SENSOR CONTINUUM

Distributed application in the compute continuum

MAURIZIO
RENZI
0294643

FRANCESCO
MASCI
0365258

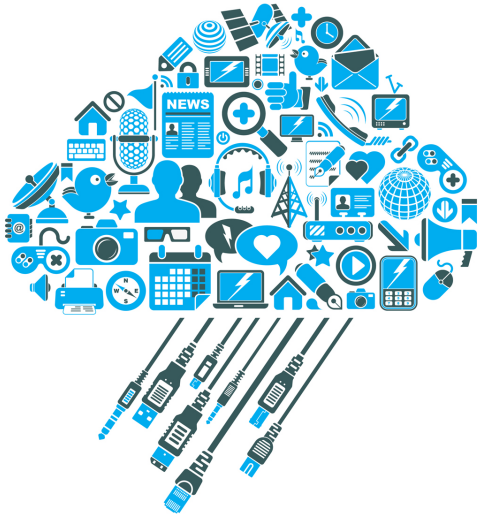
I LIMITI DEL CLOUD COMPUTING TRADIZIONALE

- ***Latenza Elevata***: Il tempo di andata e ritorno dei dati grezzi verso il cloud è un collo di bottiglia per le applicazioni real-time.

- ***Costi di Banda***: Trasferire enormi volumi di dati da migliaia di sensori diventa economicamente insostenibile.

- ***Scalabilità Ridotta***: I data center centralizzati faticano a gestire picchi di dati provenienti da fonti distribuite.

- ***Dipendenza dalla rete***: La dipendenza totale dal cloud rende il sistema vulnerabile a interruzioni di rete.



OBIETTIVI DEL PROGETTO

Elaborazione distribuita

Distribuire il carico computazionale sui vari livelli distinti per ottimizzare le risorse.

Bassa latenza

Processare i dati il più vicino possibile alla fonte per risposte in tempo reale.

Resilienza

Usare pattern robusti per garantire l’affidabilità dei dati.

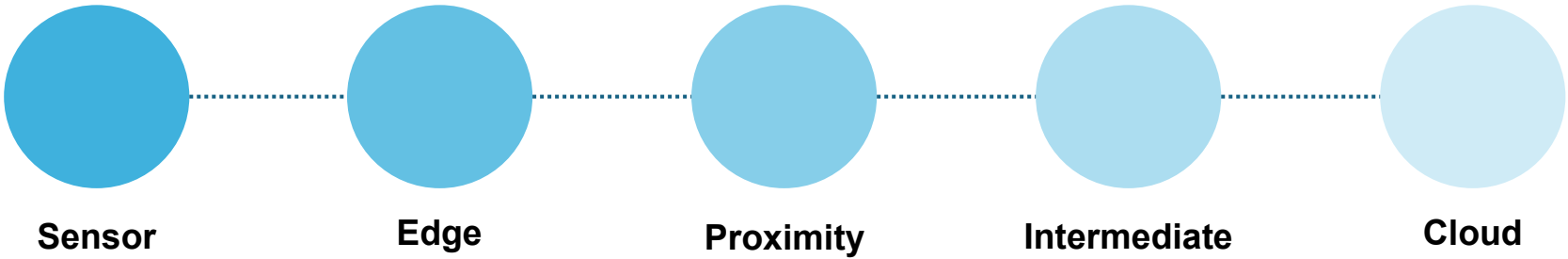
Scalabilità

Sfruttare un’architettura a microservizi per una crescita modulare.

IL PARADIGMA DEL COMPUTE CONTINUUM

- Un ecosistema coeso in cui le risorse computazionali collaborano in modo trasparente, dal dispositivo più piccolo al data center più grande.

- Questo approccio integra vari livelli di elaborazione per allocare le risorse nel punto più opportuno, eseguendo l'elaborazione più vicino alla fonte.



I LIVELLI DEL CONTINUUM

Servizi Cloud

Analisi complesse, API per client, storage a lungo termine.

AWS Lambda, API Gateway, PostgreSQL

Intermediate Fog

Aggregazione a livello di regione e persistenza dei dati.

Kafka, PostgreSQL, Batching

Proximity Fog

Aggregazione livello di zona e macrozona, buffering dei dati.

MQTT, Kafka, Outbox

Edge

Filtraggio in tempo reale e prima aggregazione.

