Laboratorio di Basi di Dati e Programmazione Web

DBMS per sysadm

Stefano Zacchiroli
zack@pps.jussieu.fr

DBMS per sysadm

Info pratiche: uso di DBMS in lab

Punti di vista

- il punto di vista “anomalo” dei sistemisti
  - i DBMS come dipendenze di altri applicativi, non prodotti finali
- una dicotomia
  1. DBMS stand-alone
     - *servizi* esposti agli utenti (localmente o via rete)
     - data storage centralizzato e gestito dal DBMS, namespace DBMS-specific
  2. DBMS embedded
     - librerie linkate a runtime con altri applicativi
     - data storage sul filesystem e gestito dall'applicazione, locazione application-specific

DBMS deployment

- i DBMS stand-alone sono software complessi
  - Li tratteremo separatamente dai DBMS embedded
- alto grado di coupling con il sistema operativo
  - servizi per molti utenti
    - accesso multiutente locale e via rete
  - consumo di risorse, potenzialmente molto elevato
    - memoria, CPU, processi
  - accesso privilegiato al file system (e.g., raw partitions)
  - scheduling della manutenzione ordinaria
- la loro amministrazione richiede privilegi
  - e.g. root (caso generale; a volte meno)

DBMS “virtualizzati”

- Per sperimentare: useremo macchine virtuali in cui siamo root
  - La nostra scelta: virtualbox OSE (Open Source Edition)
- Ognuno di noi è root
  - Potrà affrontare tutte le problematiche di amministrazione di DBMS standalone
- Avete già le immagini pronte (corso di LabSO)
  - Altrimenti: `/nfs/serraglio/srv/mfosset/dbweb/debian.vdi.gz`
  - `e.g.: gunzip -c /nfs/serraglio/srv/mfosset/dbweb/debian.vdi.gz > /public/zack/debian.vdi`
- Accesso alle macchine virtuali
  - In lab
  - In remoto: ssh -X *.cs.unibo.it virtualbox (auguri ...)
  - A casa: copiando l'immagine

Demo

Utente: `root` Password: `qwerty`
Utente: user Password: `user`

Altre info sulle slide del corso di LabSO:
http://www.cs.unibo.it/~tassi/STSL/02-installazione-vm.pdf

DBMS per sysadm

Installazione e setup iniziale

MySQL – overview

- Il primo DBMS che considereremo è MySQL
  - Componente LAMP
  - Licenze: GPL / proprietario
  - ~ 10 milioni di installazioni
  - C / C++, ~ 1 MLOCs
  - Multi-platform (incl. win32)
- Feature
  - I soliti noti: SQL 99, triggers, stored proc., cursori, writable views, ...
  - Salienti: multi-threading, multiple storage engines
- (short) Timeline
  - 1995: prima release
  - 2001: 3.23
  - 2003: 4.0
  - 2004: 4.1
    - Subquery, PREPARE
  - 2005: 5.0
    - Cursors, stored proc., triggers
  - 2008: comprato da Sun
  - 2008: 5.1
  - ...: 6.0
    - Ref. Integrity (!!)

Installazione

- (fortunatamente) non è più necessario compilare un DBMS per installarlo
  - milioni di righe di codice ...
- nelle distribuzioni GNU/Linux i grandi attori sono pacchettizzati
  - analizzeremo il caso di distribuzioni Debian-based
- l'installazione consiste quindi in
  1. installazione dei pacchetti
  2. (post installation setup manuale)
  3. creazione di utenti e database
  4. (goto 3)

MySQL – pacchetti

- pacchetto sorgente `mysql-dfsg-5.0`, binari:
  - `mysql-server-*`
    - server MySQL: demone standalone
  - `mysql-client-*`
    - top-level interattivo per query (“mysql”)
    - tool di amministrazione command line (“mysqladmin”)
  - `mysql-server`, `mysql-client`
    - Meta-package version-agnostic: attenzione agli upgrade!
  - `libmysqlclient*-dev`
    - librerie C (shared e dev) per l'accesso via API *nativa*
  - `mysql-common`
    - shared stuff, e.g. file di configurazione `/etc/mysql/*`

MySQL – pacchetti (cont.)

- attenzione: pacchetti vs metapacchetti
  - pianificate le vostre politiche di upgrade!
  - e.g. mysql-server-5.0 vs mysql-server
- documentazione?
  - non nella distro per problemi di licenza (non-free)
  - su web MySQL reference manual (version-specific) http://dev.mysql.com/doc/refman/5.0/en/
- bells and whistles
  - `mysql-admin`: GUI-based administration tool
  - `phpmyadmin`: web-based administration tool

MySQL – installation HOWTO

- As easy as

  `aptitude install mysql-server`

  - oppure

  `aptitude install mysql-server-5.0`

- ...
- post installazione
  - Leggere
  - `/usr/share/doc/mysql-server-5.0/README.Debian.gz` (!!!)

