Shopping Lists on the Cloud

g14

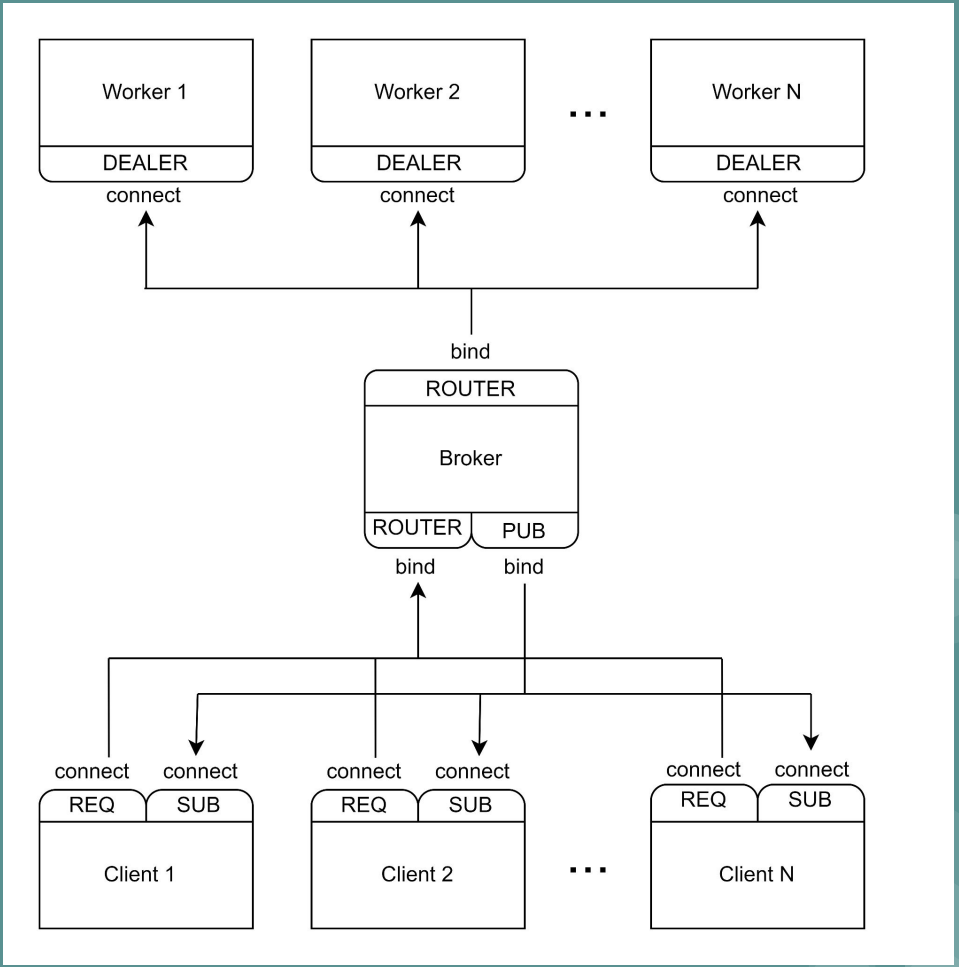
António Rego - up202108666
João Coelho - up202004846
João Mota - up202108677
Pedro Lima - up202108806

Problem Description

Key Features & Functionalities:

- **Create Shopping Lists**:
  - Users can create new shopping lists via the CLI.
  - Each list is assigned a unique ID (MD5 encrypted hash) for sharing.
- **Shared Access**:
  - Shared lists allow all users with the unique ID to add, modify, and delete items.
- **Item Management**:
  - Add, edit, or delete items.
- **Concurrency Handling**:
  - Uses a **Conflict-free Replicated Data Type (CRDT)** for real-time conflict resolution.
- **Local-First Design**:
  - Data persists locally on user devices.
  - Cloud component for data sharing and backup.

