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)

1.1.1 *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, cioè l'insieme di file fisici che effettivamente memorizzano i dati.[1] 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.

Seguono due dei vantaggi principarli offerti dal clustering, specialmente in un ambiente di database di alto volume:

- Fault tolerance (tolleranza di guasti): in caso di guasto del singolo server, il cluster offre un'alternativa, poiché esiste più di un server o istanza per gli utenti a cui connettersi.
- 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.[1]

Ci sono differenti tipi di architetture clustering, che si diversificano da come vengono memorizzati i dati e allocate le risorse. La prima modalità di clustering è conosciuta come architettura "Shared-Nothing" (SN). È un'architettura di elaborazione distribuita in cui ogni nodo/server è totalmente indipendente e autonomo, pertanto nessuno dei nodi condivide memoria o archiviazione del disco. Più generalmente, non esiste un

unico punto di contesa nel sistema.[3] Il partizionamento è tale che ogni nodo possiede un sottoinsieme dei dati, ovvero ogni nodo ha accesso esclusivo su quel particolare sottoinsieme. ((Un esempio di questa forma di clustering potrebbe essere quando un'azienda ha più data centers per un unico sito web. Con molti server in tutto il mondo, nessun singolo server è un "master".[2]))

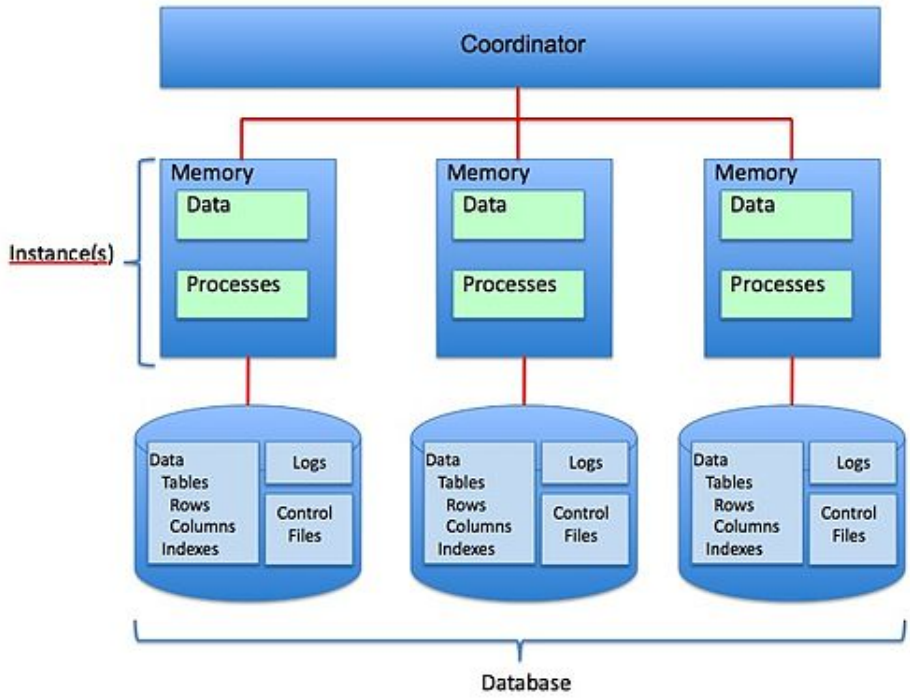

Figura 1: Architecture Shared Nothing [4]

I vantaggi dell'architettura SN rispetto a un'entità centrale che controlla la rete (un'architettura basata su controller) riguarda l'eliminazione di qualsiasi singolo punto di guasto, consentendo funzionalità di (self-healing) auto-riparazione e fornendo un vantaggio nell'offrire aggiornamenti non distruttivi.[5] **DA LEGGERE PER BENE ARTICOLO CITATO.** Shared-nothing è anche noto come "database sharding". In generale, un sistema SN divide i suoi dati in vari nodi su database diversi o può richiedere a ciascun nodo di mantenere la propria copia dei dati dell'applicazione utilizzando un qualche tipo di protocollo di coordinamento.[3]
Si oppone a quest'ultima, l'architettura nota come "shared-disk" (disco condiviso), in cui tutti i dati vengono memorizzati centralmente in un unico disco e sono accessibili da tutti i nodi di cluster.[4] In questo tipo

di struttura quindi più istanze di database vengono raggruppate in un singolo database sul disco. Nei sistemi di dischi condivisi, i blocchi (o pagine) di dati su disco possono avere un solo proprietario.((La proprietÃ dei blocchi (o pagine) viene trasferita all'istanza che sta facendo l'aggiornamento. Questo genera traffico di rete e in genere un'infrastruttura dedicata viene implementata tra le istanze di database per far fronte a questo traffico.)) L'architettura Shared-Disk è un esempio di Synchronous Multi-Master, ovvero ogni istanza del database può scrivere (cioè è un master) in modo sincrono.[4] **DA CHIEDERE E DA CONTROLLARE ULTIME FRASI. GUARDA COMMENTO per il sito**

