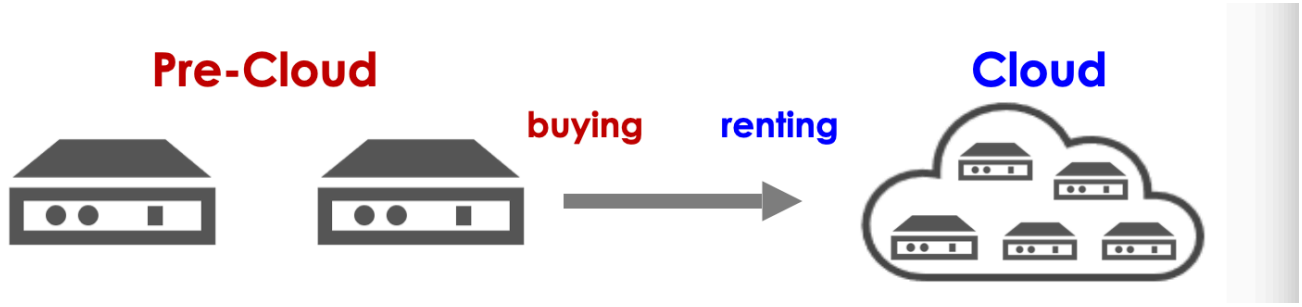
How to build your IT infrastructure?

- Buy servers
  - how many? how much memory, disk, CPU, GPU?
  - how many users (particularly for special occasions)
  - Simple solution: provision for peak load, but underutilize in most times
- Build a data center
  - where to physically put the servers?
  - what if the machines are crashed?
  - what if we need a software upgrade
  - cooling techniques
  - skilled engineers
    Soln: All is possible without clod but at what cost?

Cloud Computing



- cloud computing is about buying vs renting
- instead of buying hardware, just renting an instance from a cloud provider (Amazon AWS, Google Cloud)

> *"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."*

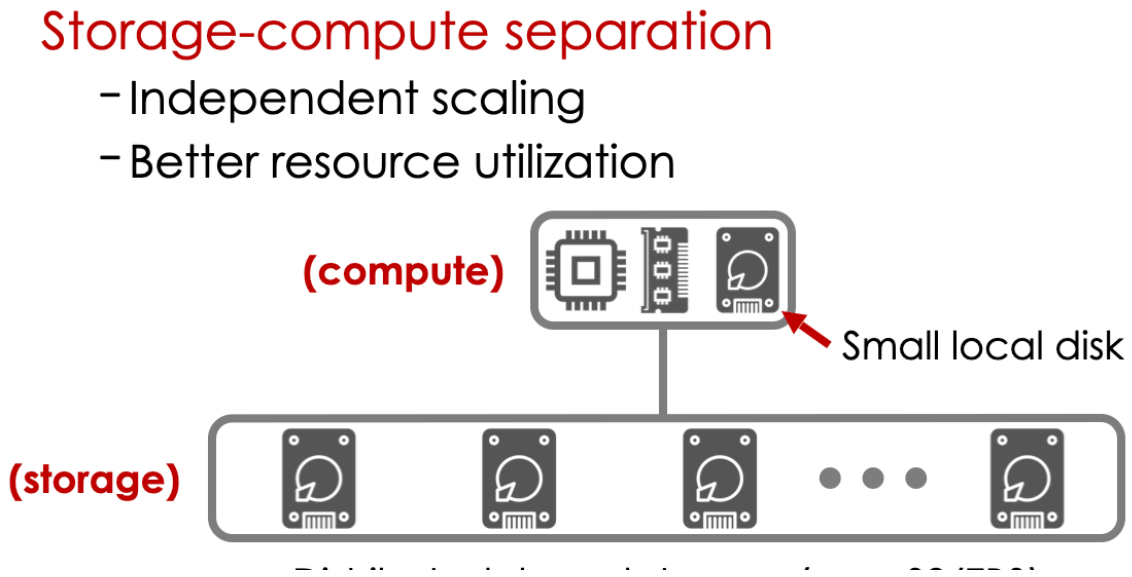
Key Characteristics

- Pay as you go
  - only pay for what you use with fine-grained metering, no up-front commitment (CPU per hour, memory per GB)
  - Elasticity

