



UNIVERSITÀ
DEGLI STUDI
FIRENZE

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*

book

DEFINIZIONE DEL PROBLEMA

(replicazione dei dati - introduzione)

Cluster di database

Un cluster è una raccolta di componenti che garantisce scalabilità e disponibilità distribuendone i costi. Un cluster di database (SQL usa il termine cluster di catalogo) è una collezione di database gestiti da una singola istanza di un server database in esecuzione. Un'istanza è la raccolta di memoria e processi che interagiscono con un database, cioè l'insieme di file fisici che effettivamente memorizzano i dati. A tal fine, è possibile creare un cluster di database per applicazioni enterprise high-end, memorizzando e elaborando informazioni sui nodi. L'architettura per un cluster di database è distinta da come le responsabilità dei dati sono condivise tra i nodi di calcolo.

Il clustering offre due vantaggi principali, specialmente in un ambiente di database di alto volume:

- Fault tolerance (tolleranza di guasti): poiché esiste più di un server o istanza per gli utenti a cui connettersi, il cluster offre un'alternativa, in caso di guasto del singolo server.
- Load balancing (bilanciamento del carico): la funzionalità di clustering è generalmente impostata per consentire agli utenti di essere assegnati automaticamente al server con il minor carico.

In un'architettura shared-nothing, il partizionamento è tale che ogni nodo di calcolo possiede un sottoinsieme dei dati. Ovvero, un nodo opererà esclusivamente su un particolare sottoinsieme. La scalabilità dipende ovviamente da una giudiziosa partizione. Il partizionamento fisico può essere che un nodo di calcolo ponga la sua porzione di dati sul proprio disco locale e che che assomiglia a un array di storage — riduce la tolleranza agli errori come un crash di nodo

Rete di pari

Sistemi di ridondanza disco (RAID)

Codice di correzione errore (erasure coding)

HARDWARE UTILIZZATO PER GLI ESPERIMENTI

SOFTWARE UTILIZZATO PER GLI ESPERIMENTI

PostgreSQL

(spiegazione cinci -audio-) Uno dei punti di forza che ha reso famoso e versatile Posgres Ã che Ã un DBMS (Database Management System Ã un sistema software progettato per consentire la creazione e manipolazione efficiente di database (ovvero di collezioni di dati strutturati)) (scritto in C). Ha dei rametti dove te riesci ad agganciarci dei software fatti ad OC. te ti agganci a un sistema di segnalistica interna e lui quindi tramite messaggi di scambio interno ti permette di lavorare (ti dice "ho fatto l'inserimento di questa tabella(INSERT), ti scatta il segnale, e te riesci a intercettare e quindi fare delle operazioni)

Pg Logical Ã un estensione di Posgres. Lui capisce quando fa inserimenti e riesce a duplicare soltando parti di database. Mentre replica di posgres per adesso ti replicherebbe tutto il DB. Invece noi si fa con solo tabelle (R) o quanti record di tabelle (per id maggiori di x) adirittura si vogliono replicare o colonne di tabelle

(spiegazione cinci -audio-)

Con il termine PostgreSQL viene denominato un particolare ORDBMS, acronimo di Object-Relational DataBase Management System, cioÃ un software relazionale e ad oggetti per la gestione di basi di dati. PostgreSQL Ã Open Source, quindi il suo codice sorgente Ã disponibile pubblicamente ed "aperto" ai contributi degli sviluppatori che volontariamente intendono partecipare alla sua implementazione.

In PostgreSQL, ma il discorso puÃ²essere allargato a tutti i gestori di database relazionali, id

L'SQL Ã quindi un linguaggio a disposizione dell'utente da impiegare per "interfacciarsi" ai gestori di database; esso ha la caratteristica di agire in modo "trasparente", infatti le operazioni svolte a carico dei dati saranno portate avanti senza bisogno che l'utilizzatore sappia in che modo essi vengono trattati.

(spiegazione cinci -audio-

(<http://www.slony.info/images/Slony-I-concept.pdf>)

Replicating schema changes is an often discussed problem and only very few database systems provide the necessary hooks to implement it. PostgreSQL does not provide the ability to define triggers called on schema changes, so a transparent way to replicate schema changes is not possible without substantial work in the core PostgreSQL system. Moreover, very often database schema changes are not single, isolated DDL statements that can occur at any time within a running system. Instead they tend to be groups of DDL and DML statements that modify multiple database objects and do mass data manipulation like updating a new column to its initial value. The Slony-I replication system will have a mechanism to execute SQL scripts in a controlled fashion as part of the replication process.