MySQL – post installation

- la fasa di post-installazione è in buona parte delegata al pacchetto
  - DBMS bootstrap
  - creazione di utenti per la manutenzione ordinaria periodica
  - ...
- manualmente è richiesto il setup della “password di root”
  - senza: tutti gli utenti possono accedere come “root”!
    - (versioni più recenti del pacchetto chiedono la password via debconf)

```
/usr/bin/mysqladmin -u root password “new-password”
```

MySQL – post installation (cont.)

- Tip
  - è consigliabile salvare la password dell'utente root (MySQL) come configurazione dell'utente root (sistema)
- Il file di configurazione `$HOME/my.cnf`

```
# an example of $HOME/.my.cnf
[client]
user = “username”
password = “new-password”
```

MySQL – il servizio di sistema

- mysql server è compatibile con la “API” Debian per i servizi di sistema
  - servizio “mysql”
  - script `/etc/init.d/mysql`

```
Usage: /etc/init.d/mysql start|stop|restart|reload|
  force-reload|status

invoke-rc.d mysql-server start

invoke-rc.d mysql-server stop

invoke-rc.d mysql-server ...
```

- soggetto alla configurazione dei runlevel
  - si veda, e.g., il pacchetto `sysv-rc-conf`

MySQL – configurazione

- Dopo l'installazione di ogni *servizio*, un buon sysadm si domanda: *« dov'è il file di configurazione? »*
- Risposta per MySQL: `/etc/mysql/my.cnf`
  - la suite MySQL include diversi eseguibili
    - e.g.: `mysql, mysqladmin, mysqld, mysqld_safe,ndbd, ndbd_mgmd` ...
  - ogni parametro di configurazione di un eseguibile può essere specificato a cmdline o inserito in `my.cnf`
  - `my.cnf` è diviso in gruppi ini-like (“[gruppo]”)
  - ogni eseguibile legge la conf di uno o più gruppi
    - in Debian anche: `/etc/mysql/conf.d/*`
  - http://dev.mysql.com/doc/refman/5.0/en/program-options.html

MySQL – configurazione (cont.)

- il servizio MySQL è incarnato nel demone “mysqld”
  - (e dallo script mysqld_safe che lo “sorveglia”...)
  - legge i gruppi “mysqld” e “server”
- http://dev.mysql.com/doc/refman/5.0/en/server-options.html
- tipologie di configurazioni per mysqld
  - data dir, runtime dir, ...
  - cache size e log level
  - networking
  - ...

MySQL – root password

- cookbook per 2 problemi comuni
  - *cambiare la password di root*

    ```
    /usr/bin/mysqladmin -u root password 'new-password'
    ```

  - *resettare la password di root* (persa)

    http://dev.mysql.com/doc/refman/5.0/en/resetting-permissions.html

    ```
    invoke-rc.d mysql stop
    SET PASSWORD FOR 'root'@'localhost' =
     PASSWORD('MyNewPassword'); # text file ~/reset
    mysqld_safe --init-file=~/reset &
    rm ~/reset
    ```

Esercizi

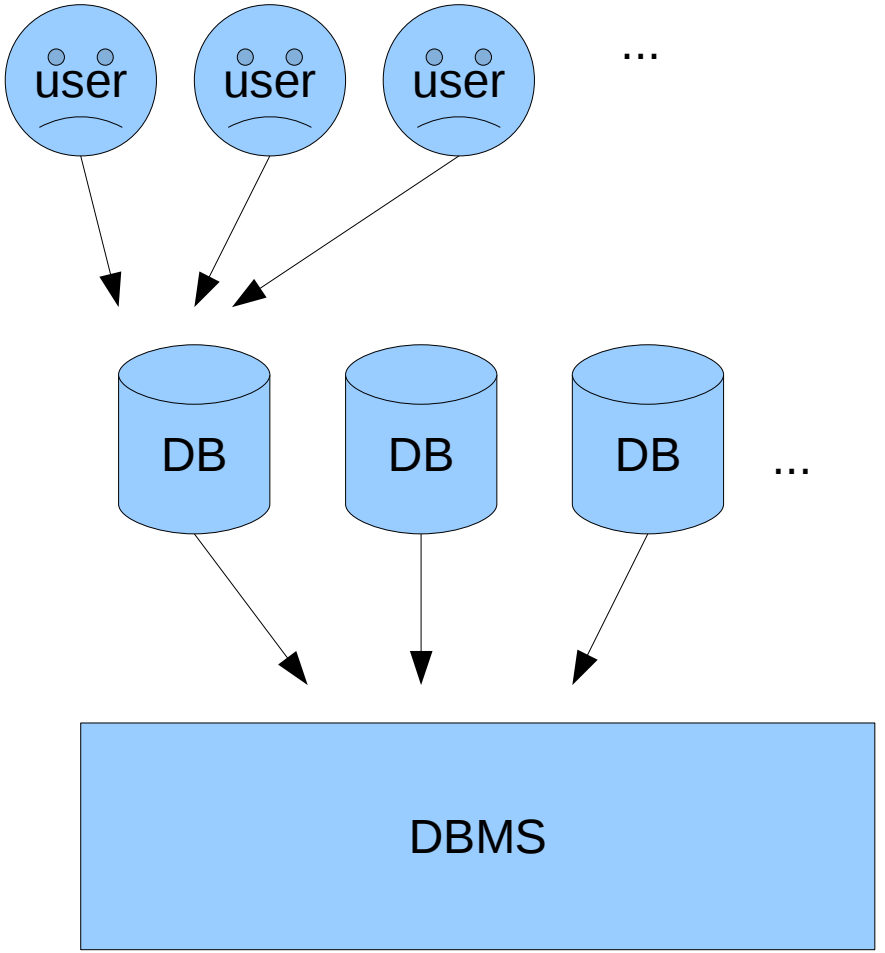
- Installate MySQL server e client
- Configurate password
  - Impostate quella di root
  - Configurate il client “mysql” in modo che non sia richiesta interattivamente
  - Diminuite l'occupazione di risorse in modo da non caricare troppo la macchina guest (e quindi quella host)
- Verificate che il client `mysql` possa connettersi

