American University Kyiv

Teaching Strategies for Solution Architecture: Utilizing Open-Source Software Projects to Enhance Professional Learning

Ivan Manzhula
MSE | 2024

AUK Experience

Practical Cases on Architectural Katas

Group work and collaboration

Industry experience and knowledge sharing

Ivan Manzhula
MSE | 2024

Problem

Gap between industry and academia

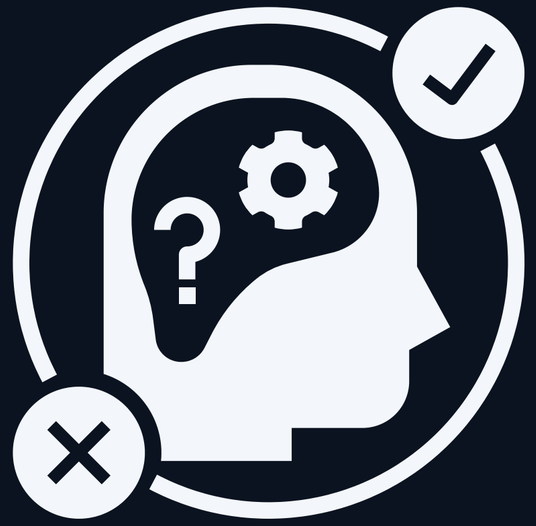
Lack of good teaching materials. Problems are usually explored in isolation, from the beginning ignoring pre-existing solution cases, redesign issues, assuming stable requirements. Reality differs significantly

Uncertainty

Complex domain research goes with non-obvious problems. Students usually face real tradeoffs, risks, demands from wide group of stakeholders after years of working experiences with high probability to fail first designs

Ivan Manzhula
MSE | 2024

Expectation



Software Architecture Course must combine modern theory and practice.

Students can gain experience working with systems larger than a few hundred lines of code.

Course in Software Architecture must emphasize soft skills such as brainstorming, facilitation, negotiation, leadership, decision making, risk analysis etc.

Ivan Manzhula
MSE | 2024

Literary Review

Huge theoretical base

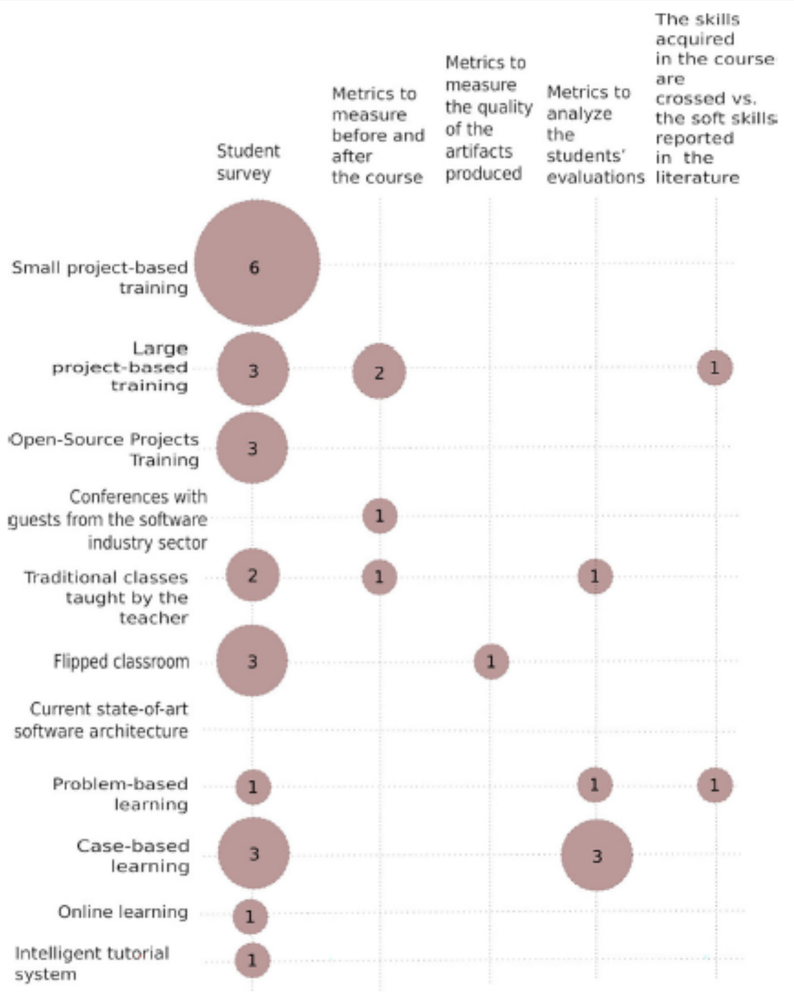
- Software Architecture in Practice
- Patterns of Software Architecture
- Software Architecture: Foundations, Theory, and Practice
- Just Enough Software Architecture: A Risk-Driven Approach
- Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives
- Design It! From Programmer to Software Architect
- Software Architecture: Visual Lecture Notes

Research Specific

- A Better Way to Teach Software Architecture
- Teaching Reform and Practice of Software Architecture for Postgraduates in Universities
- Training software architects suiting software industry needs

Ivan Manzhula
MSE | 2024

Teaching Strategies



EXAMPLE:

