Carnegie Mellon University

26 | Final Review + Systems Potpourri

Intro to Database Systems
15-445/15-645
Fall 2019

AP
Andy Pavlo
Computer Science
Carnegie Mellon University

ADMINISTRIVIA

Project #4: Tuesday Dec 10th @ 11:59pm

Extra Credit: Tuesday Dec 10th @ 11:59pm

Final Exam: Monday Dec 9th @ 5:30pm

FINAL EXAM

Who: You

What: http://cmudb.io/f19-final

When: Monday Dec 9th @ 5:30pm

Where: Porter Hall 100

Why: https://youtu.be/6yOH_FjeSAQ

FINAL EXAM

What to bring:

→ CMU ID
→ One page of handwritten notes (double-sided)
→ Extra Credit Coupon

Optional:

→ Spare change of clothes
→ Food

What not to bring:

→ Your roommate

COURSE EVALS

Your feedback is strongly needed:
→ https://cmu.smartevals.com

Things that we want feedback on:
→ Homework Assignments
→ Projects
→ Reading Materials
→ Lectures

OFFICE HOURS

Andy's hours:

→ Friday Dec 6th @ 3:30-4:30pm

→ Monday Dec 9th @ 1:30-2:30pm

All TAs will have their regular office hours up to and including Saturday Dec 14th

STUFF BEFORE MID-TERM

SQL
Buffer Pool Management
Hash Tables
B+Trees
Storage Models
Inter-Query Parallelism

TRANSACTIONS

ACID

Conflict Serializability:

→ How to check?
→ How to ensure?

View Serializability

Recoverable Schedules

Isolation Levels / Anomalies

TRANSACTIONS

Two-Phase Locking

→ Rigorous vs. Non-Rigorous
→ Deadlock Detection & Prevention

Multiple Granularity Locking

→ Intention Locks

TRANSACTIONS

Timestamp Ordering Concurrency Control
→ Thomas Write Rule

Optimistic Concurrency Control
→ Read Phase
→ Validation Phase
→ Write Phase

Multi-Version Concurrency Control
→ Version Storage / Ordering
→ Garbage Collection

CRASH RECOVERY

Buffer Pool Policies:
→ STEAL vs. NO-STEAL
→ FORCE vs. NO-FORCE

Write-Ahead Logging

Logging Schemes

Checkpoints

ARIES Recovery
→ Log Sequence Numbers
→ CLRs

DISTRIBUTED DATABASES

System Architectures

Replication

Partitioning Schemes

Two-Phase Commit

2018

Cockroach LABS	26
Google Spanner	25
mongoDB	24
Amazon Aurora	18
redis	18
cassandra	17
elasticsearch	12
HIVE	11
Scuba	10
MySQL	10

2019

Scuba	20
mongoDB	19
Cockroach LABS	18
Amazon Aurora	17
Google Spanner	17
snowflake	17
PostgreSQL	17
OceanBase	16
amazon REDSHIFT	15
elasticsearch	15

Scuba

facebook

FACEBOOK SCUBA

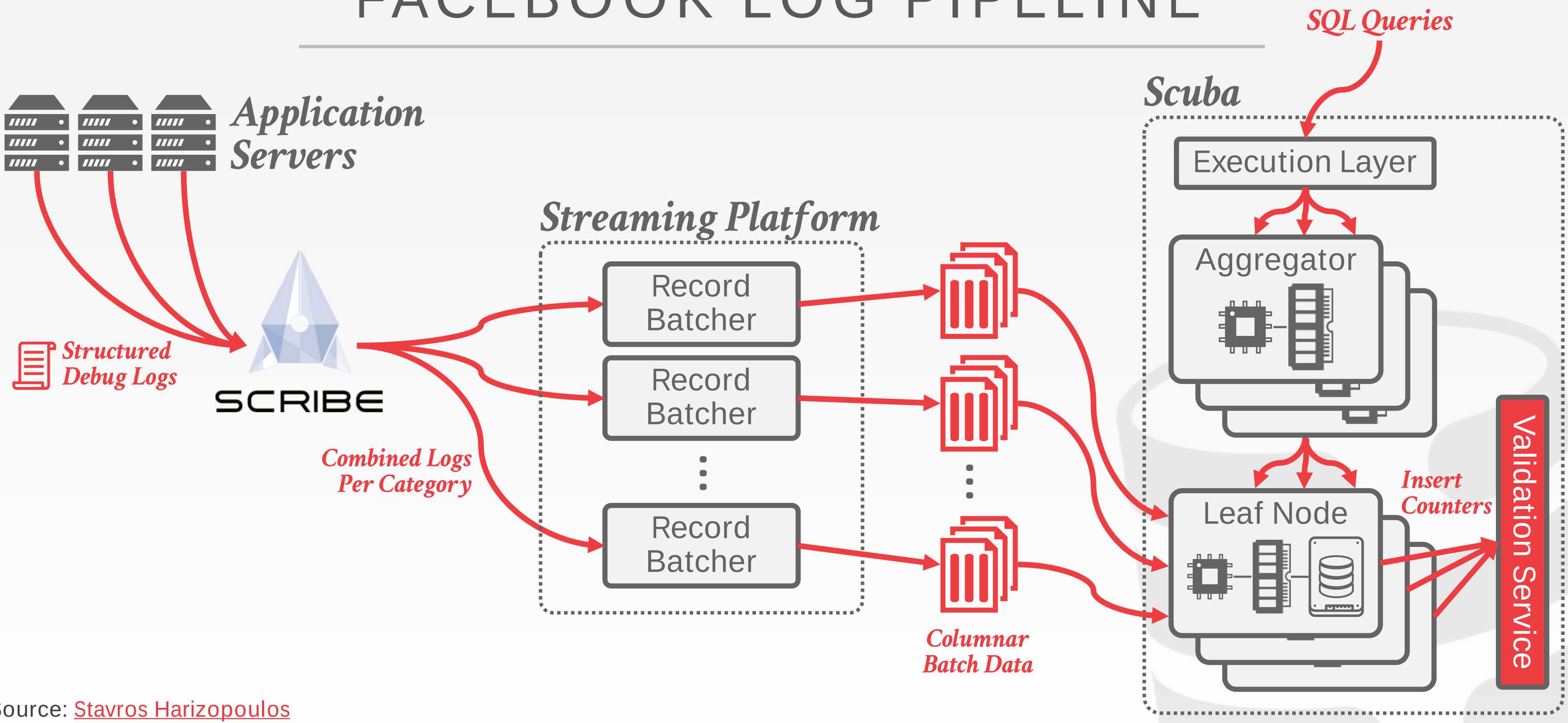
Internal DBMS designed for real-time data analysis of performance monitoring data.