DBMS per sysadm

uso, management generale

1 DBMS, molti DB, 100'000 utenti

- DBMS usage scenario nel contesto FOSS
  - Web-app (LAMP-like)
  - applicazioni stand-alone con requisiti:
    - di prestazioni
    - di data storage
    - SQL-based query capabilities
- 1 DBMS → molti DB
  - DB namespace, piatto
- 1 DB → molti utenti



Gestione dei DB

- DB diversi, per usi diversi di un DBMS
  - sono necessari meccanismi di creazione/distruzione dei vari DB che popolano il namespace di un DBMS
    - se ne occupa il DCL (Data Control Language)
    - vi sono inoltre helper tools (e.g. mysqladmin) usati come wrapper sul DCL
- tanti utenti per un DB
  - per il *principio dei privilegi minimi* sono necessarie politiche (ACL) e meccanismi (controllo di accesso)
    - DCL + helper
  - tecnicamente è necessario permettere agli utenti di accedere localmente e/o via rete
    - configurazione del DBMS

MySQL – gestione dei DB

- Creazione di database
  - `create database db_name;  -- dcl`
  - `mysqladmin create db_name  # helper`
- Rimozione di database
  - `drop database db_name;  -- dcl`
  - `mysqladmin drop db_name  # helper`
- Ispezione e uso dei database
  - `show databases;  -- mostra i db correnti`
  - `use db_name;  -- passa al db db_name`
  - E ancora
    - `show tables;`
    - `describe <TABLE>;`
    - …

Interazione con i DBMS

- gli utenti finali difficilmente interagiscono direttamente con i DBMS
  - le applicazioni mediano per loro e filtrano gli errori
    - accesso al DBMS via API
  - gli utenti finali spesso non sanno dell'esistenza di un DBMS!
- come amministratori (o programmatori) è spesso utile una interazione non mediata
  - motivazioni: test di funzionalità, debugging, auto-apprendimento
- i DBMS offrono *top-level interattivi* (console)
  - accesso mediato dalla API, ma meno “filtrato”

MySQL – shell

- “mysql” è un client per l'accesso a MySQL
  - permette uso interattivo (top-level)
    - GNU readline capabilities
  - è script-friendly: si comporta come un filtro UNIX tra standard input e standard output
  - http://dev.mysql.com/doc/refman/5.0/en/mysql.html
- esempi d'uso:

```
mysql db_name                                    # defaults
mysql -p -h host -P port -u user db_name         # TCP/IP
mysql -b db_name < script.sql > output.tab       # batch
mysql -X ...                                     # XML output
```

http://dev.mysql.com/doc/refman/5.0/en/mysql-command-options.html

MySQL – shell (cont.)

- configurazione di “mysql”
  - via gruppo “[mysql]” di my.cnf
- esempio (deja-vu?):

```
[client]
user = “username”
password = “new-password”
```

- Reminder
  - Le chiavi del file di configurazione sono 1-1 con le opzioni command line

MySQL – shell (cont.)

```
$ # da utente MySQL root
$ mysqladmin create studenti
$ mysql studenti < studenti.sql
$ mysql studenti
mysql> source studenti_data.sql;
mysql> show tables;
mysql> describe Studenti;
mysql> select * from Studenti;
mysql> ...
mysql> CTRL-D
$
```

MySQL – dump & restore

- Capita sovente di avere necessità di backup/restore di specifici database
  - Migrazioni (e.g., da un DBMS all'altro)
    - Nota: implica un certo grado di *portabilità* del backup
  - Safety measure (e.g., major release upgrade)
- `mysqldump` è la soluzione proposta da MySQL
  - Implementa backup usando statement SQL come backup encoding
  - Portabilità a scelta dell'utilizzatore
    - Opzioni per abilitare istruzioni MySQL-specific

MySQL – mysqldump

```
shell> mysqldump [options] db_name [tables]
shell> mysqldump [options] --databases db1 [db2 ...]
shell> mysqldump [options] --all-databases
```