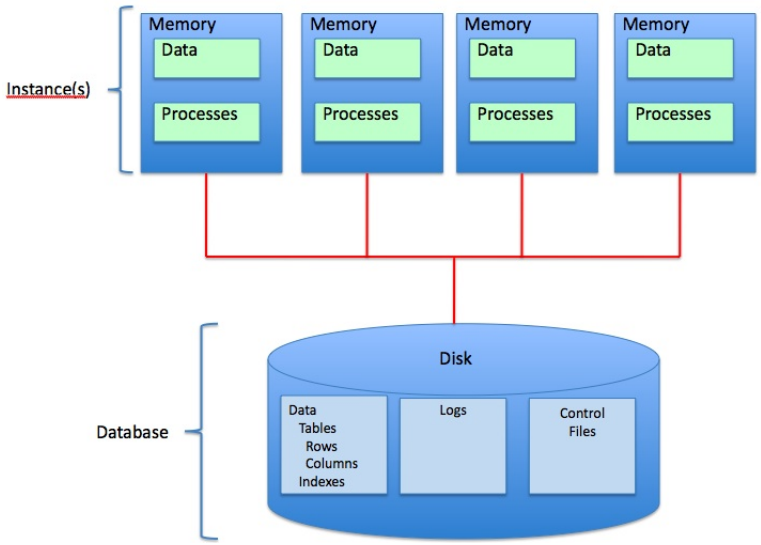

Figura 2: Architecture Shared Disk - Shared Everything [4]

In un'architettura SD, grandi reti di computer possono operare su un singolo set di dati senza la necessità di replicare o bloccare quel set di dati.[4] Shared Disk ha due vantaggi: ogni processore ha la propria memoria, il bus di memoria non è un collo di bottiglia (contrariamente all'architettura "Shared-Everything"). Inoltre il sistema offre un modo semplice per fornire un certo grado di tolleranza agli errori.

((La distinzione tra i due tipi è diventata confusa di recente con l'introduzione di grid computing o distribuzione cache. In questa configurazione, i dati sono ancora gestiti centralmente, ma controllati da un potente "server virtuale" composto da molti server che lavorano insieme come uno.[2]))

1.1.2 *Rete di pari*

1.1.3 *Sistemi di ridondanza disco (RAID)*

RAID, acronimo di "Redundant Array of Independent Disks", insieme ridondante di dischi indipendenti (originariamente "Redundant Array of Inexpensive Disks"), è una tecnologia che permette di memorizzare dati su più dischi rigidi in un computer (o collegati ad esso) in modo da garantire una gestione sicura dei dati.[7] Consente di ottenere quindi determinate caratteristiche di protezione e velocità su sistemi "casalinghi" su dischi economici, in contrapposizione a configurazioni riservate per sistemi professionali ben più costosi.

Il RAID, con modalità differenti a seconda del tipo di configurazione, tra vantaggio dai principi di ridondanza dei dati e di parallelismo in modo da ottenere:

- incrementi di prestazioni (in lettura/scrittura)
- aumenti nella capacità di memorizzazione disponibile
- miglioramenti nella tolleranza ai guasti, ne segue migliore affidabilità. [8] Il RAID rende il sistema resiliente alla perdita di uno o più hard disk, permettendo di sostituirli senza l'interruzione del servizio.

I dati vengono suddivisi in sezioni, anche noti come "stripes" di uguale lunghezza, detta l'unità del sezionamento, e scritti su dischi differenti.

La caratteristica fondamentale che identifica una configurazione RAID è l'array, che identifica il tipo di collegamento logico che c'è tra i vari dischi. Con tale criterio è determinato il livello RAID, la configurazione della tipologia di RAID e stabilito il numero minimo di hard disk che sono necessari per attivarlo. A seconda del livello RAID sono implementate diverse tecniche per ottenere maggiori prestazioni o una maggiore sicurezza dei propri dati oppure entrambe le condizioni.