→ Columnar Storage Model
→ Distributed / Shared-Nothing
→ Tiered-Storage
→ No Joins or Global Sorting
→ Heterogeneous Hierarchical Distributed Architecture

Designed for low-latency ingestion and queries.

Redundant deployments with lossy fault-tolerance.

FACEBOOK LOG PIPELINE



Source: Stavros Harizopoulos

SCUBA ARCHITECTURE

Leaf Nodes:

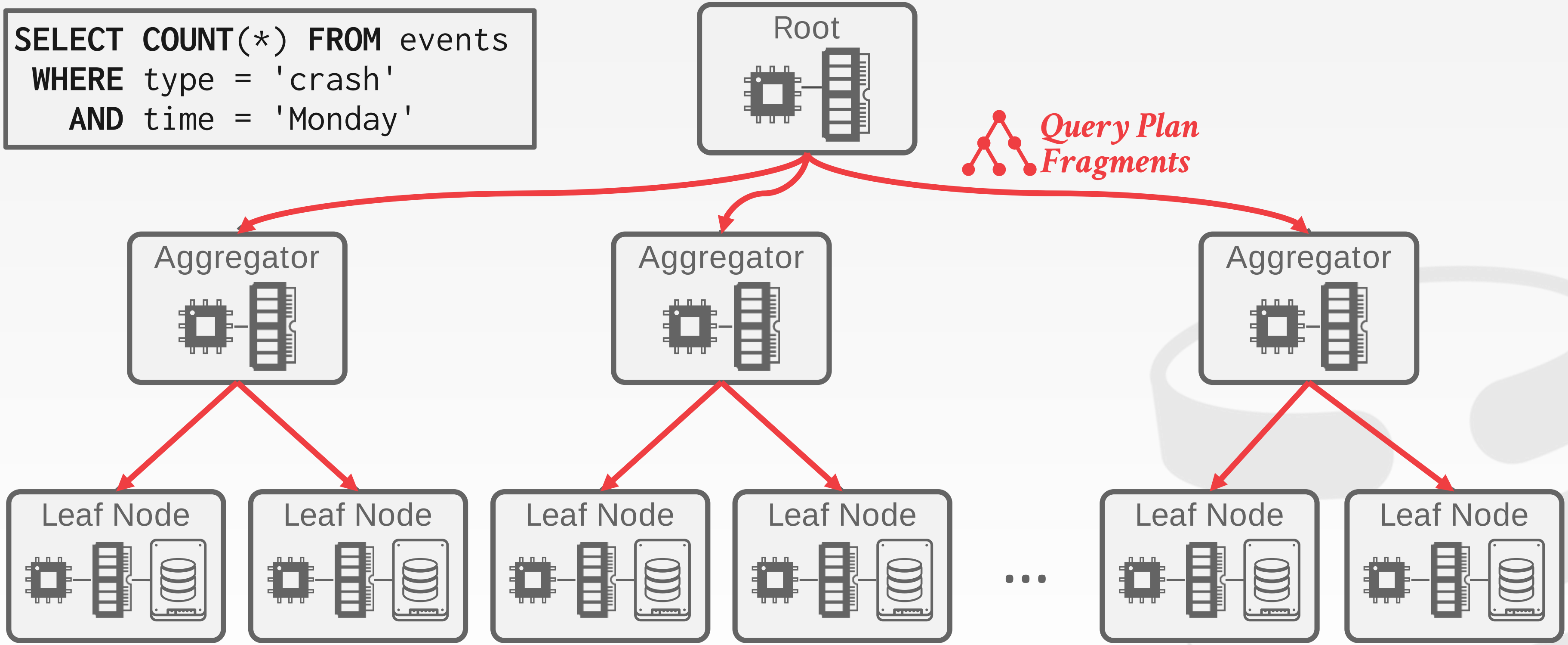
→ Store columnar data on local SSDs.
→ Leaf nodes may or may not contain data needed for a query.
→ No indexes. All scanning is done on time ranges.

Aggregator Nodes:

→ Dispatch plan fragments to all its children leaf nodes.
→ Combine the results from children.
→ If a leaf node does not produce results before a timeout, then they are omitted.