- Output: standard output
- Opzioni notevoli
  - `--opt` (default) abilita estensioni MySQL-specific; trade-off performance a scapito di portabilità
  - `--compatible=name` richiede compatibilità con DBMS specifici, e.g.: postgresql, oracle, db2, maxdb, …
  - `--xml` produce output in formato XML
  - `--no-create-db --no-create-info --no-data`
  - `--add-drop-database --add-drop-table`
- http://dev.mysql.com/doc/refman/5.0/en/mysqldump.html

Database di esempio

- Avremo a disposizione vari DB di esempio
  - Il database “studenti”, dalle slide della parte DBMS di questo corso:
    http://upsilon.cc/~zack/homepage/teaching/0809/mfosset/01-lezione-1.pdf , slide 54
    - A voi l'onore di crearlo :-)
  - Il database “world”, esempio standard di MySQL (small-sized)
    - `/nfs/serraglio/srv/mfosset/dbweb/data/world.sql`
  - Il database “employees”, esempio standard di MySQL (medium-sized), da http://launchpad.net
    - `/nfs/serraglio/srv/mfosset/dbweb/data/employees_db/`
  - Il database di IMDb .... !!!

Intermezzo – IMDb in SQL !

- IMDb (http://imdb.com) è un popolare sito Web di informazioni cinematrografiche
- la sua base dati è esportata in formato plain text periodicamente
  - http://www.imdb.com/interfaces.html
  - ftp://ftp.fu-berlin.de/pub/misc/movies/database/
- Il progetto FOSS IMDbPY offre
  - Un'interfaccia Python programmatica per l'accesso alla base dati plain text
  - Un wrapper per fare inject di tutti i dati in DBMS vari (MySQL, Postgres, SQLite, …)
  - http://imdbpy.sourceforge.net/

Intermezzo – IMDb in SQL ! (cont.)

- Qualche numero
  - 660 Mb: la base plain text gzippata
  - 4 Gb il DB MySQL (2h circa di import-time)
  - ~ 20 tabelle
  - Tuple
    - ~ 1.3M (titoli)
    - ~ 2.5M (persone)
    - ~ 20M (casting)
    - ...
- Troppo grande per essere facilmente utilizzabile nelle nostre virtual machine :-(

IMDb: i film “del millennio”

- Useremo quindi una versione ridotta del DB
  - Proiettata sui film dell'anno 2000
  - Creata ad hoc
    - Prima creando nuove tabelle filtrate
    - Poi esportandole con mysqldump
- Il risultato si trova in

`.../dbweb/2000_imdb.sql.bz2`

- ```
  create table 2000_title as
    select * from title
    where production_year = 2000 ;
  ```
- ```
  create table 2000_cast_info as
    select cast_info.id, person_id, movie_id,
      person_role_id, note, nr_order, role_id
    from cast_info, 2000_title
    where cast_info.movie_id = 2000_title.id;
  ```
- ```
  create table 2000_name as
    select name.id, name, imdb_index, imdb_id,
      name_pcode_cf, name_pcode_nf,
      surname_pcode
    from name, 2000_cast_info
    where name.id = 2000_cast_info.person_id;
  ```
- ```
  create table 2000_movie_keyword as
    select movie_keyword.id, movie_id,
      keyword_id
    from movie_keyword, 2000_title
    where
      movie_keyword.movie_id = 2000_title.id;
  ```

Esercizi

- Create il database MySQL “studenti”
  - Scrivete le query SQL che ritornano:
    - Gli indirizzi di posta elettronica degli studenti che hanno preso 30 e lode nel corso di Analisi
    - Il nome e cognome degli studenti che hanno sostenuto almeno un esame (senza duplicati)
- Importate in MySQL i database “world” e “employees”
- Importata in MySQL il database IMDb (anno 2000)
  - Scrivete le query SQL che ritornano
    - Gli attori di un film a vostra scelta (del 2000)
    - Tutti gli altri film nei quali 2 attori di un dato film hanno recitato assieme
    - … add your own query here !

Esercizi (cont.)

- Considerate il database studenti
  - Quale dei vincoli previsti dallo schema *non* è enforced da MySQL?
  - http://dev.mysql.com/doc/refman/5.0/en/create-table.html

DBMS per sysadm

controllo di accesso

Controllo di accesso

- i DBMS offrono come *servizi di sistema* accesso a larghe e complesse basi di dati
- il principio dei privilegi minimi è implementato con controllo di accesso a 2 livelli:
  1. *canali di accesso* al servizio DBMS
  2. *fine-grained access control*
     - per-db policy
     - meccanismi di gestione dell'utenza
       - Delegati agli admin dei singoli DB
     - possibili, ma non necessarie relazioni con l'utenza system-wide
       - i.e., DB admin <> root, DB user <> UNIX user

Canali di accesso

- canali di accesso
  - *locale* (e.g.: socket sul filesystem)
  - *TCP/IP* (e.g.: socket TCP)
  - ... ma potenzialmente anche Sun RPC, SOAP, ...
- il DCL dei DBMS è solitamente in grado di distinguere tra utenti locali e utenti remoti
  - non sono necessarie utenze separate per i 2 realm
- la gestione dei canali di accesso si riduce a:
  1. abilitare/disabilitare i canali di accesso
  2. gestire politiche di accesso extra-DBMS
     - e.g.: firewalling, port binding, file system permission, ...

