System Design Guidance Cheat Sheet
V2021.03.19
(Dr Yan Xu)

System Design Goals

- ➔ Simple, Easy to Understand
  - Follow standards (e.g. microservices, database)
- ➔ Easy to Extend
  - SOLID principle
- ➔ Fast
  - Bottleneck & improvements
- ➔ Secure
  - Respect secure & privacy

Existing Success Design Standards

- ➔ Microservices (*details: reference*)
  - Web API
- ➔ Monolish
  - Standalone Desktop App (e.g. Games)
  - Embedded App (limited internet)

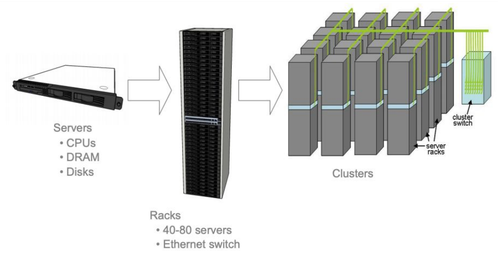
System Requirement

- ➔ Every system could be different
- ➔ MUST understand system requirements before system design:
  - Revenue critical components
    - Why CEO/CTO approves this system?
  - Future extension components
  - Relationship with existing systems
  - Team dependency and Ownership
- ➔ MUST understand system non-functional requirements:
  - Users
  - QPS
  - Storage
  - Budget

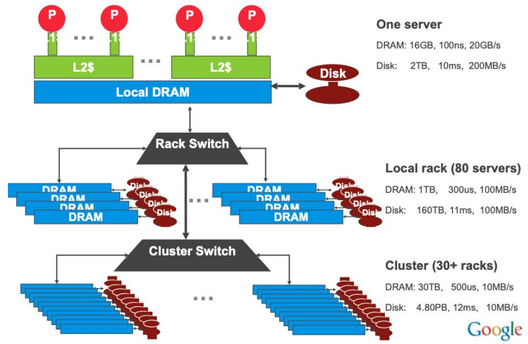
Time Unit

Unit	Equivalent
seconds (s)	1 s
milliseconds (ms)	10^{-3} s
microseconds (μs)	10^{-6} s
nanoseconds (ns)	10^{-9} s
picoseconds (ps)	10^{-12} s

Cloud Infrastructure



Speed Hierarchy (1)



(source: https://www.cs.cornell.edu/projects/ladis2009/talks/dean-keynote-ladis2009.pdf)

Speed Hierarchy (2)

L1 cache reference	0.5 ns
Branch mispredict	5 ns
L2 cache reference	7 ns
Mutex lock/unlock	25 ns
Main memory reference	100 ns
Compress 1K bytes with Zippy	3,000 ns
Send 2K bytes over 1 Gbps network	20,000 ns
Read 1 MB sequentially from memory	250,000 ns
Round trip within same datacenter	500,000 ns
Disk seek	10,000,000 ns
Read 1 MB sequentially from disk	20,000,000 ns
Send packet CA->Netherlands->CA	150,000,000 ns

(source: https://www.cs.cornell.edu/projects/ladis2009/talks/dean-keynote-ladis2009.pdf)

Database Options

QPS	Database
~1K	MySQL / PosgreSQL
~10-100K	NoSQL (e.g. MongoDB)
~1M	In-memory DB (e.g. Redis)
unlimited	Cloud Service (e.g. DynamoDB)

- QPS refers to standard small queries
- Different services could choose databases independently
- Other factors:
  - Data Size (Distributed vs Single DB)
  - Auto-scale, Replica
  - Complex Query and Index
  - Support Transactions?
  - Consistency Requirement

Backend Technology Options

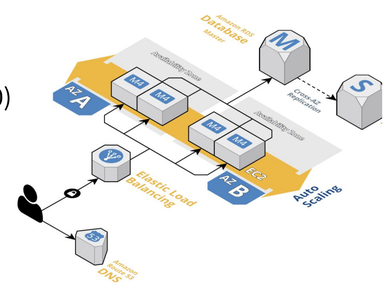
	App Types	Type System	Cross Platform	Community	Performance	Learning Curve
.NET	All	Static	No	Large	OK	Long
.NET Core	Web Apps, Web API, Console, Service	Static	Yes	Medium and growing rapidly	Great	Long
Java	All	Static	Yes	Huge	OK	Long
node.js	Web Apps, Web API	Dynamic	Yes	Large	Great	Medium
PHP	Web Apps, Web API	Dynamic	Yes	Large	OK -	Medium
Python	All	Dynamic	Yes	Huge	OK -	Short

Other factors:

1. Team Skills (e.g. Golang, Functional Language)

Network Infrastructure (AWS example)

- ➔ Routing
- ➔ Load Balancer
- ➔ Auto-Scaling
- ➔ Delay (p99, p90)



Web FrontEnd Options

Angular VS React

Angular	React
• Full-blown Framework	• UI-Centric Library
• Long learning curve	• Short learning curve

Mobile Technology Options

	Native	Hybrid	Cross Platform
Development Language & IDE	**iOS** – Objective-C or Swift, with X-Code & iOS SDK **Android** – Java with Android Studio & Android SDK	Thin wrapper around HTML, JavaScript, CSS	Xamarin (C#, Visual Studio) React Native (JavaScript)
Access to Phone's Features	Full control, no limits	Very limited	Catch-up with latest versions
User Experience	Exceptional	Inferior	Good, with limitations

Other Key Concepts

- ➔ Stateless
  - Being stateless is the key for scalability and easy design
  - Being stateless may make system diagram complex
- ➔ Server Warm Up
  - typically required to reduce initial delay
- ➔ Loose Coupling
  - Minimize the # of connections between services
  - Minimize the # endpoints between services
- ➔ Cache
  - **Time to live (TTL)**
  - Browser Cache, Cache Service, Server Cache, DB Cache

General System Design Advices (Microservices)

- ➔ Design for Product Iteration (or Feature Priority)
  - Do not image features in future
- ➔ Design for Growth (3X)
  - Scalability in Server and DB
  - Load balancing logic
- ➔ Design for Failure
  - Reasonable response to end users
  - Failover logic
- ➔ Sufficient Monitoring
  - HTML-based status
  - Performance
  - Error
  - Infrastructure