Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Soluzioni di Database Resilienti per Reti IoT

Antonio Francesco Gentile, Davide Macrì, Emilio Greco

RT-ICAR-CS-25-03 **Gennaio 2025**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 8-9C, 87036 Rende, Italy, URL: www.icar.cnr.it
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: www.icar.cnr.it
– Sezione di Palermo, Via Ugo La Malfa, 153, 90146 Palermo, URL: www.icar.cnr.it

Indice generale

Introduzione

Le reti IoT (Internet of Things) sono caratterizzate dalla generazione continua e massiva di dati temporali, richiedendo soluzioni di database che garantiscano non solo scalabilità, ma anche resilienza e continuità operativa. In contesti di rete così dinamici e distribuiti, è essenziale adottare architetture in grado di assicurare un'elevata disponibilità e una gestione efficiente dei dati. Questo rapporto tecnico esamina tre soluzioni di database resilienti e scalabili per infrastrutture IoT: **InfluxDB**, **MariaDB (Galera Cluster)** e **PostgreSQL (Cluster con Patroni)**.

Ogni tecnologia affronta specifiche esigenze delle reti IoT. **InfluxDB**, progettato come database time-series, è ottimizzato per la gestione e l'archiviazione di dati temporali provenienti da dispositivi IoT, offrendo replica automatica e bilanciamento del carico. **MariaDB con Galera Cluster** è una soluzione relazionale che garantisce consistenza immediata attraverso una replica sincrona, ideale per applicazioni in cui l'integrità dei dati è prioritaria. **PostgreSQL con Patroni**, invece, combina la potenza di un database relazionale con la gestione del failover automatico, rendendolo adatto a scenari in cui è richiesta alta affidabilità e transazioni ACID.

Questo documento si concentra sull'analisi di ciascuna soluzione, descrivendone l'architettura, le caratteristiche principali e le modalità di configurazione per ottenere una resilienza ottimale. Saranno inoltre affrontati aspetti critici come backup, disaster recovery e prevenzione di guasti, evidenziando come queste tecnologie possano essere integrate per soddisfare i requisiti delle reti IoT.

L'obiettivo del rapporto è fornire una guida pratica e comparativa, orientata a progettisti e gestori di infrastrutture IoT, per scegliere e implementare la soluzione più adatta alle proprie esigenze operative e strategiche.

InfluxDB: Database Time Series per Reti IoT

InfluxDB rappresenta una soluzione particolarmente efficace per affrontare le sfide poste dalla gestione di dati in reti IoT. Le reti IoT generano un flusso continuo e massivo di dati temporali, provenienti da dispositivi e sensori distribuiti. Questa caratteristica richiede un'infrastruttura capace di gestire in modo efficiente l'archiviazione, l'interrogazione e la protezione di tali dati, anche in presenza di cambiamenti dinamici della rete. InfluxDB, progettato come database time-series, si adatta perfettamente a queste esigenze, grazie alla sua architettura ottimizzata per la gestione di dati cronologici e al supporto nativo per scalabilità e resilienza.

Uno degli aspetti distintivi di InfluxDB è la sua capacità di garantire continuità operativa in scenari complessi. Grazie alla replica automatica dei dati e al bilanciamento del carico all'interno del cluster, è possibile distribuire il traffico tra i nodi, riducendo il rischio di colli di bottiglia e assicurando che i dati siano sempre accessibili. In contesti IoT, dove le informazioni sono fondamentali per decisioni in tempo reale, questa capacità rappresenta un vantaggio competitivo.

Architettura e Caratteristiche

InfluxDB si basa su un'architettura modulare, che separa i componenti di gestione dei meta-dati, dei dati e del log delle operazioni. Questa separazione consente una gestione più efficiente delle risorse, migliorando le prestazioni in ambienti che richiedono un'elaborazione rapida e affidabile di grandi volumi di dati temporali. L'integrazione di funzionalità come il supporto nativo per query temporali avanzate e la gestione di continuous queries lo rende particolarmente potente per applicazioni IoT che necessitano di analisi dei dati in tempo reale.