MySQL – canali di accesso

- MySQL offre 2 canali di accesso
  - file system socket
  - porta TCP/IP
- `/etc/mysql/my.cnf`, gruppo “mysqld” contiene le configurazione dei canali di accesso
- i default:

```
socket = /var/run/mysqld/mysqld.sock


bind-address = 127.0.0.1
port = 3306
# skip-networking
```

Utenza

- i DBMS offrono una gestione dell'utenza separata rispetto all'utenza di sistema
  - rationale:
    - le basi dati sono complesse, spesso hanno necessità di controllo di accesso diverse da quelle di sistema
    - le entità da controllare sono nel dominio del DBMS, non noto al sistema host
    - gli utenti dei vari db spesso non hanno controparti negli utenti di sistema (caso notevole: web apps)
- è comunque spesso possibile “ereditare” utenti di sistema

Controllo di accesso in SQL

- SQL offre un meccanismo di controllo di accessi
  - basato su:
    - *authorization IDs* (nomi utente)
    - privilegi per effettuare operazioni su tabelle
    - due statement nel DCL: GRANT e REVOKE
- al singolo DBMS viene demandata
  - gestione degli authorization ID
  - più (ovviamente) estensioni e restrizioni DBMS-specific
- analizziamo inizialmente il meccanismo nativo di SQL

Privilegi

- i privilegi di SQL
  1. select
  2. insert
  3. delete
  4. update
  5. references
  6. usage
  7. trigger
  8. execute
  9. under

- 1-4 si applicano a tabelle (o viste) con la semantica ovvia
- 5 permette di referenziare una tabella come FOREIGN KEY
- 6: “uso” in altre dichiarazioni
- 7 definizione di trigger
- 8 esecuzione di stored procedure
- 9 sottotipaggio

Controllo dei privilegi

- ogni query SQL richiede un insieme di privilegi per essere portata a termine
  - Esempio:

```
INSERT INTO Studio(name)
   SELECT DISTINCT studioName
   FROM movie
   WHERE studioName NOT IN
      (SELECT name
       FROM Studio)
```

  - privilegi richiesti:
    - insert su Studio
    - select su Studio (non implicato)
- *tutti* i privilegi sono necessari per completare la query con successo

Authorization id

- ogni query SQL viene eseguita da un agente che impersona un authorization ID: il *current authorization ID*
  - viene solitamente stabilito all'atto della connesione al DB (via API o altri client)
    - e.g. `mysql -u user db_name < foo.sql`
    - fa si che il current authorization ID per l'esecuzione delle query contenute nel file foo.sql sia “user”
  - può cambiare in corso d'opera con appositi statement SQL
    - Usati di rado

GRANT statement

- lo statement GRANT permette ad un utente (i.e. un agente in esecuzione con un certo authorization ID) di *delegare* privilegi ad altri utenti
  - Tipicamente, esiste un utente super-user che possiede tutti i privilegi
    - è necessario per il bootstrap del processo di delega
- La delega è persistente (sticky), ma revocabile

GRANT statement (cont.)

```
GRANT <privilege list> ON <db element> TO <user list>
  [WITH GRANT OPTION]
```

- `db element` rappresenta una entità referenziabile del db
  - e.g. una tabella, un campo, l'intero db
- `privilege list` rappresenta la lista dei privilegi che si vuole delegare
- `user list` rappresenta la lista degli utenti ai quali delegare i privilegi
- `WITH GRANT OPTION`, se presente, permette agli utenti designati di delegare a loro volta i privilegi ottenuti ad altri utenti

- GRANT è eseguibile solo da utenti che possiedono *tutti* i privilevi da delegare

REVOKE statement

- sintassi:

```
REVOKE <privilege list> ON <db element> FROM <user
  list>
```

- semantica intuitiva e duale a quella di GRANT

MySQL – utenza

- Il sistema di privilegi di MySQL si occupa di
  0. Autenticare connessioni di utenti locali e remoti
     - Outcome booleana: connessione permessa o rifiutata
  1. Associare gli utenti che si sono connessi ad un insieme di privilegi
  2. Verificare, query per query, che i privilegi necessari ad eseguire la query siano associati all'utente che si è connesso
- Nota:
  - (0) è una funzionalità addizionale rispetto a quanto previsto da SQL standard
  - È implementato da molti DBMS come controllo addizionale di sicurezza