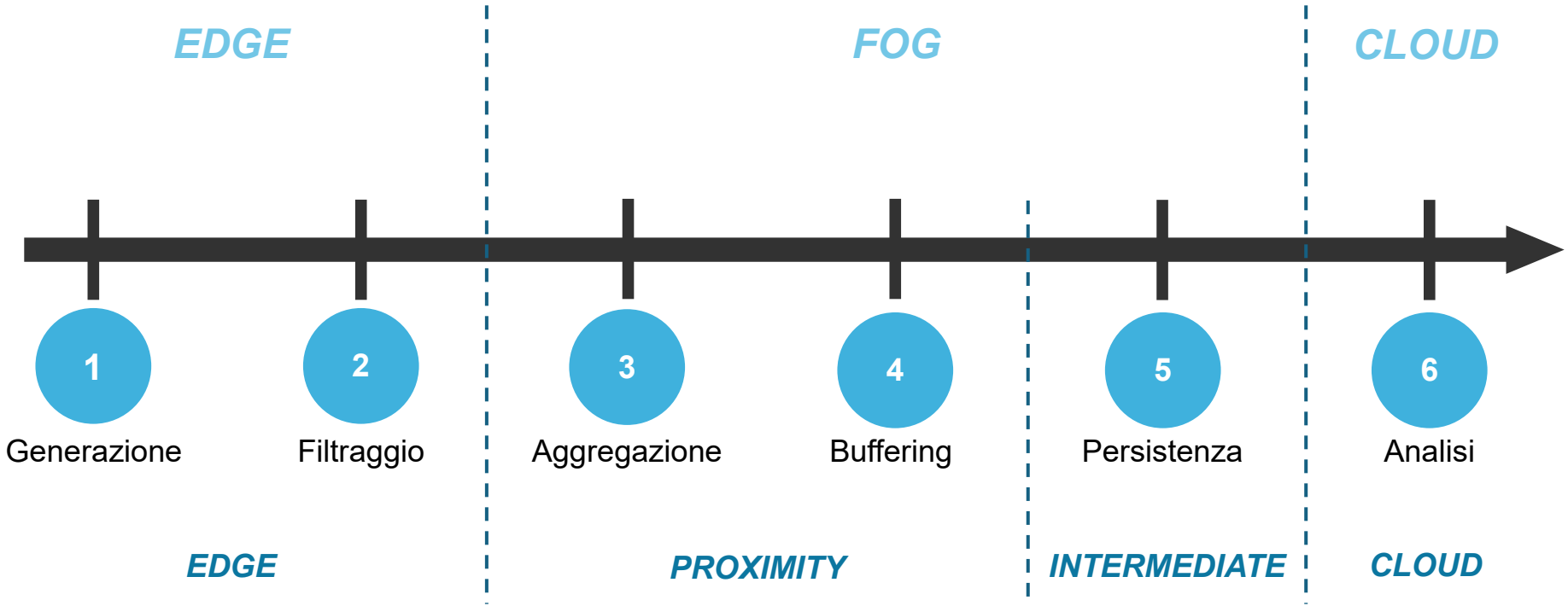
Redis, MQTT

Sensor

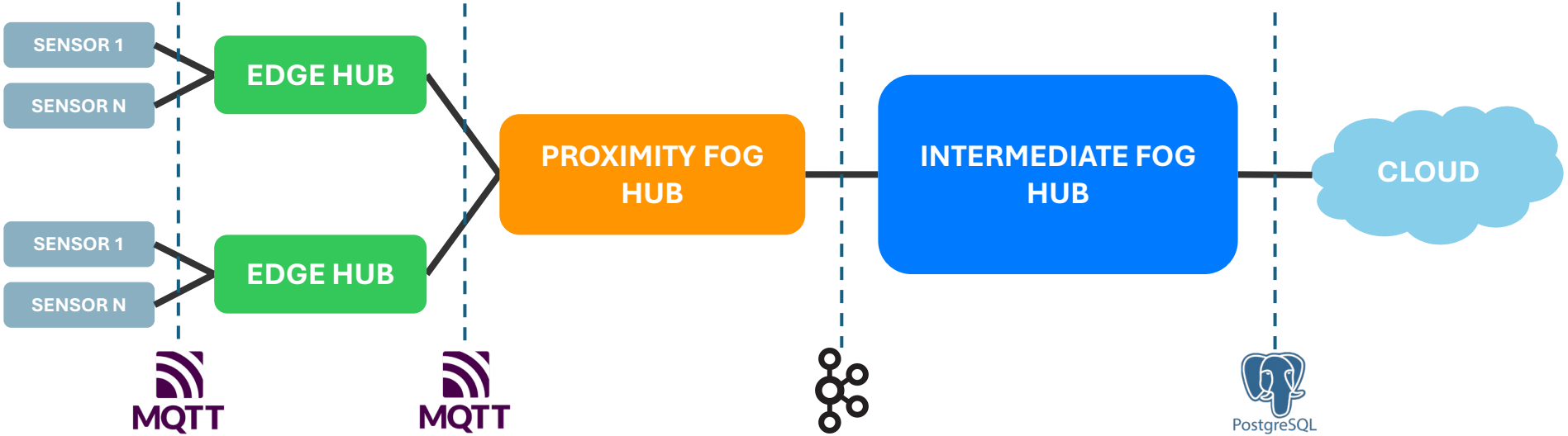
Generazione e invio dei dati grezzi di misurazione.

