Chapter #1 – Reliable, Scalable, and Maintainable Data Applications

- Data-intensive application uses a large amount of **dynamic** data and needs to accomplish various tasks such as:
    - Store data so that they, or another application, can find it again later (databases)
    - Remember the result of an expensive operation, to speed up reads (caches)
    - Allow users to search data by keyword or filter it in various ways (search indexes)
    - Send a message to another process, to be handled asynchronously (stream processing)
    - Periodically crunch a large amount of accumulated data (batch processing)

- Requires reliable, scalable, and maintainable data systems.

    - Reliability - The system should continue to work correctly (performing the correct function at the desired level of performance) even in the face of adversity (hardware or software faults, and even human error).

    - Scalability - As the system grows (in data volume, traffic volume, or complexity), there should be reasonable ways of dealing with that growth.

    - Maintainability - Over time, many different people will work on the system (engineering and operations, both maintaining current behavior and adapting the system to new use cases), and they should all be able to work on it productively.

Thinking About Data Systems

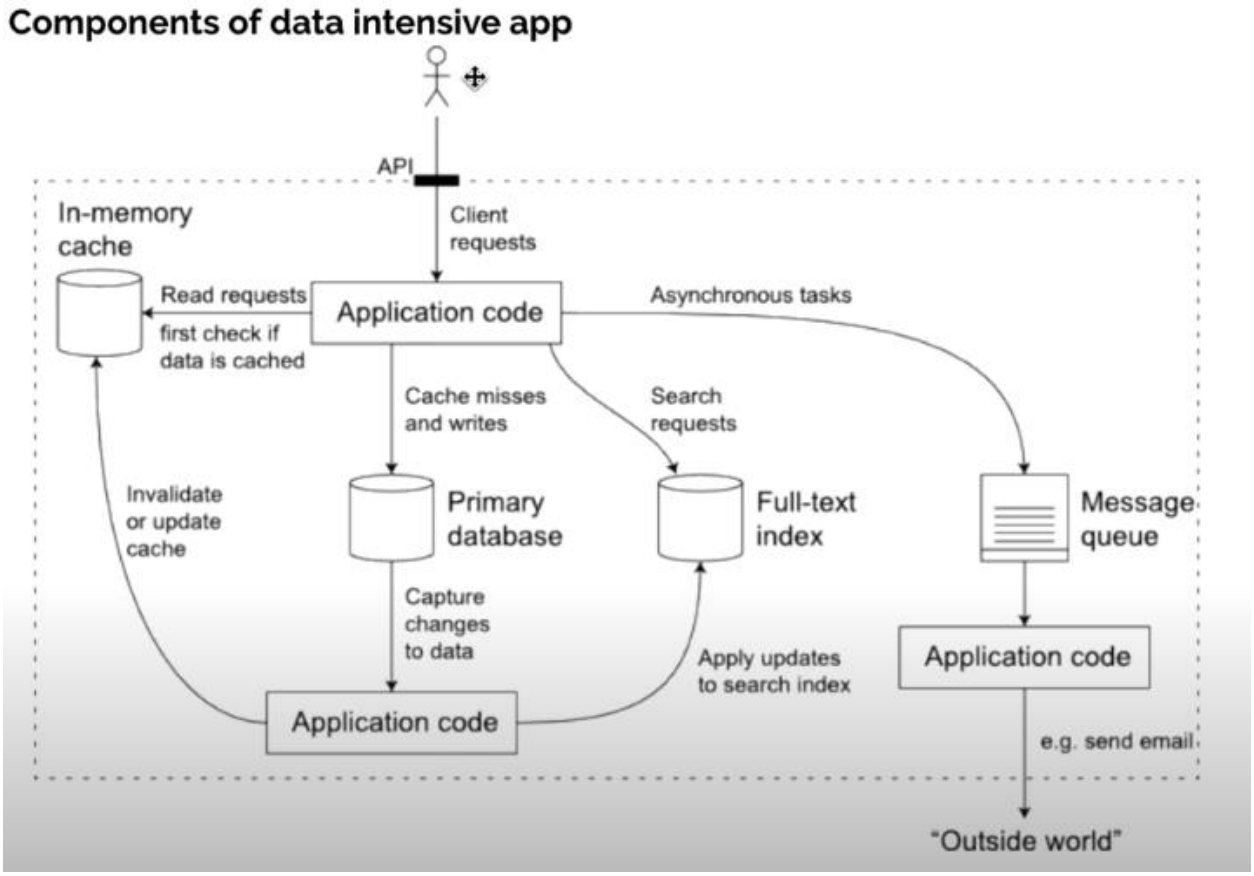

Figure 1-1. One possible architecture for a data system that combines several components.

Reliability

Reliability includes:

- Fault tolerance
- No unauthorized access
- Chaos testing
- Full machine failures
- Bugs – Automating tests
- Staging/testing environment
- Quick roll-back

Scalability

Scalability includes:

- Handle higher traffic volume
- Traffic load with peak # of reads, writes and simultaneous users
- Capacity planning
- Response time vs throughput
- End user response time
- 90^{th}+ percentile SLO/A server level objectives and agreements
- Scaling up (a few more powerful machines)
- Scaling out (many more smaller machines)

Maintainability

Maintainability includes:

- Operable: Configurable and testable
- Simple: easy to understand and ramp up
- Evolvable: easy to change