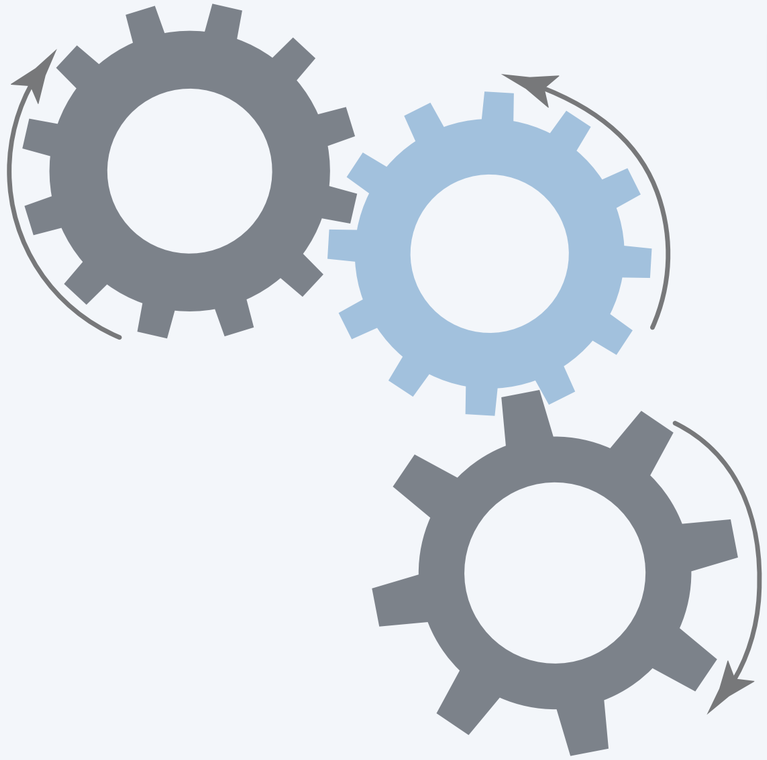
1. Divide students into groups / teams
2. Assign roles: senior / cognitive / junior architects
3. Senior architects - make decisions
4. Cognitive architects - challenge these decisions
5. Junior architects - model these decisions

Pantoja Yépez, W.L., Hurtado Alegría, J.A., Bandi, A. et al. Training software architects suiting software industry needs: A literature review. Educ Inf Technol (2023). https://doi.org/10.1007/s10639-023-12149-x

Use existing mature open-source projects

Pros 1

Real-world settings, long-term project allows study of changing requirements impact and architecture refactoring



Pros 2

Learning, recovering architecture can potentially help students find bugs or implement new features, thus contributing to the community and bring real industrial impact

Pros 3

Opportunity to communicate with open-source and industrial practitioners

Pros 4

Numerous design decisions within one project, observability of real treadoffs

Challenges

1. **Complexity**: can be resolved by proactive extraction of design decisions from source code and their usage for teaching examples

2. Course structure changes: new content organization and additional materials. Design decisions can be collected based on quality attributes, viewpoints, software development lyfecycles