Simulazione dei dataset reali, MQTT

IL VIAGGIO DI UNA MISURAZIONE

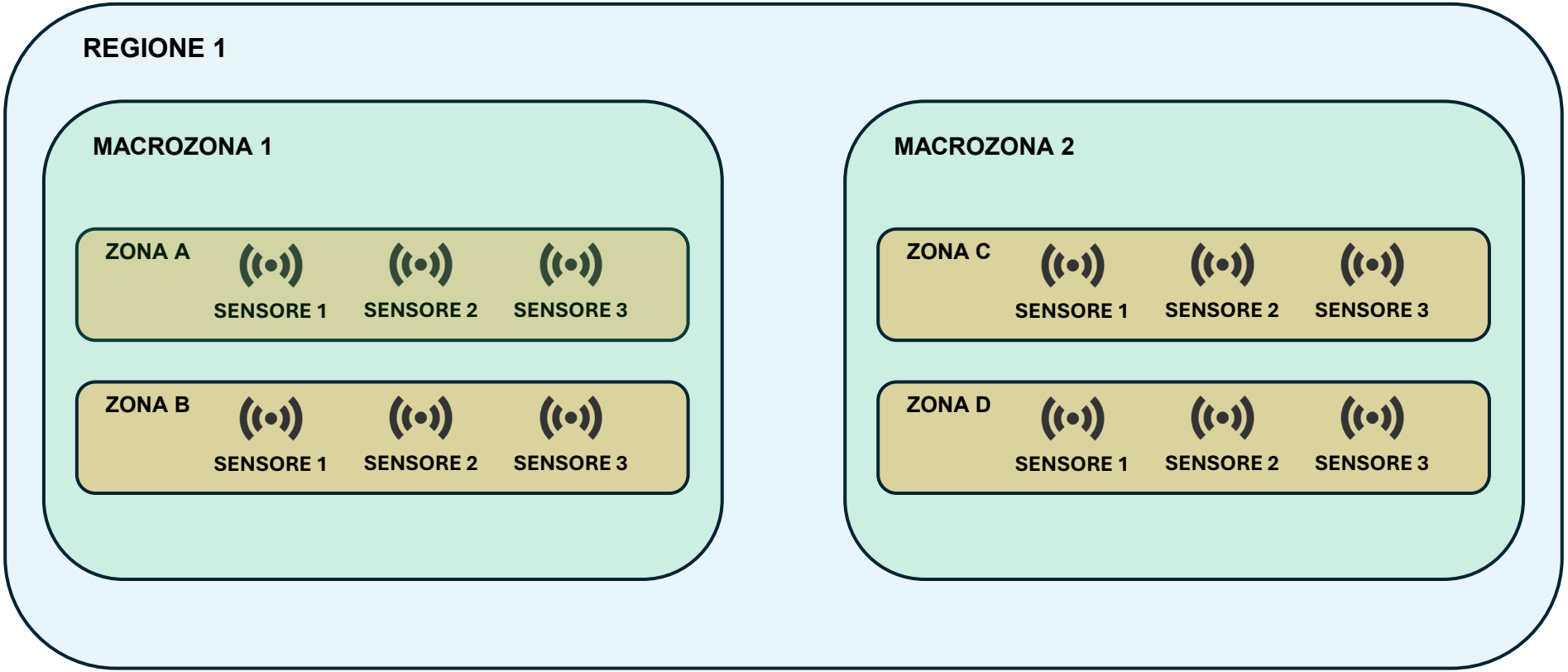


IL CUORE DEL SISTEMA



- ***Edge Hub***: Filtraggio intelligente alla fonte
- ***Proximity Hub***: Garanzia di consegna dei dati verso la rete Internet
- ***Intermediate Hub***: Persistenza e ottimizzazione in scrittura

ARCHITETTURA GERARCHICA DEL SISTEMA



STACK TECNOLOGICO

RUNTIME

- Go
- Docker & Docker Compose
- React

COMUNICATION

- MQTT
- Apache Kafka
- HTTP

STORAGE

- PostgreSQL: TimescaleDB & PostGIS
- Redis
- Amazon Aurora

CLOUD

- CloudFormation & S3
- Lambda & API Gateway
- Route 53
- Amplify

PATTERN ARCHITETTURALI CHIAVE

TRANSACTIONAL OUTBOX

Garantisce la consegna affidabile dei messaggi e previene la perdita di dati in caso di fallimenti temporanei della rete o dei servizi

RICEVITORE IDEMPOTENTE

Assicura che i messaggi duplicati vengano processati una sola volta, mantenendo la consistenza dei dati

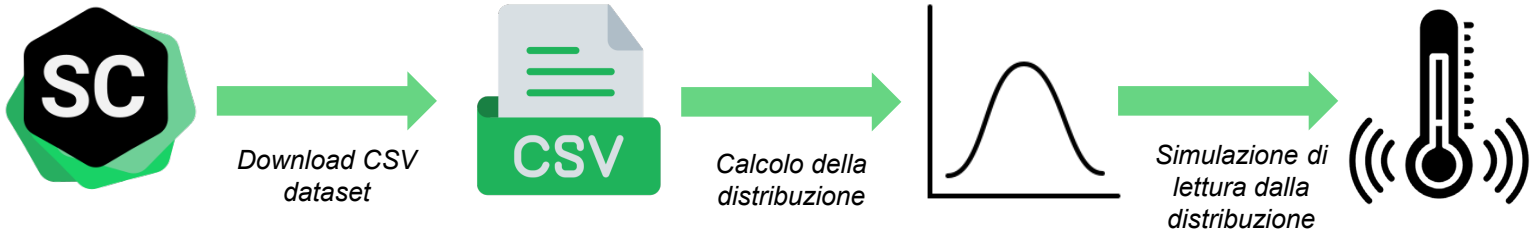
MICROSERVIZI

Permette lo sviluppo, il deployment e la scalabilità indipendente di ogni componente del sistema