MySQL – authorization id

- L'identità in MySQL è determinata dalla coppia:
  1. host dal quale proviene la connessione
  2. username specificato da chi richiede la connessione
     - N.B. in alcuni client (e.g. “mysql”) uno username non specificato ha come default prima il file di configurazione, se esiste, poi l'utente di sistema. Ciò non implica alcuna correlazione tra utenti di sistema e utenti MySQL
- Rationale:
  - macchine diverse = realm di protezione diversi

```
SELECT CURRENT_USER();    -- utente corrente
```

MySQL – controlli

- il controllo di accesso effettuato da “`mysqld`” si divide in 2 fasi:
  1. controllo del permesso di connettersi *(connection verification)*
  2. controllo query per query dei privilegi *(request verification)*
- in entrambi le fasi il server fa affidamento su tabelle del db “mysql” dette *grant table*
  - “user”, “db”, “host” (coarse grained access control)
  - “tables_priv”, “columns_priv”, ... (fine grained)
  - ogni tabella contiene *scope columns* (il contesto della riga) e *privilege columns* (i privilegi garantiti)

MySQL – privilegi

- in aggiunta ai privilegi di SQL MySQL offre privilegi molto fini per controllare
  - chi può creare e rimuovere elementi del db (CREATE/DROP)
  - azioni su viste
  - azioni amministrative sul db (e.g., shutdown)
    - si, è possibile effettuarlo via DCL (?!?!)
  - azioni su indici
- http://dev.mysql.com/doc/refman/5.0/en/privileges-provided.html

MySQL – connection verification

- una connessione da user@host (con password pwd) è accettata se
  - nella tabella “user” esiste una riga t.c. Host=“host”, User=“user” (, Password=“pwd”)
- notazioni
  - tutti i campi supportano le usuali wildcard di SQL
    - “%” (sequenza arbitraria di caratteri)
    - “_” (un carattere arbitrario)
  - host specificati come IP supportano netmask
    - AKA: CIDR notation
    - e.g. “192.168.0.0/255.255.255.0”

MySQL – request verification

- tabelle
  - la tabella “user” stabilisce i privilegi DBMS wide
    - e.g. se in “user” viene garantito il privilegio “DELETE” l'utente può cancellare righe dalle tabelle di tutti i DB
  - “db” e “host” garantiscono privilegi DB-specific
  - “tables_priv”, “columns_priv”, ... DB-element specific
- GRANT e REVOKE possono essere usati per modificare tutti i privilegi d'accesso visti in MySQL
  - … ma è consigliato l'uso di statement specifici per l'utenza

MySQL – gestione dell'utenza

- creazione di utenti (senza privilegi)

```
CREATE USER user [IDENTIFIED BY [PASSWORD] 'password']
```

  - poi GRANT/REVOKE ....
- rimozione di utenti

```
DROP USER user
```

- password

```
SET PASSWORD [FOR user] = PASSWORD('some password')
```

- ispezione dei privilegi

```
SHOW GRANTS [FOR user]
```

- http://dev.mysql.com/doc/refman/5.0/en/account-management-sql.html

Esercizi

- Nel db “studenti” precedentemente creato impostare i permessi come segue
  1. creare un account “student-admin” con tutti i privilegi possibili sul db (tranne la grant option)
  2. creare un account “segretario” (con password) che possa ispezionare e modificare le tabelle “Studenti” e “Corsi” e che inoltre possa ispezionare la tabella “Esami”
  3. creare un account “docente” (con password) che possa inserire nuove righe nella tabella “Esami”
  - l'account “admin” deve potere accedere solo da localhost, gli altri solo dalla rete 192.168.0.0/24
- Verificare con query effettuate via mysql -u / -p che i permessi configurati siano corretti

DBMS per sysadm

MySQL: storage engine

MySQL – storage engine

- Nello schema del nostro esempio molti constraint sono stati ignorati
  - check, foreign key, ...
  - perché?
    - perché lo *storage engine* di default per le tabelle non li supporta (!!)
- Molte altre feature di MySQL sono storage-dependent
  - transazioni, fulltext index, ...
- Molte di queste feature sono supportate dallo storage InnoDB

MySQL – scelta dello storage

- lo storage engine stabilisce la rappresentazione fisica dei dati in “memoria”
  - vari trade-off: efficienza, compattezza, feature
- lo storage engine viene scelto all'atto di creazione di una tabella

```
CREATE TABLE tbl_name (create_definition,...)
  [table_option ...]

table_option:

    ENGINE = engine_name

    ...
```

- Può essere cambiato utilizzando ALTER TABLE
  - Potenzialmente molto time-consuming
- http://dev.mysql.com/doc/refman/5.0/en/create-table.html
- lo storage di default è MyISAM

MySQL – storage engines

- alcuni storage engine di MySQL
  - bdb (BerkeleyDB): transaction-safe, page locking, *deprecato*
  - csv (comma separated value) !!!
  - InnoDB: transaction-safe, row locking, foreign keys
  - memory: heap representation, memory only
  - MyISAM: MySQL default, portabile
  - NDB: clustered, fault-tolerant
- http://dev.mysql.com/doc/refman/5.0/en/storage-engines.html