La configurazione del file principale (`/etc/influxdb/influxdb.conf`) consente di personalizzare ogni nodo del cluster, ottimizzando la gestione delle risorse e le prestazioni del sistema. Ad esempio, è possibile definire le directory per i dati e i log, configurare le policy di retention per l'archiviazione automatica e specificare impostazioni di rete per l'accesso remoto.

Distribuzione e Resilienza del Cluster

La distribuzione di InfluxDB su un cluster di nodi garantisce scalabilità e tolleranza ai guasti, rispondendo così a due requisiti fondamentali delle reti IoT. Un tipico cluster è composto da almeno tre nodi, con replica automatica dei dati tra di essi. Questa configurazione assicura che, anche in caso di guasto di un nodo, il sistema possa continuare a funzionare senza interruzioni.

Ad esempio, in un'architettura IoT con sensori distribuiti in diverse aree geografiche, i dati possono essere raccolti e distribuiti tra i nodi del cluster, mantenendo un alto livello di disponibilità. Il bilanciamento del carico tra i nodi ottimizza l'elaborazione delle richieste, garantendo tempi di risposta rapidi anche in presenza di carichi elevati.

Backup, Disaster Recovery e Continuità Operativa

InfluxDB supporta strategie di backup avanzate per proteggere i dati e garantire la continuità operativa in caso di guasti o interruzioni. Tramite il comando nativo `influx backup`, è possibile eseguire backup incrementali o completi, mentre il comando complementare `influx restore` consente di ripristinare rapidamente i dati in caso di necessità. Per le reti IoT, dove la perdita di dati può compromettere processi critici, questa funzionalità è essenziale.

Oltre ai backup, la replica automatica dei dati su più nodi all'interno del cluster offre un ulteriore livello di protezione. Questa funzionalità riduce significativamente il rischio di perdita di dati e garantisce che le applicazioni IoT possano continuare a funzionare anche in scenari di emergenza.

InfluxDB si inserisce perfettamente nelle architetture di rete IoT descritte nell'introduzione, rispondendo in modo efficace alla necessità di scalabilità, resilienza e gestione continua dei dati. La sua capacità di gestire flussi di dati temporali in tempo reale, unita a funzionalità avanzate di clustering e backup, lo rende ideale per applicazioni IoT in cui l'affidabilità e le prestazioni sono fondamentali. Con un'architettura distribuita, una gestione efficiente delle risorse e strumenti per il disaster recovery, InfluxDB rappresenta una soluzione chiave per realizzare infrastrutture IoT moderne e resilienti.

Un esempio di configurazione di un nodo include:

```
[meta]
  dir = "/var/lib/influxdb/meta"
  retention-autocreate = true
  logging-enabled = true

[data]
  dir = "/var/lib/influxdb/data"
  wal-dir = "/var/lib/influxdb/wal"
  query-log-enabled = true

[cluster]
  shard-writer-timeout = "5s"
  max-concurrent-queries = 0
  max-points-per-write = 5000
```

```
[http]
  enabled = true
  bind-address = ":8086"
  flux-enabled = true

[continuous_queries]
  enabled = true
  recompute-previous-n = 2
```

Questo file definisce i parametri critici per il funzionamento del database. La sezione `[meta]` gestisce i meta-dati, inclusa la creazione automatica delle policy di retention, mentre la sezione `[data]` specifica i percorsi per i dati e il log del write-ahead (WAL), fondamentale per garantire l'integrità dei dati durante le scritture. La sezione `[cluster]` configura le impostazioni di timeout e i limiti per le operazioni in un ambiente distribuito, migliorando le prestazioni in reti IoT.

Ad esempio, in una rete IoT con sensori distribuiti in diverse regioni, un cluster InfluxDB può archiviare i dati generati da ciascun gruppo di dispositivi in shard separati, replicandoli tra i nodi per garantire l'affidabilità. Ogni nodo comunica con gli altri attraverso un protocollo interno che sincronizza le informazioni e bilancia il carico.