Event Driven: Apache Kafka

Source Code: Java vs Scala

Complex Structure

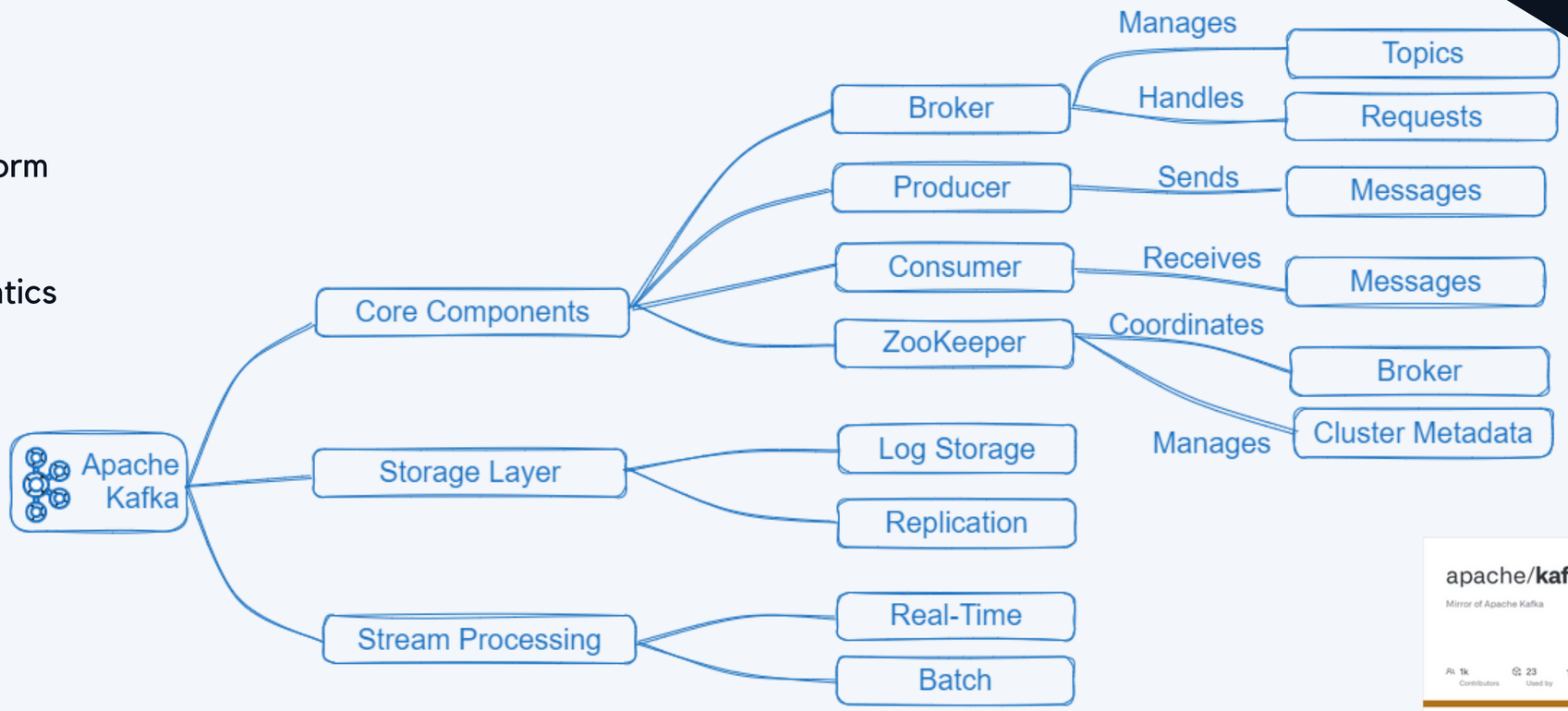
Multiple Design Patterns Combined

Popular and Widely Used

Distributed streaming platform

Topic-based data model

At least one delivery semantics



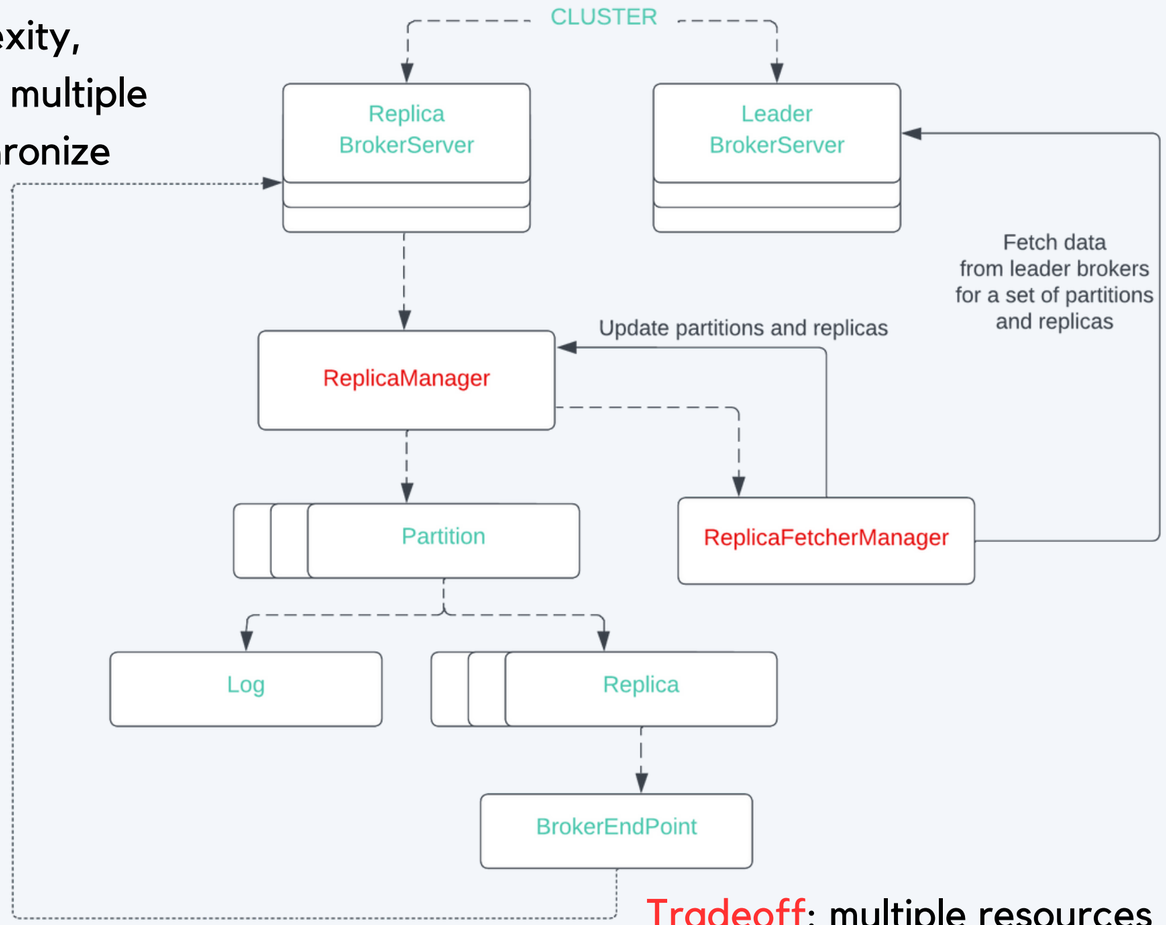
Ivan Manzhula

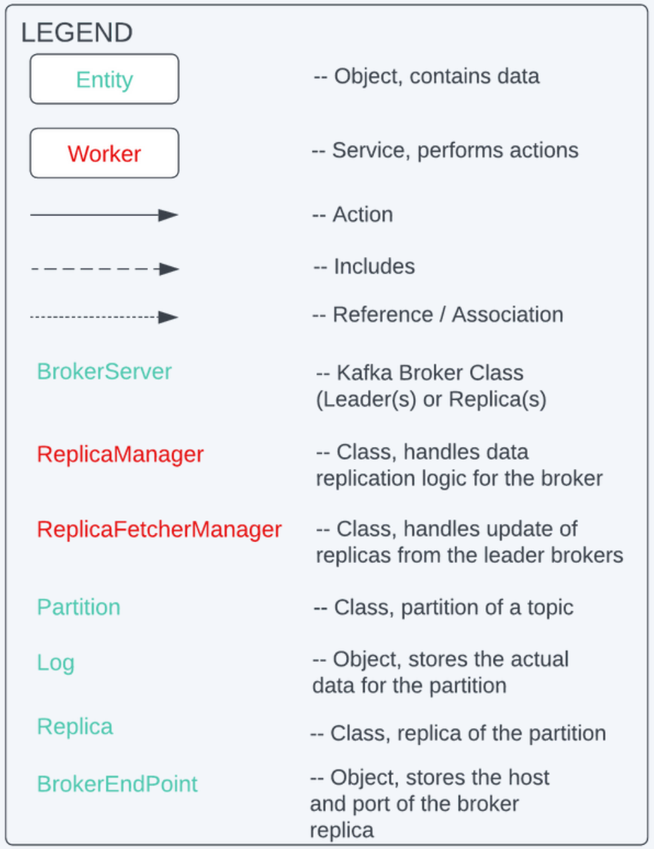
MSE | 2024

High Availability: Replication

Data is replicated across multiple brokers within the Kafka cluster to ensure data availability and prevent data loss in case of hardware failures