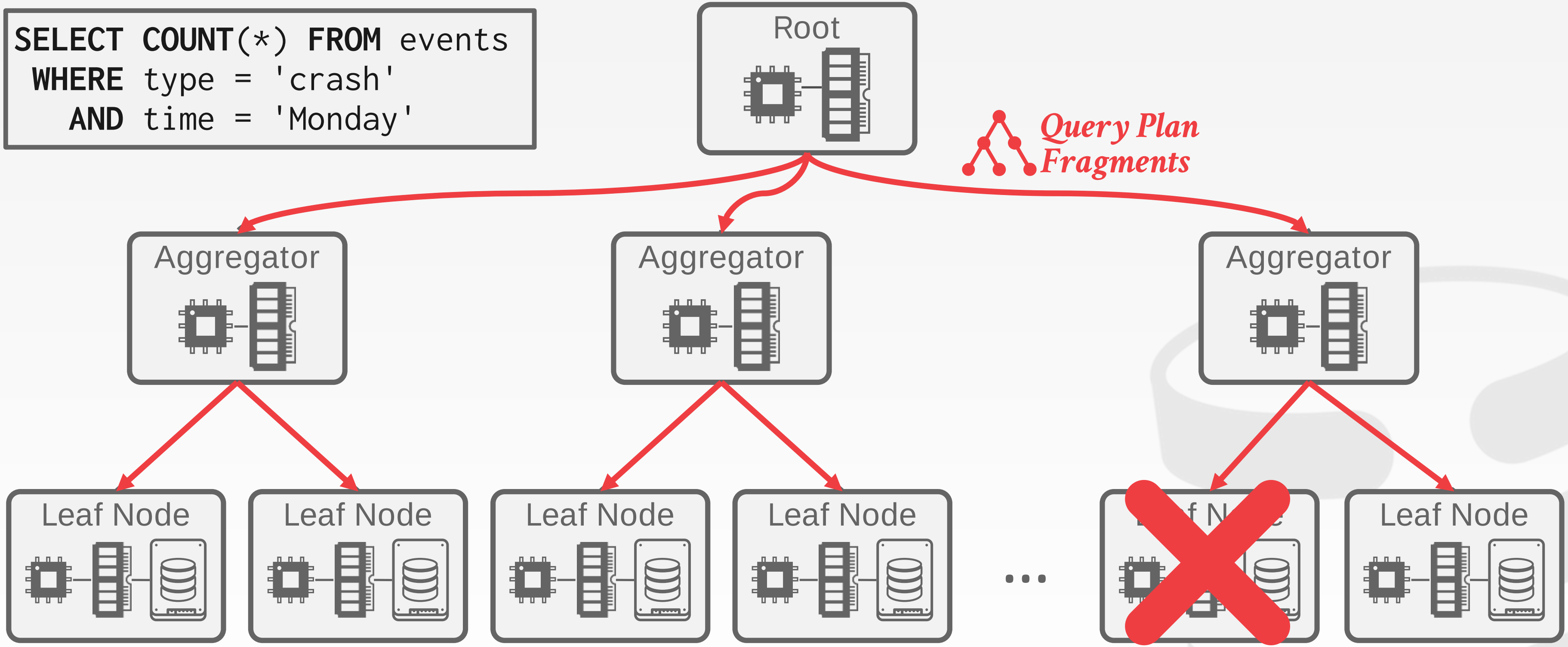
SCUBA ARCHITECTURE

```
SELECT COUNT(*) FROM events
 WHERE type = 'crash'
   AND time = 'Monday'
```



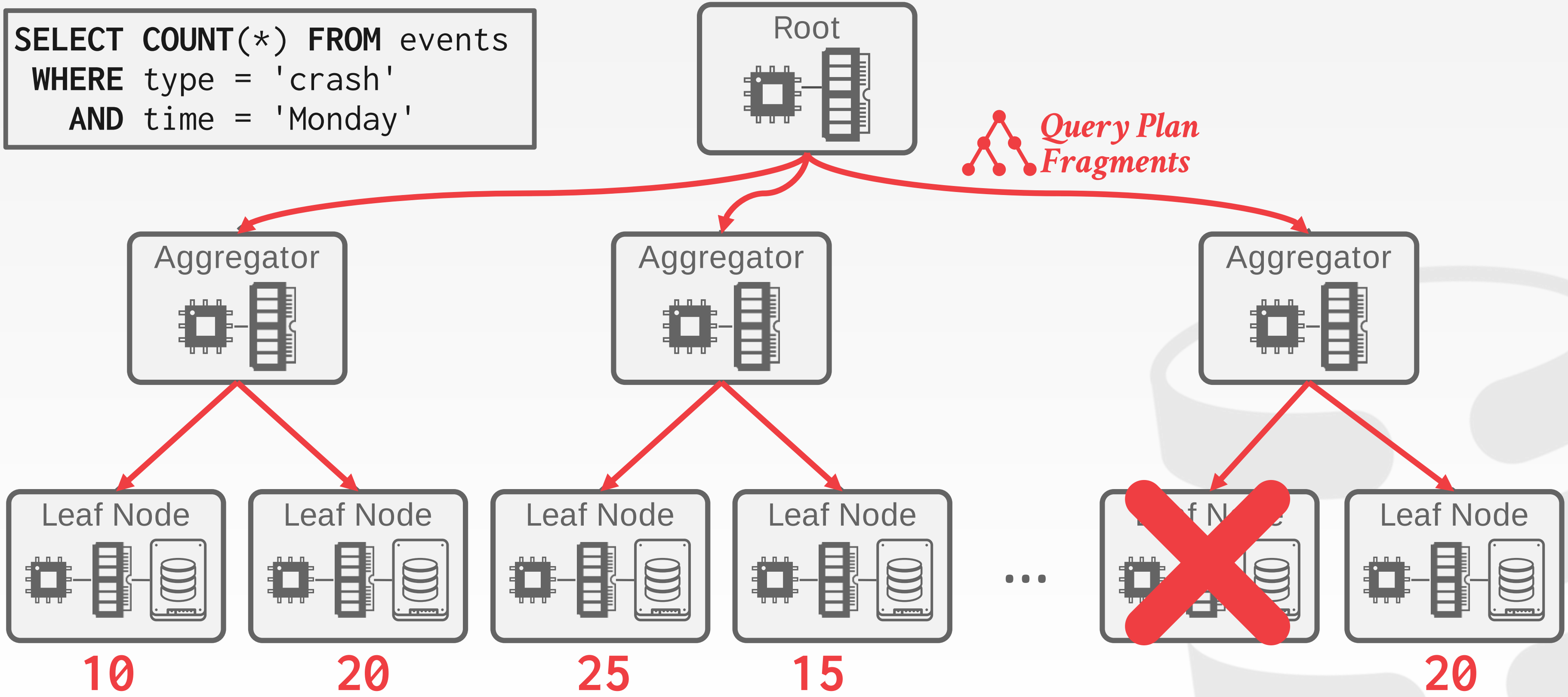
SCUBA ARCHITECTURE

```
SELECT COUNT(*) FROM events
 WHERE type = 'crash'
   AND time = 'Monday'
```



