KubeCon | CloudNativeCon

Europe 2022

WELCOME TO VALENCIA

KubeCon | CloudNativeCon Europe 2022

Adapting TiKV for Cloud Storage

Xinye Tao (@tabokie), PingCAP

TiKV

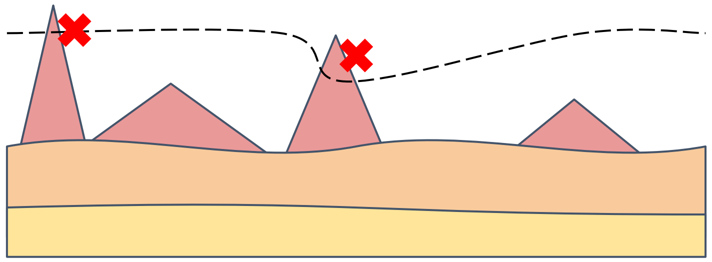
- A **distributed** transactional key-value storage engine
  - Scale out to hundreds of nodes
  - Replication of both WAL and data files

TiKV + Cloud Storage

- A **distributed** storage engine for TiDB
  - Scale out to hundreds of nodes
  - Replication of both WAL and data files
- **Cloud-based** disk: Elastic Block Storage (AWS), Persistent Disk (GCP), Managed Disk (Azure)
  - Under the hood, they are connected to remote shared hardware and replicated across different zones
    - High latency
    - Provisioned performance (bandwidth, IOPS)
    - Service degradation and outage

Challenges

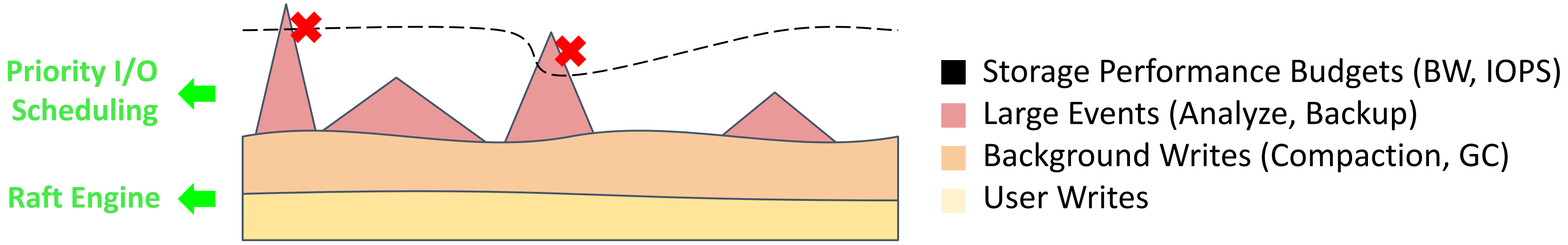
- Scalability
    - Likelihood of rare events increases on a larger scale
    - Storage performance is more likely to degrade on a larger scale
- Cost
    - User are more sensitive to read/write amplification
    - There is an additional cost to replicating data flow



- ■ Storage Performance Budgets (BW, IOPS)
- 🟥 Large Events (Analyze, Backup)
- 🟧 Background Writes (Compaction, GC)
- 🟨 User Writes

Challenges

- Scalability
  - Likelihood of rare events increases on a larger scale
  - Storage performance is more likely to degrade on a larger scale
- Cost
  - User are more sensitive to space amplification and write amplification
  - There is an additional cost to replicating data flow


Priority I/O Scheduling
Raft Engine
■ Storage Performance Budgets (BW, IOPS)
🟥 Large Events (Analyze, Backup)
🟧 Background Writes (Compaction, GC)
🟨 User Writes

Raft Engine

- A lightweight log store in Rust (https://github.com/tikv/raft-engine)
- [x] <Primary Goal> write less than RocksDB, both foreground and background
- [ ] <Secondary Goal> a more performant engine than RocksDB

Raft Engine ≈ Index + Logs

- A lightweight log store in Rust (https://github.com/tikv/raft-engine)
- [x] <Primary Goal> write less, both foreground and background
    - An **in-memory index** keeps track of all active log entries
        - No need to sort the log files
        - No need for GC to read deleted data
    - Log entries are **compressed** and appended to log files
    - Test results* showed **30%** reduction in server write I/Os

** Ran TPC-C 50 threads on a 3 node cluster, AWS r5.2xlarge with gp3 disk*

Priority I/O Scheduling

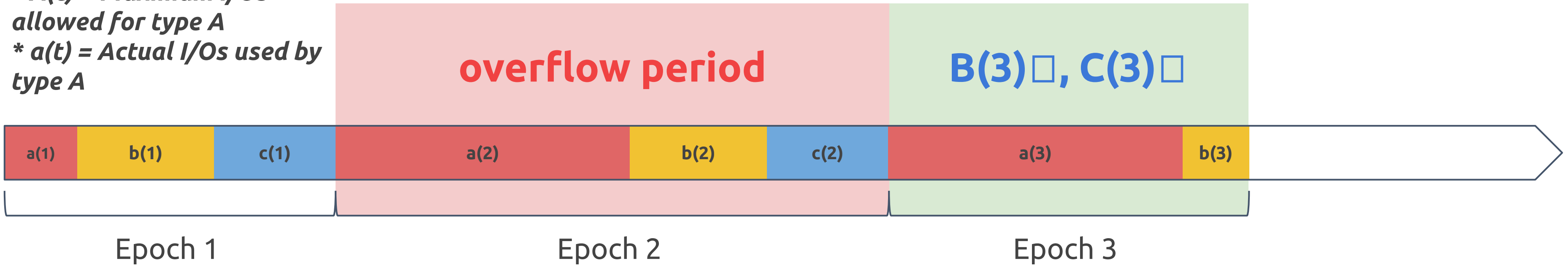
- A non-intrusive way to prioritized I/O requests (#9197)
  - No userland I/O queue, zero overhead
- Read I/O accounting and Rust async support (#11969)
  - Pulling stats from `*proc/task/io*` when needed

A { foreground, replication, load-balance }

B { flush, level-zero-compaction }

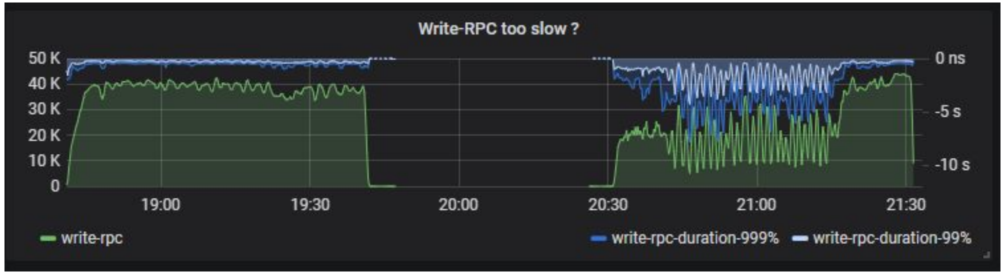
C { compaction, import, export }

Notes:
**** A(t) = Maximum I/Os allowed for type A***
**** a(t) = Actual I/Os used by type A***



Priority I/O Scheduling

- A non-intrusive way to prioritized I/O requests (#9197)
- I/O intensive tasks are well constrained



Importing data while running TPC-C workload w/o I/O scheduling

More on the way

- CPU Limiting: proactive back pressure for low-resource environments (#12151)
- Raft Witness: a write-only node that only replicates logs (raft-rs#145)
- …