Concept Provider-Subscriber

Master to multiple cascaded slaves The basic structure of the systems combined in a Slony-I installation is a master with one or more slave nodes. Not all slave nodes must receive the replication data directly from the master. Every node that receives the data from a valid source can be configured to be able to forward that data to other nodes. There are three distinct ideas behind this capability. The first is scalability. One database, especially the master that receives all the update transactions from the client applications, has only a limited capability to satisfy the slave nodes queries during the replication process. In order to satisfy the need for a big number of read-only slave systems it must be possible to cascade. The second idea is to limit the required network bandwidth for a backup site while keeping the ability to have multiple slaves at the remote location. The third idea is to be able to configure failover scenarios. In a master to multiple slave configuration, it is unlikely that all slave nodes are exactly in the same synchronization status when the master fails. To ensure that one slave can be promoted to the master it is necessary that all remaining systems can agree on the status of the data. Since a committed transaction cannot be rolled back, this status is undoubtedly the most recent sync status of all remaining slave nodes. The delta between this one and every other node must be easily and fast generated and applied at least to the new master (if that's not the same system) before the promotion can occur. **Nodes, Sets and forwarding** The Slony-I replication system can replicate tables and sequence numbers. Replicating sequence numbers is not unproblematic and is discussed in more detail in section

2.3. Table and sequence objects are logically grouped into sets. Every set should contain a group of objects that is independant from other objects originating from the same master. In short, all tables that have relationships that could be expressed as foreign key constraints and all the sequences used to generate any serial numbers in these tables should be contained in one and the same set. Figure 1 illustrates a replication configuration that has 2 data sets with different origins. To replicate both data sets to NodeC it is not required that Node C really communicates with the origin of Set 1. This scenario has full redundancy for every node. Obviously if Node C fails, the masters of Set 1 and Set2 are still alive, no problem. If Node A fails, Node B can get promoted to the master of both sets. The tricky situation is if Node B fails. In the case Node B fails, Node C needs to get promoted to the master of Set 2 and it must continue replicating Set 1 from Node A. For that to be possible, Node A must have knowledge about Node C and its subscription to Set 1. Generally speaking, every node that stores replication log information must keep it until all subscribers of the affected set are known to have replicated that data. To simplify the logic, the configuration of the whole network with all nodes, sets and subscriptions will be forwarded to and stored on all nodes. Because the sets, a node is not subscribed to must not even exist in its database, this does not include the information about what tables and sequences are included in any specific set.

Pg Logical

L'estensione pglogical fornisce la replica logica di streaming per PostgreSQL, utilizzando un modulo di pubblicazione / sottoscrizione. Si basa sulla tecnologia sviluppata come parte del Progetto BDR.

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

Come funziona pglogical? Pglogical utilizza le funzioni di Decodifica Logica aggiunte da 2ndQuadrant (e disponibili da PostgreSQL 9.4). Pglogical funziona ancora più veloce con PostgreSQL 9.5 successive, con basso overhead su

Pglogical si basa molto sulle caratteristiche introdotte nell'ambito dello sviluppo BDR, tra cui:

Decodifica logica Slot di replica Lavoratori di sfondo statico Origini di replica Impegnano timestamp Messaggi WAL logici Replica pglogical Bi-Directional Replication (BDR)? No. pglogical non fornisce funzionalità complete di replica multi-master e un supporto di modifica dello schema coerente, come fa la BDR. Pglogical incorpora le funzionalità di BDR e le lezioni apprese da BDR per produrre una soluzione più semplice e più semplice da utilizzare per la replica unidirezionale, utilizzata da

Lo sviluppo di BDR continuerà per quelli che richiedono piena capacità multi-master, riutilizzando gran parte del codice da pglogical.

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

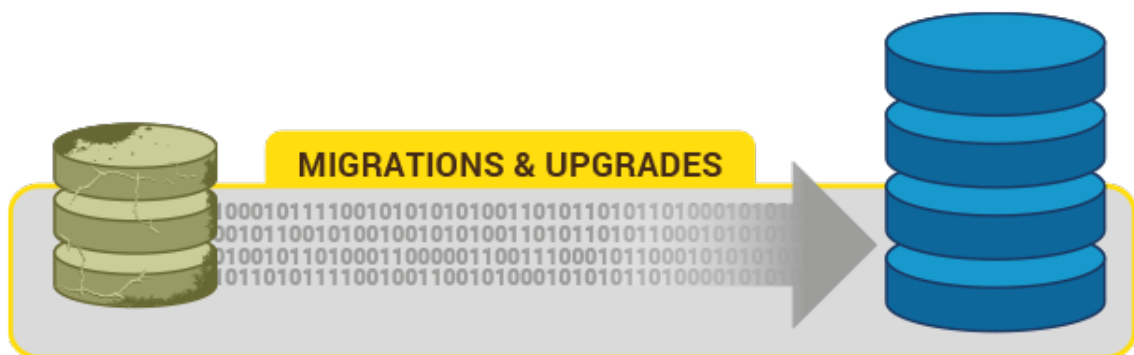


Figura 1: Migrazione e aggiornamenti PostgreSQL [?]