MySQL – MyISAM

- `CREATE TABLE t (i INT) ENGINE = MYISAM;`
- on disk
  - ogni tabella è rappresentata da 3 file:
    - .frm (table format), .MYD (data), .MYI (index)
    - `ls /var/lib/mysql/`*`dbname/`*
- tradeoff: performance a discapito di feature
  - non sono supportate foreign key e check constraint
  - migliori performance in lettura di altri storage
    - è uno dei motivi per il quale, *nella conf. di default*, MySQL è più performante di altri DBMS
  - sono supportati indici fulltext
- http://dev.mysql.com/doc/refman/5.0/en/myisam-storage-engine.html

MySQL – InnoDB

- `CREATE TABLE t (i INT) ENGINE = INNODB;`
- feature
  - row-level locking, ACID-transactions, foreign keys
  - nessun limite sulle dimensioni delle tabelle
  - alte prestazioni su dati voluminosi (ordine dei TB)
- on disk
  - tablespace privato (per tabella o per db): molti file o anche partizioni raw
    - Nota: partizioni raw → soluzioni di backup non filesystem-level
- configurazione
  - deve essere abilitato in my.cnf, è il default
- http://dev.mysql.com/doc/refman/5.0/en/innodb.html

Esercizi

- ricreate il database “studenti” degli esempi precedenti utilizzando InnoDB come storage engine
- quali constraint sono ora enforced da MySQL?
- cosa manca?

DBMS per sysadmin

introduzione all'amministrazione di PostgreSQL

Postgres – overview

- Postgres è l'altro grande attore tra i DBMS FOSS
  - Solido, estensibile
  - Licenza: BSD
  - C, ~ 500 KLOCs
  - Multi-platform
- Feature
  - I soliti noti …
  - Type system
  - Table inheritance
  - MVCC, no read locks
  - Programmabilità in *SQL
    - Built-in, perl, python, ...
- (short) Timeline
  - 1982: ingres @ Berkeley
  - 1985-88: research
  - 1988: prototype
  - 1991: 3.0
  - 1994: Postgres95
  - 1996: PostgreSQL
    - SQL (!)
  - 1997: 6.0
  - 2000: RedHat backing
  - 2005: commercial support by Pervasive Software
  - … yearly major releases

Postgres – pacchetti

- pacchetto sorgente `postgresql-8.1`, binari:
  - `postgresql-8.1`
    - server: demone standalone
  - `postgresql-client-8.1`
    - client console-like e tool amministrativi
  - libpq5*, libpq*-dev
    - librerie (shared e non) per l'accesso via API nativa
  - `postgresql-common`
    - (source package: `postgresql-common`)
    - shared stuff, e.g. periodic maintenance (cron)
    - management di più versioni di Postgres

Postgres – pacchetti (cont.)

- binari:
  - `libecpg*`, `libpgtypes*`
    - librerie per *Embedded PostgreSQL for C (EPCG)*
      - sviluppo in C con query SQL (a Postgres) verbatim
        - a-la “clipper”, per chi ha funeste memorie
      - non è una versione di Postgres embedded!
  - `postgresql-server-dev-8.1`
    - librerie (dev part) per implementare estensioni SSI di Postgres (ad esempio in C)
    - non per client application
  - `postgresql-contrib-8.1`
    - estensioni di terze parti
    - e.g.: GIST, crypto support, fulltext search, XML storage, tipi di dato per ISBN/DOI, ...

Postgres – pacchetti (cont.)

- binari:
  - `postgresql-pl{tcl,perl,python}*`
    - supporto per l'implementazione di stored procedure in tcl/perl/python
  - `postgresql-doc-8.2`
    - documentazione in formato HTML

      `/usr/share/doc/postgresql-doc-8.2/`
- metapacchetti
  - `postgresql, postgresql-{client,doc,contrib}`

Postgres – installation HOWTO

- facile:

```
aptitude install postgresql
```

  - oppure

```
aptitude install postgresql-8.1
```

- creati automaticamente durante il postinst
  - un cluster “main”
  - un superuser “postgres”
    - cui corrisponde un utente di sistema “postgres” che può accedere come superuser a Postgres

Postgres – servizio di sistema

- il demone è integrato come servizio di sistema e implementa la usuale API Debian
  - servizio “postgresql-X.Y”
    - e.g. postgresql-8.1
  - usual stuff

    `invoke-rc.d postgresql-8.1 start/stop/...`
    - soggetto alla configurazione dei runlevel
- file di configurazione system-wide
  - `/etc/postgresql/`*`X.Y/cluster/`*`.conf`
    - e.g.: `/etc/postgresql/8.1/main/postgresql.conf`

Postgres – gestione dei db

- l'amministrazione di postgres solitamente avviene usando l'utente (di sistema) postgres
  - è possibile delegare l'uso di questo utente ad utenti diversi da root con tecniche usuali
    - sudo, password protected account, ...
  - è possibile che altri utenti di sistemi diventino superuser postgres
- nuovi DB si possono creare con il comando createdb
  - nel PATH dell'utente postgres

```
createdb [mydb]     # default: username (di sistema)
dropdb mydb         # duale: rimuove un db
```

