory for fast access, and persistent key-value stores, which store data on disk or in a distributed file system for durability. Key-value stores can be used in a variety of applications, including caching, session management, and real-time analytics.
- Key-value stores are generally simpler to use and more scalable than other types of databases, such as relational database management systems (RDBMS). However, they are not as well-suited for storing complex, structured data that requires advanced querying capabilities.

In a distributed system, key-value stores can be used to store data that needs to be accessed quickly and consistently across multiple nodes. They can also be used to store metadata and other auxiliary data that is used by the system. It is important to choose the right type of key-value store for the specific requirements of the system, taking into account factors such as scalability, performance, and durability.

3. Blob storage & Databases

Blob storage and database systems are two different types of storage systems that can be used to store and manage data.

Blob storage, also known as object storage, is a type of storage system that is designed to store large amounts of unstructured data, such as documents, images, videos, and audio files. Blob storage systems are typically highly scalable and can handle a large number of requests concurrently. They are often used to store data that is accessed frequently, such as media files or user-generated content.

- Database systems, on the other hand, are designed to store structured data that is organized in a specific way.
- There are several different types of database systems, including relational database management systems (RDBMS), NoSQL databases, and in-memory databases.
- Database systems are typically used to store data that needs to be queried and accessed in a structured way, such as customer records or financial transactions.

Blob storage and database systems can be used together in a distributed system to store and manage different types of data. For example, a distributed system might use a blob storage system to store unstructured data such as user-generated content, and a database system to store structured data such as customer records and transactions. It is important to choose the right type of storage system for each type of data, taking into account the specific requirements of the system and the needs of the users.

4. Rate limiters

Rate limiters are system design components that are used to limit the rate at which a system or application processes requests or performs certain actions. This can be useful in a number of different scenarios, such as when a system needs to protect itself from being overloaded by too many requests, or when an organization wants to prevent a specific user or group of users from making excessive requests that could impact the performance of a system.

There are many different types of rate limiters, and the specific type that is used in a given system will depend on the specific requirements of the system. Some common types of rate limiters include:

- **Request rate limiters** are used to limit the number of requests that a system or application processes within a given time period.

- **Action rate limiters** are used to limit the number of times that a specific action or operation can be performed within a given time period.
- **User rate limiters** are used to limit the rate at which a specific user or group of users can make requests to a system or application.
- **Token bucket rate limiters** are used to limit the rate at which requests are processed by a system by allowing a certain number of requests to be processed in each time period, with any excess requests being held in a “bucket” until the next time period.

In general, rate limiters are a useful component of many system designs, and can play a key role in ensuring that a system is able to handle a high volume of requests without being overwhelmed. By limiting the rate at which requests are processed, rate limiters can help prevent a system from being overloaded or degraded, and can help ensure that it is able to provide consistent and reliable performance.

5. Monitoring System

A monitoring system is a system design component that is used to collect, analyze, and report on various metrics and performance data related to a system or application. This can be useful in a number of different scenarios, such as when a system needs to track its own performance and availability, or when an organization needs to monitor the performance of its systems and applications to ensure that they are meeting their desired service levels.

There are many different types of monitoring systems, and the specific type that is used in a given system will depend on the specific requirements of the system. Some common types of monitoring systems include:

- **Network monitoring systems,** are used to monitor the performance of a network and its various components, such as routers, switches, and servers.
- **System monitoring systems,** are used to monitor the performance of a computer system and its various components, such as the CPU, memory, and disk usage.
- **Application monitoring systems**, are used to monitor the performance of specific applications or services, such as web servers or databases.
- **Infrastructure monitoring systems,** are used to monitor the performance of the underlying infrastructure on which a system or application is running, such as virtual machines or containers.

In general, a monitoring system is a critical component of many system designs and can play a key role in ensuring that a system is performing well and meeting its desired service levels. By providing real-time visibility into the performance of a system, a monitoring system can help identify and troubleshoot issues as they arise, and can provide valuable insights into the overall health and availability of a system.

6. Distributes system messaging queue

A distributed system messaging queue is a system that enables the exchange of messages between different nodes in a distributed system. Messaging queues allow nodes to communicate asynchronously, decoupling the sender and receiver of a message and enabling each node to operate independently.