History of RAID? non credo Il RAID funziona mettendo i dati su più dischi e consentendo operazioni di input/output (I/O) di sovrapporsi in modo equilibrato, migliorando le prestazioni. Poiché l'utilizzo di più dischi aumenta il tempo medio tra i guasti (MTBF), memorizzare i dati ridondantemente aumenta anche la tolleranza agli errori.

per proteggere i dati nel caso di un guasto di unità.

spiegazione cinci -audio- Nelle macchine non distribuite per garantire la ridondanza dei dati si scrive su piu dischi, sincrono (i dati vengono

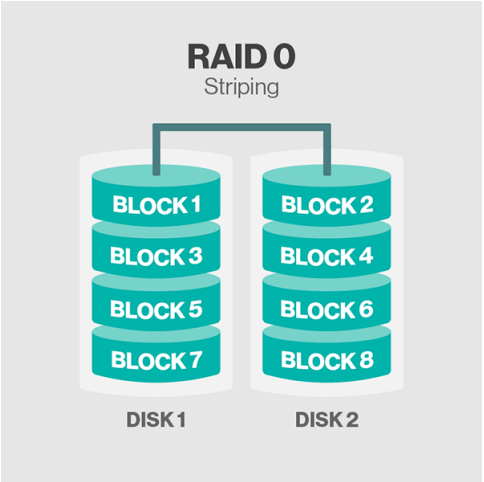

Figura 3: Striping - Sezionamento senza ridondanza [7]

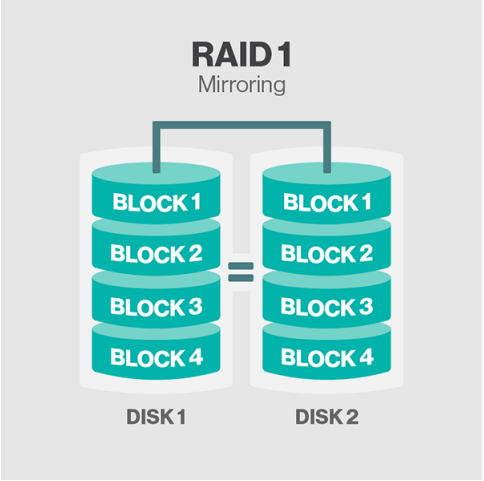

Figura 4: Mirroring - Replicazione [7]

scritti simultaneamente) - come raid mirroring ci puo essere o un affarino (scheda) chiamato raid controller. Ha una memoria che puoi configurarlo.. puoi dirgli tra disco 1 e disco 2 imponi un raid 0. Quindi che succede? Da una macchina, cioe dal processore, tutte le volte che viene fatta una scrittura su disco, il raid controller la splitta e viene fatta in entrambi i dischi. quindi hai due dischi clonati (mirroring). Stessa cosa del REPLICA.

1.1.4 *Codice di correzione errore (erasure coding)*

http://searchstorage.techtarget.com/definition/erasure-coding

1.2 HARDWARE UTILIZZATO PER GLI ESPERIMENTI

Inserire foto hardware con descrizione

1.3 SOFTWARE UTILIZZATO PER GLI ESPERIMENTI

1.3.1 *PosgreSQL*

(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

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 pubbli-

camente ed "aperto" ai contributi degli sviluppatori che volontariamente intendono partecipare alla sua implementazione.

In PostgreSQL, ma il discorso puÃ² essereallargatoatuttiigestoridididatabaserelazion

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 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.

1.3.2 *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 thatâs 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.

1.3.3 *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 meto-

do 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 piÃ¹ *velocecon PostgreSQL 9.5 esuccessive, con bassoverheadsuen*

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 funzionalitÃ complete di replica multi-master e un supporto di modifica dello schema coerente, come fa la BDR. Pglogical incorpora le funzionalitÃ di BDR e le lezioni apprese da BDR per produrre una soluzione piÃ¹ semplice e piÃ¹ semplice da utilizzare per la replica unidirezionale, utilizzabile da piÃ¹ per

Lo sviluppo di BDR continuerÃ per quelli che richiedono piena capacitÃ 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 funzionalitÃ complete di replica multi-master e un supporto di modifica dello schema coerente, come fa la BDR. Pglogical incorpora le funzionalitÃ di BDR e le lezioni apprese da BDR per produrre una soluzione piÃ¹ semplice e piÃ¹ semplice da utilizzare per la replica unidirezionale, utilizzabile da piÃ¹ per