CACHE ASIDE

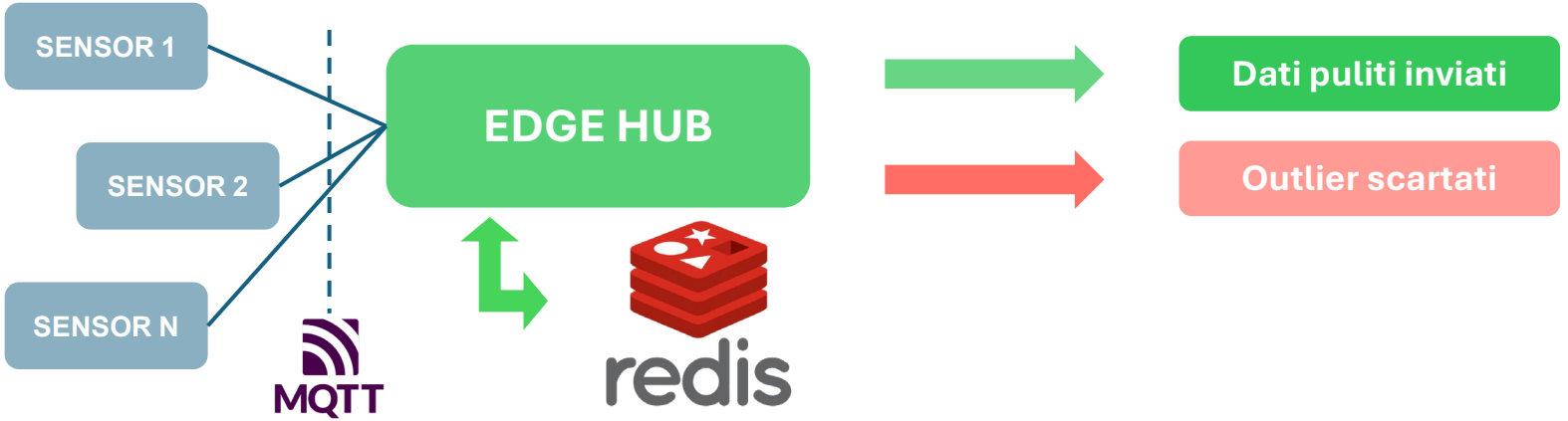
Utilizzo di redis per velocizzare l’accesso ai dati recenti e utilizzo di cache di tipo timescaldb mantenere riferimento ai dati ancora pendenti

SENSOR AGENT: SIMULAZIONE REALISTICA



- **Simulazione realistica:** Sfruttando un dataset reale, le letture riflettono una distribuzione di valori reali.
- **Aggiornamento distribuzione:** La distribuzione viene ricalcolata su base oraria per avere dei valori coerenti e dinamici.
- **Aggiornamento dataset:** Il dataset viene scaricato ad ogni avvio in modo da utilizzare dati aggiornati.

EDGE HUB: FILTRAGGIO INTELLIGENTE



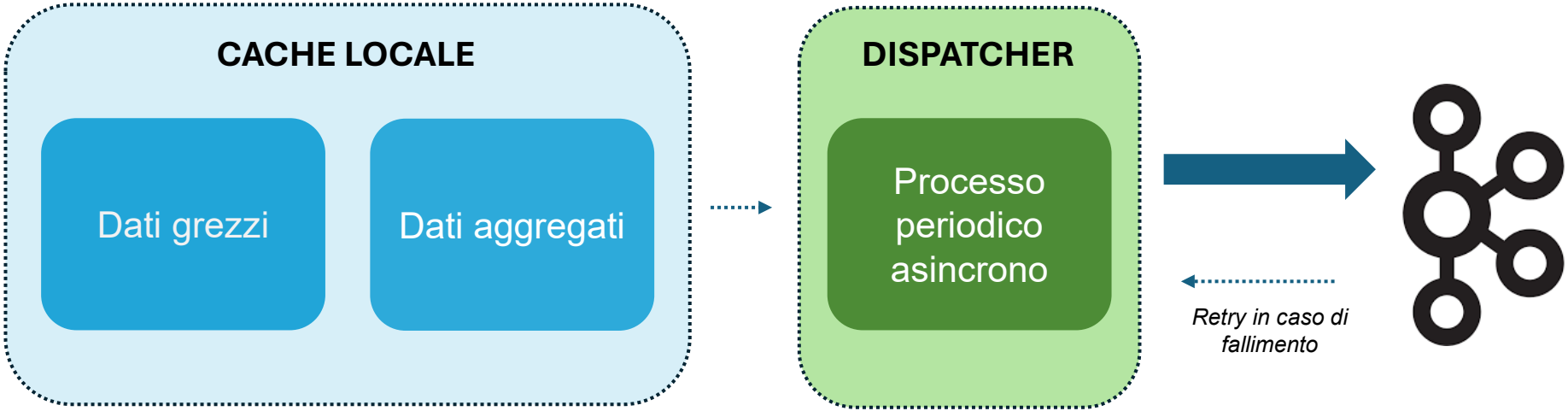
- **Riduzione del Traffico di Rete:** Solo i dati significativi e validati vengono inoltrati al livello successivo.

