**Designing Data-Intensive Applications**

**Part I. Foundations of Data Systems**

1. **Reliable, Scalable, and Maintainable Applications**

A data-intensive application need databases, caches, search indexes, stream processing, batch processing.

**Thinking About Data Systems**

Factors that may influence the design of a data system:

* The skills and experience of the people involved
* Legacy system dependencies
* The time-scale for delivery
* The organization’s tolerance of different kinds of risk
* Regulatory constraints & etc.

**Reliability:**

Tolerating hardware & software faults, human errors. (fault-tolerant or resilient: continuing to work correctly, even when things go wrong)

* The application performs the function that the user expected.
* It can tolerate the user making mistakes or using the software in unexpected ways.
* Its performance is good enough for the required use case, under the expected load and data volume.
* The system prevents any unauthorized access and abuse.

Hardware Faults: Add redundancy to the individual components in order to reduce the failure rate of the system. When one component dies, the redundant component can take its place while the broken component is replaced. (Rather random and independent: Hard disks crash, RAM becomes faulty, the power grid has a blackout, someone unplugs the wrong network cable.)

Software Errors: Carefully thinking about assumptions and interactions in the

system; thorough testing; process isolation; allowing processes to crash and restart;

measuring, monitoring, and analyzing system behavior in production. (Harder to anticipate, and tend to cause many more system failures due strong correlations: software bug, runaway process, system that slows down, Cascading failures)

Human Errors: (i.e. configuration errors by operators)

* Design systems (well-designed abstractions, APIs, and admin interfaces)
* Decouple places (sandbox)
* Test thoroughly (Automated testing)
* Recover quick and easy (roll back configuration changes, roll out new code gradually and provide tools to recompute data)
* Set up detailed and clear monitoring (performance metrics and error rates)
* Implement good management practices and training.

**MINE:**

Through the reading of the first small part of the first chapter, I have learned many possible faults that may happened during an application developing or implementing and several reliable solutions to build a fault-tolerant system.

(2023.02.11)

---------------------------------------------------------------------------------------------------------------------------------

**Scalability:** Measuring load & performance, latency percentiles, throughput.

Describing Load

Describing Performance

Approaches for Coping with Load

**Maintainability:** Operability, Simplicity & Evolvability.

Operability: Making Life Easy for Operations

Simplicity: Managing Complexity

Evolvability: Making Change Easy

**Summary**

1. **Data Models and Query Languages**
2. **Storage and Retrieval**
3. **Encoding and Evolution**