Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea

IMPLEMENTAZIONE E TESTING DI UN CLUSTER DI DATABASE SU UNA RETE DI PARI

TITOLO INGLESE

LINDA LUCIANO

Relatore: *Lorenzo Bettini*
Correlatore: *Correlatore*

Anno Accademico 2016-2017

INDICE

ELENCO DELLE FIGURE

"Inserire citazione"
— Inserire autore citazione

1

DEFINIZIONE DEL PROBLEMA

(replicazione dei dati - introduzione)

Cluster di database

Rete di pari

Sistemi di ridondanza disco (RAID)

Codice di correzione errore (erasure coding)

HARDWARE UTILIZZATO PER GLI ESPERIMENTI

SOFTWARE UTILIZZATO PER GLI ESPERIMENTI

PG Logical

PG Logical è un sistema logico di replica implementato come estensione di PostgreSQL. Completamente integrato, non richiede alcun triggers o programmi esterni. Questa alternativa alla replica fisica è un metodo altamente efficiente per replicare i dati utilizzando un modello di publish/subscribe per la replica selettiva.

I vantaggi offerti da Pg Logical sono i seguenti:

- Replica sincrona
- Replica ritardata
- Risoluzione dei conflitti configurabili
- Capacità di convertire lo standby fisico in una replica logica
- Può pubblicare i dati da PostgreSQL a un abbonato Postgres-XL

- Le sequenze possono essere replicate
- Nessun trigger significa ridurre il carico di scrittura sul Provider
- Nessuna re-esecuzione di SQL significa overhead e latenza ridotti per il Sottoscrittore
- Il sottoscrittore non è in ripristino di riposo caldo, in modo da poter utilizzare tavoli temp, non sbloccati o normali
- Non è necessario annullare le query per consentire alla replica di continuare la riproduzione
- Il sottoscrittore (Subscriber) può avere diversi utenti e protezione, indici diversi, impostazioni di parametri diversi
- Replica solo un database o un sottoinsieme di tabelle, noto come set di replica (Replication Sets)
- Replicare in versioni o architetture di PostgreSQL, consentendo aggiornamenti a bassa o zero-downtime
- Più server a monte in un singolo subscriber per l'accumulo di cambiamenti

Casi di uso: I diagrammi che seguono descrivono i gestori di database delle funzioni che sono in grado di eseguire con PgLogical.

Migrare e aggiornare PostgreSQL con tempi di inattività quasi a zero

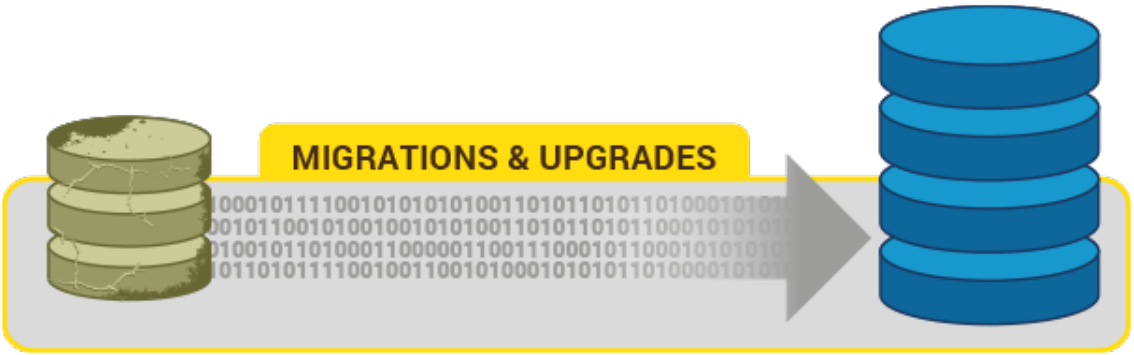

Figura 1: Migrazione e aggiornamenti PostgreSQL [1]

Accumulare le
modifiche provenienti da server di database scartati in un data warehouse