Design


Worker 1
DEALER
connect
Worker 2
DEALER
connect
...
Worker N
DEALER
connect
bind
ROUTER
Broker
ROUTER
PUB
bind
bind
connect
connect
REQ
SUB
Client 1
connect
connect
REQ
SUB
Client 2
...
connect
connect
REQ
SUB
Client N

ZMQ

- Using a functionally REQ/REP messaging pattern, where the client sends a request to the server and waits for a reply;
- Uses the nlohmann JSON library for C++ [8] to handle lists and communication between endpoints;
- Uses a Load Balancing Message Broker [5] - Communication with a centralized server (broker) that manages task distribution to workers through a ROUTER-ROUTER structure;
- Publisher/Subscriber Pattern [4] - Uses PUB (Publisher) and SUB (Subscriber) sockets for broadcasting and receiving messages on specific topics.

Broker

- Routes incoming client requests to the appropriate worker nodes based on a consistent hashing mechanism;
- When workers join the system, the broker registers them and integrates their virtual nodes into the consistent hash ring;
- If the number of available workers is below the replication factor (3), the broker forwards requests to all available workers to ensure data availability and continuity;
- Worker replies are forwarded back to the corresponding clients;
    - Responses can be published to a **subscriber** socket (PUB/SUB architecture) - allows for real-time monitoring and event-driven processing.

Cloud

- Cloud-side architecture inspired by Amazon Dynamo [3] for high availability and scalability.
- Utilizes consistent hashing with a ring-based structure of workers to distribute and manage data efficiently.
    - Each worker is assigned multiple virtual nodes;
    - Hashing uses the OpenSSL EVP library for C++ [7];
- Replication to allow for fault-resistance and data availability in the event of worker failure.
- Threads;

CRDT

Composable Map, based on the paper:

Guarantees in the presence of CRDT composition [2]

The Composable map allows:

- *get(key)*
- *put(key,obj)*
- *rem(key)*
- *merge* of two replicas of a given map

In the paper, the values stored can be any CRDT. It also allows for any mechanism to determine the order of events.

In our case, the keys are always strings and the values are always Counter CRDTs described on the right. It uses Vector Clocks to encode ordering.

Embedded Resettable Counter, from the paper:

The problem with embedded CRDT counters and a solution [1]

$$
\begin{aligned}
\mathsf{Counter} &= (\mathbb{I} \times \mathbb{N} \hookrightarrow \mathbb{N} \times \mathbb{N}) \times (\mathbb{I} \hookrightarrow \mathbb{N}) \\
\bot &= (\{\}, \{\}) \\
\mathsf{inc}_i(s) &= \mathsf{upd}_i(s, (1, 0)) \\
\mathsf{dec}_i(s) &= \mathsf{upd}_i(s, (0, 1)) \\
\mathsf{upd}_i((m, c), u) &= (m'\{d \mapsto m'(d) + u\}, c') \text{ where } d = (i, c'(i)), \\
&\quad (m', c') = \begin{cases} \mathsf{fresh}_i((m, c)) & \text{if } (i, c(i)) \notin \mathsf{dom}\, m \\ (m, c) & \text{otherwise} \end{cases} \\
\mathsf{fresh}_i((m, c)) &= (m\{(i, c(i) + 1) \mapsto (0, 0)\}, c\{i \mapsto c(i) + 1\}) \\
\mathsf{reset}_i((m, c)) &= (\{\}, c) \\
\mathsf{value}_i((m, c)) &= \sum_{d \in \mathsf{dom}\, m} \mathsf{fst}\, m(d) - \mathsf{snd}\, m(d) \\
(m, c) \sqcup (m', c') &= (\{d \mapsto m(d) \sqcup m'(d) \mid d \in \mathsf{dom}\, m' \cap \mathsf{dom}\, m\} \cup \\
&\quad \{((j, n), v) \in m \mid n > c'(j)\} \cup \{((j, n), v) \in m' \mid n > c(j)\}, \\
&\quad c \sqcup c')
\end{aligned}
$$