There are several different types of messaging queues, including:

- **Point-to-point queues:** In this type of queue, messages are delivered to a specific recipient.
- **Publish-subscribe queues:** In this type of queue, messages are published to a topic and are delivered to all subscribers to that topic.
- **Hybrid queues:** Hybrid queues combine elements of both point-to-point and publish-subscribe queues, allowing messages to be delivered to specific recipients or to all subscribers to a topic.

Distributed system messaging queues can be used to enable communication between different components of a distributed system, such as microservices or distributed applications. They can also be used to decouple different parts of the system, allowing each component to operate independently and improving the system's resilience and scalability.

There are several tools and frameworks available for implementing distributed system messaging queues, including Apache Kafka, RabbitMQ, and Amazon Simple Queue Service (SQS). It is important to choose a messaging queue that meets the specific requirements of your system, taking into account factors such as scalability, performance, and fault tolerance.

7. Distributed unique id generator

A distributed unique ID generator is a system that generates unique identifiers (IDs) that can be used to identify objects or entities in a distributed system. These IDs are typically used to uniquely identify items in a database or to provide a stable identifier for a resource that is accessed over the network.

There are several approaches to generating distributed unique IDs :

- Using a centralized service
- Using a distributed consensus algorithm
- Using timestamps

8. Distributes search

Distributed search refers to the practice of using multiple nodes or servers to index and search large datasets in a distributed system. Distributed search can be used to improve the performance and scalability of search operations, as it allows for parallel processing of search queries and the distribution of data across multiple nodes.

There are several approaches to implementing distributed search, including:

- **Using a distributed search engine:** A distributed search engine is a search platform that is designed to scale horizontally across multiple nodes. These systems typically use a distributed index to store the data being searched, allowing for parallel processing of search queries. Examples of distributed search engines include Elasticsearch and Apache Solr.
- **Using a database with search capabilities:** Some databases, such as MongoDB and Cassandra, have built-in search capabilities that allow for the indexing and searching of data stored in the database. These systems can be used to implement distributed search in a distributed system.
- **Using a cloud-based search service:** Cloud-based search services, such as Amazon Elasticsearch Service and Google Cloud Search, can be used to implement distributed search in a distributed system. These services are typically highly scalable and fault-tolerant, and they can be a good choice for organizations that do not want to manage their own search infrastructure.

In a distributed system, it is important to choose a distributed search solution that meets the specific requirements of the system, taking into account factors such as scalability, performance, and cost.

9. Distributed logging services

Distributed logging refers to the practice of collecting, storing, and analyzing log data from multiple sources in a distributed system. This can be useful for tracking the health and performance of a distributed system, as well as for debugging issues that may arise.

There are several approaches to implementing distributed logging, including:

- Using a centralized logging service
- Using a distributed logging system
- Using a cloud-based logging service

10. Distributes task scheduler

A distributed task scheduler is a system that is responsible for scheduling and executing tasks in a distributed system. A task scheduler can be used to automate the execution of tasks at regular intervals, on a specific schedule, or in response to certain events.

There are several approaches to implementing a distributed task scheduler, including:

- **Using a standalone task scheduler:** A standalone task scheduler is a separate system that is responsible for scheduling and executing tasks in a distributed system. This approach can be simple to implement and allows for flexibility in terms of the types of tasks that can be scheduled. However, it can be more complex to manage and may require additional infrastructure.
- **Using a built-in task scheduler:** Some distributed systems, such as container orchestration platforms or cloud-based serverless platforms, have built-in task schedulers that can be used to schedule tasks within the system. This approach can be simpler to implement and manage but may be less flexible in terms of the types of tasks that can be scheduled.
- **Using a cloud-based task scheduler:** Cloud-based task schedulers, such as Amazon Simple Notification Service (SNS) or Google Cloud Scheduler, can be used to schedule tasks in a distributed system. These services are typically highly scalable and fault-tolerant, and they can be a good choice for organizations that do not want to manage their own task-scheduling infrastructure.

It is important to choose a distributed task scheduler that meets the specific requirements of the system, taking into account factors such as scalability, performance, and cost.