        - users can release resources if not needed or request additional resources if needed, ideally automatically (severless computing)
    - Virtualization
        - all the resources (cpu, memory, disk, network) are virtualized, no software is bounded to hardware
        - resources are pooled to serve multiple users (multi-tenancy)
    - Automation
        - No human interaction (start/stop a machine, crash, backup)
        - everything is managed by cloud providers

Cloud Native Databases

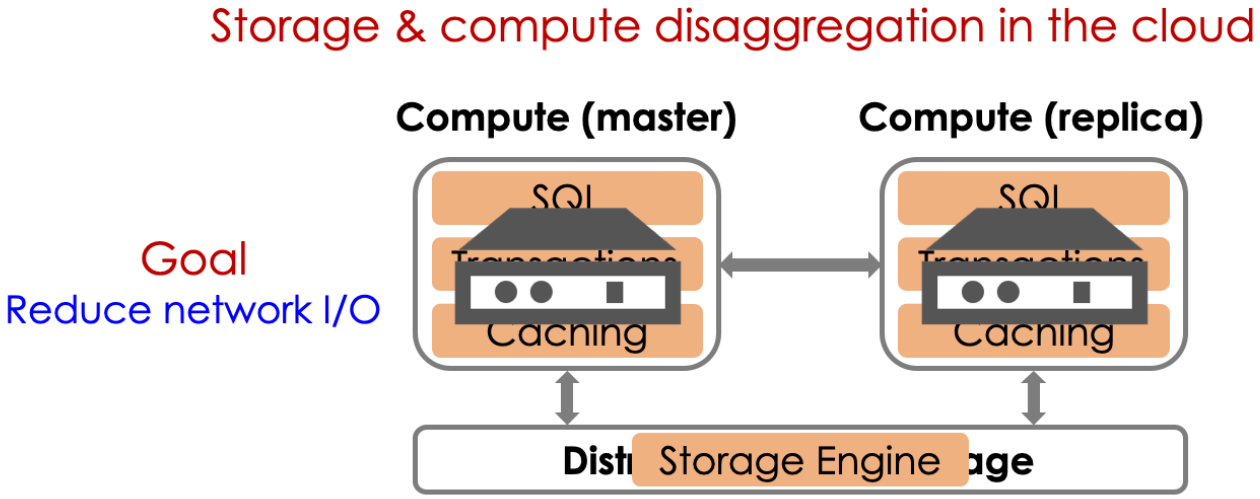
- Treat each cloud machine the same as in-house machine
- run existing database systems directly
- Cloud-native dbs are re-architectured to fully leverage the cloud infrastructure
    - resource/storage disaggregation
    - resource/storage pooling



Implications

- DB software level needs to be aware of the underlying hardware-level resource disaggregation
    - software level disaggregation
    - in order to enable more optimizations
- Distributed database architecture needs to be changed
    - from shared nothing to shared storage

