DEVOXX™ UNITED STATES

Baratine

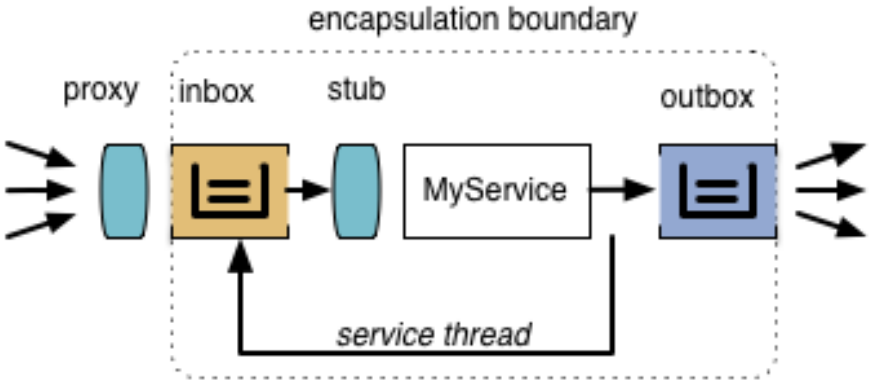
Operational Service Views

DEVOXX™ UNITED STATES

Architecture Overview

DEVOXX™ UNITED STATES

Baratine Service



Key Concepts:

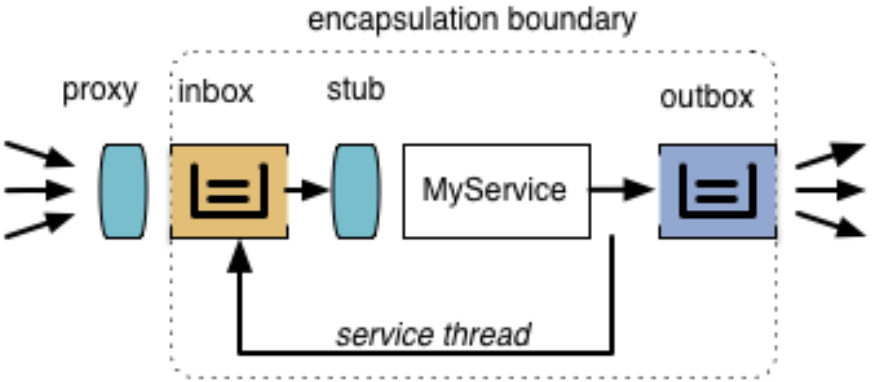
1. Strong encapsulation boundaries.

2. Each service has its own thread and efficient lock-free inbox.

3. Inside a service, application code can be single-threaded.

DEVOXX™ UNITED STATES

Baratine Service



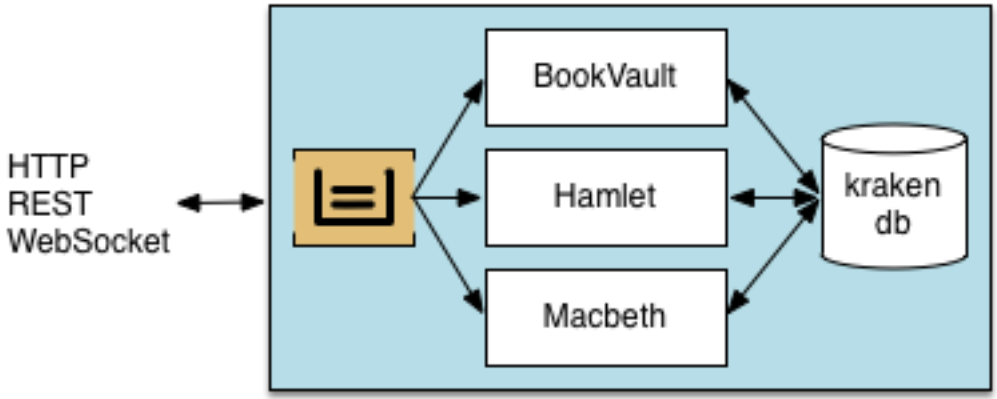
4. Persistent services load and store to a document-style key/value database.

