ARCHITECTURES

COMPUTER SCIENCE 4 BIG DATA

CONTENT

SYSTEMS DESIGN PRIMER

Read the systems design primer thoroughly!

Answer the following questions in a few sentences:

WHAT IS THE DIFFERENCE BETWEEN LATENCY AND THROUGHPUT?

https://github.com/kwbln/system-design-primer#latency-vs-throughput

https://community.cadence.com/cadence_blogs_8/b/sd/posts/understanding-latency-vs-throughput

LATENCY

- time to perform some action or to produce some result
- measured in units of time: hours, minutes, seconds, nanoseconds or clock periods

THROUGHPUT

- number of such actions or results per unit of time
- measured in units of whatever is being produced (cars, motorcycles, I/O samples, memory words, iterations) per unit of time.
- "memory bandwidth" is sometimes used to specify the throughput of memory systems

AIM

- maximal throughput with acceptable latency.

EXAMPLE

The following manufacturing example should clarify these two concepts:

An assembly line is manufacturing cars. It takes eight hours to manufacture a car and that the factory produces one hundred and twenty cars per day.

The latency is: 8 hours.

The throughput is: 120 cars / day or 5 cars / hour.

~~WHAT~~ WHEN WOULD YOU DESIGN A AP OR A CP SYSTEM?

https://github.com/kwbln/system-design-primer#availability-vs-consistency

In a distributed computer system, you can only support two of the following guarantees:

CONSISTENCY

- Every read receives the most recent write or an error

AVAILABILITY

- Every request receives a response, without guarantee that it contains the most recent version of the information

PARTITION TOLERANCE

- The system continues to operate despite arbitrary partitioning due to network failures

NETWORKS AREN'T RELIABLE, SO YOU'LL NEED TO SUPPORT PARTITION TOLERANCE. YOU'LL NEED TO MAKE A SOFTWARE TRADEOFF BETWEEN CONSISTENCY AND AVAILABILITY.

CP - CONSISTENCY AND PARTITION TOLERANCE

- Waiting for a response from the partitioned node might result in a timeout error.
- CP is a good choice if your business needs require atomic reads and writes.

AP - AVAILABILITY AND PARTITION TOLERANCE

- Responses return the most recent version of the data available on a node, which might not be the latest. Writes might take some time to propagate when the partition is resolved.
- AP is a good choice if the business needs allow for eventual consistency or when the system needs to continue working despite external errors.

WHAT IS REPLICATION, FAIL-OVER AND HOW DOES REDIS REPLICATION WORK?

Two main patterns to support high availability: fail-over and replication.

FAIL-OVER

https://github.com/kwbln/system-design-primer#fail-over

ACTIVE-PASSIVE

- heartbeats are sent between the active and the passive server on standby
- If the heartbeat is interrupted, the passive server takes over the active's IP address and resumes service.
- The length of downtime is determined by whether the passive server is already running in 'hot' standby or whether it needs to start up from 'cold' standby. Only the active server handles traffic.
- master-slave failover

ACTIVE-ACTIVE

- both servers are managing traffic, spreading the load between them.
- If the servers are public-facing, the DNS would need to know about the public IPs of both servers. If the servers are internal-facing, application logic would need to know about both servers.
- master-master failover

FAIL-OVER DISADVANTAGE(S)

- Fail-over adds more hardware and additional complexity.
- There is a potential for loss of data if the active system fails before any newly written data can be replicated to the passive

REPLICATION

https://github.com/kwbln/system-design-primer#replication

MASTER-SLAVE

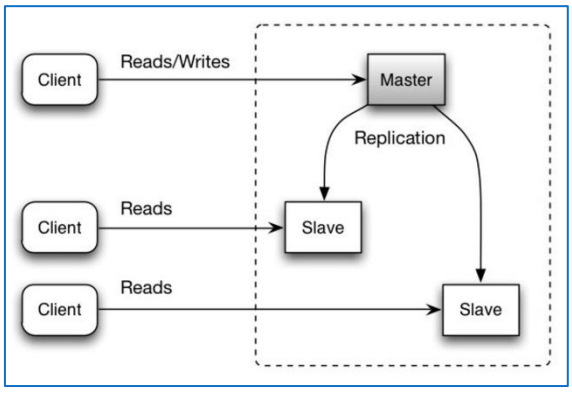
https://github.com/kwbln/system-design-primer#master-slave-replication