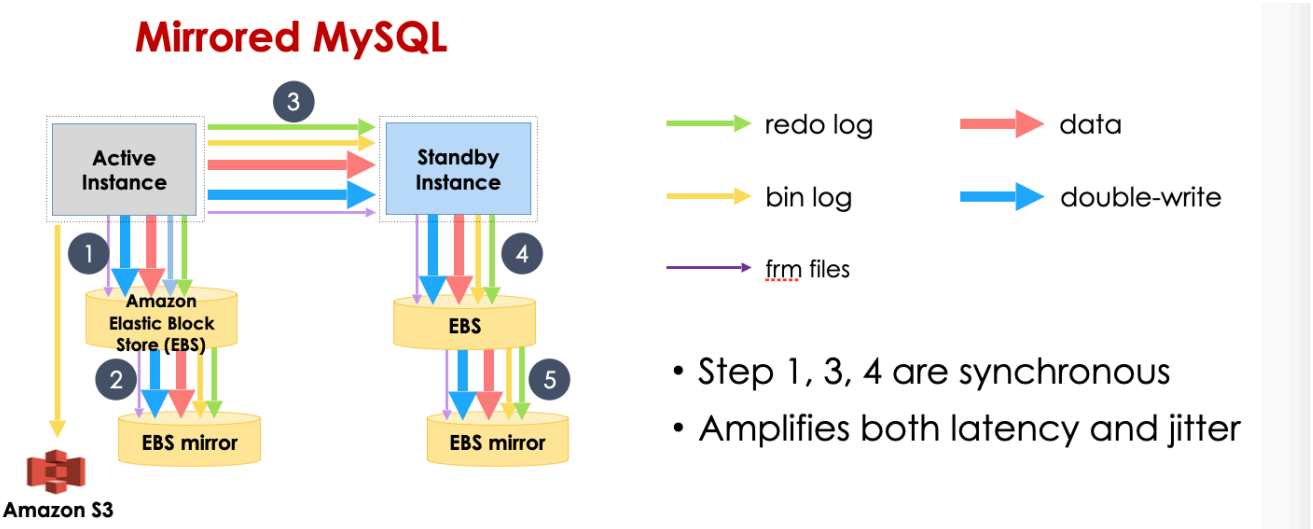
System Architecture



Main idea:

- Log is the database
  - only write redo logs on network
  - push log applicator to storage tier
- Asynchronous processing
  - materialize pages in background
- Buffer cache
  - to avoid network I/O
  - can rad pages upon cache miss

I/O traffic in traditional DB



I/O traffic in Aurora

- Only write redo log records

- all steps asynchronous
- 4/6 quorum storage
- 7.7X less data movement

Writes

- writes (trxns) send logs to storage asynchronously
- durability: each log is durable (ack) with 4/6 quorums
- Volume Durable LSN (VDL)
  - Log records can be lost, out of order
  - VDL: the largest one with all prior LSNs are durable

Transaction Commits

- transaction commits asynchronously
- when a transaction commits, mark its commit LSN
- commit only if VDL >= commit LSN

Replication: Scalability

- 1 writer and up to 15 reader instance
- To keep data consistent between writer and readers
  - writer sends logs to readers at the same time
  - once the reader receives logs, it will check if the page is in the cache
  - replication lag: 20ms

Reads (Caching)

- Each reader instance has a buffer (cache)
- upon read, check cache first
  - the cache is supposed to contain the latest data pages
  - except replication lags
    What if the cache is full?
- always evict a clean page: a page that's durable (pageLSN <= VDL)
- Why? O.w need to write dirty pages to storage, which increases network overhead

Crash Recovery

- if writer (master) is crashed, detected by HM (health monitor), promote a reader to writer first, and perform recovery
  - what if the failed writer comes back? -> contact HM
  - sometimes two master -> many unexpected issues