5. Services use their data in-memory; loading and storing is transparent.

6. Services own their data in an object-oriented model. No additional locks are needed to protect against other services modifying their data

DEVOXX™ UNITED STATES

Database Persistence



1. In-memory persistent services are built-in to Baratine, using its asynchronous high-performance internal database Kraken.

2. An Application service operates on its data in-memory, improving performance and simplifying code.

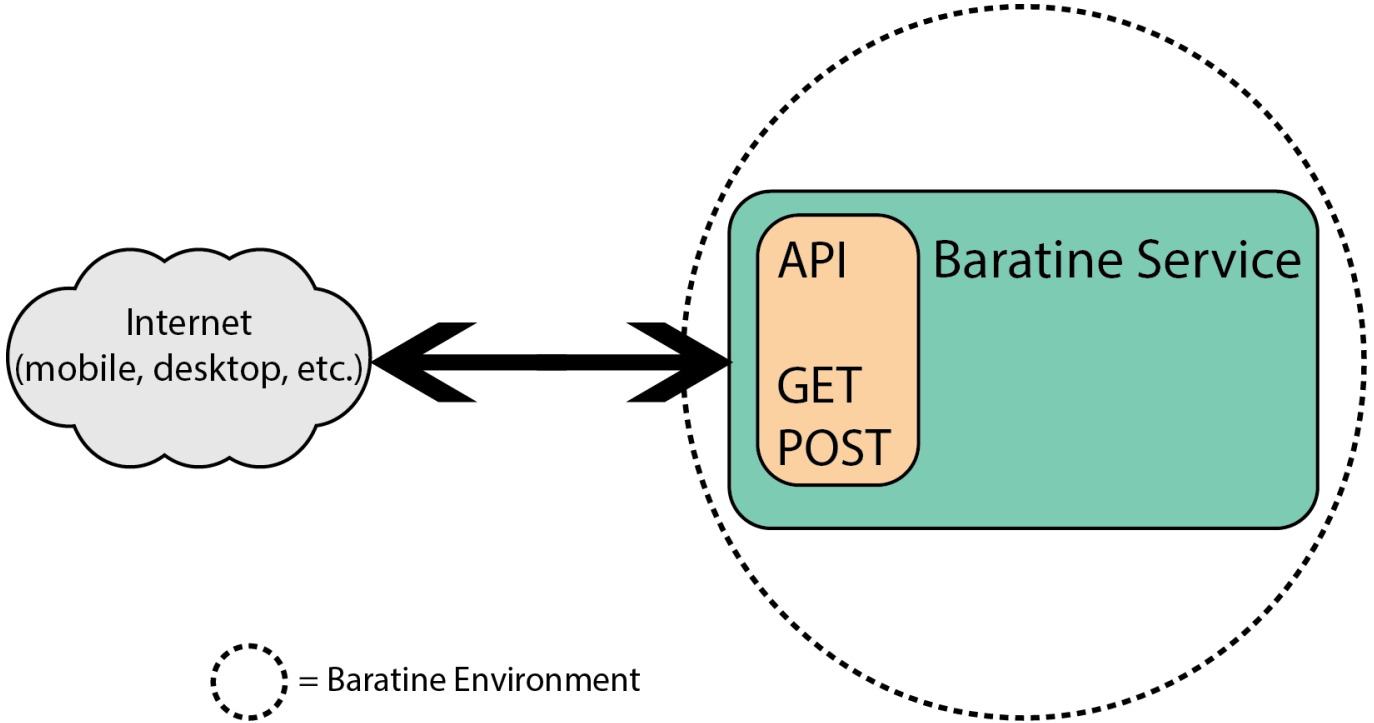
3. The data is automatically loaded and saved to the internal document-oriented database as part of the asset lifecycle.

