What is BonsaiDb?

Presented by Jonathan Johnson aka @Ecton (He/Him)

What is BonsaiDb?

- ACID-compliant, transactional document database
- Grows with you:
  - Local-only (like SQLite)
  - Client/Server (+ replicas -- coming later)
  - Client/Cluster (coming later)
- Feature-rich:
  - Map/Reduce powered views
  - PubSub
  - Atomic Key/Value Storage
  - Extensible Role-Based Access Control
- Built with Rust, for Rust
  - ... but could be used with other languages
- Non-commercial and open source (MIT + Apache 2.0)

The Goals of BonsaiDb

- Simplify Development
- Simplify Deployment
- Simplify Maintenance
- Be @Ecton's ideal way to build "backend" services

Simplifying Development

- Solve common app development problems:
  - Reliable storage
  - Reasonable performance
  - Job queuing, scheduling, and monitoring*
  - Metrics/Time Series support*
  - Log Aggregation*
  - Monitoring and Alerting*
- Easier testing: Most features work in local-only mode

Simplifying Deployment

- Embedded within your application
- As simple as deploying your own app
- May eventualy offer a platform similar to Cloudflare Workers/AWS Lambda

Simplifying Maintenance

- Included administration and monitoring tools*
- Ability to report metrics and alerts to third party tools*
- Incremental* backup/restore
- "Easy" clustering*

Ecton's App Development vision

- I want to build apps and games
- After adopting Rust, I don't want to use any other language
- BonsaiDb aims to make backend development in Rust easier
- Still evaluating the best GUI approach for tooling

Minority Game: An BonsaiDb + Gooey example

- Simple implementation of a multiplayer game theory game
- Each player chooses to either stay in or go out
- Players that choose the less popular option are rewarded
- A second set of buttons allows broadcasting your intention
  - But you can bluff!

Building a High Scores Service

```rust
#[derive(Collection, Default, Debug, Serialize, Deserialize, Clone)]
#[collection(authority = "minority-game", name = "player", views = [PlayerByScore])]
pub struct Player {
    pub choice: Option<Choice>,
    #[serde(default)]
    pub tell: Option<Choice>,
    pub stats: PlayerStats,
}

#[derive(Debug, Serialize, Deserialize, Clone)]
pub struct PlayerStats {
    pub happiness: f32,
    pub times_went_out: u32,
    pub times_stayed_in: u32,
    #[serde(default)]
    pub times_lied: u32,
    #[serde(default)]
    pub times_told_truth: u32,
}
```

Building a High Scores Service

```rust
#[derive(View, Debug, Clone)]
#[view(collection = Player, name = "by-score", key = u32, value = PlayerStats)]
pub struct PlayerByScore;

impl CollectionViewSchema for PlayerByScore {
  type View = Self;

  fn map(
      &self,
      player: CollectionDocument<<Self::View as View>::Collection>,
  ) -> ViewMapResult<Self::View> {
      player
          .header
          .emit_key_and_value(player.contents.stats.score(), player.contents.stats)
  }
}
```

Building a High Scores Service

```
let top_players = db
        .view::<PlayerByScore>() // Begin the PlayerByScore query
        .descending()            // Reverse the order of the results
        .limit(10)               // Only return the first 10 results
        .query()                 // Retrieve the results
        .await                   // Non-async interface available, also
        .unwrap();
```

- Views are *indexed* as needed.
- *Indexing* calls the View's `map()` function for each changed document
- Our map function *emits* a single key-value pair:
  - `PlayerStats::score()` is the *Key*.
  - `PlayerStats` is the *Value*.
- A View is always sorted by its Key.
- `top_players` is a `Vec<Map<u32, PlayerStats>>`

What's next for BonsaiDb?

- Current Project: Optimizing multithreaded transaction performance
- Next Major Project: Replication
- Always looking for new contributors

Questions?

Here are some useful links if you want to learn more or connect with us:

Websites

BonsaiDb: bonsaidb.io

Gooey: gooey.rs

Minority Game Demo: minority-game.gooey.rs

Talk with us

Discord: discord.khonsulabs.com

Forums: community.khonsulabs.com