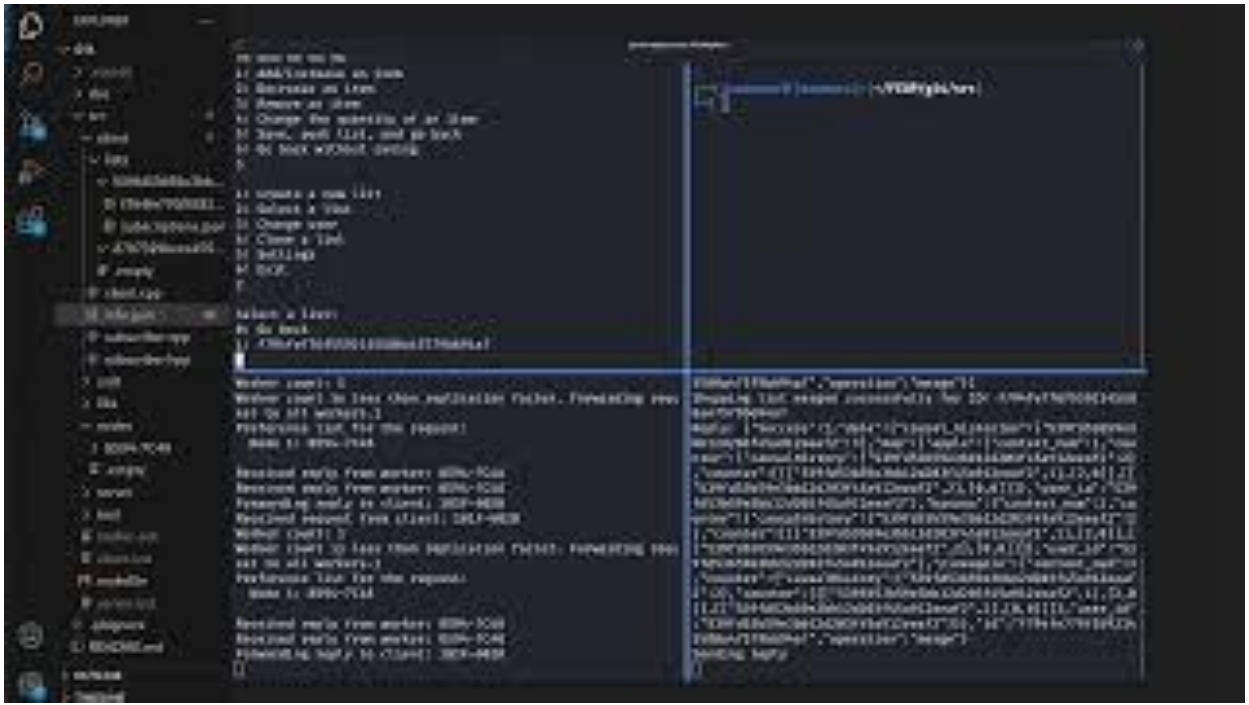
Limitations

- If the broker fails, it is not possible to connect to the cloud;
- There is no tag to mark products as bought;
- Products are only represented as strings, not as complex objects;
- Program needs to be restarted in order to have a connection if client is started in local mode.
- Threads can lead to race conditions;