DEVOXX™ UNITED STATES

Use Case #1: API

DEVOXX UNITED STATES

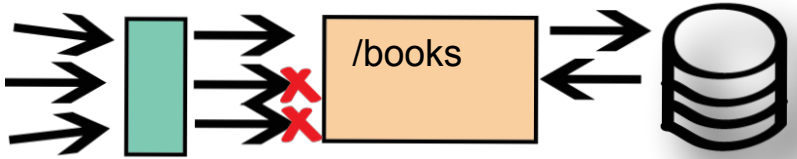
API Engine



DEVOXX UNITED STATES

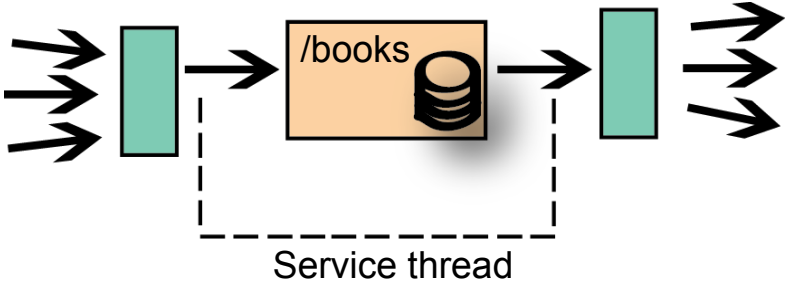
Bookstore REST service

Traditional



1. Operations must be persisted to db
2. Objects must be **synchronized** for multiple requests
3. No thread ownership of data