Tradeoff: complexity, necessity to manage multiple replicas and synchronize them



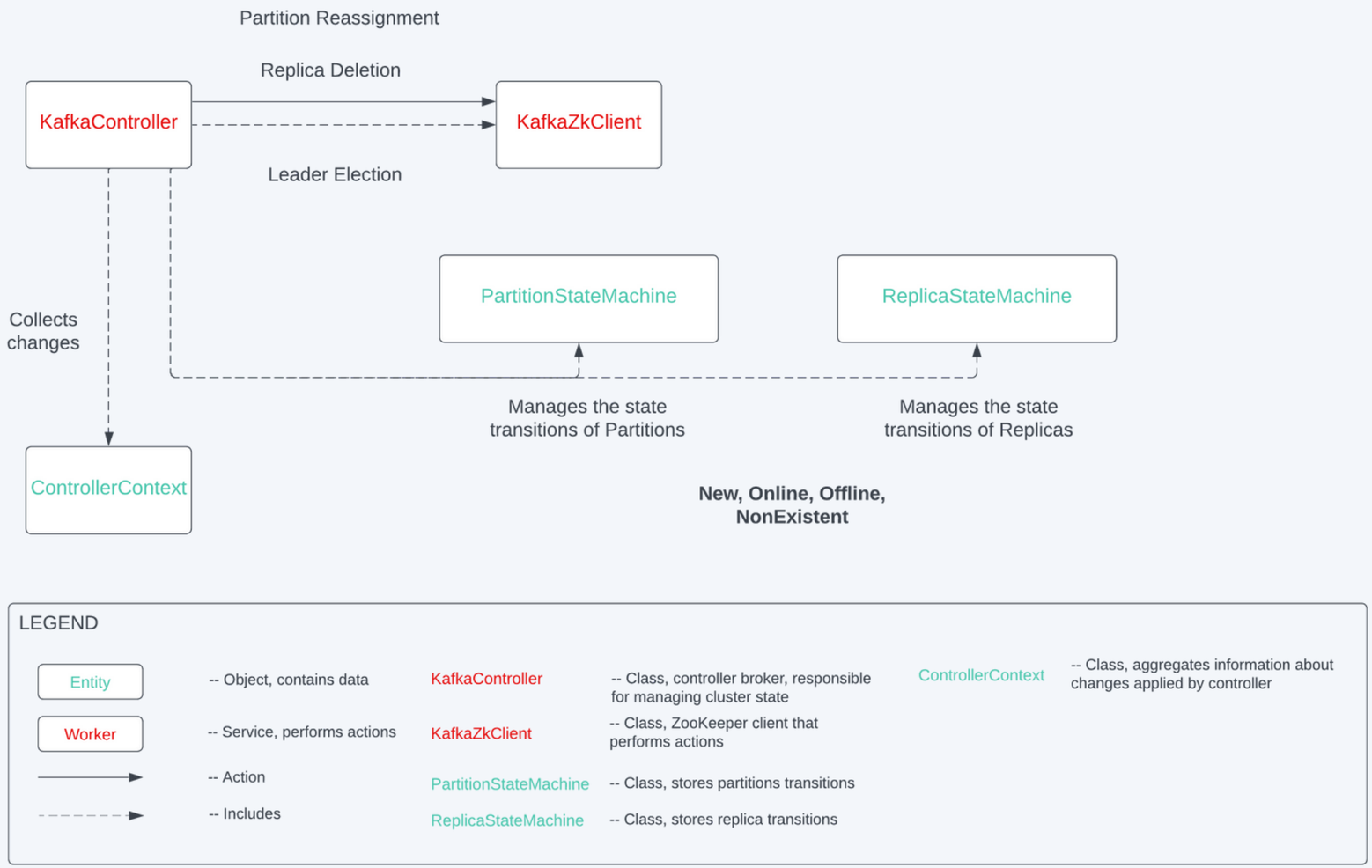


Tradeoff: multiple resources reflecting same information in many objects

High Availability: Leader election

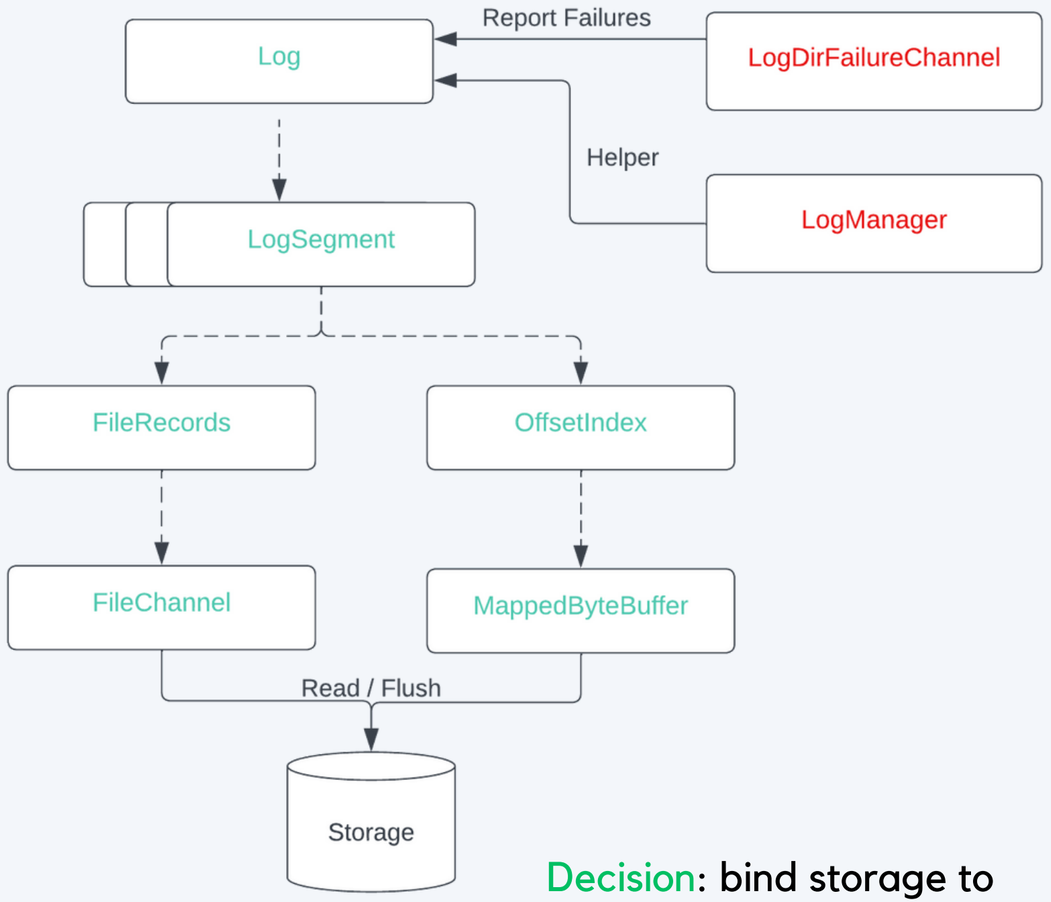
For each topic partition, a leader broker is responsible for replicating data to other followers and handling read/write requests. This ensures data availability and fault tolerance even if the leader fails.