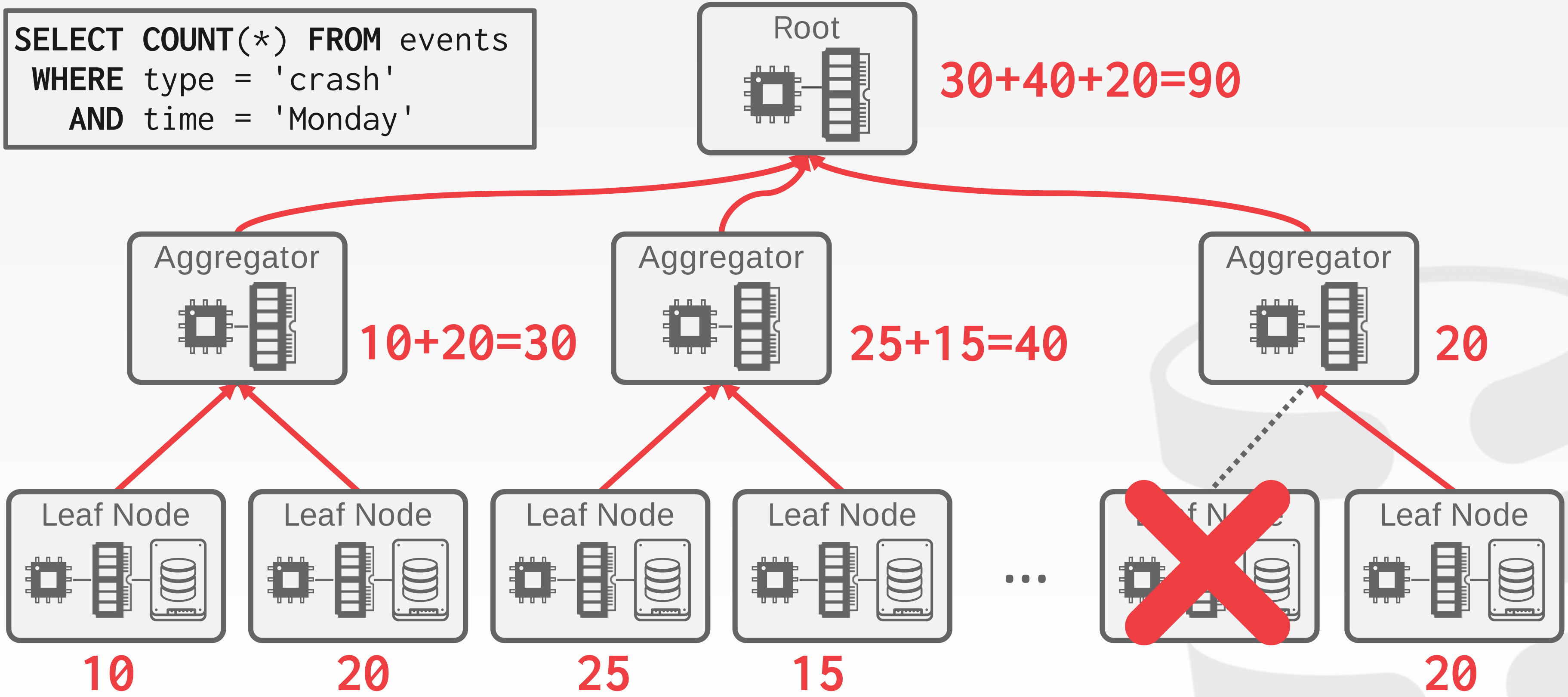
SCUBA ARCHITECTURE

```sql
SELECT COUNT(*) FROM events
 WHERE type = 'crash'
   AND time = 'Monday'
```



SCUBA ARCHITECTURE

```
SELECT COUNT(*) FROM events
 WHERE type = 'crash'
   AND time = 'Monday'
```



FAULT TOLERANCE

Facebook maintains multiple Scuba clusters that contain the same databases.

Every query is executed on all the clusters at the same time.

It compares the amount of missing data each cluster had when executing the query to determine which one produced the most accurate result.

→ Track the number of tuples examined vs. number of tuples inserted via Validation Service.

mongoDB

MONGODB

Distributed **document** DBMS started in 2007.

→ Document → Tuple

→ Collection → Table/Relation

Open-source (Server Side Public License)

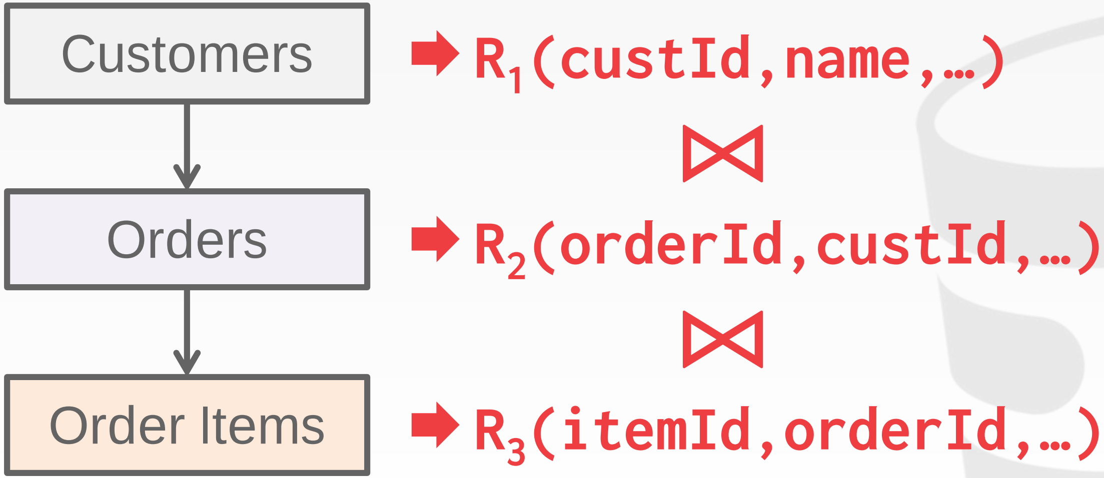
Centralized shared-nothing architecture.