MariaDB con Galera Cluster: Database Relazionale con Replica Sincrona

MariaDB con Galera Cluster rappresenta una soluzione avanzata per la gestione di database relazionali, progettata per garantire consistenza, resilienza e alta disponibilità. Questa tecnologia è particolarmente adatta per applicazioni IoT che richiedono coerenza immediata dei dati tra più nodi e capacità di gestione simultanea delle operazioni di lettura e scrittura. Galera Cluster implementa un meccanismo di replica sincrona, assicurando che ogni nodo del cluster sia sempre aggiornato, indipendentemente da quale nodo gestisca una determinata richiesta.

L’architettura di Galera Cluster è pensata per fornire un sistema di database distribuito, in cui tutti i nodi possono agire come punti di ingresso per le operazioni. Questo approccio elimina la necessità di un nodo primario, migliorando la scalabilità e riducendo i colli di bottiglia. La replica sincrona assicura che i dati siano immediatamente disponibili su tutti i nodi, garantendo coerenza e resilienza anche in caso di guasti. Queste caratteristiche rendono MariaDB con Galera Cluster ideale per ambienti IoT, dove l'integrità dei dati è essenziale per il funzionamento continuo delle applicazioni.

Architettura e Caratteristiche

La replica sincrona è il cuore del funzionamento di Galera Cluster. Quando un nodo riceve una richiesta di scrittura, la modifica viene propagata agli altri nodi prima di essere confermata. Questo processo assicura che non vi siano discrepanze tra i dati memorizzati nei diversi nodi. Inoltre, ogni nodo è in grado di gestire sia le richieste di lettura che di scrittura, aumentando la capacità del cluster di rispondere a carichi elevati.

Per prevenire situazioni di split-brain, in cui i nodi possono divergere in assenza di un consenso chiaro, Galera Cluster utilizza un sistema di quorum. La maggioranza dei nodi deve essere attiva e comunicare tra loro affinché le operazioni di scrittura siano accettate. Questo meccanismo garantisce che il cluster rimanga consistente anche in caso di perdita temporanea di connessione tra i nodi.

Configurazione di Galera Cluster

La configurazione di un cluster MariaDB con Galera inizia con la personalizzazione del file di configurazione principale, solitamente situato in `/etc/mysql/my.cnf`. Questo file contiene tutte le impostazioni necessarie per abilitare e gestire il cluster.

Un esempio di configurazione per il nodo 1 è il seguente:

```
[mysqld]
binlog_format=ROW
default-storage-engine=innodb
innodb_autoinc_lock_mode=2
bind-address=0.0.0.0

[galera]
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so
wsrep_cluster_name="galera_cluster"
wsrep_cluster_address="gcomm://10.0.0.1,10.0.0.2,10.0.0.3"
wsrep_node_address="10.0.0.1"
wsrep_node_name="node1"
wsrep_sst_method=rsync
wsrep_sst_auth=user:password
```

Descrizione della configurazione:

- **[mysqld]**: Questa sezione configura il motore di archiviazione (InnoDB), il formato del log binario e altre impostazioni generali richieste per abilitare la replica.
- **[galera]**: Qui vengono definite le impostazioni specifiche per Galera Cluster, inclusi il nome del cluster (wsrep_cluster_name), gli indirizzi IP dei nodi (wsrep_cluster_address) e il metodo di trasferimento dello stato (wsrep_sst_method), utilizzato per sincronizzare nuovi nodi con il cluster.

Ogni nodo del cluster avrà una configurazione simile, con modifiche a **`wsrep_node_address`** e **`wsrep_node_name`** per identificare correttamente ciascun nodo.

Distribuzione del Cluster MariaDB

Un cluster Galera è solitamente composto da almeno tre nodi, che collaborano per fornire un sistema altamente disponibile e tollerante ai guasti. Una distribuzione tipica potrebbe essere rappresentata come segue:

In questo schema, ogni nodo è connesso agli altri, e le richieste di lettura e scrittura possono essere inviate a qualsiasi nodo. La replica sincrona garantisce che ogni modifica venga propagata immediatamente agli altri nodi.