Tradeoff: despite states and caches leader election can impact performance



High Availability: Data persistence

Data is written to disk and periodically flushed to ensure durability even if the system crashes



LEGEND

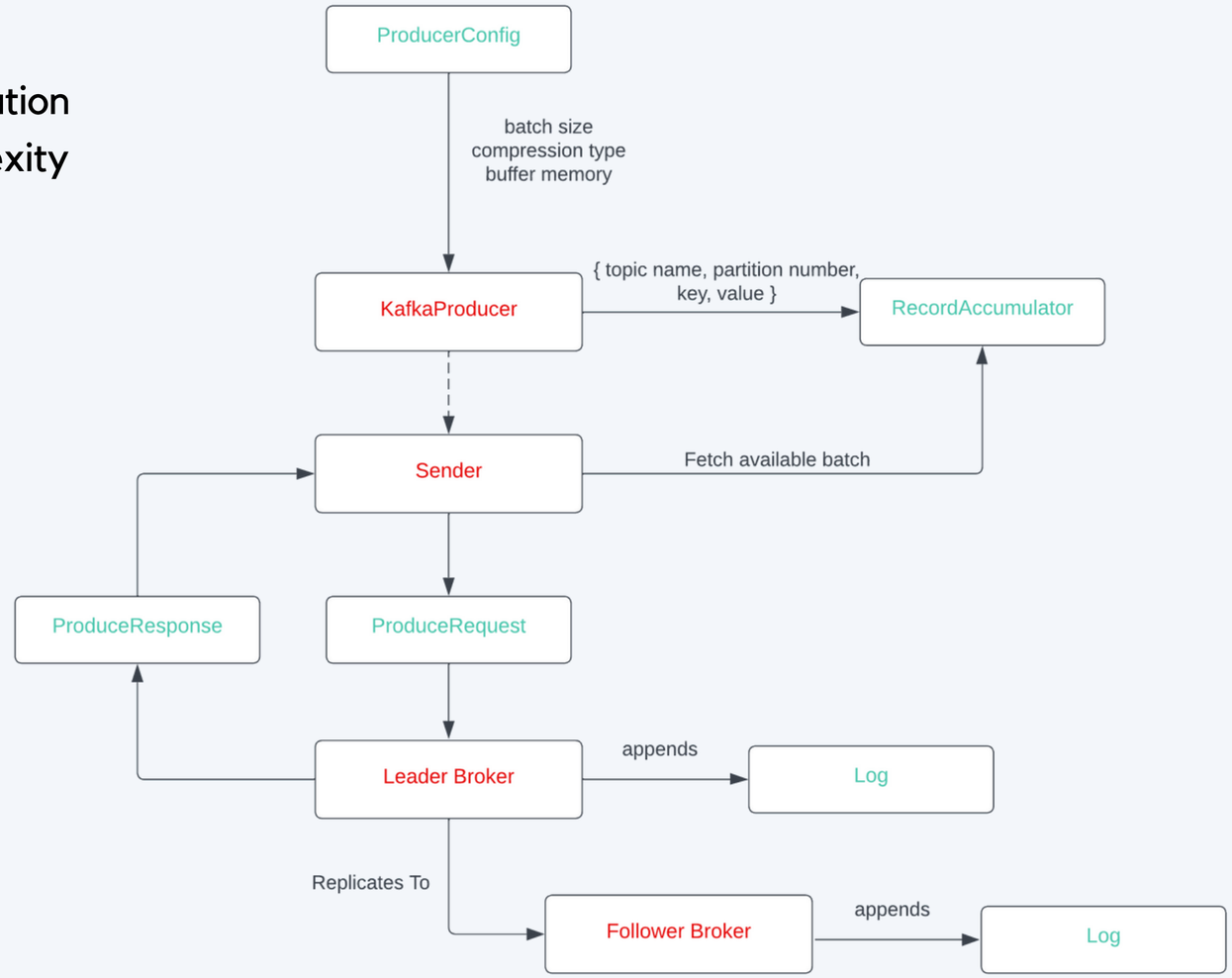
Entity	-- Object, contains data
Worker	-- Service, performs actions
→	-- Action
- - - →	-- Includes
LogDirFailureChannel	-- Object, reports any failure in the log directory, to handle retry
LogManager	-- Class, helper function to process log data
Log	-- Object, stores the actual data for the partition.
LogSegment	-- Class, represents a physical file that stores the data
FileRecords	-- Class, stores the actual record in a file channel
OffsetIndex	-- Class, stores the index of the offsets in the file
FileChannel	--Object, low-level IO operations on the file
MappedByteBuffer	--Object, memory mapped access to the file

Decision: bind storage to data object instead of using central resources

High Performance: Producer batching

Producers can batch data into larger messages for more efficient network transfer and processing