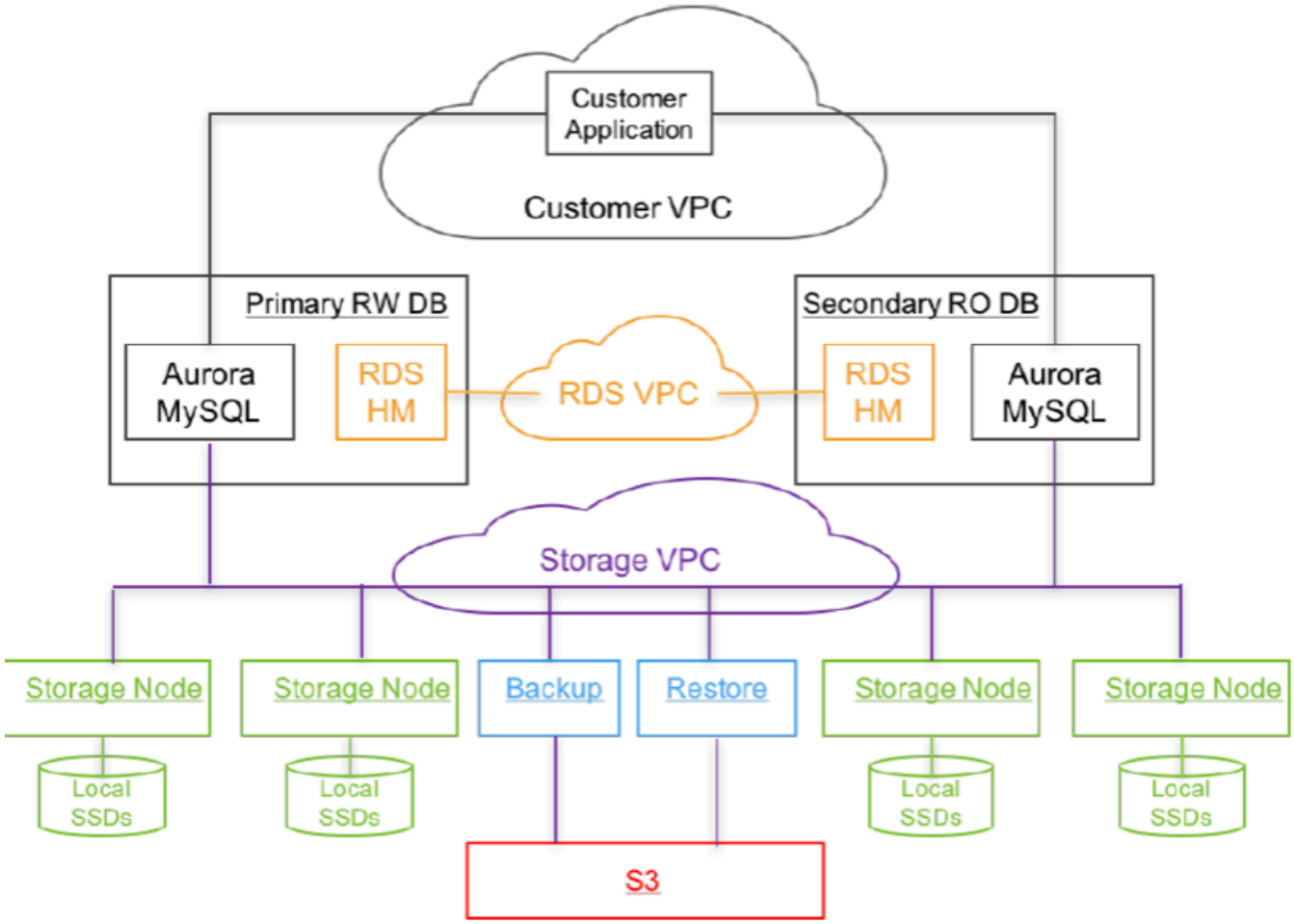
Crash Recovery

Traditional Databases

- Have to replay logs since the last checkpoint
- Typically 5 minutes between checkpoints
- Single-threaded in MySQL; requires a large number of disk accesses

Aurora

- No need to generate pages during recovery (very fast)
- Just need to re-establish VDL ➔ make sure storage is consistent
- Generate pages asynchronously, in parallel
- DB engine undo partial transactions
- Typically a few seconds



- Up to 5x faster than Cloud MySQL, but how about MySQL with local SSDs?

Comments on Aurora

- **New way of building cloud DB systems**
  - Monolithic (since 1970s) ➔ disaggregation
  - Hardware & software
- **Widely adopted in industry**
  - Microsoft Socrates DB
  - Alibaba PolarDB
  - Tencent CynosDB
  - Huawei TaurusDB
  - …