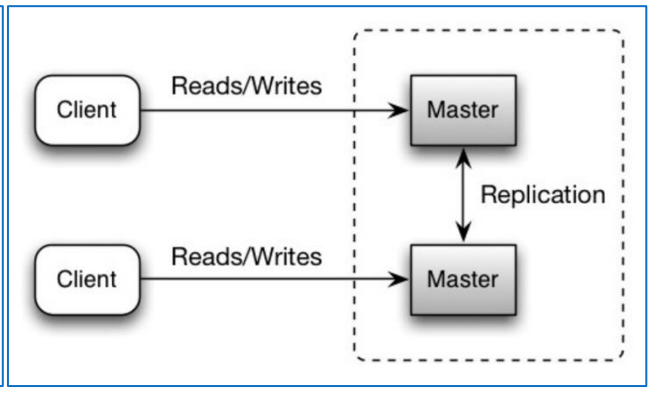
- The master serves reads and writes, replicating writes to one or more slaves, which serve only reads.
- Slaves can also replicate to additional slaves in a tree-like fashion. If the master goes offline, the system can continue to operate in read-only mode until a slave is promoted to a master or a new master is provisioned.
- Disadvantage
  - Additional logic is needed to promote a slave to a master.
  - (+ general disadvantages of partition)

MASTER-MASTER

https://github.com/kwbln/system-design-primer#master-master-replication

- Both masters serve reads and writes and coordinate with each other on writes.
- If either master goes down, the system can continue to operate with both reads and writes.
- Disadvantage(s)
  - You'll need a load balancer or you'll need to make changes to your application logic to determine where to write.
  - Most master-master systems are either loosely consistent (violating ACID) or have increased write latency due to synchronization.
  - Conflict resolution comes more into play as more write nodes are added and as latency increases.
  - (+ general disadvantages of partition)





GENERAL REPLICATION DISADVANTAGE(S):

- There is a potential for loss of data if the master fails before any newly written data can be replicated to other nodes.

- Writes are replayed to the read replicas. If there are a lot of writes, the read replicas can get bogged down with replaying writes and can't do as many reads.
- The more read slaves, the more you have to replicate, which leads to greater replication lag.
- On some systems, writing to the master can spawn multiple threads to write in parallel, whereas read replicas only support writing sequentially with a single thread.
- Replication adds more hardware and additional complexity.

REDIS REPLICATION

https://redis.io/topics/replication

https://redislabs.com/blog/redis-manage-storage-replication/

Asynchronous replication, with asynchronous slave-to-master acknowledges of the amount of data processed

- A master can have multiple slaves.
- Slaves are able to accept connections from other slaves. Aside from connecting a number of slaves to the same master, slaves can also be connected to other slaves in a cascading-like structure. Since Redis 4.0, all the sub-slaves will receive exactly the same replication stream from the master.
- Redis replication is non-blocking on the master side. This means that the master will continue to handle queries when one or more slaves perform the initial synchronization or a partial resynchronization.

WHAT WOULD BE THE PERFECT DATABASE / DATABASE MODEL FOR YOUR SWT PET PROJECT IF YOU WOULD HAVE TO SCALE LARGE AND HAVING SOME 10.000 CLIENTS?

https://github.com/kwbln/system-design-primer#database

https://www.mongodb.com/scale/nosql-vs-relational-databases

I would use a Document store, because my pet project works with json files, e.g. MongoDB.

MongoDB Scaling:

- It’s much cheaper to scale a NoSQL database than a relational database because you can add capacity by scaling out over cheap, commodity servers.
- Relational databases, on the other hand, require a single server to host your entire database.
- To scale, you need to buy a bigger, more expensive server.

WHAT ARE THE ADVANTAGES AND DISADVANTEGES OF (WEB) CACHING?

https://github.com/kwbln/system-design-primer#cache

ADVANTAGES

Caching improves page load times and can reduce the load on your servers and databases.

Databases often benefit from a uniform distribution of reads and writes across its partitions. Popular items can skew the distribution, causing bottlenecks. Putting a cache in front of a database can help absorb uneven loads and spikes in traffic.

DISADVANTAGES

- Need to maintain consistency between caches and the source of truth such as the database through cache invalidation.
- Cache invalidation is a difficult problem, there is additional complexity associated with when to update the cache.
- Need to make application changes such as adding Redis or memcached.
- additional disadvantages depending to the caching type, e.g:
  - write-behind
    - There could be data loss if the cache goes down prior to its contents hitting the data store.
    - It is more complex to implement write-behind than it is to implement cache-aside or write-through.
  - refresh-ahead
    - Not accurately predicting which items are likely to be needed in the future can result in reduced performance than without refresh-ahead.