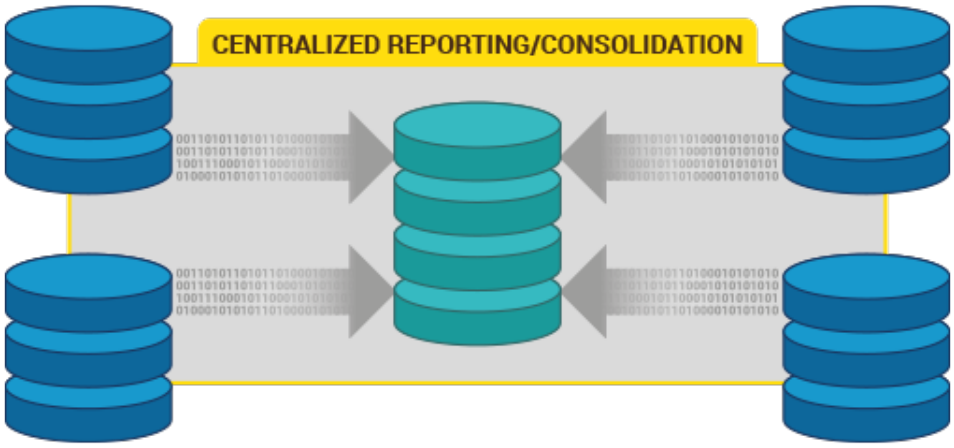

Figura 2: Aggregazione [2]

Copiare tutti o una selezione di tabelle di database ad altri nodi di un cluster

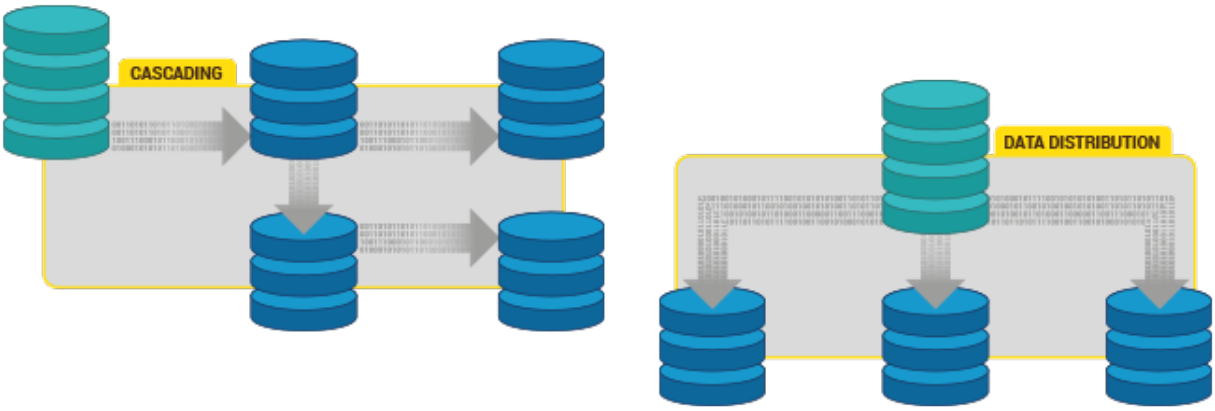

Figura 3: A cascata e distribuzione dati [2]