Tradeoff: communication and recovery complexity



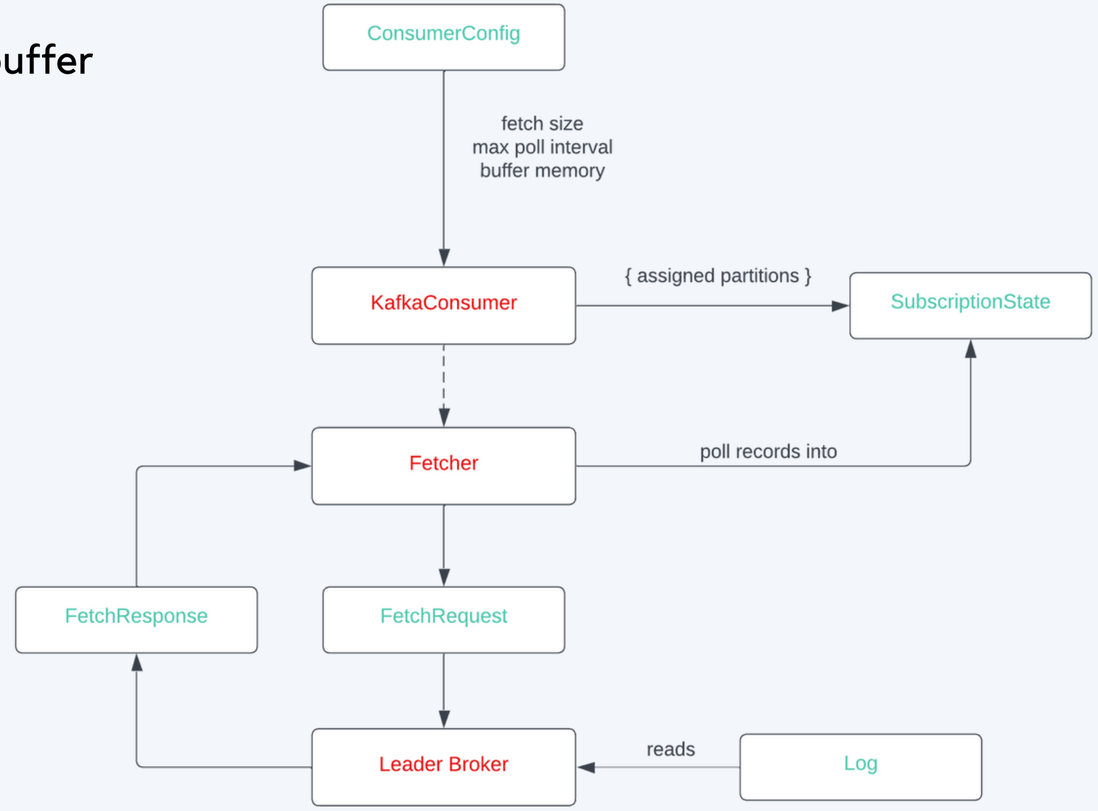
LEGEND

Entity	-- Object, contains data
Worker	-- Service, performs actions
⟶	-- Action
- - - - - - - ⟶	-- Includes
KafkaProducer	-- Class, main interface for sending data to Kafka topics
Sender	-- Object, handles actual sending of data to the brokers
Log	-- Object, stores the actual data for the partition.
RecordAccumulator	-- Object for batches of records that are ready to be sent to brokers
ProducerConfig	-- Object, stores configuration parameters for the producer
ProduceResponse	-- Object, contains status from the append operation
ProduceRequest	--Object, lcontains batches of records for each partition
Broker	--Class, handles producer request processing

High Performance: Consumer buffering

Consumers can buffer data to reduce the number of disk reads and improve consumer performance

Decision: use buffer



LEGEND

Entity	-- Object, contains data
Worker	-- Service, performs actions
→	-- Action
- - - - - - - →	-- Includes
KafkaConsumer	-- Class, main interface for consuming data from Kafka topics
Fetcher	-- Object, periodically requests records from leader brokers
Log	-- Object, stores the actual data for the partition.
SubscriptionState	-- Objec, stores subscription information of the consumer
ConsumerConfig	-- Object, stores configuration parameters for the consumer
FetchResponse	-- Object, contains records and offset for each partition
FetchRequest	--Object, lcontains topic names, partition ids and offsets of the records to be fetched
Broker	--Class, handles producer request processing