Backup e Disaster Recovery

Per proteggere i dati e garantire la continuità operativa, MariaDB con Galera Cluster supporta strumenti avanzati di backup, tra cui **MariaBackup**. Questo strumento consente di eseguire backup completi o incrementali senza interrompere le operazioni del cluster. I backup possono essere utilizzati per ripristinare nodi danneggiati o per ripristinare l'intero cluster in caso di guasto critico.

Un esempio di comando per creare un backup è:

```
mariabackup --backup --target-dir=/path/to/backup --user=user --password=password
```

Il ripristino di un nodo può essere effettuato utilizzando il comando complementare:

```
mariabackup --prepare --target-dir=/path/to/backup
```

L’integrazione di backup regolari con una strategia di disaster recovery garantisce che il cluster possa essere rapidamente ripristinato in caso di emergenza.

MariaDB con Galera Cluster si distingue come una soluzione affidabile per la gestione di database relazionali in ambienti IoT, dove l’integrità dei dati e la continuità operativa sono fondamentali. La replica sincrona, il supporto per la gestione simultanea di letture e scritture e il sistema di quorum per prevenire lo split-brain rendono questa tecnologia particolarmente adatta per reti IoT complesse. La capacità di eseguire backup e ripristino senza interruzioni, unita alla resilienza intrinseca del cluster, assicura che i dati siano sempre disponibili e protetti.

PostgreSQL con Patroni Cluster: Database Relazionale con Failover Automatico

PostgreSQL, uno dei database relazionali open-source più potenti e versatili, si distingue per il supporto nativo alle transazioni ACID, che garantiscono affidabilità e coerenza dei dati. Quando configurato con **Patroni**, PostgreSQL può essere implementato in un cluster con funzionalità avanzate come il failover automatico e la replica sincrona. Questo rende la soluzione ideale per applicazioni IoT in cui è necessario garantire alta disponibilità, resilienza e protezione dei dati.

Il sistema Patroni utilizza **etcd**, un servizio distribuito per la gestione della configurazione e del coordinamento, per monitorare lo stato del cluster e coordinare le operazioni tra i nodi. Questa combinazione garantisce che in caso di guasto del nodo primario (master), uno dei nodi standby venga promosso automaticamente, assicurando la continuità del servizio senza intervento manuale.

Architettura e Caratteristiche

La configurazione di PostgreSQL con Patroni segue un'architettura master-slave, con un nodo primario che gestisce tutte le operazioni di scrittura e nodi standby che replicano i dati in tempo reale. La replica sincrona garantisce che le modifiche vengano propagate immediatamente ai nodi standby, minimizzando la possibilità di perdita di dati.

Il ruolo di Patroni è quello di monitorare costantemente lo stato del nodo master e dei nodi standby. Utilizzando etcd come strumento di coordinamento, Patroni garantisce che, in caso di guasto del nodo master, uno dei nodi standby venga promosso automaticamente al ruolo di master. Questo meccanismo di failover automatico è fondamentale per le applicazioni IoT, dove l'interruzione del servizio potrebbe avere gravi conseguenze.

Configurazione di Patroni Cluster

La configurazione del cluster Patroni si basa su un file YAML che specifica i parametri per il coordinamento tra i nodi, la replica e la gestione del database. Un esempio di file di configurazione per un nodo è il seguente:

```
scope: postgres_cluster
name: node1

etcd:
  host: 10.0.0.1:2379,10.0.0.2:2379,10.0.0.3:2379

restapi:
  listen: 10.0.0.1:8008
  connect_address: 10.0.0.1:8008

postgresql:
  listen: 10.0.0.1:5432
  connect_address: 10.0.0.1:5432
  data_dir: /var/lib/postgresql/12/main
  replication:
    username: replica_user
    password: replica_password
  superuser:
    username: postgres
    password: postgres_password
  parameters:
    max_wal_senders: 10
    wal_keep_segments: 8
```

```
    synchronous_commit: "on"

bootstrap:
  dcs:
    postgresql:
      use_pg_rewind: true
  initdb:
  - encoding: UTF8
  - data-checksums
```