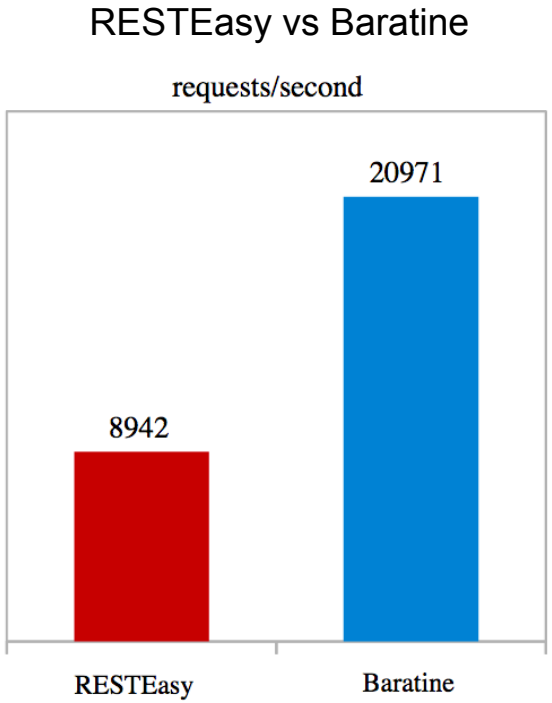
Baratine



1. Operations done in-memory
2. No locking/blocking threads
3. Single thread ownership of data

DEVOXX UNITED STATES

Bookstore REST service



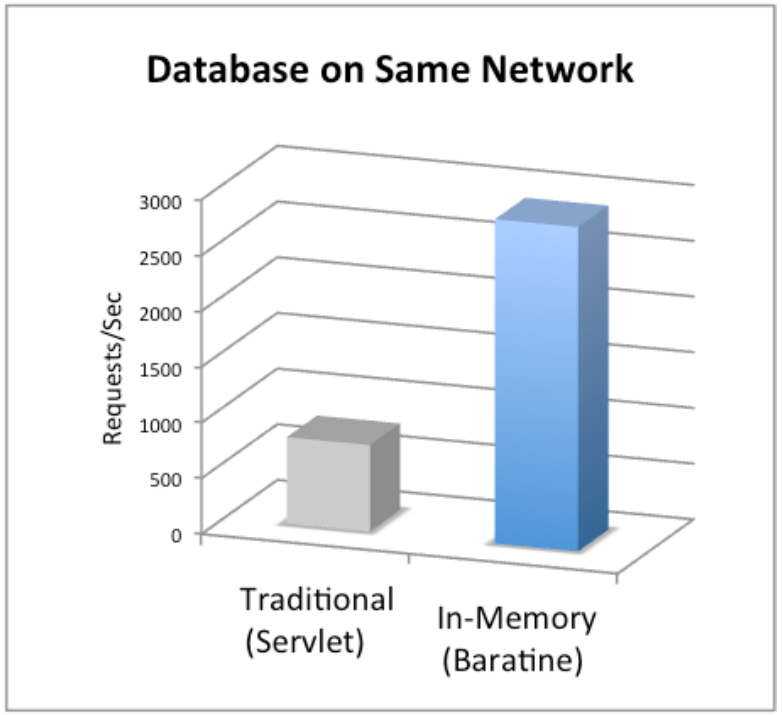
Baratine:

- Over **2x** the performance
- Can be scaled to match cores on CPU
- Persistence added with single annotation

```
// best out of 10 runs
$ wrk "http://127.0.0.1:8080/books"
```

DEVOXX UNITED STATES

In-Memory Operations



Benchmark requirements:

- One update per request
- Data persisted (MySQL)

The numbers represented are for a database that is on the same network.

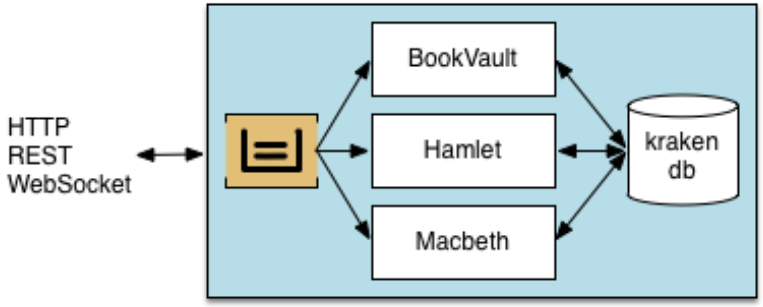
Baratine is much faster because it doesn't need to touch the database on each and every request - it just uses its data that is residing in-memory.

DEVOXX™ UNITED STATES

In-Memory Operations

1. Data loaded once
    1.1. Only removed on an LRU

2. Large data sets shard
    2.1. Data owned per service

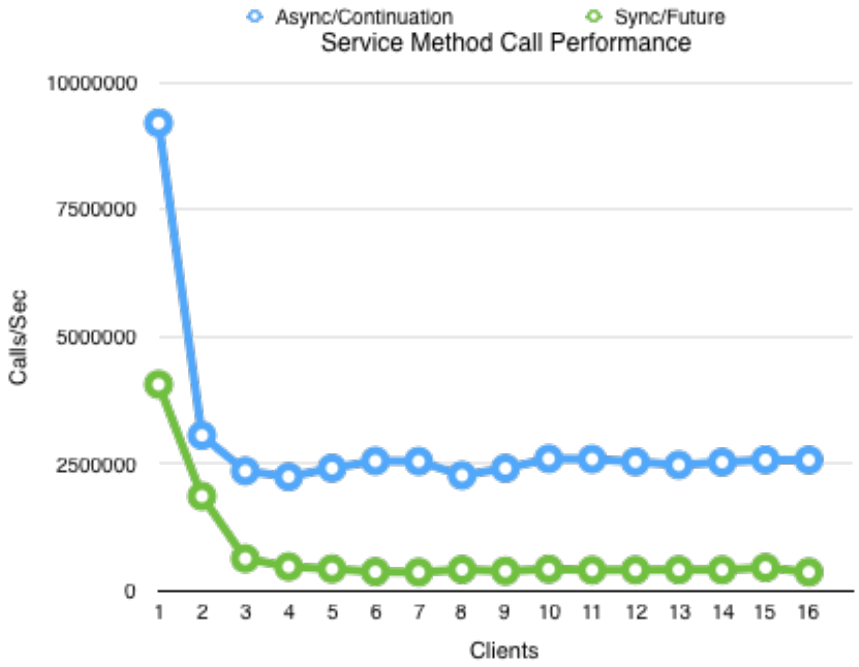


3. Write-through in place
    3.1. Baratine saves the fields of the asset to an embedded, fully reactive document database named Kraken (No data schema)

