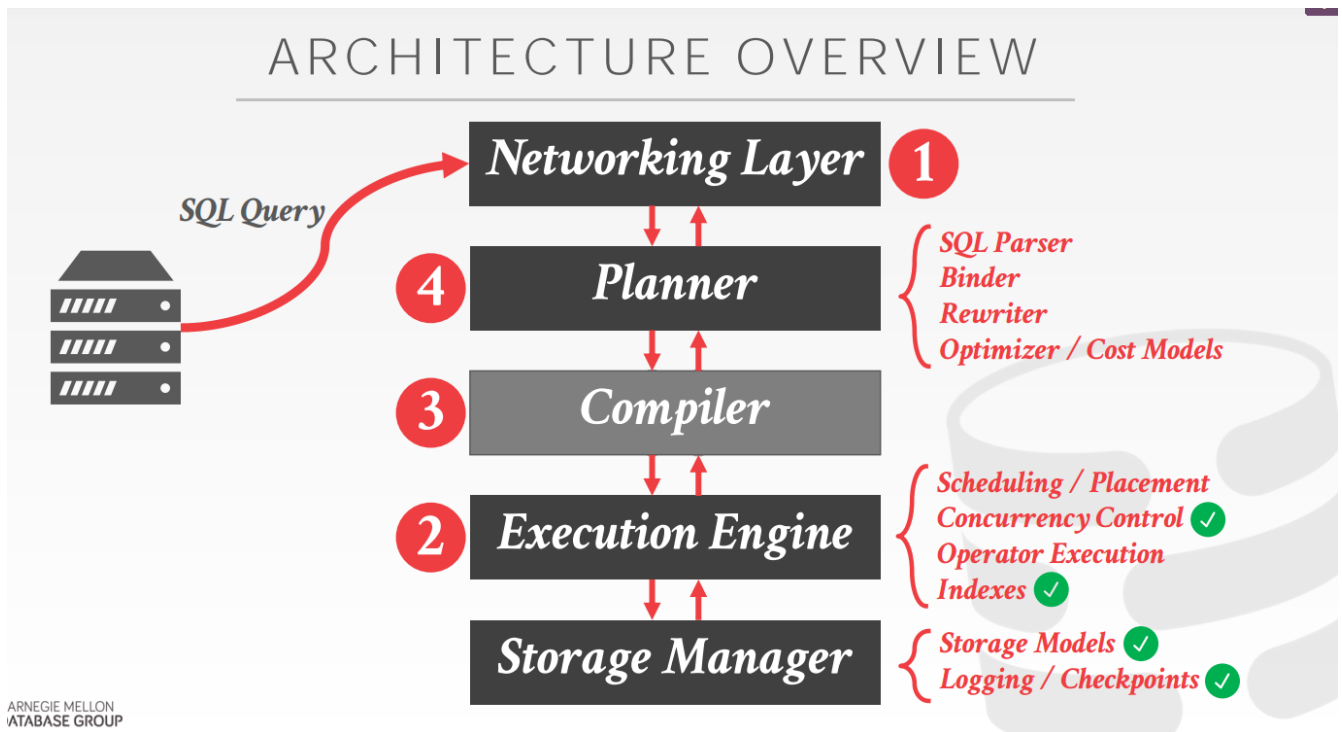


# Networking Protocol



上图给出了SQL执行过程中的经过的架构总结

- Networking Layer
- Planner

SQL Parser -> Binder -> Rewriter -> Optimizer / Cost Models

- Compiler
- Execution Engine

Scheduling/Placement ,Concurrency Control , Operator Execution ,Indexes

- Storage Manager

Storage Models/Logging/Checkpoint

## JDBC

# JAVA DATABASE CONNECTIVITY

---

## **Approach #1: JDBC-ODBC Bridge**

→ Convert JDBC method calls into ODBC function calls.

## **Approach #2: Native-API Driver**

→ Convert JDBC method calls into native calls of the target DBMS API.

## **Approach #3: Network-Protocol Driver**

→ Driver connects to a middleware that converts JDBC calls into a vendor-specific DBMS protocol.