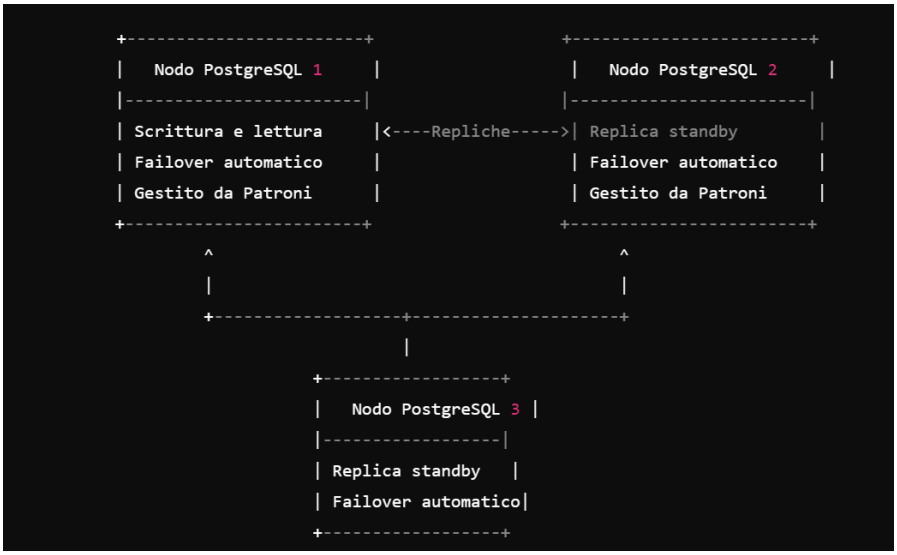
Descrizione della configurazione:

- **scope e name**: Definiscono il cluster e il nome del nodo.
- **etcd**: Specifica gli indirizzi dei nodi etcd utilizzati per il coordinamento.
- **restapi**: Configura l'interfaccia API REST per monitorare e gestire il nodo.
- **postgresql**: Specifica i parametri relativi al database, inclusi gli indirizzi di connessione, la directory dei dati, le credenziali per la replica e le impostazioni per la replica sincrona.
- **bootstrap**: Contiene le configurazioni iniziali, come la codifica dei dati e l'abilitazione dei checksums.

Questa configurazione è replicata sugli altri nodi del cluster, con variazioni nei parametri **`name`**, **`listen`** e **`connect_address`** per riflettere le differenze tra i nodi.

Distribuzione del Cluster PostgreSQL

Un cluster PostgreSQL configurato con Patroni può essere visualizzato come segue:



In questa configurazione, etcd coordina i nodi e monitora il loro stato. Patroni garantisce che in caso di guasto del nodo master (Node 1), uno dei nodi standby (Node 2 o Node 3) venga promosso a master.

Backup e Disaster Recovery

La protezione dei dati in un cluster PostgreSQL con Patroni è gestita tramite **pgBackRest**, uno strumento avanzato per il backup e il ripristino. pgBackRest supporta backup incrementali, che riducono il tempo e lo spazio necessari per le operazioni di backup, e backup point-in-time (PITR), che consentono di ripristinare il database a uno stato specifico.

Un esempio di comando per eseguire un backup completo:

```
pgbackrest --stanza=mydb backup
```

E un esempio per il ripristino point-in-time:

```
pgbackrest --stanza=mydb restore --type=time --target="2025-01-01 12:00:00"
```

Queste funzionalità garantiscono che, in caso di guasto o perdita di dati, il sistema possa essere ripristinato rapidamente, minimizzando i tempi di inattività.

PostgreSQL con Patroni Cluster si distingue per la sua capacità di combinare la potenza di un database relazionale con funzionalità avanzate di failover e replica. Questo lo rende ideale per applicazioni IoT che richiedono alta affidabilità, consistenza dei dati e resilienza a guasti. Grazie a Patroni e etcd, il cluster può rispondere rapidamente ai guasti, mantenendo la continuità operativa senza intervento manuale. L'integrazione con strumenti come pgBackRest per backup e ripristino rafforza ulteriormente la sicurezza dei dati, rendendo questa soluzione adatta a contesti in cui la protezione e la disponibilità delle informazioni sono fondamentali.

