Chapter 1 : Reliable, Scalable and Maintainable Applications

What is Data Intensive Application?

Amount of data and complexity of data is huge.

They devote most of their processing time to I/O operations as compared to compute intensive applications.

What is common in data intensive applications?

- Databases
- Caches
- Indexes
- Message streams
- Batch processing

What problems arise when we design data systems or service?

- Ensuring data remains correct and complete.
- Consistent good performance to clients.
- Handle increase in load.
- Defining the good API structure for your service.

Let’s jump into

RELIABILITY

SCALABILITY

MAINTAINABILITY

Reliability

- System should continue to work correctly in any fault.
- Performant under expected load and volume.
- Prevents any unauthorized access and abuse.

Fault vs Failure

- Fault is one component deviating from its specifications
- We can design a fault tolerant system.

- Failure is entire system stops providing the service.
- We cannot design a failure tolerant system.

Faults

- One component deviating from its specification.
- Fault tolerant / Resilient systems
- Hardware faults
  - Hard disk crash
  - RAM crash
  - Power grid blackout
- Software Errors
  - Software bug like a bad input
  - Service that becomes unresponsive
- Human Errors
  - Configuration errors

How can we make systems more Reliable?

- Well designed Abstractions, APIs, admin interfaces.
- Provide separate environment for testing and experimenting.
- Have unit, integration, manual and automation tests.
- Make systems easy to rollback in case of faults.
- Set up monitoring for performance metrics.

Scalability

System's ability to cope with increased load.

Load can be defined in terms of:

Number of requests per second

Ratio of reads vs writes

Number of concurrent users

Scalability problem in Twitter

- In Twitter, each user can follow multiple people.
- A person can be followed by multiple people.
- Let’s take A celebrity’s example – he/she might have a million followers.
- How would they update the timelines for all the users who follow this celebrity? It’s a fan-out problem.

