Software Architecture - Analyzing Architectures

Why Evaluate?

- Tells important properties of the system.

When to Evaluate?

- At the earliest stages, When choosing between architectures.

Techniques

- Presentation.
- Formal Reviews/Walkthroughs.
- Scenarios.
- Prototypes/POC.
- Skeleton Systems.

Presentation

- **Informal** explanation to stakeholders.
- Make the audience think deeply of the architectural decisions.
- Only a shallow level of analysis is possible.

Formal Reviews and Structured Walkthroughs

- Get a group to review.
- Has roles (Moderator, Presenter, Reviewer)
- Send the review items beforehand (early)

Evaluation using Scenarios

- **ATAM** (Architecture Trade-off Analysis Method)
- **CBAM** (Cost Benefits Analysis Method)
- **SAAM** (Software Architecture Analysis Method)
- **SBAR** (Scenario Based Architecture Re-
- Engineering)

Prototypes & Proof-of-Concept Systems

- Done to mitigate risks and user interfaces.
- Is a functional sub-set of the system.
- Sometimes can be sophisticated therefore expensive and time-consuming.

- **POC** - Code designed to prove that a risky part of the architecture is feasible.

Skeleton Systems

- For validation implements the system' structure but contains only a minimal subset of the functionality.

- Most expensive form of validation

Software Architecture Short- note (Scalability)

Factors

- **Scalability**
  - Number of users / sessions / transactions / operations the entire system can perform
- **Performance**
- **Responsiveness**
- **Availability**
- **Downtime Impact**
  - The impact of a downtime of a server/service/resource - number of users, type of impact etc
- **Cost**
- **Maintenance Effort**

Vertical Scaling

- Scaling up
- increasing resources without changing no of nodes.

- **Advantages**
  - Simple to implement
  - Easier + Quicker than re-designing the software

- **Disadvantages**
  - Finite limit
  - Hardware does not scale linearly (diminishing returns for each incremental unit)
  - **Requires downtime**
  - **Increases Downtime Impact**

Incremental costs increase exponentially.

Other Factors for Scalability

- Caching
- CDNs
- Asynchronous Communication

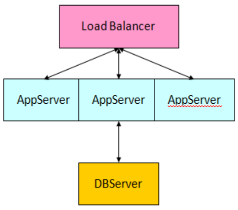
Vertical Partitioning

- Deploying each service on a separate node

- **Positives**
  - Increases per application Availability
  - Task-based specialization, optimization and tuning possible
  - No changes to App required
  - Flexibility increases

- **Negatives**
  - Sub-optimal resource utilization
  - May not increase overall availability
  - Finite Scalability

Horizontal Scaling



Scaling out, Load balancer could be HW, SW

Sticky Session

- Asymmetric load distribution
- Request of specific user always sent to same server
- **Downtime impact - Loss of session data.**

Central Session Storage

- Shared session store cluster
- Single point of failure
- Session read write Network, Disk I/O

Clustered Session Management

- No SPOF.
- Instant reads.
- Network I/O for writes and increase exponentially as nodes are added.
- Rare chance of stale data.

Database Replication

Master – Slave

- Writes are sent to a single master which replicates the data to multiple slave nodes (application needs to be changed)
- Simple setup
- No conflict management required
- Asynchronous/Synchronous

Multi-Master

- Writes can be sent to any of the multiple masters which replicate them to other masters and slaves
- Conflict Management required
- Deadlocks possible if same data is simultaneously modified at multiple places.

Data Partition

- Divide the data
- **Vertical Partitioning**
- Divide by tables/columns
- **Horizontal Partitioning**
- Divide by rows
- Joins, etc. are affected