WHAT IS THE DIFFERENCE BETWEEN RPC AND REST?

https://github.com/kwbln/system-design-primer#remote-procedure-call-rpc

https://github.com/kwbln/system-design-primer#representational-state-transfer-rest

https://apihandyman.io/do-you-really-know-why-you-prefer-rest-over-rpc/

https://www.smashingmagazine.com/2016/09/understanding-rest-and-rpc-for-http-apis/

RPC	REST
Remote procedure call	Representational state transfer
In an RPC, a client causes a procedure to execute on a different address space, usually a remote server. The procedure is coded as if it were a local procedure call, abstracting away the details of how to communicate with the server from the client program. Remote calls are usually slower and less reliable than local calls so it is helpful to distinguish RPC calls from local calls.	REST is an architectural style enforcing a client/server model where the client acts on a set of resources managed by the server. The server provides a representation of resources and actions that can either manipulate or get a new representation of resources. All communication must be stateless and cacheable.
RPC style endpoints are great when you want only one job done well. This makes it useful to one or two app clients because it is niche a service. RPC endpoints can implement business logic inside the service, given that it only does one thing. This adds simplicity and clarity to the service.	For a REST endpoint, you must treat it like a resource that provides domain data. The reward is you are now segregating data into separate domains. This makes it useful for when you have any number of apps requesting data. This approach attempts to decouple data from application or business logic.
The answer to which style to implement is the usual “it depends.” A distributed large scale system may benefit from REST while a smaller monolithic one does not. MVC systems with a basic CRUD can benefit from RPC as long as there is little need to scale.	

HOW MANY CORES DOES THE STACKOVERFLOW SERVERS HAVE, WITH WHAT CHIP HZ AND HOW MANY MB L2 CACHE?

https://nickcraver.com/blog/2016/02/17/stack-overflow-the-architecture-2016-edition/

https://nickcraver.com/blog/2016/03/29/stack-overflow-the-hardware-2016-edition/

(Servers Running Stack Overflow & Stack Exchange Sites)

SQL SERVERS (STACK OVERFLOW CLUSTER)

2 DELL R720XD SERVERS, EACH WITH:

- Dual E5-2697v2 Processors (12 cores @2.7–3.5GHz each)
- 384 GB of RAM (24x 16 GB DIMMs)
- 1x Intel P3608 4 TB NVMe PCIe SSD (RAID 0, 2 controllers per card)
- 24x Intel 710 200 GB SATA SSDs (RAID 10)
- Dual 10 Gbps network (Intel X540/I350 NDC)

https://ark.intel.com/de/products/75283/Intel-Xeon-Processor-E5-2697-v2-30M-Cache-2-70-GHz-

SQL SERVERS (STACK EXCHANGE "…AND EVERYTHING ELSE" CLUSTER)

2 DELL R730XD SERVERS, EACH WITH:

- Dual E5-2667v3 Processors (8 cores @3.2–3.6GHz each)
- 768 GB of RAM (24x 32 GB DIMMs)
- 3x Intel P3700 2 TB NVMe PCIe SSD (RAID 0)
- 24x 10K Spinny 1.2 TB SATA HDDs (RAID 10)
- Dual 10 Gbps network (Intel X540/I350 NDC)
- 20 MB SmartCache

https://ark.intel.com/products/83361/Intel-Xeon-Processor-E5-2667-v3-20M-Cache-3-20-GHz-

WEB SERVERS

11 DELL R630 SERVERS, EACH WITH:

- Dual E5-2690v3 Processors (12 cores @2.6–3.5GHz each)
- 64 GB of RAM (8x 8 GB DIMMs)
- 2x Intel 320 300GB SATA SSDs (RAID 1)
- Dual 10 Gbps network (Intel X540/I350 NDC)
- 30MB SmartCache

https://ark.intel.com/de/products/75283/Intel-Xeon-Processor-E5-2697-v2-30M-Cache-2-70-GHz-

SERVICE SERVERS (WORKERS)

2 DELL R630 SERVERS, EACH WITH:

- Dual E5-2643 v3 Processors (6 cores @3.4–3.7GHz each)
- 64 GB of RAM (8x 8 GB DIMMs)
- 20 MB SmartCache

https://ark.intel.com/de/products/81900/Intel-Xeon-Processor-E5-2643-v3-20M-Cache-3-40-GHz-

1 DELL R620 SERVER, WITH:

- Dual E5-2667 Processors (6 cores @2.9–3.5GHz each)
- 32 GB of RAM (8x 4 GB DIMMs)
- 2x Intel 320 300GB SATA SSDs (RAID 1)
- Dual 10 Gbps network (Intel X540/I350 NDC)
- 15 MB SmartCache

https://ark.intel.com/de/products/64589/Intel-Xeon-Processor-E5-2667-15M-Cache-2-90-GHz-8-00-GT-s-Intel-QPI-

REDIS SERVERS (CACHE)