Concurrency Control:

→ OCC with multi-granular locking

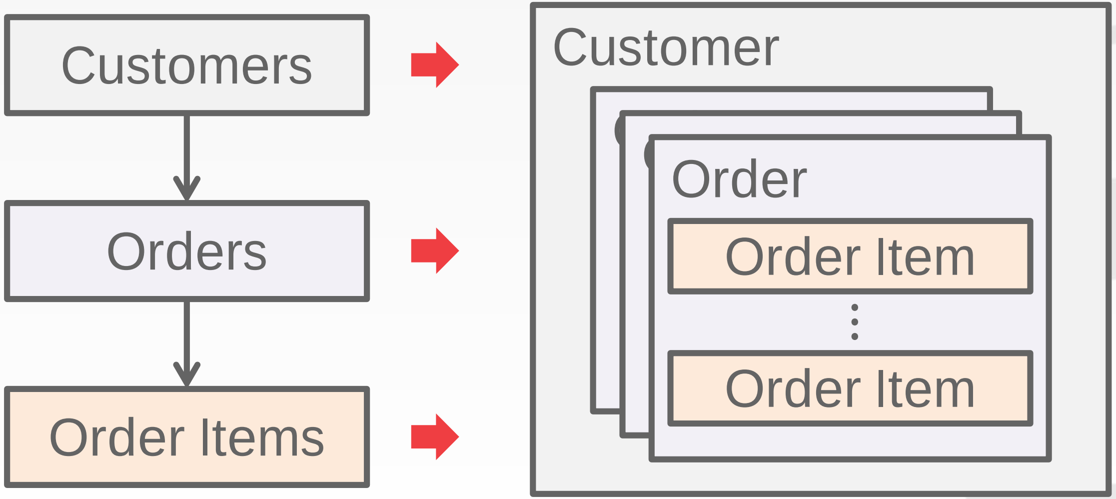
PHYSICAL DENORMALIZATION

A customer has orders and each order has order items.



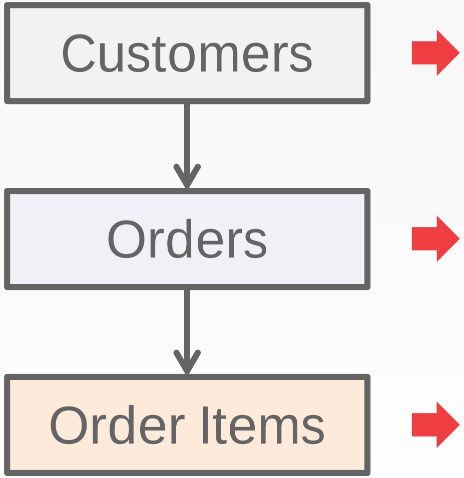
PHYSICAL DENORMALIZATION

A customer has orders and each order has order items.



PHYSICAL DENORMALIZATION

A customer has orders and each order has order items.




```
{
  "custId": 1234,
  "custName": "Andy",
  "orders": [
    { "orderId": 9999,
      "orderItems": [
        { "itemId": "XXXX",
          "price":  19.99 },
        { "itemId": "YYYY",
          "price":  29.99 },
      ] }
  ]
}
```

QUERY EXECUTION

JSON-only query API

No cost-based query planner / optimizer.
→ Heuristic-based + "random walk" optimization.

JavaScript UDFs (not encouraged).

Supports server-side joins (only left-outer?).

Multi-document transactions.

DISTRIBUTED ARCHITECTURE

Heterogeneous distributed components.

→ Shared nothing architecture

→ Centralized query router.

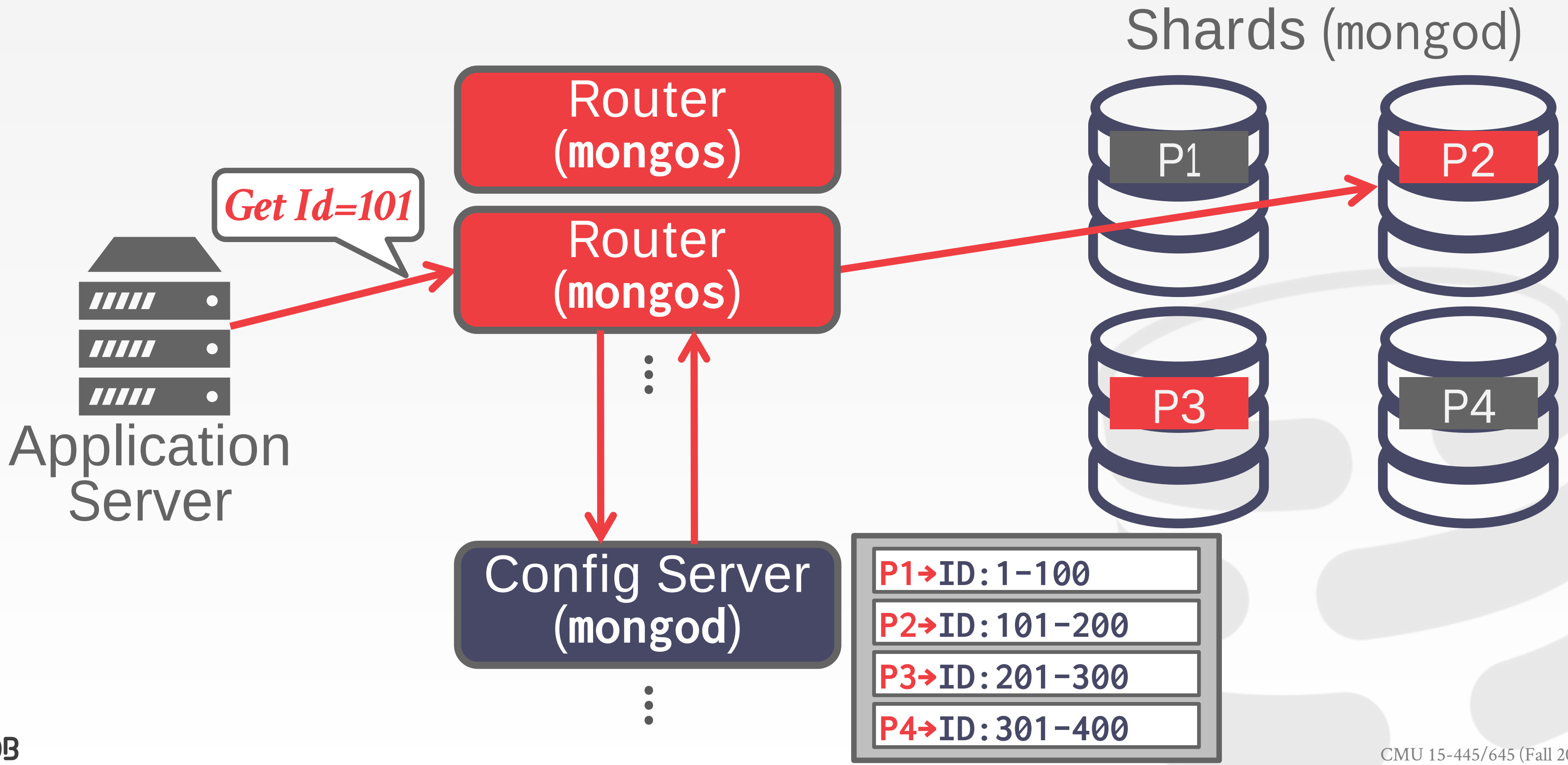
Master-slave replication.