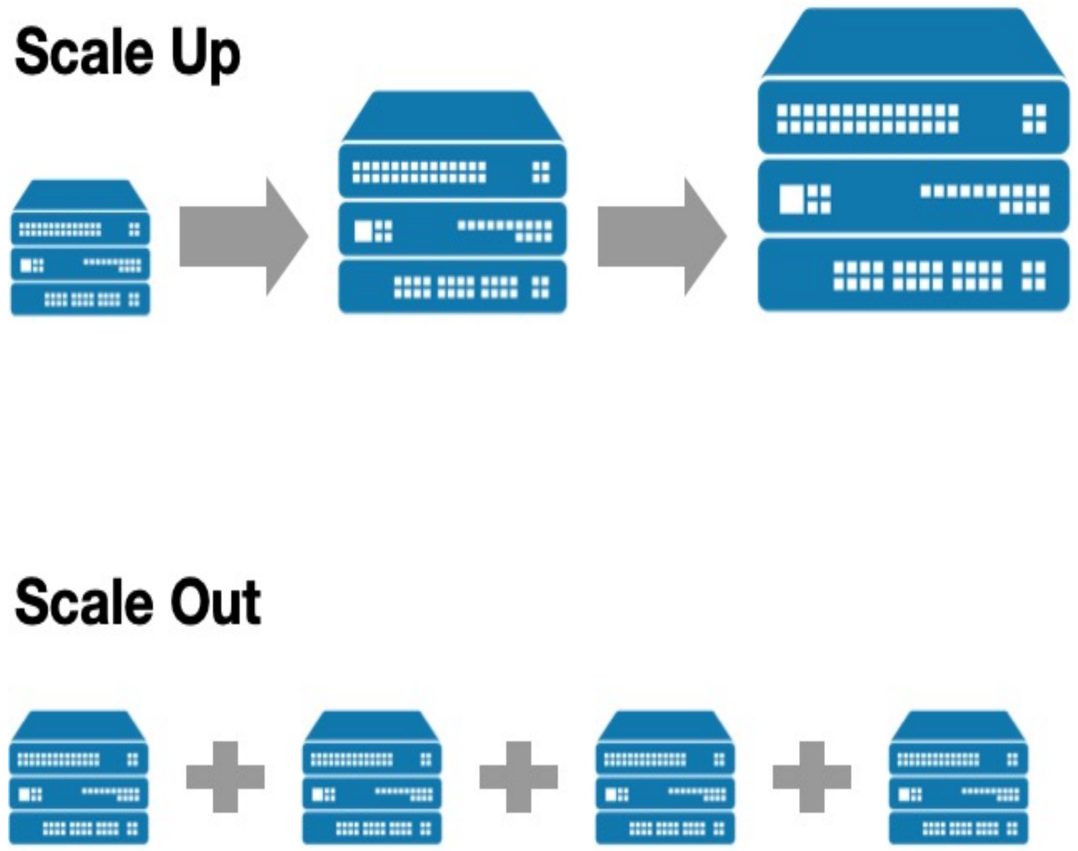
Performance

Latency: Request is waiting to be handled, during which it is latent, awaiting service.

Response time: Time what client sees, includes actual time to process the request, network delays and queuing delays.

Performance is measured by percentiles like TP90, TP99 etc.

Percentiles are used in SLA(Service level agreement), contracts that define the expected performance and availability of a service.



How to cope with load?

- Vertical Scaling
- Increasing the power/ a big machine.

- Horizontal Scaling
- Multiple smaller machines

Maintainability

Why is it important?

- Costly operation

What does it involve?

- Fixing bugs
- Investigating failures
- Adapting to new platforms
- Work with outdated systems

3 Design principles

Operability

Simplicity

Evolvability

Operability

- Monitoring the health of the system and quickly restoring.
- Tracking system failures and degraded performance.
- Keeping software up to date.
- Keeping track how one system affects the other.
- Establishing good deployment, configuration practices.
- Avoiding dependency on individual machines.
- Good documentation.

Simplicity

What is a complex system?

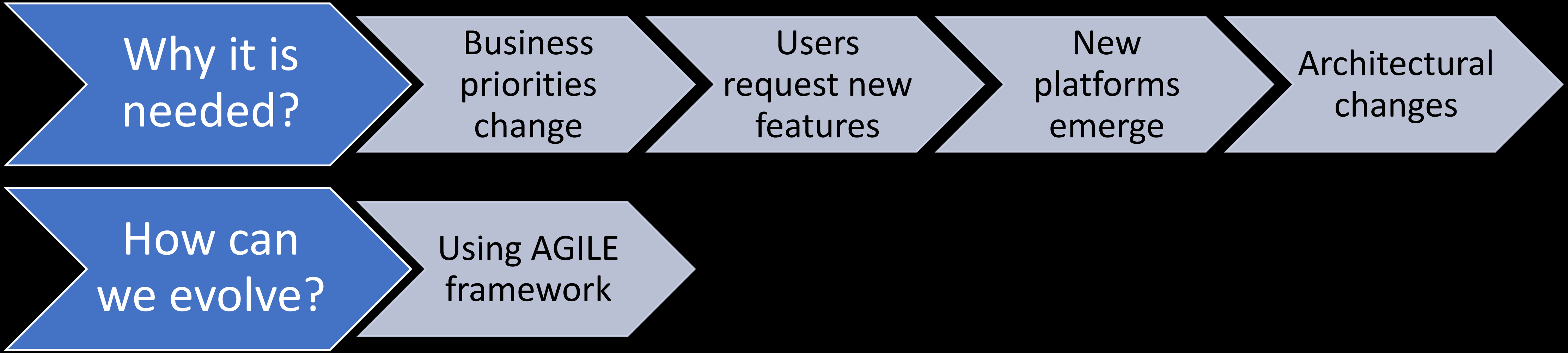
- Tight couple of modules
- Inconsistent terminology and naming conventions.
- Hacks
- Tangled dependencies

Why to make a simple system?

- Improved maintainability

How to remove complexity?

- Using abstraction



Evolvability

Making it easy for engineers to make changes into the system in the future.

Summary

- Reliability
- Scalability
- Maintainability

Thank You!