lecture 4

Tuesday, October 15, 2024 11:47 AM

RDMA (Remote direct Memory access)

- >100 Gbps bandwidth
- Single digit microsecond latency
- Communications bypass Kernel
- One-sided Read / write verbs bypass remote machine's CPU.
- Servers use queue pairs to connect with each other.
- Two-sided send/recieve verbs.

key-value Interfaces.

- Get (Key) → Value

  the table should return you the latest value of the key.
- Put (Key, value) → void

  allows to overwrite the key with a value it provides.
- iterate (Start key) → iterator

  when you scan a data base, given this Starting from the Start key & the next key returns must be in sorted order.

LSM Tree & levelDB

level DB organize the data into 2 parts; one is in memory & one is on the disk; everything is like a table file & all the files are immutable. To search for a file run binary search on the files & locate the file that contains the key. When "put" request comes in, it simply appends it to the active memtable, & append a log record.

When "get" request comes in, it starts from the memtables, searching all the tables at level 0. If the memtable does not contain the key, it will go to the SSTables level 1. Return immediately once key is found.

Iterate/scan, is more expensive because the scan needs to first locate the star key, & the look for next key.

levelDB & RocksDB are monolithic software just like MySQL, you can download & run it on your laptop, assuming you have fixed amount of CPU, DRAM, etc. However in modern data centers, all the hardware resources are disaggregate; to give higher throughput.

LSM-tree components.

- LST-tree component, LTC: implements memtables & SSTables, identifying SSTables that should be compacted (like a garbage collector— to remove all the dublicates); performs compaction or off-loads it to the Stoc.
- logging Component, LogC: implements logging to survive crash.
- Storage Component, StoC: They essentially store, retrieve & manage variable size blocks; each storage component can be plugged in different storage hardwares like DRAM.

All these components are connected with the RDMA switch.

Challenges

- writes are stalleds because when tables are full you have no way to put the additional value in the memtable anymore; & have to wait for memtable to flush the disk before the "put" can be served.
- large memory slows down scan; construct dynamic range at runtime based on workload (↑ Solution)
- large memory slows down gets; as they'll have to search a lot of memtables & a lot of SSTables; use a lookup index. (↑ Solution)
- Random requests collide & reference the same STOC; Scatter blocks of a SSTable across multiple StoCs (↑ Solution)
- logging; replicating log Records across node. (↑ Solution)
- Skewed access pattern wastes disk bandwidth; DRange enables LTC to write 65% less data to StoCs with skewed data access.

Nova-LSM

Scale components independently for different workloads. using lookup index & range index to fasten reads. Scatter blocks of a SSTable across disks using power-of-d. Replicate log records via RDMA. Recovers GBs of memtables under a few seconds.

Today's Query Optimizer components.

- Cardinality estimation is very critical to estimate how many tuples are going to be produced after processing each of the other nodes in the plan.
- Analytical cost model try to mimic the real execution time.
- Plan enumeration will enumerate the possible alternatives to execute an operator.

Why do plan changes lead to drastic performance changes?

A slight change of bind values produces completely different query plans; The new plan may be very slow due to the inherent sub-optimal components

* The error from one component propagates to other components.

3 objectives of keples:

- Improve total query runtime.
- Be Robust; minimize regressions.
- Fast inference time.

Kepler has 2 main parts: Training & deployment.

In Training, the objective that kepler really want to evaluate all of these candidate plans & to figure out from them & use this data to fit into the model & check if a new query comes in among all the templates, which one to be picked?

Input to the model is a set of encoded values & the output is one query plan, it produces a score for every plan, & every candidate will pick the plan with the highest score.

key Insights from RCE.

It is a powerful & efficient way of interfacing with classical query optimizers; only needs to build a covering set of good plans, instead of identifying a single best plan; only pertubs what is relevant to the optimizer; Uniform random multiplicative pertubations are sufficient.

Training Query Execution

There are several ways to speed up training data collection.

- Use minimal set of plans so that every query instance has a near-optimal plan.
- Aggresively time out plans once we know they're suboptimal.
- Collecting training data is almost 100% embarrassingly parallel.

Transactions

Concurrency Control Protocols.