- **Minore Carico Computazionale:** Gli Hub Proximity e Intermediate non sprecano risorse per processare dati inutili.

- **Qualità dei Dati Migliore:** L'analisi sui livelli superiori parte da una base dati più pulita e affidabile.

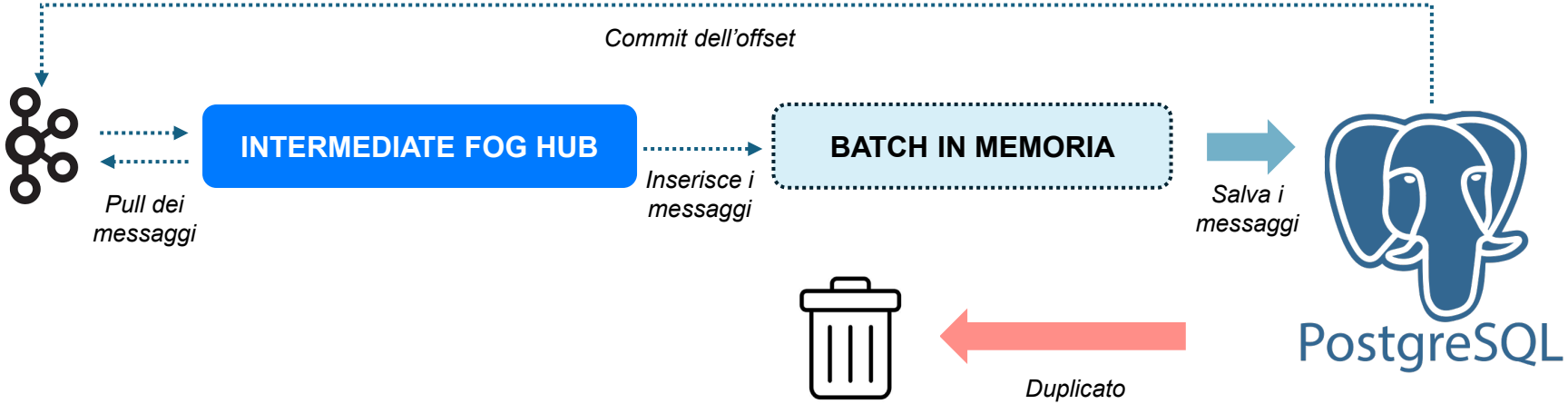
PROXIMITY HUB: GARANZIA CONSEGNA DATI

- ***Problema***: La comunicazione tra servizi distribuiti in rete può fallire. Come evitare la perdita di dati?



- ***Risultato***: Garanzia di consegna "at-least-once". I dati vengono salvati localmente prima di ogni tentativo di invio. Se l'invio fallisce, il dispatcher riproverà in seguito, garantendo affidabilità in caso di assenza di rete

INTERMEDIATE HUB: BATCHING E CONSISTENZA



- **Consistenza Assoluta**: Ogni record viene archiviato una e una sola volta.
- **Robustezza del Sistema**: Il sistema è immune a problemi di rete che causano ritrasmissioni.
- **Operazione Atomica**: Il controllo e il salvaggio avvengono in un'unica transazione sicura, prevenendo stati inconsistenti.

CLOUD SERVICES: ACCESSO E ANALISI AVANZATA

Lambda
Logica backend serverless per l'elaborazione delle richieste API e analisi periodiche

API Gateway
Espone le funzionalità del backend come un' API RESTful sicura e scalabile

Route 53
Gestione del DNS per mappare un dominio pubblico all'API Gateway e ai vari database regionali

Amazon Aurora
Database gestito per lo storage a lungo termine di metadati globali

CloudFormation
Automazione del deployment dell'infrastruttura cloud per coerenza e riproducibilità