4. Journaling for reliability/failover
    4.1. The inbox is backed by a repayable journal for failover

DEVOXX™ UNITED STATES

Baratine Performance



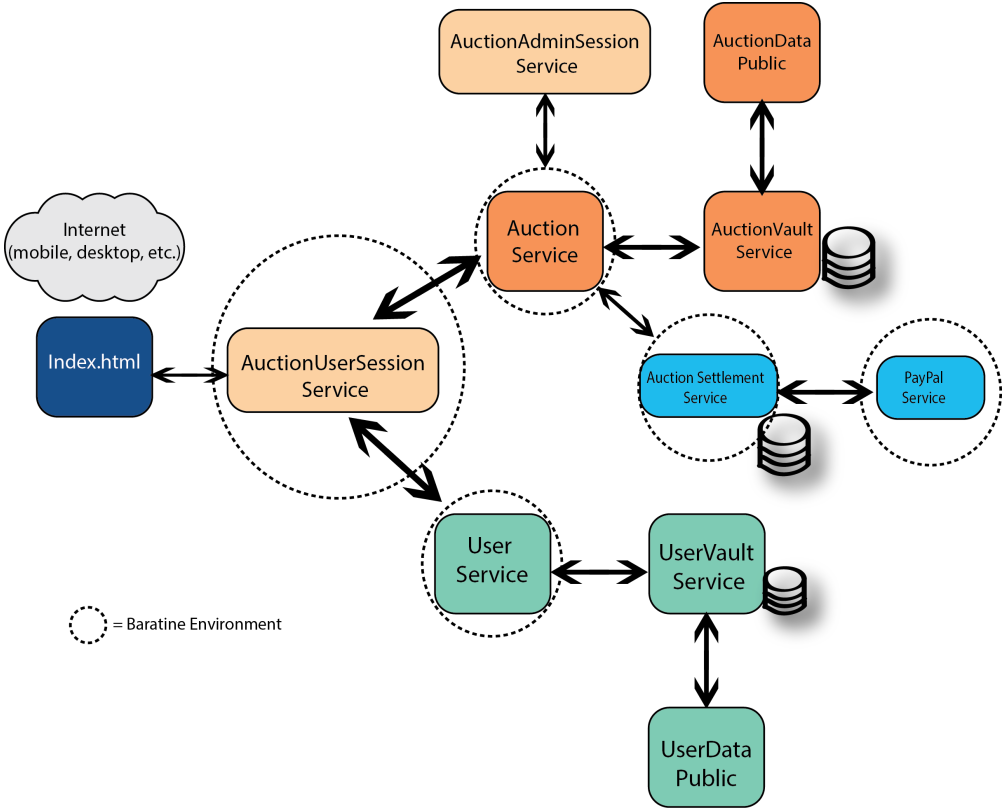
- Continuation/async performance is consistently better than async/future, especially under load.
- Under light load, async performance is around 9M method calls/sec while future performance is 4M calls/sec
- As the load increases future performance drops to 0.5M calls/sec. Async performance drops as well but stays over 2.2M calls/sec.

DEVOXX™ UNITED STATES

Use Case #2: Microservice

DEVOXX UNITED STATES

Auction Microservice



The application uses the following Baratine provided services

- DatabaseService
- Store (key - value)
- Timer
- Event
- WebSocket

DEVOXX™ UNITED STATES

Questions?

DEVOXX™ UNITED STATES

Thank You!

baratine.io