noDB

Daniel Walter

Outline

- What Storage options do we have
- How to scale storage
- How to choose the right™one
- Performance measurement
- Examples

Why does it matter?

- Performance
- Complexity
- Reliability
- Maintainability

Scenarios

- Configuration
- Media Server
- Monitoring data

Storage Options, an incomplete List

- Relational Databases
- NoSQL (K/V, Wide Column, Document Store)
- local, remote or distributed Files
- Data Lakes, Data Lakehouses, Data Warehouses
- specialized Storage options like Graph databases, etc.

What influences our choice

- Structured vs unstructured Data
- Data access patterns
- Scalability requirements
- Data freshness requirements and update frequency

Scenarios

- Configuration
  - Structured data, smaller than 10GB
  - Read-write, with mostly reads
- Media Server
  - Unstructured data, each entry can be multiple GBs
  - Mostly read, updates on Metadata part only
- Monitoring data
  - Structured data, small entries but lots of them
  - Read-write with constant writes, but writes are typically append-only

How to scale storage

- Vertical scaling of the storage system
- Horizontal scaling options
  - Sharding
  - Distribution
  - Multi Layer approach
- Move to hosted storage aka monetary scaling
- Reduce latency by adding a caching layer

Measuring Performance

Rob's Rules #1 and #2

Benchmarking

- Measure the performance of a single task / function
- Language specific tooling
  - Go has builtin benchmark support via the test framework
  - Python has `pyperf` and `timeit`
  - Rust has `test::Bencher` in Unstable or `criterion.rs`
  - For C++ you can use `google/benchmark`
  - Javascript has `deno bench`
- Ideal while developing to decide which algorithms to use
- E.g. compare parser implementations, serialization speed, etc

Benchmarking — Scenarios

- Configuration
  - Benchmarks can help to identify the best config file format.
  - E.g. load and parse JSON vs Protobuf vs reading data from SQLite
- Media Server
  - For Metadata same as above
  - Benchmark various Filesystems to identify which one solves the content delivery best
- Monitoring data
  - In this particular case we should benchmark the write path, but results might not be reliable

Profiling

- Capture performance data while the application is running
- Complete view of the application allows for identifing bottlenecks
- Very useful to find what needs to be optimized
- Well known tools are `gperftools`, `perf`, and `dtrace`

Loadtesting

- Send well defined mix of requests to the application and observe behaviour
- Needs additional ways to measure and record observations, typically done using monitoring or even profiling.
- Should be done on setups similar to Prod

Conclusion

Configuration

- Small amount of mostly read data
- Data is structured and the structure is not expected to change outside a release cycle
- Use either JSON or Protobuf to store the data to the file. Both normally outperform SQLite for this task.

Media Server

- Multi Layer Approach for large content files
  - 2 different storage solutions for Metadata and Content
  - Content should be stored as Files either local on the server or on a storage solution
  - Metadata will be stored in either a Database or in a File
- For smaller content
  - Use a single Document Store

Monitoring Data

- Large amount of data, which is mostly append only.
- Low latency for recent metrics, can tolerate higher latency for older data
- Multi-level approach for timeseries data seems to be the best choice
- Recent data in memory, shard if needed
- Older data can be stored on disk and old data can be downsampled to save storage

Questions?