Funzionalità: Analisi di trend annuali e correlazione tra macrozone

GESTIONE E MANUTENZIONE

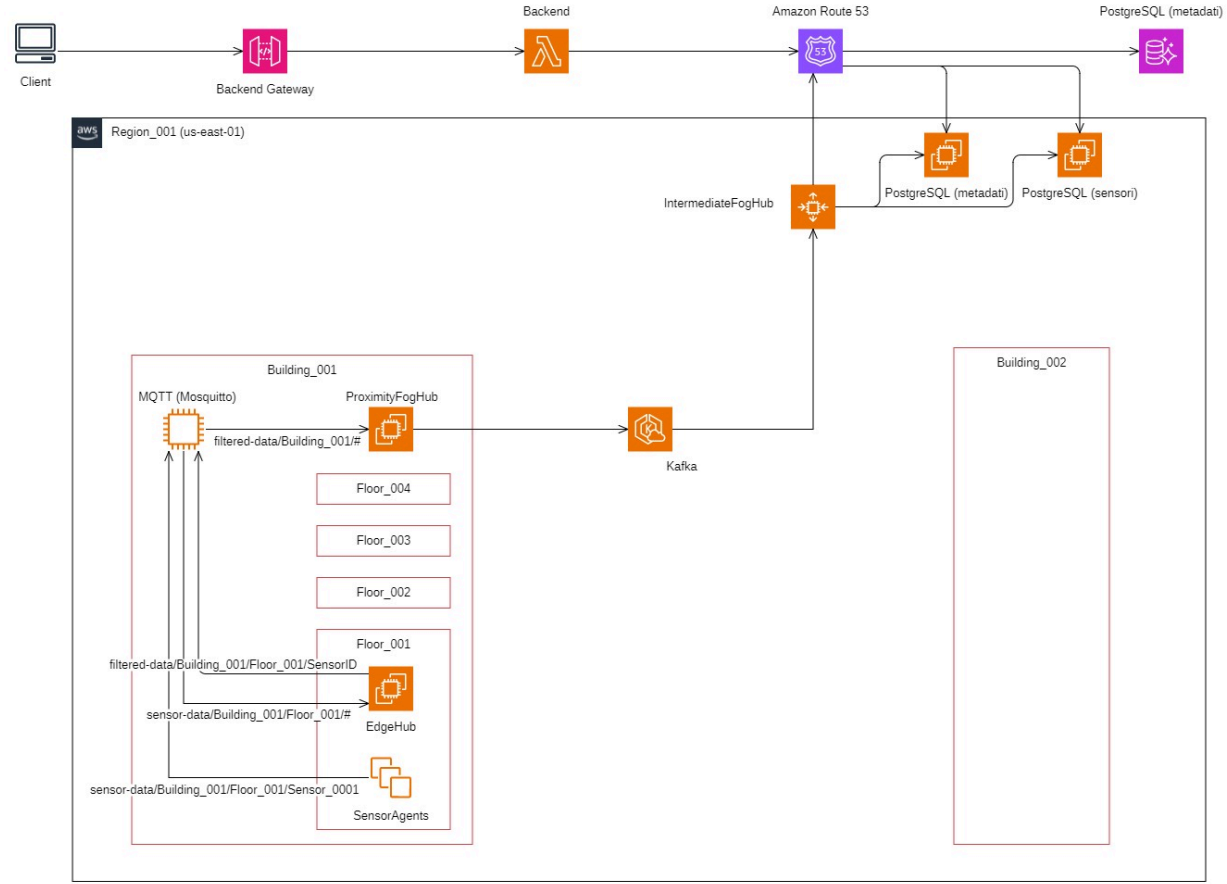
HEARTBEAT

- Ogni Hub notifica al livello superiore di essere ancora in funzione correttamente.
- I sensori non inviano il messaggio, ma viene dedotto dall'ultima lettura rilevazione effettuata.
- Questi metadati vengono mantenuti in un database specifico per ogni regione.

CONFIGURATION

- All'avvio, ogni componente invia al livello superiore un messaggio di registrazione al sistema.
- Se il componente è già registrato, viene solamente aggiornato il suo stato.
- Questi metadati vengono mantenuti in un database specifico per ogni regione.

ARCHITETTURA DI DEPLOYMENT



TEST FUNZIONALE

```
[ec2-user@ip-10-0-1-34 ~]$ ./extract_logs.sh
=== MESSAGGI SENSORI ===
[INFO]    macrozone=build-0001 sensor=sensor-agent-01 service=sensor-agent zone=floor-001 2025/09/27 02:54:31 simulator.go:55: Sensor reading: 24.93125538484429
[INFO]    macrozone=build-0001 sensor=sensor-agent-01 service=sensor-agent zone=floor-001 2025/09/27 02:54:36 simulator.go:55: Sensor reading: 24.977288130685743
[INFO]    macrozone=build-0001 sensor=sensor-agent-01 service=sensor-agent zone=floor-001 2025/09/27 02:54:41 simulator.go:55: Sensor reading: 24.914916298465492
[INFO]    macrozone=build-0001 sensor=sensor-agent-01 service=sensor-agent zone=floor-001 2025/09/27 02:54:46 simulator.go:55: Sensor reading: 24.86190042359859
---------------------------------------
=== MESSAGGI NEGLI HUB (filtrati per sensor-agent-01) ===
>>> Hub: zone-hub-filter-floor-001-01
[INFO]    hub=zone-hub-filter-01 macrozone=build-0001 service=edge_hub_filter zone=floor-001 2025/09/27 02:54:31 sensor-handler.go:21: Processing data for sensor sensor-agent-01 - value: 24.93125538484429, timestamp: 1758941671
[INFO]    hub=zone-hub-filter-01 macrozone=build-0001 service=edge_hub_filter zone=floor-001 2025/09/27 02:54:31 sensor-handler.go:40: Data is valid for sensor: sensor-agent-01
[INFO]    hub=zone-hub-filter-01 macrozone=build-0001 service=edge_hub_filter zone=floor-001 2025/09/27 02:54:46 sensor-handler.go:21: Processing data for sensor sensor-agent-01 - value: 24.86190042359859, timestamp: 1758941686
[INFO]    hub=zone-hub-filter-01 macrozone=build-0001 service=edge_hub_filter zone=floor-001 2025/09/27 02:54:46 sensor-handler.go:40: Data is valid for sensor: sensor-agent-01
---------------------------------------
>>> Hub: zone-hub-filter-floor-001-02
[INFO]    hub=zone-hub-filter-02 macrozone=build-0001 service=edge_hub_filter zone=floor-001 2025/09/27 02:54:36 sensor-handler.go:21: Processing data for sensor sensor-agent-01 - value: 24.977288130685743, timestamp: 1758941676
[INFO]    hub=zone-hub-filter-02 macrozone=build-0001 service=edge_hub_filter zone=floor-001 2025/09/27 02:54:36 sensor-handler.go:40: Data is valid for sensor: sensor-agent-01
[INFO]    hub=zone-hub-filter-02 macrozone=build-0001 service=edge_hub_filter zone=floor-001 2025/09/27 02:54:41 sensor-handler.go:21: Processing data for sensor sensor-agent-01 - value: 24.914916298465492, timestamp: 1758941681
[INFO]    hub=zone-hub-filter-02 macrozone=build-0001 service=edge_hub_filter zone=floor-001 2025/09/27 02:54:41 sensor-handler.go:40: Data is valid for sensor: sensor-agent-01
---------------------------------------
```