Auto-sharding:

→ Define 'partitioning' attributes for each collection (hash or range).

→ When a shard gets too big, the DBMS automatically splits the shard and rebalances.

MONGODB CLUSTER ARCHITECTURE



STORAGE ARCHITECTURE

Originally used **mmap** storage manager
→ No buffer pool.
→ Let the OS decide when to flush pages.
→ Single lock per database.

MongoDB v3 supports pluggable storage backends
→ **WiredTiger** from BerkeleyDB alumni.
http://cmudb.io/lectures2015-wiredtiger
→ **RocksDB** from Facebook ("MongoRocks")
http://cmudb.io/lectures2015-rocksdb

Cockroach LABS

COCKROACHDB

Started in 2015 by ex-Google employees.

Open-source (BSL – MariaDB)

Decentralized shared-nothing architecture using range partitioning.

Log-structured on-disk storage (RocksDB)

Concurrency Control:

→ MVCC + OCC

→ Serializable isolation only

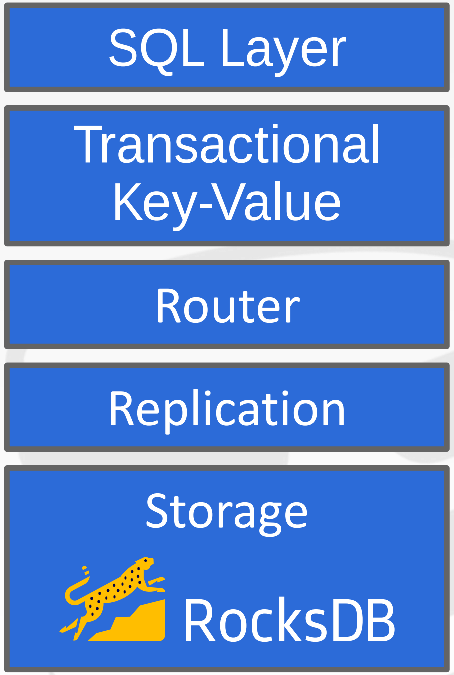
DISTRIBUTED ARCHITECTURE

Multi-layer architecture on top of a replicated key-value store.

→ All tables and indexes are store in a giant sorted map in the k/v store.

Uses RocksDB as the storage manager at each node.

Raft protocol (variant of Paxos) for replication and consensus.



CONCURRENCY CONTROL

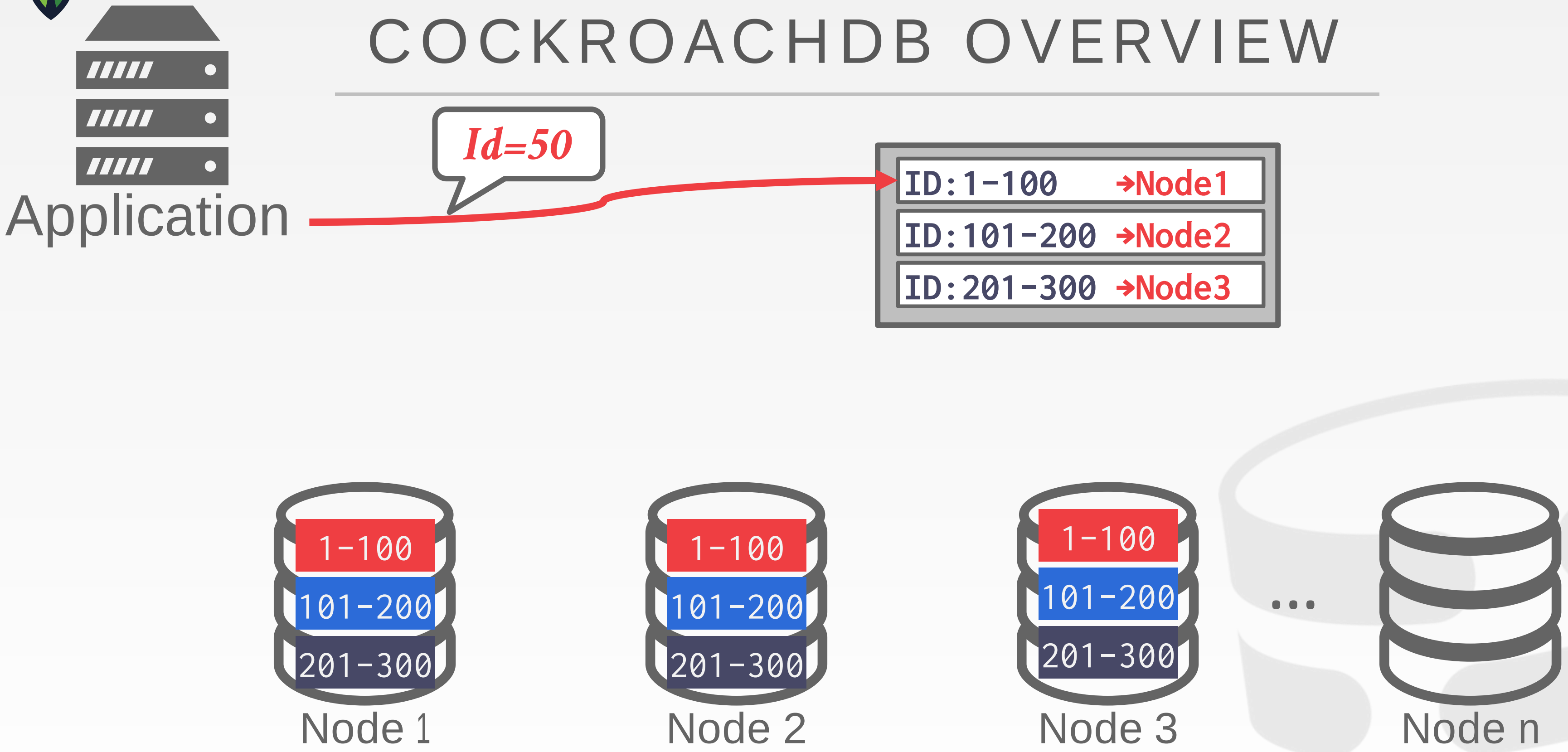
DBMS uses hybrid clocks (physical + logical) to order transactions globally.