Le modifiche del database in tempo reale ad altri sistemi

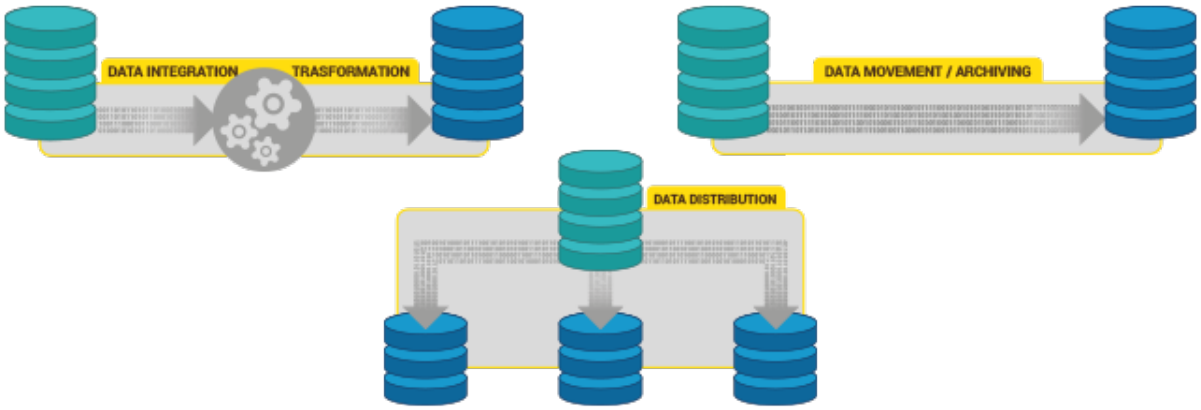

Figura 4: A cascata e distribuzione dati [2]

2

DEFINIZIONE DEL PROGETTO

Simulazione di un filesystem distribuito (Dati e Metadati)

CONSIDERAZIONI STATISTICHE SULLA RIDONDANZA SUL DATO

CONSIDERAZIONI STATISTICHE SULLA RIDONDANZA DEL METADATO

RESILIENZA AI CAMBIAMENTI DI RETE

3

DEFINIZIONE DEL QUADRO SPERIMENTALE

LANCIO IN CONFIGURAZIONE 1

LANCIO IN CONFIGURAZIONE 2

LANCIO IN CONFIGURAZIONE 3

4

CONCLUSIONI E POSSIBILI EVOLUZIONI

UTILIZZO DI DISCHI SSD

UTILIZZO DI PROCESSORI DUAL CORE

5

ESERCIZI

1. Scrivere le possibili evoluzioni del programma

   ```
   co X: = X+2 // X: = X+1 oc
   ```

   assumendo che ciascun assegnamento è realizzato da tre azioni atomiche che caricano X in un registro (`Load R X`), incrementano il valore del registro (`Add R v`) e memorizzano il valore del registro((`Store R X`). Per ciascuna delle esecuzioni risultanti dallÕinterleaving delle azioni atomiche descrivere il contenuto dopo ogni passo della locazione condivisa X e dei registri privati, R_1 del processo che esegue il primo assegnamento ed R_2 per il processo che esegue il secondo assegnamento. Se assuma che il valore iniziale di X sia 50.

2. Si definisca il problema della *barrier synchronization* e si descrivano per sommi capi i differenti approcci alla sua soluzione. Se ne fornisca quindi una soluzione dettagliata utilizzando i semafori.

3. Considerare n api ed un orso che possono avere accesso ad una tazza di miele inizialmente vuota e con una capacità di k porzioni. L'orso dorme finchè la tazza è piena di k-porzioni, quindi mangia tutto il miele e si rimette a dormire. Le api riforniscono in continuazione la tazza con una porzione di miele finchè non si riempie; l'ape che aggiunge la k-esima porzione sveglia l'orso. Fornire una soluzione al problema modellando orso ed api come processi e utilizzando un monitor per gestire le loro operazioni sulla tazza. Prevedere che le api possano eseguire l'operazione *produce-honey* anche concorrentemente.

4. Descrivere le primitive di scambio messaggi send e receive sia sincrone che asincrone ed implementare

```
- synch_send(v:int)
- send(v:int)
- receive(x:int)
```

utilizzando le primitive di LINDA.

BIBLIOGRAFIA

[1] 2ndQuadrant Ltd - *pglogical* (Cited on pages 3 and 9.)

[2] Autore - *Titolo* - altre informazioni (Cited on pages 3, 9, and 10.)