Conclusioni

Nell'ambito delle reti IoT, la resilienza, la coerenza dei dati e la capacità di recupero rapido da guasti rappresentano requisiti fondamentali per garantire la continuità operativa e la sicurezza delle applicazioni. L'architettura distribuita e dinamica tipica di queste reti richiede soluzioni di database avanzate, in grado di affrontare carichi elevati e di adattarsi rapidamente a condizioni mutevoli. Questo rapporto ha analizzato tre tecnologie chiave che rispondono a tali esigenze, ciascuna con caratteristiche e punti di forza distinti.

InfluxDB, progettato come database time-series, è particolarmente adatto alla gestione di dati temporali generati in modo massivo dai dispositivi IoT. La sua architettura scalabile e la capacità di bilanciamento automatico del carico lo rendono ideale per applicazioni che richiedono l'archiviazione e l'analisi in tempo reale di grandi volumi di dati cronologici. La resilienza è assicurata dalla replica dei dati e dal supporto per backup completi e incrementali, che garantiscono una gestione sicura e affidabile anche in scenari complessi.

MariaDB Galera Cluster si distingue per la sua capacità di fornire una consistenza immediata attraverso la replica sincrona. Questo approccio garantisce che i dati siano sempre coerenti tra i nodi del cluster, eliminando la necessità di operazioni di riconciliazione post-scrittura. La gestione del quorum previene problemi di split-brain, assicurando che il cluster rimanga operativo solo quando la maggioranza dei nodi è attiva. Queste caratteristiche rendono MariaDB Galera Cluster una scelta eccellente per applicazioni IoT in cui la coerenza dei dati è essenziale.

PostgreSQL con Patroni offre una combinazione unica di affidabilità e flessibilità, con funzionalità avanzate come il failover automatico e la replica sincrona. Patroni utilizza etcd per il coordinamento

e il monitoraggio del cluster, garantendo che i guasti dei nodi vengano gestiti senza interruzioni manuali. Inoltre, l'integrazione con strumenti come pgBackRest per backup e ripristino consente di proteggere i dati e di ripristinare il sistema rapidamente, minimizzando i tempi di inattività. Questo lo rende ideale per applicazioni IoT che richiedono alta affidabilità e conformità alle transazioni ACID.

Ogni soluzione analizzata in questo rapporto può essere adattata per soddisfare le esigenze specifiche di un'infrastruttura IoT, a seconda della natura dei dati gestiti e dei requisiti operativi. InfluxDB eccelle nella gestione di dati temporali e scalabilità, MariaDB Galera Cluster si focalizza sulla consistenza immediata e sulla resilienza del cluster, mentre PostgreSQL con Patroni offre una gestione avanzata del failover e una protezione completa dei dati.

In conclusione, la scelta della soluzione più adatta dipende dalle priorità specifiche del sistema IoT in questione. Combinando queste tecnologie, è possibile costruire un'infrastruttura distribuita, scalabile e resiliente, capace di rispondere efficacemente alle sfide poste dalle moderne reti IoT. Questo rapporto fornisce una guida pratica per orientarsi tra le diverse opzioni, aiutando progettisti e gestori di infrastrutture IoT a prendere decisioni informate per garantire la massima disponibilità e protezione dei dati.

Bibliografia

1. Influxdb uffial website. **How InfluxDB Works with IoT Data**. Disponibile online: https://www.influxdata.com/blog/how-influxdb-iot-data/, ultima consultazione il 11 Gennaio 2025.
2. MariaDB uffial website. **MariaDB Galera Cluster**. Disponibile online: https://mariadb.com/kb/en/galera-cluster/, ultima consultazione il 11 Gennaio 2025.
3. **Galera Cluster.** Disponibile online: https://galeracluster.com/library/documentation/ , ultima consultazione il 11 Gennaio 2025.
4. **Patroni**. Disponibile online: https://patroni.readthedocs.io/en/latest/, ultima consultazione il 11 Gennaio 2025.
5. **PostgreSQL**. Disponibile online: https://www.postgresql.org/docs/, ultima consultazione il 11 Gennaio 2025.