## **Approach #4: Database-Protocol Driver**

→ Pure Java implementation that converts JDBC calls directly into a vendor-specific DBMS protocol.

第四种效率最高，用语言本身实现原生的Driver

## **Networking Protocols**

---

- All DBMS implement wire protocol based on TCP/IP

# DATABASE NETWORKING PROTOCOLS

---

All major DBMSs implement their own proprietary wire protocol over TCP/IP.

A typical client/server interaction:

- Client connects to DBMS and begins authentication process. There may be an SSL handshake.
- Client then sends a query.
- DBMS executes the query, then serializes the results and sends it back to the client.

我们的关注点在于序列化反序列化，这也是性能优化的着重点。

- 设计要点

## Row/Column Layout

### ROW VS. COLUMN LAYOUT

---

ODBC/JDBC are inherently row-oriented APIs.

→ Server packages tuples into messages one tuple at a time.

→ Client must deserialize data one tuple at a time.

But modern data analysis software operates on matrices and columns.

One potential solution is to send data in vectors.

→ Batch of rows organized in a column-oriented layout.

## Compression

### COMPRESSION

---

**Approach #1: Naïve Compression**

**Approach #2: Columnar-Specific Encoding**

More heavyweight compression is better when the network is slow.

Better compression ratios for larger message chunk sizes.

## Data Serialization

- Approach1: Binary Encoding

eg: ProtoBuf

Client handles endian conversion.

You also need to store the extra metadata about the show the serialized messages.

可能会导致一条很长的数据，分多次发送，但是元数据是相同的，你要复制传输相同的元数据多次。

- Approach2: Text Encoding

Convert all binary values into strings.

## STRING HANDLING

### Approach #1: Null Termination

- Store a null byte ( `'\0'` ) to denote the end of a string.
- Client scans the entire string to find end.

### Approach #2: Length-Prefixes

- Add the length of the string at the beginning of the bytes.

### Approach #3: Fixed Width

- Pad every string to be the max size of that attribute.

不同节点之间的传输：

# REPLICATION PROTOCOLS

DBMSs will propagate changes over the network to other nodes to increase availability.

- Send either physical or logical log records.
- Granularity of log record can differ from WAL.

Design Decisions:

- Replica Configuration
- Propagation Scheme

可以分为两种策略：主从型和多主型

## Approach #1: Master-Replica

- All updates go to a designated master for each object.
- The master propagates updates to its replicas without an atomic commit protocol.
- Read-only txns may be allowed to access replicas.
- If the master goes down, then hold an election to select a new master.