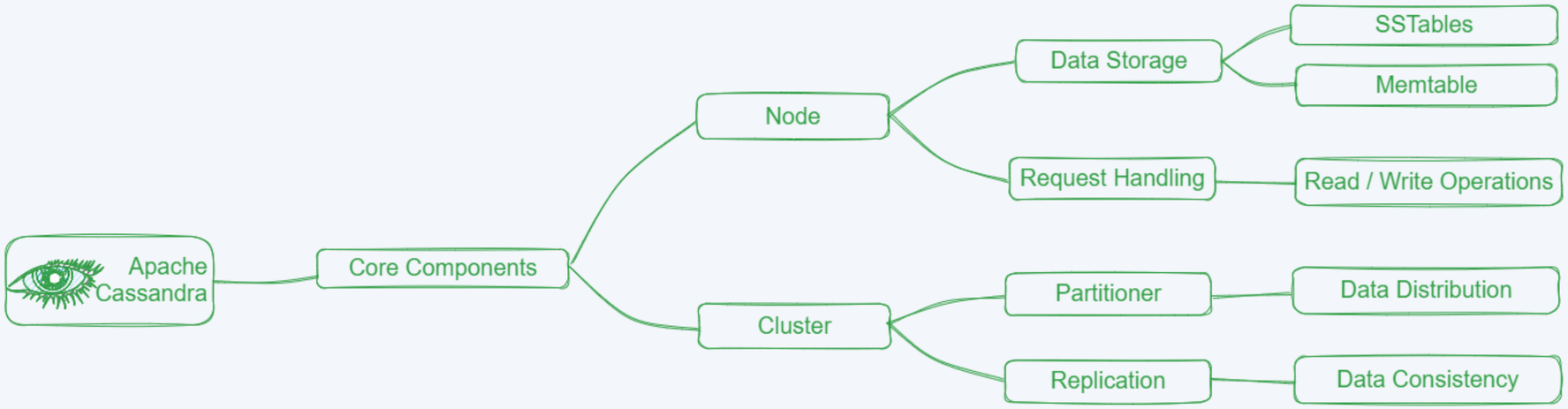
Distributed Database: Apache Cassandra

Source Code: Java

NoSQL Database

Row-based data model

Tunable consistency: eventual consistency <-> strong data integrity



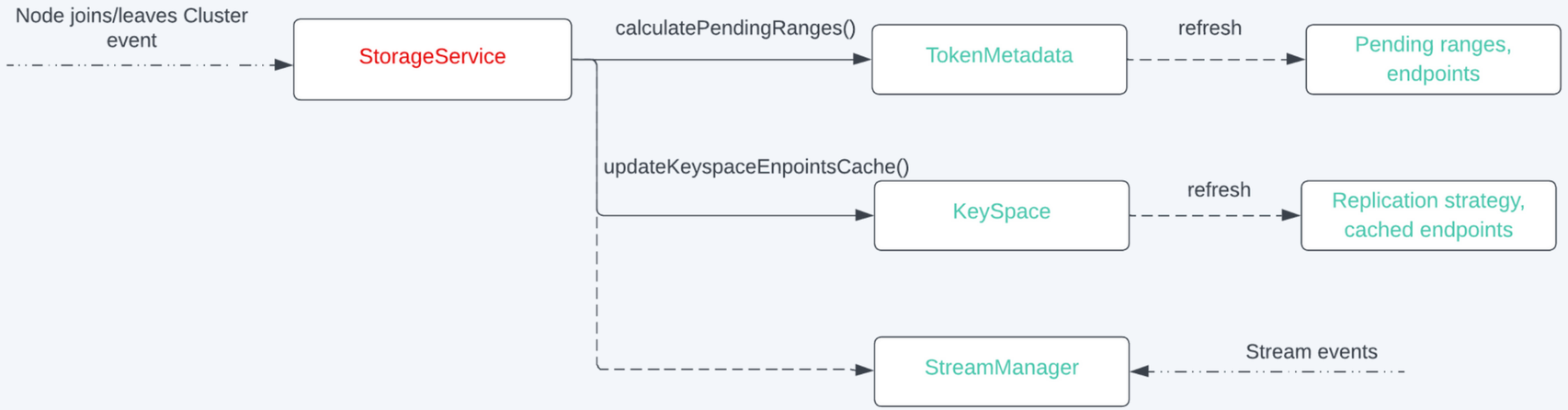
Ivan Manzhula
MSE | 2024

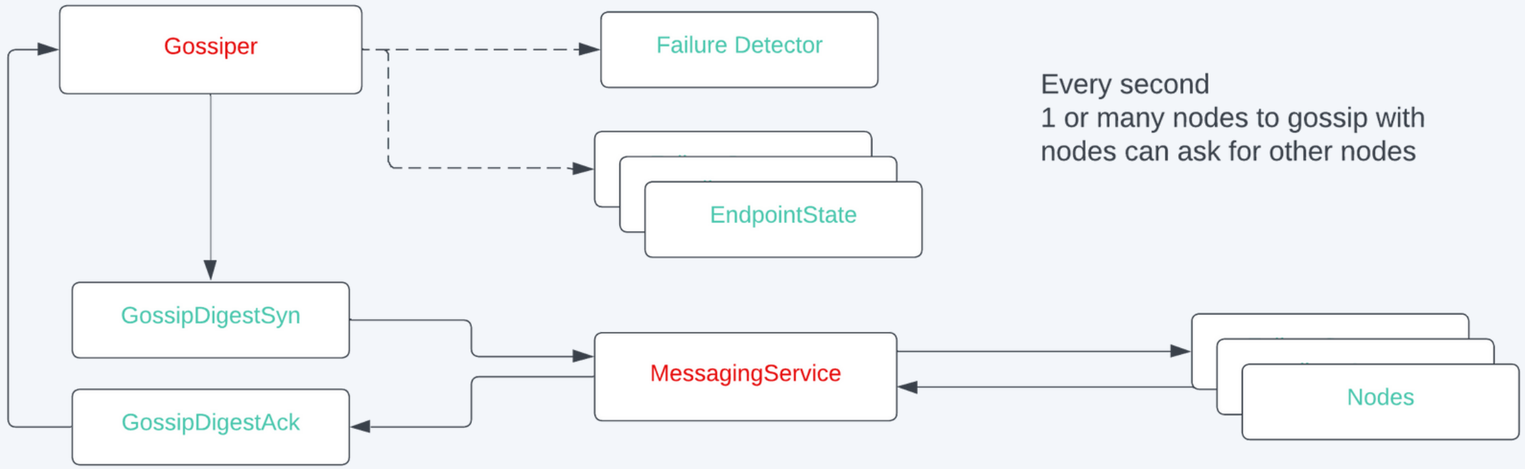
https://github.com/apache/cassandra

High Availability: Automatic failover and Gossip protocol

When a node fails, Cassandra automatically rebalances the data and workload across remaining nodes, minimizing downtime and data loss.
Nodes continuously communicate with each other through a gossip protocol, sharing information about their state and ensuring data consistency.

Tradeoff: temporary data inconsistency during failover; failover spikes when nodes handle increased workload