2 DELL R630 SERVERS, EACH WITH:

- Dual E5-2687W v3 Processors (10 cores @3.1–3.5GHz each)
- 256 GB of RAM (16x 16 GB DIMMs)
- 2x Intel 520 240GB SATA SSDs (RAID 1)
- Dual 10 Gbps network (Intel X540/I350 NDC)
- 25 MB SmartCache

https://ark.intel.com/de/products/81909/Intel-Xeon-Processor-E5-2687W-v3-25M-Cache-3-10-GHz-

ELASTICSEARCH SERVERS (SEARCH)

3 DELL R620 SERVERS, EACH WITH:

- Dual E5-2680 Processors (8 cores @2.7–3.5GHz each)
- 192 GB of RAM (12x 16 GB DIMMs)
- 2x Intel S3500 800GB SATA SSDs (RAID 1)
- Dual 10 Gbps network (Intel X540/I350 NDC)
- 20 MB SmartCache

https://ark.intel.com/de/products/64583/Intel-Xeon-Processor-E5-2680-20M-Cache-2-70-GHz-8-00-GT-s-Intel-QPI-

HAPROXY SERVERS (LOAD BALANCERS)

2 DELL R620 SERVERS (CLOUDFLARE TRAFFIC), EACH WITH:

- Dual E5-2637 v2 Processors (4 cores @3.5–3.8GHz each)
- 192 GB of RAM (12x 16 GB DIMMs)
- 6x Seagate Constellation 7200RPM 1TB SATA HDDs (RAID 10) (Logs)
- Dual 10 Gbps network (Intel X540/I350 NDC) - Internal (DMZ) Traffic
- Dual 10 Gbps network (Intel X540) - External Traffic
- 15 MB SmartCache

https://ark.intel.com/de/products/75792/Intel-Xeon-Processor-E5-2637-v2-15M-Cache-3-50-GHz-

2 DELL R620 SERVERS (DIRECT TRAFFIC), EACH WITH:

- Dual E5-2650 Processors (8 cores @2.0–2.8GHz each)

- 64 GB of RAM (4x 16 GB DIMMs)
- 2x Seagate Constellation 7200RPM 1TB SATA HDDs (RAID 10) (Logs)
- Dual 10 Gbps network (Intel X540/I350 NDC) - Internal (DMZ) Traffic
- Dual 10 Gbps network (Intel X540) - External Traffic
- 20 MB SmartCache

https://ark.intel.com/de/products/64590/Intel-Xeon-Processor-E5-2650-20M-Cache-2-00-GHz-8-00-GT-s-Intel-QPI-

GPU

How would you get a simple GPU "Hello World" Example run on Google or AWS? (theoretically. Not practically!)

HOW TO USE GOOGLE DEEP LEARNING VM IMAGES (NOT TESTED)

The Google Deep Learning images are a set of prepackaged VM images with a deep learning framework ready to be run out of the box, including e.g. :

- numpy, sklearn, scipy, pandas, …
- Jupyter environments (Lab and Notebook)
- Nvidia packages (GPU images only).

https://cloud.google.com/solutions/creating-a-virtual-gpu-accelerated-windows-workstation
https://cloud.google.com/deep-learning-vm/
https://cloud.google.com/deep-learning-vm/docs/quickstart-marketplace

https://blog.kovalevskyi.com/deep-learning-images-for-google-cloud-engine-the-definitive-guide-bc74f5fb02bc
https://blog.kovalevskyi.com/semi-managed-jupyter-lab-with-access-to-google-cloud-resources-cc6f9e439416

STEPS

1. Create an instance
    a. Go to the Google Cloud Platform (GCP) Marketplace
    b. Choose or create a project in GCP (with activated billing!)
    c. Go to the Deep Learning VM Marketplace
    d. Launch on compute engine + choose your project
    Hints:
        - not every GPU is available in every region
        - check the quotas and metrics to protect you from using to much resources
    e. Give your VM instance a name on the deployment page
    f. Customize your machine if necessary (core, memory, CPU, GPU)
    g. Choose an OS
    h. Choose a DL Image
    i. Set boot type and size, identity and API access
    j. Deploy
2. Access your instance
    a. Go to the VM instances
    b. Choose your VM and how you would like to access it
3. Start working with it
    a. Open cloud shell from GCP

b. Compute JupyterLab (take some time for installation and reboot)
c. Start JupyterLab via Button in the cloud shell
d. Play around for example with Google's BigQuery – coding examples you can find here

BASIC FUNCTIONS

1. Start the instance
    a. Go to the VM instances
    b. Select the checkbox next to DL VM
    c. Click start
2. Stop the instance
    a. Go to the VM instances
    b. Select the checkbox next to DL VM
    c. Click stop
3. Delete the instance
    a. Go to the Deployments page
    b. Select the checkbox next to DL VM
    c. Click delete