Log Edge Hub e sensore

Cache Proximity Fog Hub

sensor_me...ents_cache [Proximity Cache | EC2]

```
WHERE time='2025-09-27 02:54:00.000000 +00:00' AND macrozone_name = 'build-0001' AND zone_name = 'floor-001'  ORDER BY sensor_id
```

	time	macrozone_name	zone_name	sensor_id	type	value	status
1	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-01	temperature	24.956080924477025	sent
2	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-02	temperature	15.335099468639196	sent
3	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-04	humidity	50.974773327092905	sent
4	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-05	temperature	9.163060531930608	sent
5	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-06	temperature	9.925166674947588	sent
6	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-07	pressure	61772.69000000004	sent
7	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-08	temperature	8.281947850850152	sent
8	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-09	temperature	19.28335225792949	sent

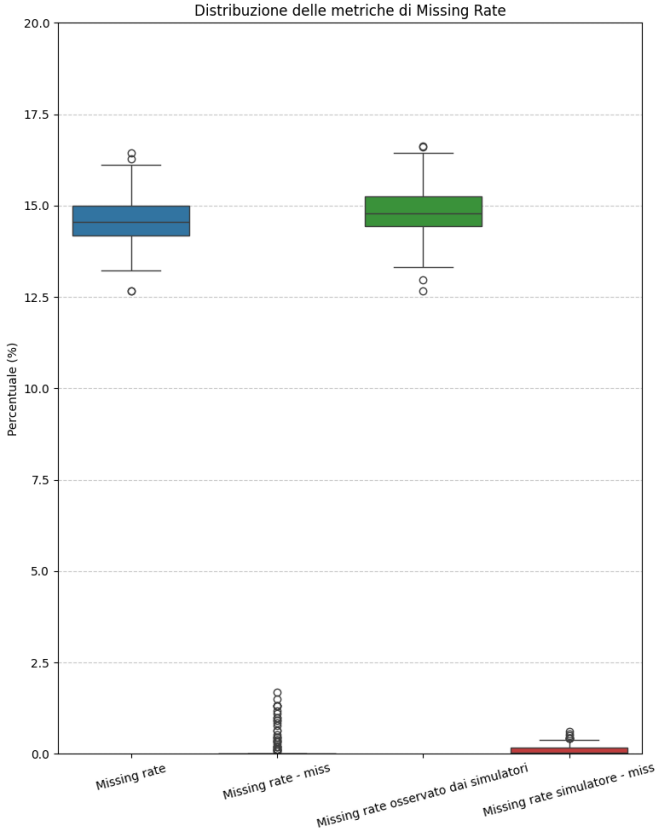
sensor_measurements [Sensor DB | EC2]

```
WHERE time='2025-09-27 02:54:00.000000 +00:00' AND macrozone_name = 'build-0001' AND zone_name = 'floor-001'  ORDER BY sensor_id
```

	time	macrozone_name	zone_name	sensor_id	type	value
1	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-01	temperature	24.956080924477025
2	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-02	temperature	15.335099468639196
3	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-04	humidity	50.974773327092905
4	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-05	temperature	9.163060531930608
5	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-06	temperature	9.925166674947588
6	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-07	pressure	61772.69000000004
7	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-08	temperature	8.281947850850152
8	2025-09-27 02:54:00.000000 +00:00	build-0001	floor-001	sensor-agent-09	temperature	19.28335225792949

Database regionale

TEST DELLE PERFORMANCE

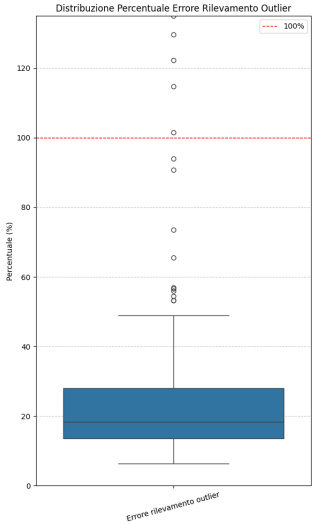


- È stato calcolato il Miss Rate dei pacchetti, definito come la percentuale di pacchetti generati dai sensori che non raggiungono l'Edge Hub.

- Questo rate è stato determinato confrontando il numero atteso di pacchetti inviati (sulla base dei parametri di simulazione) e il numero reale di pacchetti inviati con il numero di pacchetti effettivamente ricevuti.