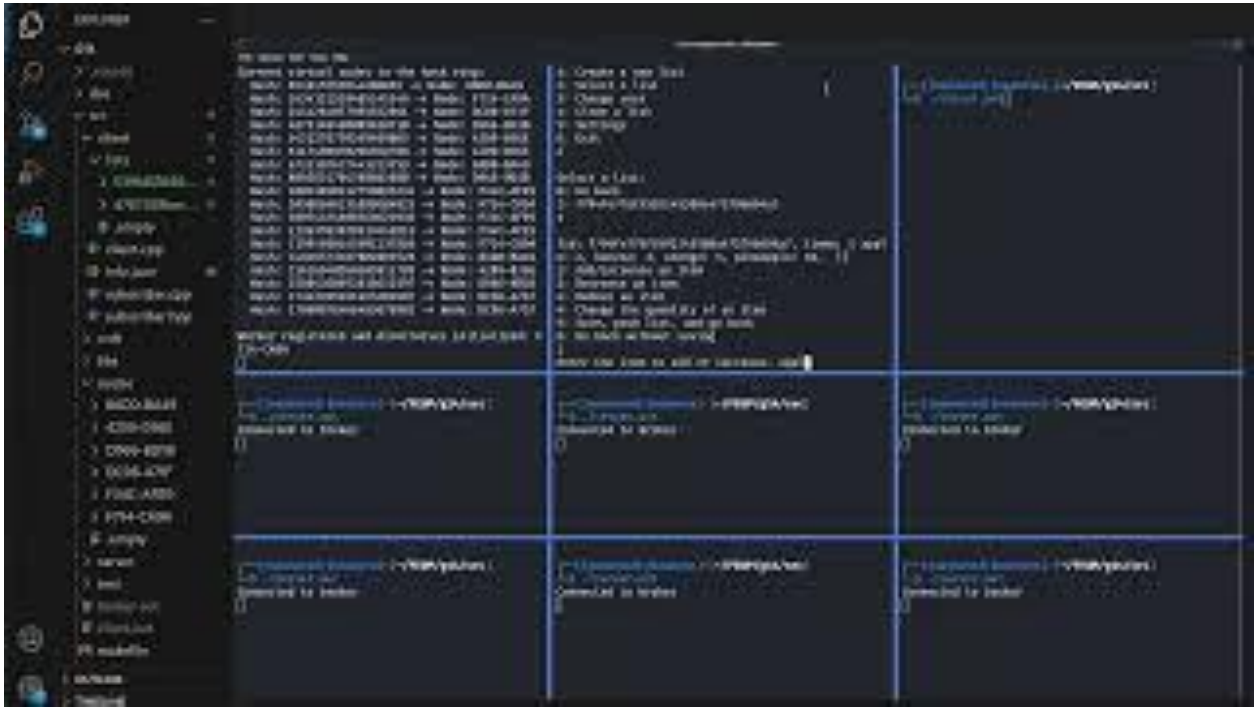
Test Cases

Offline Client

List Cloning and CRDT in action (Multiple clients alter same list)



Replication



Bibliography

[1] C. Baquero, P. Almeida, C. Lerche (2016). *The problem with embedded CRDT counters and a solution*. PAPOC 2016. Retrieved from https://repositorium.sdum.uminho.pt/bitstream/1822/51503/1/Problem-Solution-Counters-PAPOC2016.pdf.

[2] A. Bieniusa, N. Preguiça, C. Baquero, et al. (2016). *European Seventh Framework Programme ICT call 10 - Guarantees in the presence of CRDT composition*. Retrieved from https://pages.lip6.fr/syncfree/attachments/article/46/WP3-report.pdf#page=31.

[3] G. DeCandia, D. Hastorun, M. Jampani, et al. (2007). Dynamo: Amazon's Highly Available Key-Value Store. *Proceedings of the 21st ACM Symposium on Operating Systems Principles (SOSP 2007)*. Retrieved from https://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf.

[4] P. Hintjens (n.d.). *The Dynamic Discovery Problem*. In *ZeroMQ Guide* (Chapter 2). Retrieved from https://zguide.zeromq.org/docs/chapter2/#The-Dynamic-Discovery-Problem.

[5] P. Hintjens (n.d.). *A Load Balancing Message Broker*. In *ZeroMQ Guide* (Chapter 3). Retrieved from https://zguide.zeromq.org/docs/chapter3/#A-Load-Balancing-Message-Broker.

[6] cppreference contributors. (n.d.). *cppreference: Standard C++ Library Reference*. Retrieved from https://en.cppreference.com.

[7] OpenSSL Project. (n.d.). *OpenSSL EVP API Manual*. Retrieved from https://wiki.openssl.org/index.php/EVP

[8] N. Lohmann (n.d.). *JSON for Modern C++*. GitHub. Retrieved from https://github.com/nlohmann/json.