→ Synchronized wall clock with local counter.

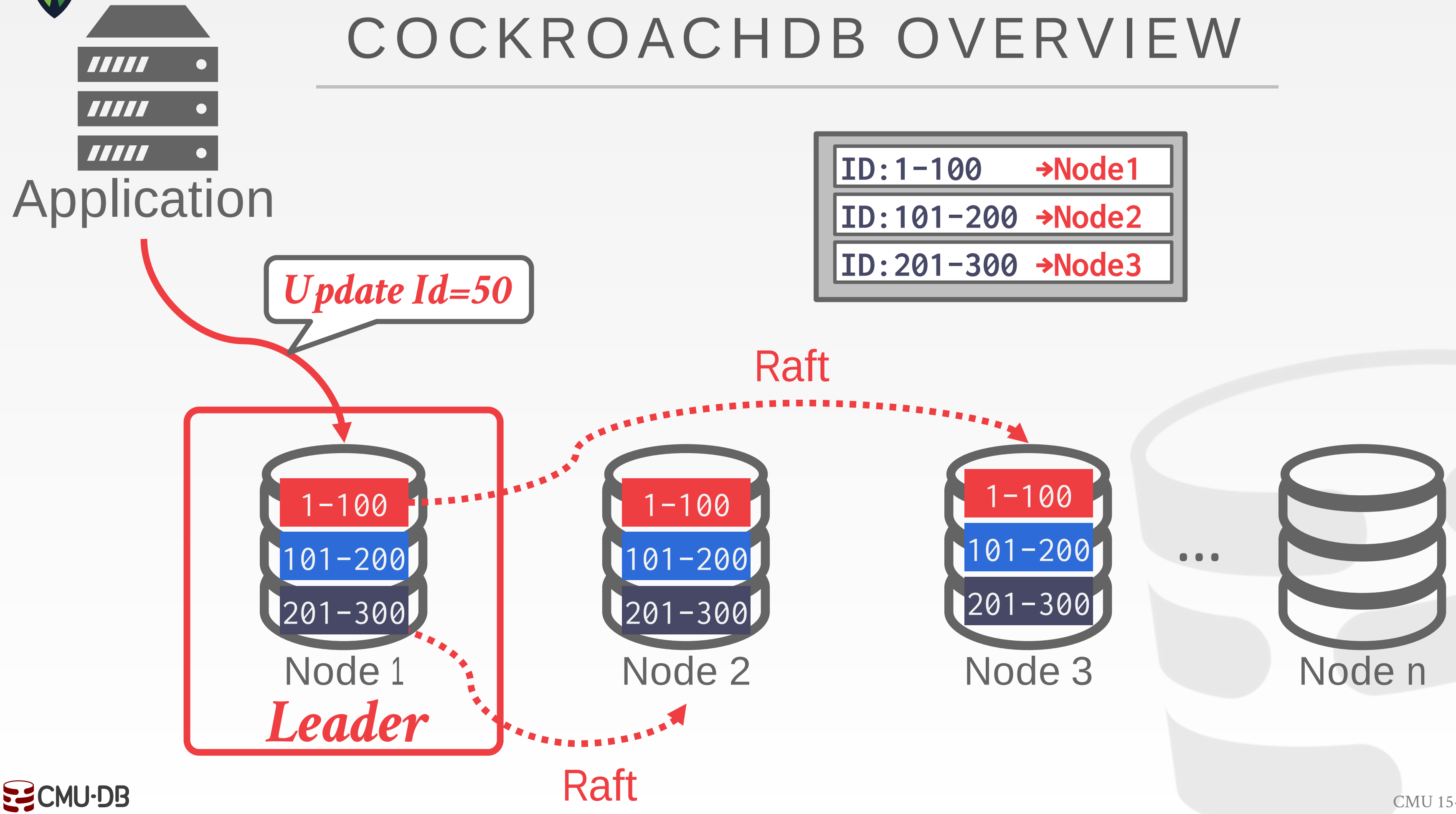
Txns stage writes as "intents" and then checks for conflicts on commit.

All meta-data about txns state resides in the key-value store.

