Chapter 1

Definizione del problema

(replicazione dei dati - introduzione)

1.0.1 Cluster di database

Un cluster è una raccolta di componenti che garantisce scalabità 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, cio 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 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 pu essere che un nodo di calcolo pone la sua porzione di dati sul proprio disco locale e che tutti i nodi siano collegati tramite una rete ad alta velocità. Tale schema - che assomiglia a una rete di storage - riduce la tolleranza agli errori come un crash di nodi rende inaccessibile una parte del database. Pertanto, la partizione fisica pu

effettivamente condividere i dischi in modo che un nodo possa "assumere" se un altro nodo si blocca. Questa architettura di gran lunga la pi popolare; IBM DB2, Microsoft SQL Server, Cluster MySQL e MPP di Bizgres (basati su PostgreSQL) seguono questo schema. L'aspetto negativo di questa configurazione è la difficoltà di installazione e manutenzione. In un'architettura shared-everything (o shared-disk), qualsiasi nodo di calcolo pu operare in qualsiasi parte del database. Il layout fisico di solito prevede la memorizzazione in rete, in cui tutti i nodi comunicano con un disco condiviso separato (spesso un RAID) tramite un'interconnessione ad alta velocità (tradizionalmente Fibre Channel). Ancora una volta, la partizione logica è la chiave per la scalabilità. L'ovvio difetto di un'architettura condivisa è che ci può essere un'enorme contesa quando numerosi nodi tentano di comunicare contemporaneamente con il disco. Come tale, l'unico importante fornitore di database che offre un'architettura condivisa in tutto è Oracle, il cui prodotto RAC (Real Application Cluster) costruisce una cache condivisa su ciascuno dei nodi di calcolo tramite "Cache Fusion". Questa tecnologia utilizza la rete ad alta velocità per mantenere la coerenza della cache. Entrambe le architetture possono richiedere tre diverse interconnessioni: Ethernet per la gestione, InfiniBand o Myrinet per la comunicazione internode e Fibre Channel per la comunicazione in blocchi. Data l'andamento della commodificazione, è possibile che iWARP o Myri-10G emergano come singola interconnessione per un database cluster.

1.0.2 Rete di pari

1.0.3 Sistemi di ridondanza disco (RAID)

1.0.4 Codice di correzione errore (erasure coding)

1.1 Hardware utilizzato per gli esperimenti

1.2 Software utilizzato per gli esperimenti

1.2.1 PosgreSQL

(spiegazione cinci -audio-) Uno dei punti di forza che ha reso famoso e versatile Posgres che un DBMS (Database Management System 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 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, cio un software relazionale e ad oggetti per la gestione di basi di dati. PostgreSQL Open Source, quindi il suo codice sorgente disponibile pubblicamente ed "aperto" ai contributi degli sviluppatori che volontariamente intendono partecipare alla sua implementazione.

In PostgreSQL, ma il discorso pu essere allargato a tutti i gestori di database relazionali, i dati vengono rappresentati attraverso delle tabelle e le tabelle sono gestite utilizzando un linguaggio di alto livello chiamato SQL, acronimo di Structured Query Language. Nel caso specifico, l'ORDBMS avr il compito di gestire le operazioni di archiviazione e di salvaguardia dell'integrit dei dati allocati.

L'SQL 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 ver y fe w 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, ver y often database schema chages 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 scr ipts in a controlled fashion as part of the replication process.

1.2.2 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 slaves 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 networ k 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 scenar ios. 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 undoubtly 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 thats 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 shor t, all tables that have relationships that could be expressed as foreign key constraints and all the sequences used to generate any ser ial 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 date 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 infor mation 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 networ k with all nodes, sets and subscriptions will be forwarded to and stored on all nodes. Because the sets,anode is not subscribed to must not even exist in its database, this does not include the infor mation about what tables and sequences are included in any specific set.

1.2.3 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 pi veloce con PostgreSQL 9.5 e successive, con bassi overhead su entrambi i provider e abbonati.

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 funzionalit complete di replica multi-master e un supporto di modifica dello schema coerente, come fa la BDR. Pglogical incorpora le funzionalit di BDR e le lezioni apprese da BDR per produrre una soluzione pi semplice e pi semplice da utilizzare per la replica unidirezionale, utilizzabile da pi persone per una vasta gamma di casi di utilizzo. BDR stato progettato innanzitutto per il multi-master della rete a n-way, e questo stato difficile adattarsi bene alla replica mono-master a senso unico.

Lo sviluppo di BDR continuer per quelli che richiedono piena capacit 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.

`DA METTERE? DIFFERENZA TRA PG LOGICAL E BDR` Replica pglogical Bi-Directional Replication (BDR)? No. pglogical non fornisce funzionalit complete di replica multi-master e un supporto di modifica dello schema coerente, come fa la BDR.

Pglogical incorpora le funzionalit di BDR e le lezioni apprese da BDR per produrre una soluzione pi semplice e pi semplice da utilizzare per la replica unidirezionale, utilizzabile da pi persone per una vasta gamma di casi di utilizzo. BDR stato progettato innanzitutto per il multi-master della rete a n-way, e questo stato difficile adattarsi bene alla replica mono-master a senso unico.