Migrare e aggiornare PostgreSQL con tempi di inattività quasi a zero

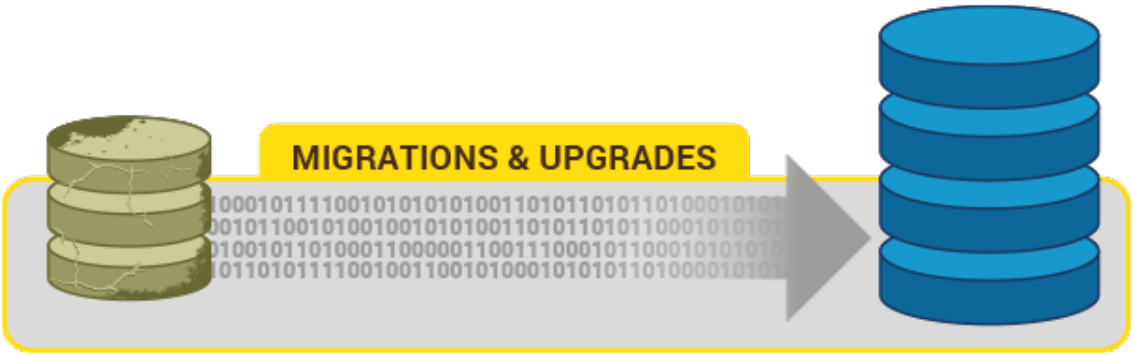

Figura 5: Migrazione e aggiornamenti PostgreSQL [9]

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

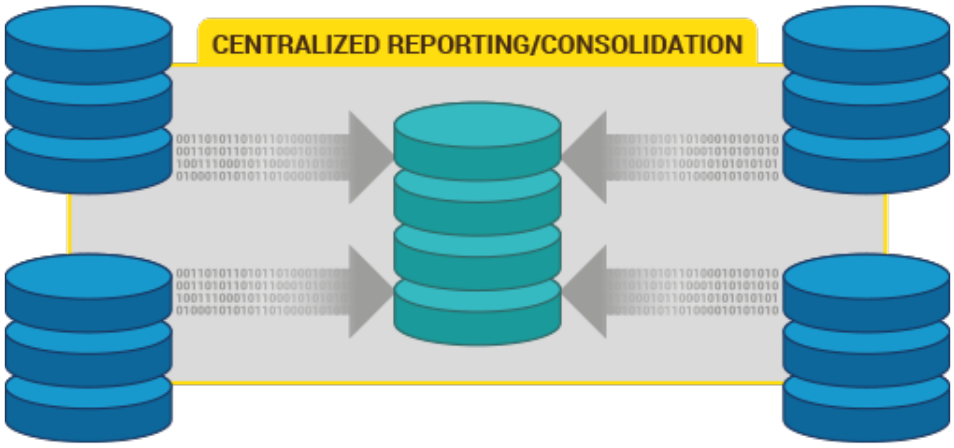

Figura 6: Aggregazione [10]

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

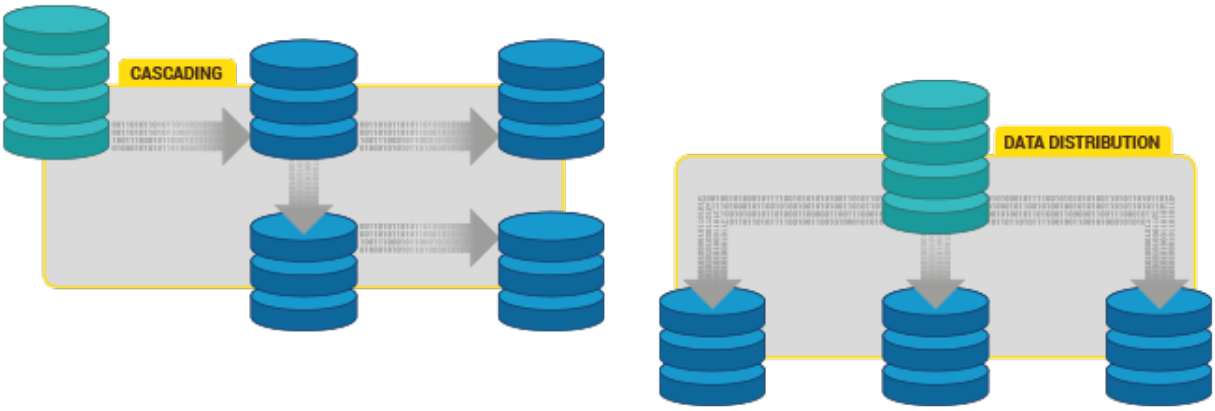

Figura 7: A cascata e distribuzione dati [10]