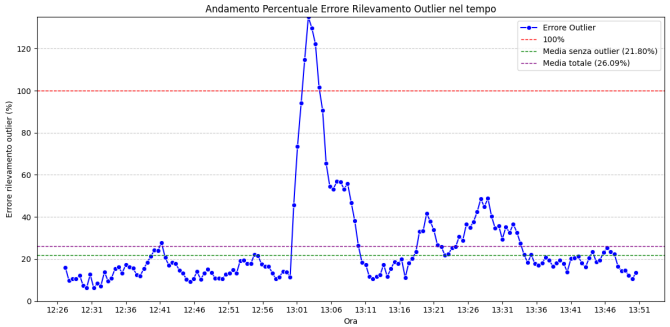
- L'analisi del Miss Rate mostra valori che si attestano stabilmente attorno al 15%. Tale risultato è in linea con i parametri della simulazione, che introducevano artificialmente una probabilità del 15% di non generare o perdere un pacchetto.

TEST DELLE PERFORMANCE

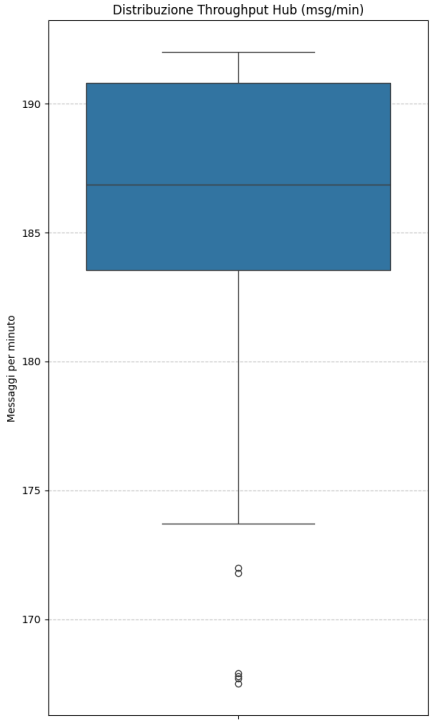


- È stato valutato l'errore percentuale commesso dagli hub nel rilevamento degli outlier.
- Questo errore è calcolato confrontando gli outlier identificati dal sistema con gli outlier introdotti intenzionalmente dal simulatore.
- L'hub tende a sovrastimare il numero di outlier di circa il 20%, mostrando un risultato accettabile per un sistema edge che privilegia la cautela (scartare dati potenzialmente validi) rispetto all'introduzione di valori anomali nel sistema.

- I picchi di errore si osservano in corrispondenza del cambio dell'ora e sono dovuti alla natura del simulatore, che ricalcola i parametri di media e deviazione standard per i sensori su intervalli orari.

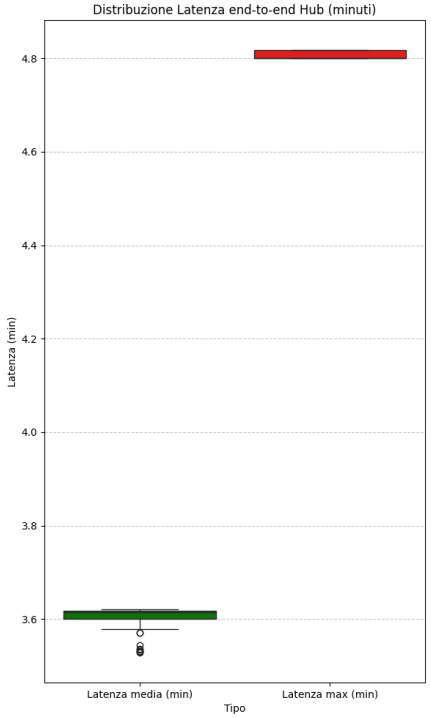


TEST DELLE PERFORMANCE



- Il throughput atteso dal sistema, basato sull'attività sincrona di 200 sensori (2 Macrozone x 2 Zone x 50 Sensori), che inviano dati in tempo reale ogni minuto, è di 200 pacchetti/minuto.

- La latenza end-to-end media osservata si mantiene stabilissima attorno ai 3.6 minuti. Questa latenza è il risultato intenzionale delle diverse fasi di elaborazione e buffering che garantiscono la robustezza e la coerenza temporale dei dati.



CONCLUSIONI E SVILUPPI FUTURI

RISULTATI OTTENUTI

- Dimostrata l'efficacia del paradigma Compute Continuum.
- Realizzato un sistema resiliente e scalabile in Go.
- Ottimizzata la latenza tramite elaborazione distribuita.
- Garantita l'integrità dei dati con pattern architetturali robusti.

SVILUPPI FUTURI

- Introduzione di algoritmi di Machine Learning sull' Edge per la predizione degli outlier.
- Implementazione di notifiche via mail per la gestione dei sensori unhelathy.
- Design di una pipeline *fast lane* per i dati prioritari.

GRAZIE PER L'ATTENZIONE

Distributed application in the compute continuum

https://github.com/F-masci/SensorContinuum

MAURIZIO
RENZI
0294643

FRANCESCO
MASCI
0365258