

Distributed Multi-Warehouse Inventory Manager

Enrico Cornacchia

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
ALMA MATER STUDIORUM – Università di Bologna

Corso di Distributed Systems

Academic Year 2025/2026

Il Problema della Centralizzazione

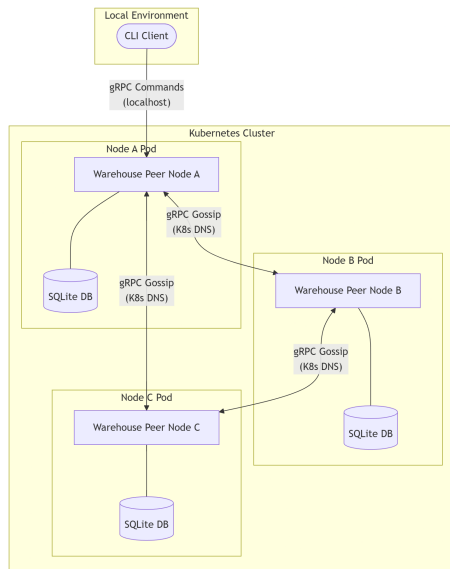
- **Sistemi Tradizionali:** Si basano su un database centralizzato.
- **Vulnerabilità:** Single Point of Failure e bottlenecks legati alla latenza.
- **Il Limite Architetturale:** Se un magazzino perde la connessione alla rete principale, tutte le operazioni locali si bloccano completamente.

Visione del Progetto

- **Scelta Progettuale:** Prioritizzare **AP** (Availability e Partition Tolerance) nel Teorema CAP.
- **Architettura Decentralizzata:** Rimozione del DB centrale in favore di una rete di nodi peer-to-peer indipendenti.
- **Risultato:** *Eventual Consistency*. Le operazioni locali non si fermano mai, anche in caso di partizione di rete (Network Partition).

Architettura del Sistema

- **Stato Locale:** Ogni nodo mantiene il proprio stato utilizzando un database SQLite integrato.
- **Comunicazione Asincrona:** Protocollo Gossip basato su gRPC.
- **Assenza di un Master:** Ogni nodo ha pari priorità nella rete.



La Sfida della Concorrenza

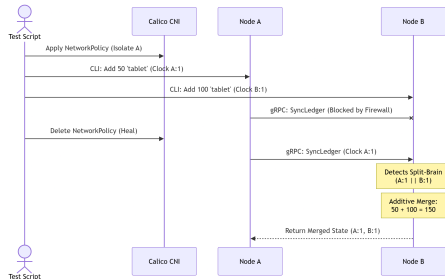
- **Lo Scenario:** Il Magazzino A e il Magazzino B aggiornano la disponibilità dello stesso articolo contemporaneamente, mentre la rete è partizionata.
- **Perché i Timestamp falliscono:** Il "clock drift" rende inaffidabile il tempo assoluto nei sistemi distribuiti.
- **Necessità:** Un meccanismo per determinare se gli eventi sono sequenziali o se si sono verificati in modo concorrente durante uno *split-brain*.

Meccanismo Core: Vector Clocks

- **Concetto:** Tracciare la *causalità logica* degli eventi invece del tempo assoluto.
- **Struttura:** Un array di contatori (es. [A:2, B:1, C:0]).
- **Rilevamento Conflitti:** Confrontando i Vector Clocks, il sistema può dimostrare matematicamente l'avvenuta concorrenza delle operazioni isolate, prevenendo sovrascritture accidentali.

Riconciliazione dello Stato

- **Trigger:** Ripristino della connessione di rete.
- **Rilevamento:** I nodi scambiano i Vector Clocks. Il sistema rileva lo *split-brain* in caso di operazioni concorrenti.
- **Risoluzione:** Merge additivo delle quantità per unire gli stati senza perdita di dati.



Tecnologie e Infrastruttura

- **Linguaggio:** Go (primitive di concorrenza avanzate e supporto nativo gRPC).
- **Containerizzazione:** Docker.
- **Orchestrazione:** Kubernetes (Minikube).
- **Networking:** Project Calico (per simulare il sezionamento della rete a livello OSI 3).

Validazione e Test Automatizzati

Script Bash progettati per validare l'infrastruttura:

- ❶ **Esecuzione Normale:** Discovery dei peer e sincronizzazione dello stato gRPC.
- ❷ **Node Failure:** Tolleranza ai crash con recupero automatico del ledger da disco.
- ❸ **Network Partition:** Utilizzo di Calico per isolare un nodo, iniezione di dati contrastanti (*split-brain*), e successiva validazione della riconciliazione automatica tramite Vector Clocks.

Conclusioni

- **Obiettivi Raggiunti:** Decentralizzazione, Fault Tolerance, ed Eventual Consistency.
- **Dimostrazione:** Il sistema resiste attivamente a disconnessioni prolungate garantendo la continuità del servizio locale.