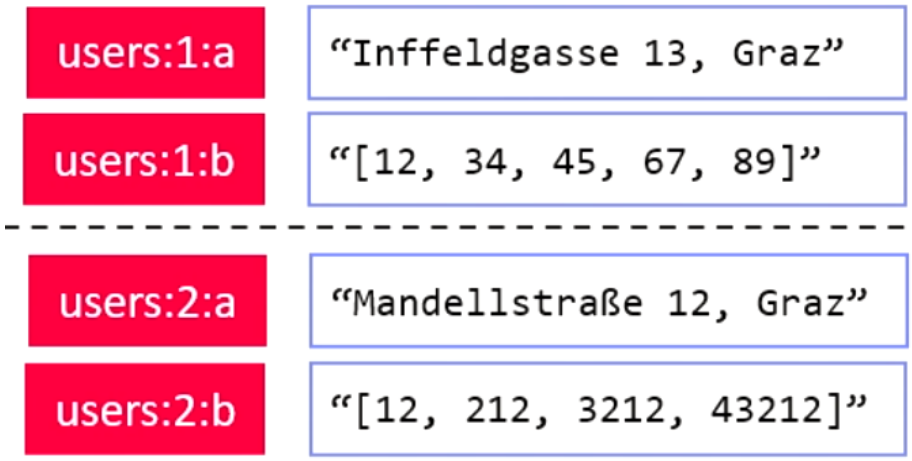
Motivation

- basic key-value mapping
  - simple API
  - complex data models
- reliability at massive scale
  - cloud computing

System Architecture

- key-value pairs map different/flexible datatypes
- API for CRUD Operations
- scalablity via sharding
  - horizontal partitioning



- example systems

Example Systems

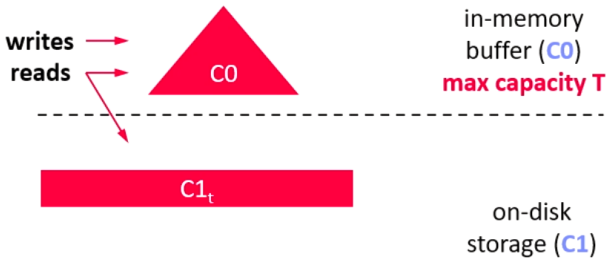
- **Redis Data Types**
  - Redis is not a plain KV-store, but "data structure server" with persistent log (**appendfsync no/everysec/always**)
  - Key: ASCII string (max 512MB, common key schemes: comment:1234:reply.to)
  - Values: strings, lists, sets, sorted sets, hashes (map of string-string), etc
- **Redis APIs**
  - **SET/GET/DEL:** insert a key-value pair, lookup value by key, or delete by key
  - **MSET/MGET:** insert or lookup multiple keys at once
  - **INCRBY/DECBY:** increment/decrement counters
  - Others: EXISTS, LPUSH, LPOP, LRANGE, LTRIM, LLEN, etc
- **Other systems**
  - Classic KV stores (AP): Riak, Aerospike, Voldemort, LevelDB, RocksDB, FoundationDB, Memcached
  - Wide-column stores: Google BigTable (CP), Apache HBase (CP), Apache Cassandra (AP)

Log-Structured Merge Tree

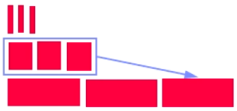
- data structure used in
  - (e.g., BigTable, DynamoDB, LevelDB, Riak, RocksDB, Cassandra, HBase)
- approach
  - buffer writes in memory
  - flushes data as sorted run
  - compaction - merges sorted runs into larger runs of next level

System Architecture

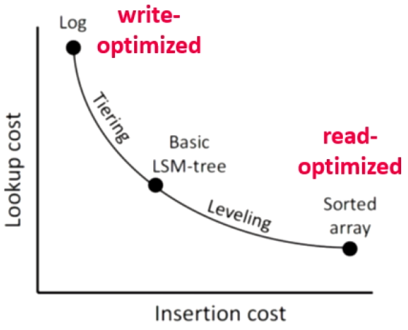
- Writes in C0
- Reads against C0 and C1 (w/ buffer for C1)
- Compaction (rolling merge): sort, merge, including deduplication



- **LSM Tiering**
  - Keep up to T-1 runs per level L
  - Merge all runs of L_i into 1 run of L_{i+1}
    - L1
    - L2
    - L3



- **LSM Leveling**
  - Keep 1 run per level L
  - Merge run of Li with Li+1
    - L1
    - L2
    - L3



[Niv Dayan: Log-Structured-Merge Trees, **Comp115 guest lecture, 2017**]

[Stratos Idreos, Mark Callaghan: Key-Value Storage Engines (Tutorial), **SIGMOD 2020**]

[[Data Models]]