Le modifiche del database in tempo reale ad altri sistemi

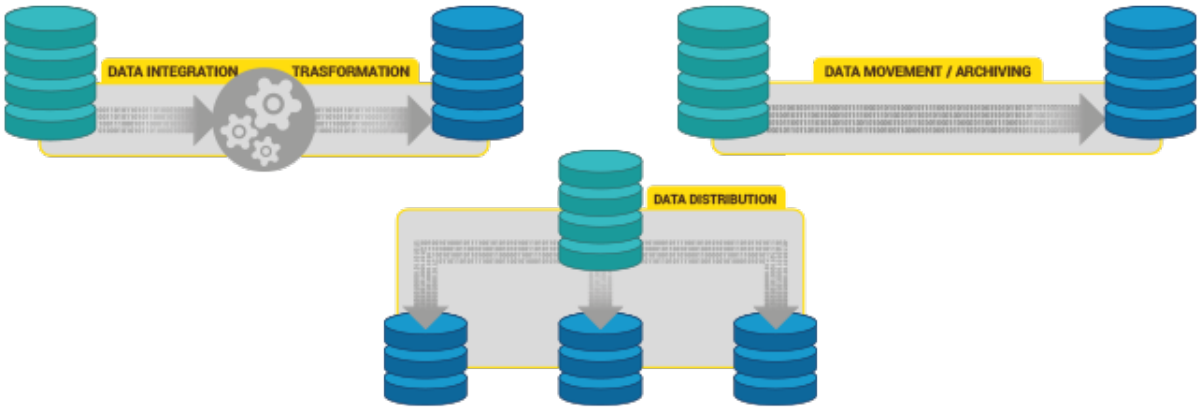

Figura 8: A cascata e distribuzione dati [10]

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

DA RIVEDERE - http://www.slony.info/images/Slony-I-concept.pdf
Failover: While it is relatively easy to tell in a master to multiple slave scenar io which of the slaves is most recent at the time the master fails, it is near ly impossible to tell the actual row delta between two slaves. So in the case of a failing master, one slave can be promoted to the master, but all other slaves need to be re-synchronized with the new master. Perfor mance: Stor ing the logging infor mation in one or ver y fe w rotating log tables means that the replication engine can retrieve the actual data for one replication step with ver y fe w quer ies that select from one table only. In contrast to that a system that fetches the current values from the application tables at replication time needs to issue the same number of quer ies per replicated table and these queries will be joining the log table(s) with the application data table. It is obvious that this systems perfor mance will be reverse proportional to the number of replicated tables. At some time the complete delta to be applied, which can not be split as pointed out already, will cause the PostgreSQL database system to require less optimal than in memory hash join query plans to deal with the number of rows returned by these queries and the replication system will be unable to ever catch up unless the wor kload on the master drops significantly

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] Techopedia - *Definition - What does Clustering mean?* (Cited on page 7.)

[2] Techopedia - *Techopedia explains Clustering* (Cited on pages 8 and 9.)

[3] Wikipedia, the free encyclopedia - *Shared nothing architecture* (Cited on page 8.)

[4] DA RIGUARDARE - *https://en.wikibooks.org/wiki/Oracle_and_DB2,_Comparison_and_Com*

[5] Dave Wright - *The Advantages of a Shared Nothing Architecture for Truly Non-Disruptive Upgrades solidfire.com. 2014-09-17. Retrieved 2015-04-21* (Cited on page 8.)

[6] Ben Stopford - *Shared Nothing v.s. Shared Disk Architectures: An Independent View". November 24, 2009. Retrieved November 1, 2012.*

[7] SearchStorage - *RAID (redundant array of independent disks)* (Cited on pages 3, 10, 11, and 12.)

[8] Derek Vadala - *Managing RAID on Linux, O' Reilly, 2002* (Cited on page 10.)

[9] 2ndQuadrant Ltd - *pglogical* (Cited on pages 3 and 18.)

[10] Autore - *Titolo* - altre informazioni (Cited on pages 3, 18, and 19.)