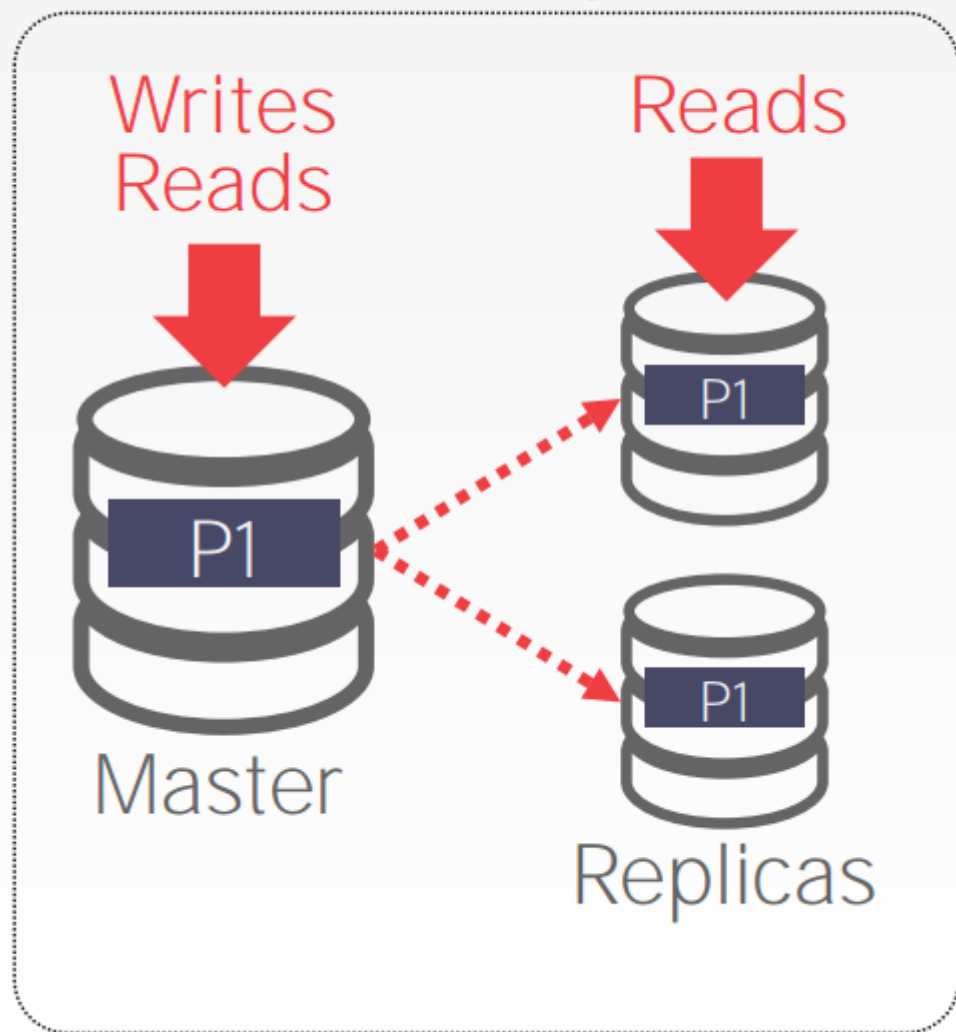
## Approach #2: Multi-Master

- Txns can update data objects at any replica.
- Replicas must synchronize with each other using an atomic commit protocol.