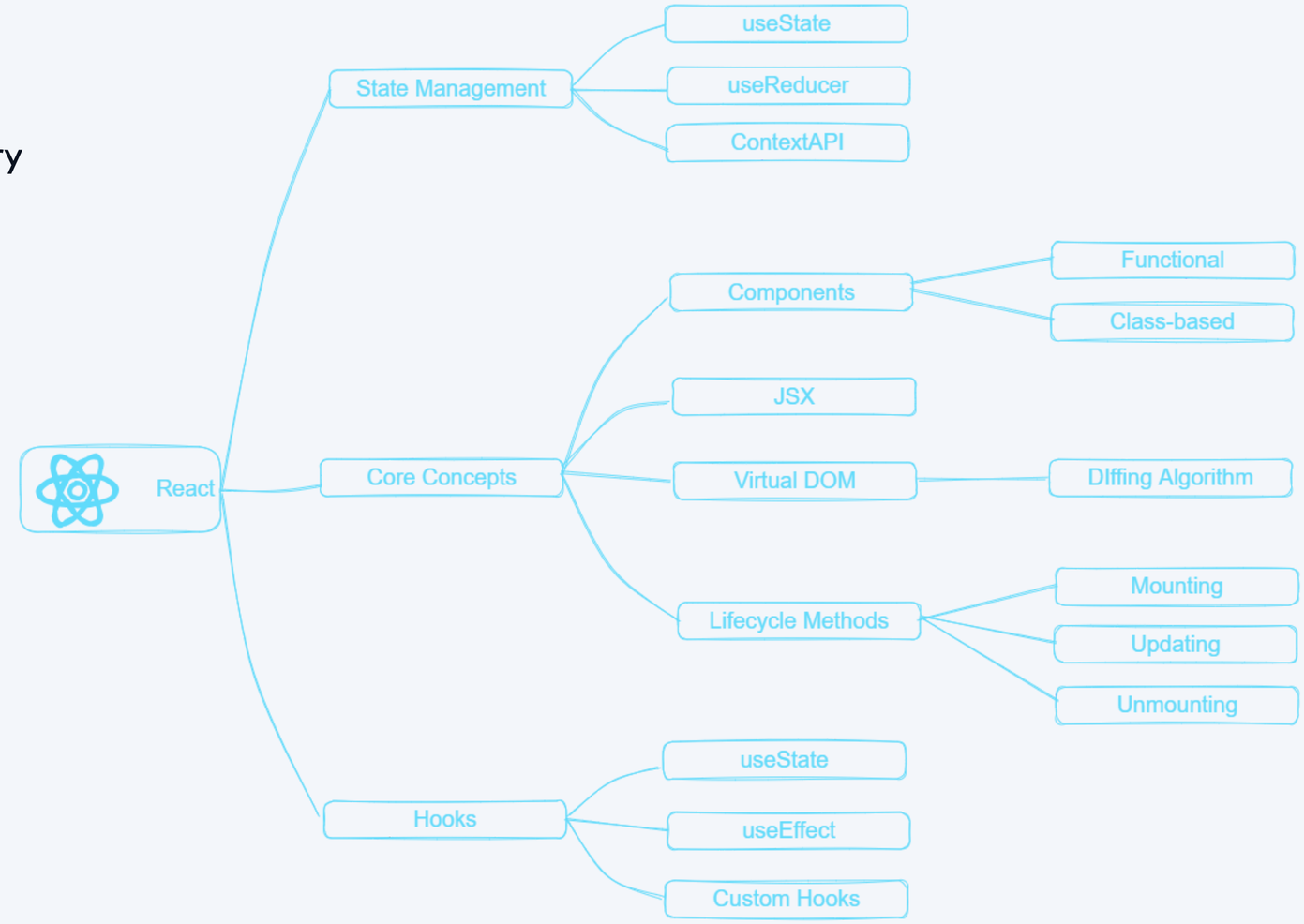


Every second
1 or many nodes to gossip with
nodes can ask for other nodes

UI Library: React

Source Code: JavaScript

Most Popular Front-End Library



Ivan Manzhula

MSE | 2024

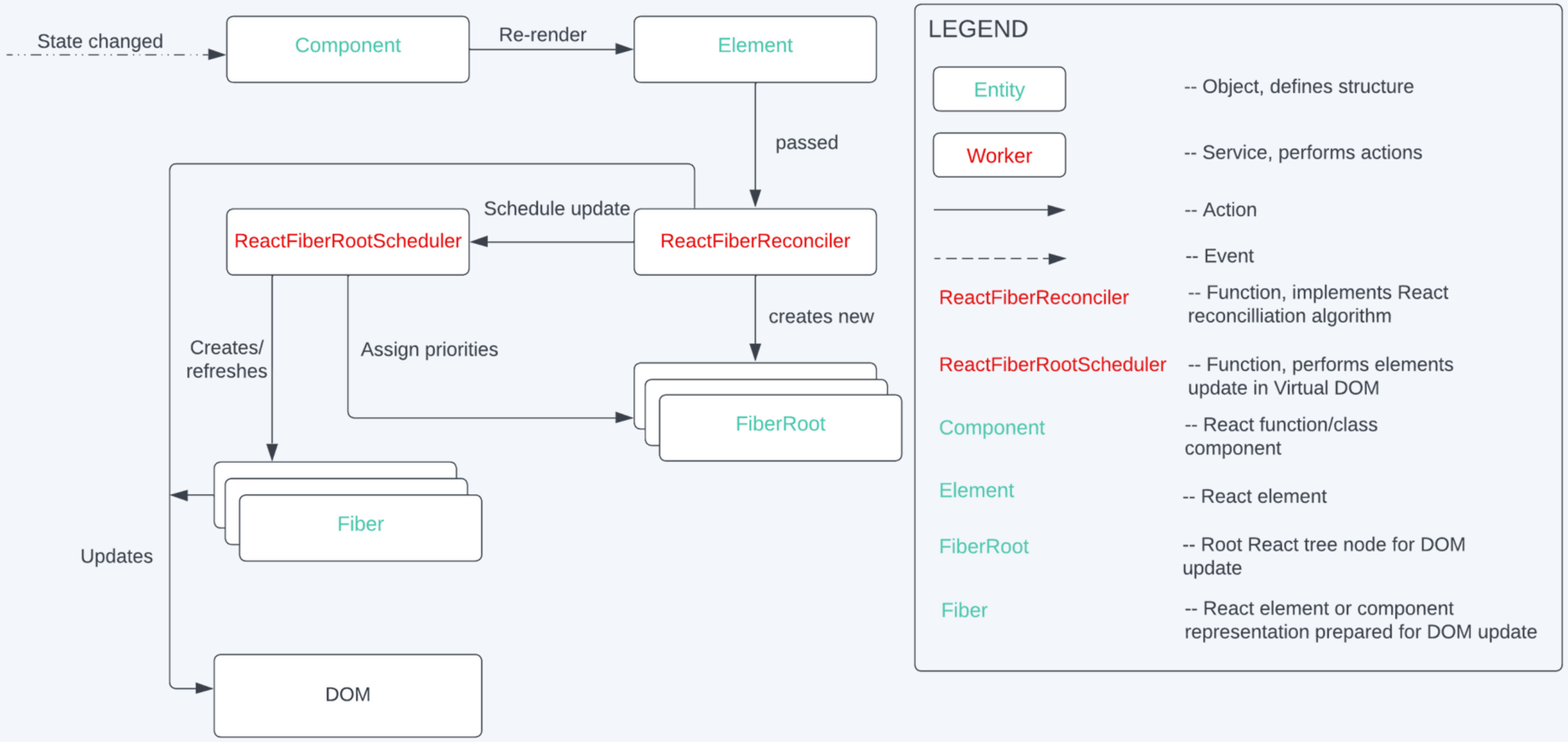
Performance: Virtual DOM

React uses a virtual DOM, which is an in-memory representation of the UI

When the state changes, React efficiently updates the virtual DOM and then re-renders only the parts of the UI that have changed

Decision: scheduling and updates based on Unit of Work

Tradeoff: Virtual DOM maintenance - requires more resources



American University Kyiv

Thank You

For Your Attention

Ivan Manzhula
MSE | 2024