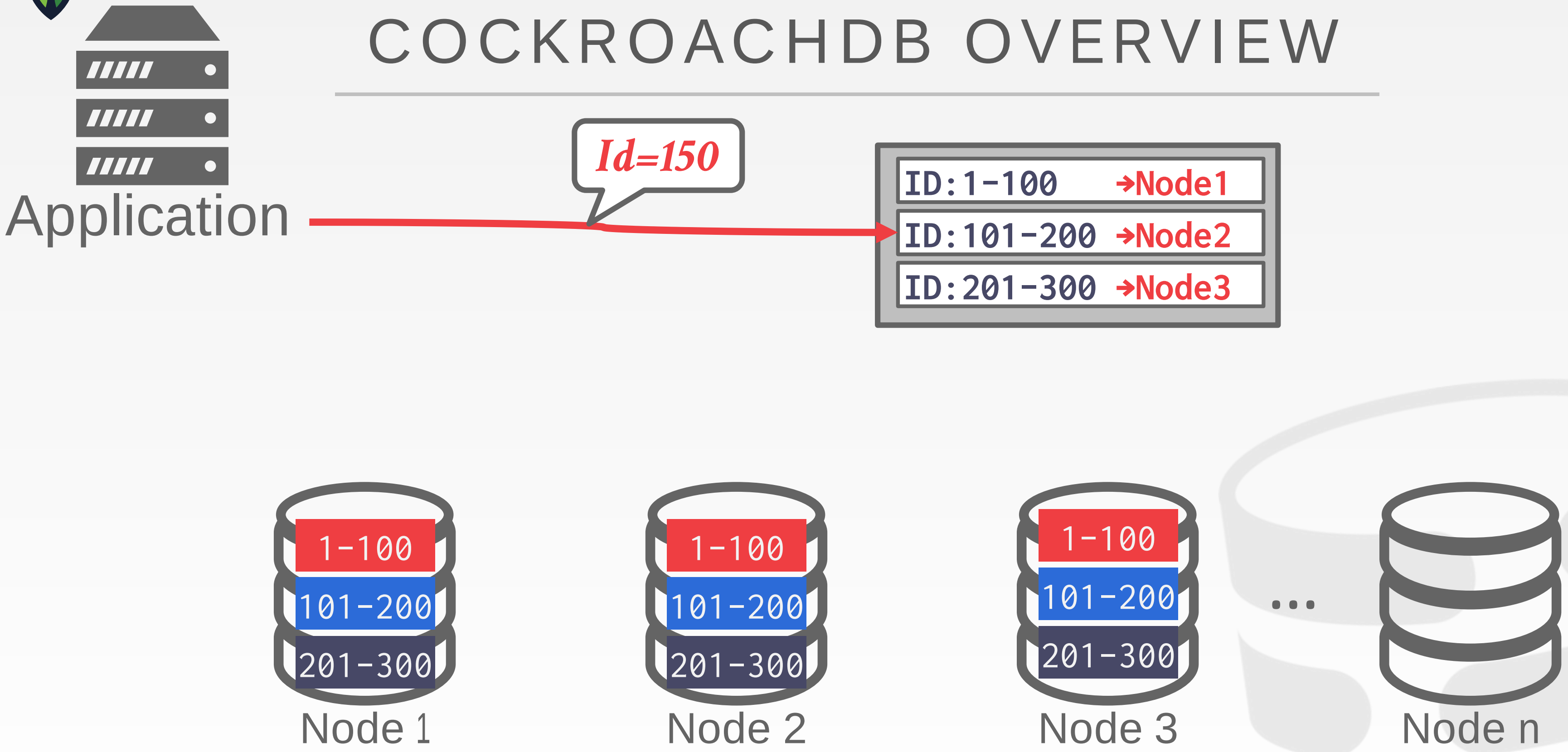
COCKROACHDB OVERVIEW



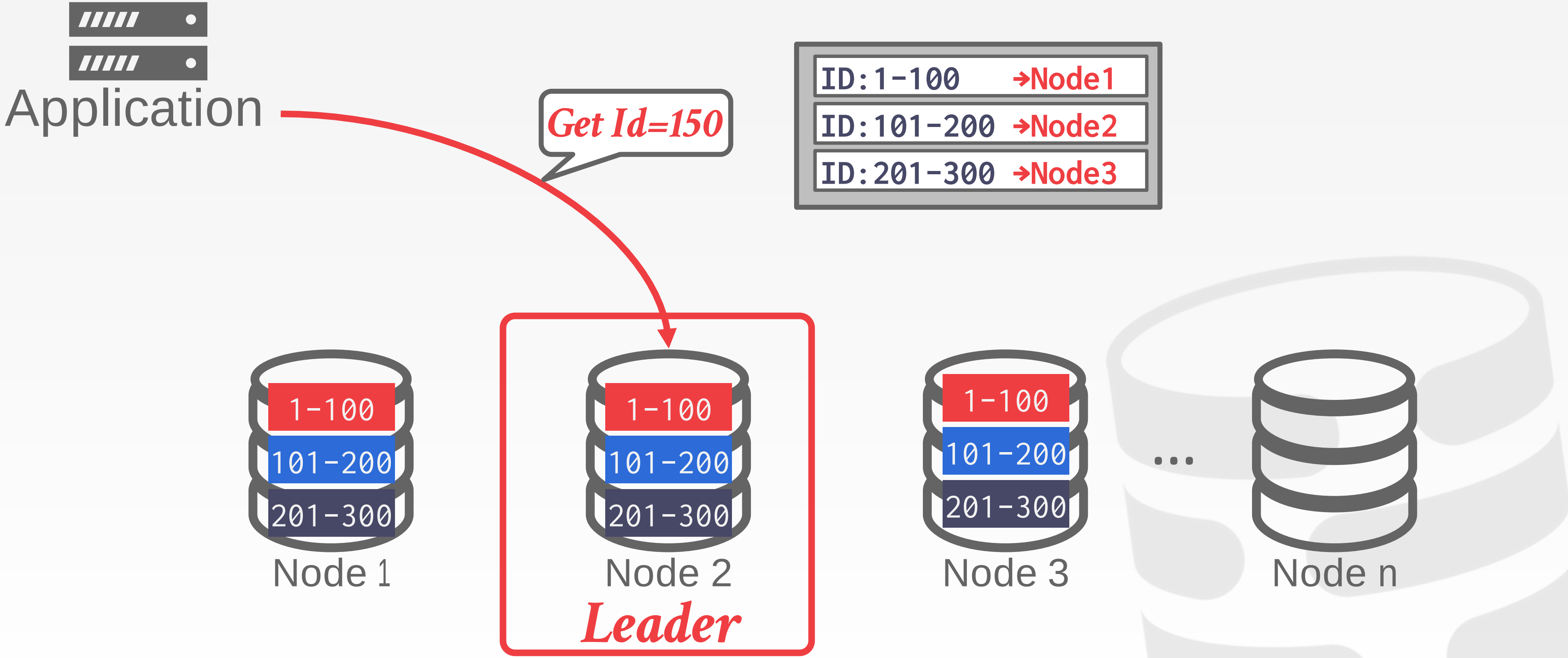
COCKROACHDB OVERVIEW



COCKROACHDB OVERVIEW



COCKROACHDB OVERVIEW



ANDY'S CONCLUDING REMARKS

Databases are awesome.
→ They cover all facets of computer science.
→ We have barely scratched the surface…

Going forth, you should now have a good understanding how these systems work.

This will allow you to make informed decisions throughout your entire career.
→ Avoid premature optimizations.