主从型：

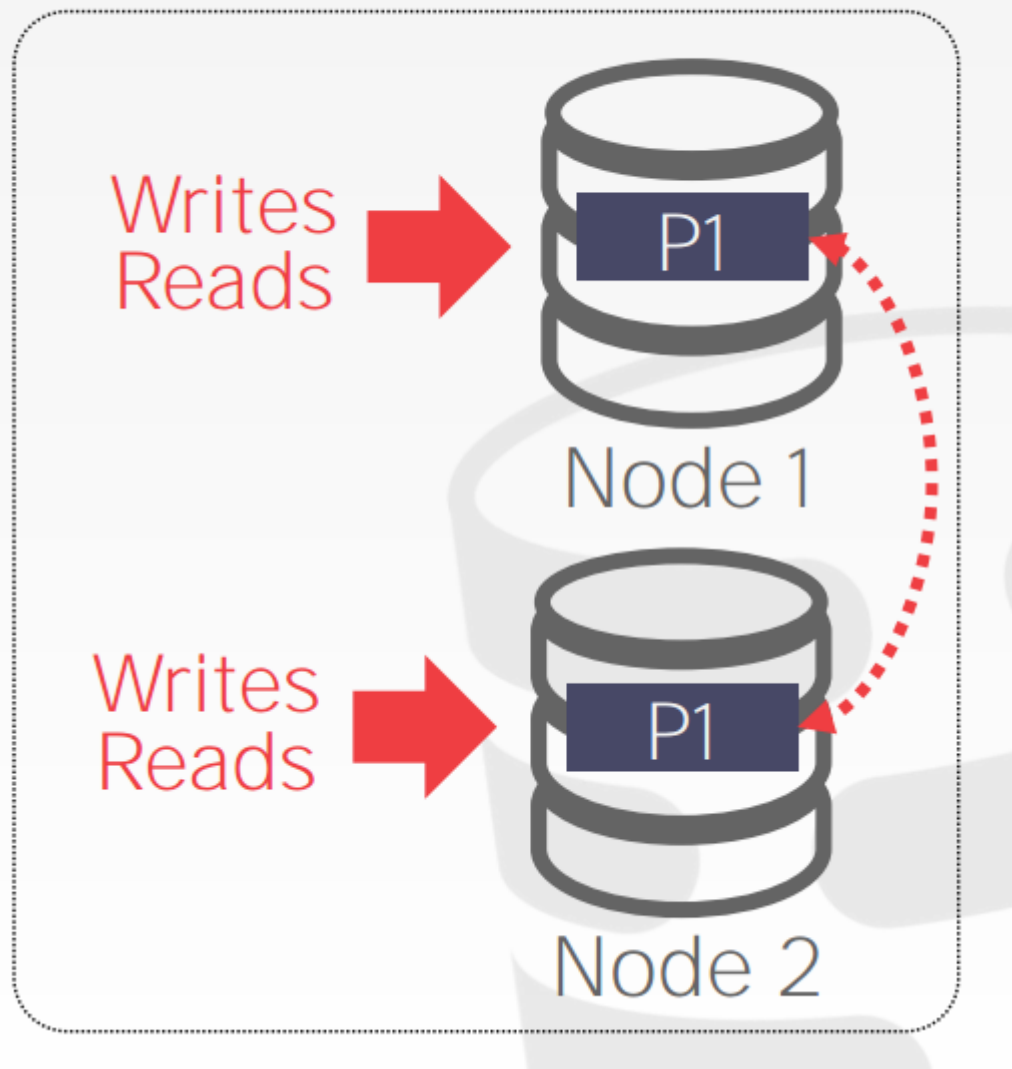
主写读从，主节点不需要通过原子提交协议与从节点做同步

## *Master-Replica*



多主型:

# Multi-Master



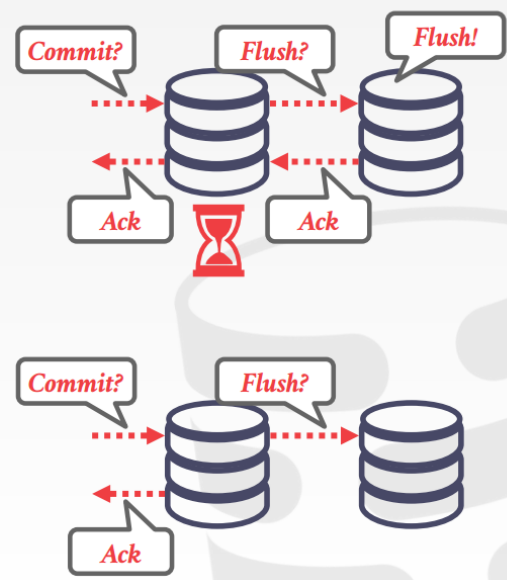
数据同步策略也可以分为同步与异步两种:

## Approach #1: Synchronous

→ The master sends updates to replicas and then waits for them to acknowledge that they fully applied (i.e., logged) the changes.

## Approach #2: Asynchronous

→ The master immediately returns the acknowledgement to the client without waiting for replicas to apply the changes.



- 同步：主节点同步到副本的时候，必须保证副本数据落盘并且返回ack给主节点，当主节点收到所有的ack之后就可以向客户端返回事务提交成功的通知。
- 异步：类比于Kafka的In-sync Replica机制，不需要等待副本落盘之后并返回ack

但其实DBMS的network protocol并不是制约其性能的原因之一，TCP/IP协议栈其实也很慢：

- 上下文切换/中断代价高
- 数据复制开销大
- 内核代码中latch很多

解决方法：

## Kernel Bypass Methods

直接从网卡拿数据，避免数据拷贝和OS TCP/IP 栈

- Approach1:

DPDK(Data Plane Development Kit)

1.消除Data复制开销/系统调用开销

2.直接从网卡拿数据

- Approach2:

RDMA(Remote Direct Memory Access)

直接从远程host根据地址获取memory的内容

## REMOTE DIRECT MEMORY ACCESS

Read and write memory directly on a remote host without going through OS.

→ The client needs to know the correct address of the data that it wants to access.

→ The server is unaware that memory is being accessed remotely (i.e., no callbacks).