Scalability

Arnav Gupta

December 5, 2024

Contents

1 Scalability

Service: different from simple program since it listens for requests from clients/users and may handle multiple requests concurrently

Requests usually delivered as messages that arrive over a network. Service runs constantly, waiting for requests and processing them.

A service is scalable if it can easily handle growth in the number of concurrent users/requests.

Measured through **work throughput**:

- requests/queries per second
- concurrent users
- monthly active users

Ignore cost, just consider scale achieved.

Scaling challenges:

- limited speed from one machine
- coordinating multiple machines
- sharing data among machines

- failure probability
- high latency from worldwide users
- authentication of service components
- software updates without downtime

Vertical scaling: make machine bigger and stronger

Horizontal scaling: add more machines

Computer performance affected by number/speed of CPU cores, RAM, disk type, number of disks, type of network connectivity, GPUs, TPUs, etc.

Shared Memory Parallelism: a single process can have multiple threads which execute concurrently while sharing the same memory

Cloud computing resources are elastic since size and quantity of resources and can be quickly changed.

Vertical scaling is not scalable:

- easy to write programs since most OSs have multithreading
- less communication required with other machines
- cannot handle big loads
- cannot be scaled easily (must replace entire machine)
- single point of failure
- bad price/performance ratio

2 HTTP and Web Servers

To convert a program to a service:

1. listen to requests on the network
2. run many copies of program concurrently
3. use queues to store unhandled requests and unsent responses
    (a) allows competing threads to share single network socket

With parallel threads, even if only a single CPU core, multiple threads can run since the app may block to request data from disk or network (IO), and in the same time, another thread can run.

HTTP: client-server data exchange protocol

Request specifies:

- human readable header with URL, method, and more
- optional body with raw data

Response includes:

- human readable header with response code and more
- optional body