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

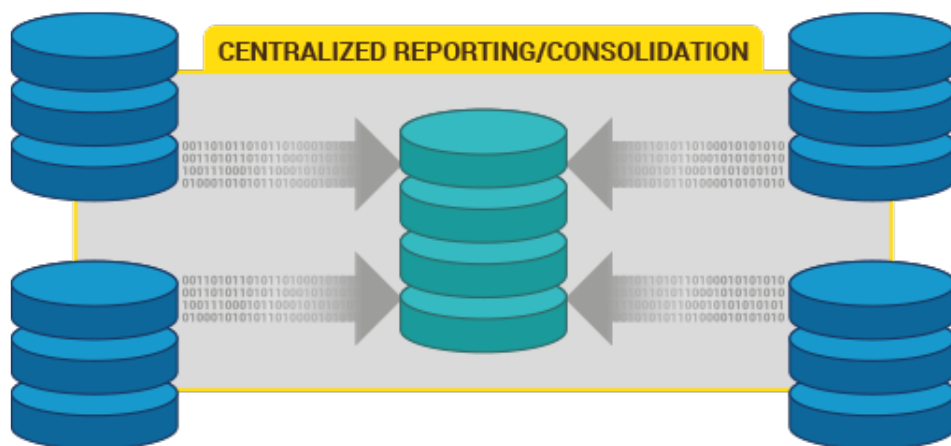


Figura 2: Aggregazione [?]

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

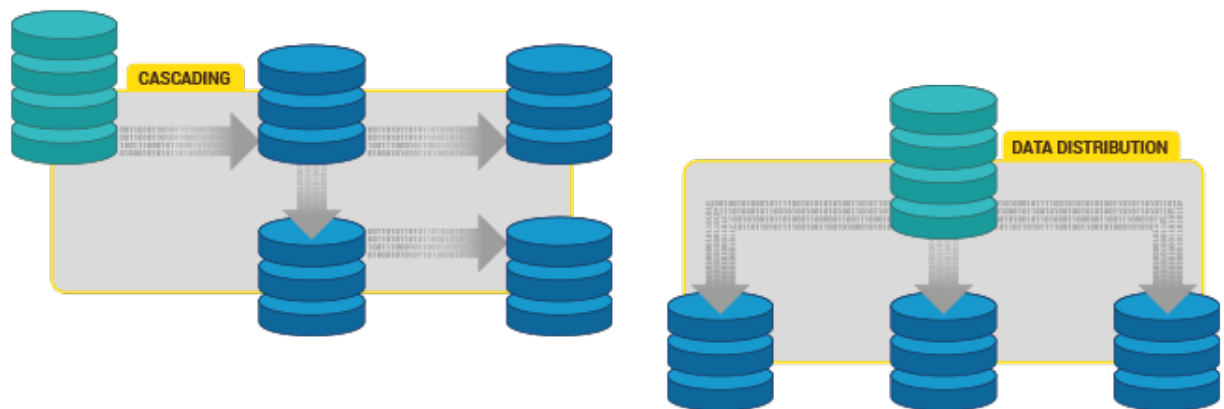


Figura 3: A cascata e distribuzione dati [?]
Le modifiche del database in tempo reale ad altri sistemi

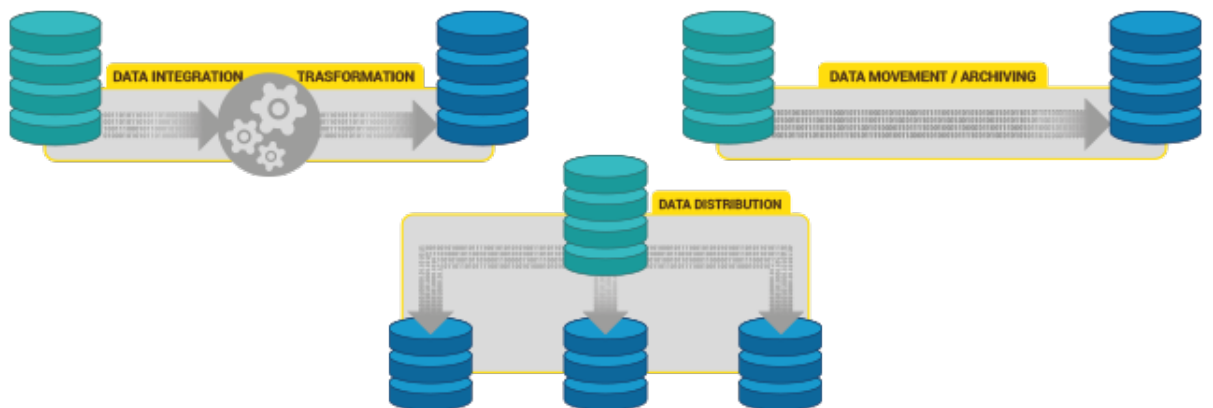


Figura 4: A cascata e distribuzione dati [?]

DA METTERE? DIFFERENZA TRA PG LOGICAL E BDR Replica pglogical Bi-Directional Replication (BDR)? No. pglogical non fornisce funzionalità complete di replica multi-master e un supporto di modifica dello schema coerente, come fa la BDR. Pglogical incorpora le funzionalità di BDR e le lezioni apprese da BDR per produrre una soluzione più semplice e più semplice da utilizzare per la replica unidirezionale, utilizzabile da più per.