Postgres – shell

- “psql” è un client per l'accesso a Postgres
  - permette uso interattivo (top-level)
- uso:

  `psql [--password] [`*`mydb`* `[`*`username`*`]]`
- prompt shell like:

  `postgres=#  -- prompt for super users`
- permette query interattive SQL e offer comandi non-SQL (*psql commands*) per altri task
  - e.g.: ispezione dei db disponibili: `\l`
- help in linea: (`\h` per SQL), (`\?` per psql)

Esercizi

- Installate Postgres server e client
  - Verificate se sono necessarie eventuali configurazioni manuali post-installazione
- Create i DB equivalenti a quelli creati in precedenza con MySQL
  - Se necessario, adattare gli schemi
  - È importabile il dump di IMDb? Se non lo è, rendetelo tale!
  - Migrate un database da MySQL a Postgres utilizzando mysqldump + psql
- Quali dei vincoli espressi negli schemi non sono enforced?

Postgres – utenza

- l'utenza gestita da Postgres è indipendente dagli utenti di sistema
- è basata su *roles* (gli “utenti” di Postgres)
  - creazione

```
CREATE ROLE name;   -- SQL
createuser name     -- helper
```

  - rimozione

```
DROP ROLE name;     -- SQL
dropuser name       -- helper
```

  - ispezione

```
SELECT rolname FROM pg_roles;      -- SQL
\du                                -- helper
```

Postgres – utenza (cont.)

- per motivi di bootstrap il ruolo “postgres” è predefinito
  - più in generale: nome di chi ha creato il cluster
- ogni connessione a Postgres è effetuata in un ruolo ben preciso
- il processo di *autenticazione* stabilisce l'identità di un utente
  - Outcome booleano: permesso o divieto di connettersi

Postgres – autenticazione

- è controllata dal file di conf “pg_hba.conf”
  - file testuale, insieme di record, uno per riga
    - campi (ordinati):
      1. tipo di connessione
      2. IP address range
      3. db name
      4. user name
      5. metodo di autenticazione
- semantica:
  - ad ogni connessione: viene selezionato il primo record i cui campi da 1 a 4 corrispondono al tentativo di connessione
  - l'utente viene autenticato con il metodo del campo 5

Postgres – auth (cont.)

- possibili formati dei record di pg_hba.conf

```
local      database  user  auth-method  [auth-option]
host       database  user  CIDR-address  auth-method  [auth-option]
hostssl    database  user  CIDR-address  auth-method  [auth-option]
hostnossl  database  user  CIDR-address  auth-method  [auth-option]
```

- tipi di connessione
  - local (locale via socket)
  - host{ssl,nossl} (TCP/IP con/senza SSL)
  - host (TCP/IP whatever)

Postgres – auth (cont.)

- record di pg_hba.conf

```
host      database  user  CIDR-address  auth-method  [auth-option]
```

- database identifica un db
  - valori notevoli: “all”, “sameuser”
- user identifica un utente
  - valori notevoli: “all”, “@file”, “+group”
- CIDR-address identifica un (range di) IP
  - ip address singolo (x.y.z.w)
  - address range (x.y.z.w/netmask)

Postgres – auth (cont.)

- record di pg_hba.conf

```
host      database  user  CIDR-address  auth-method  [auth-option]
```

- auth-method, metodo di autenticazione:
  - trust / reject
    - permetti/rifiuta l'accesso incondizionatamente
  - md5 / crypt / password
    - password based, con vari tipi di password che vengono comunicati lungo la connessione (occhio allo sniffing)
  - Altri: ident, pam, krb5, ldap

Postgres – password

- le pasword possono essere associate agli utenti all'atto della creazione via createuser
  - di default ad un utente non viene associata nessuna password
    - i suoi tentativi di connessione che richiedono un metodo di autenticazione password based falliranno !
    - sane default
- è possibile cambiare la password di un utente utilizzando ALTER ROLE (SQL)
  - e.g.: `ALTER ROLE davide WITH PASSWORD 'hu8jmn3';`

Postgres – privilegi

- GRANT e REVOKE come usuale
- inoltre ad ogni ruolo sono associati *attributi*
  - creati alla creazione del ruolo, modificabili con ALTER ROLE
  - esempi notevoli:
    - LOGIN (permette ad un utente di connettersi)
      - c'è di default per utenti creati con “CREATE USER” o a cmdline
    - SUPERUSER (non sottosta a controllo di permessi)
    - CREATEDB (permette di creare nuovi db)
    - CREATEROLE (permette di creare nuovi ruoli)

Postgres – gruppi

- i ruoli di Postgres possono simulare gruppi di utenti
  - e.g.

```
CREATE ROLE joe LOGIN INHERIT;
CREATE ROLE admin NOINHERIT;   -- no LOGIN
CREATE ROLE wheel NOINHERIT;   -- no LOGIN
GRANT admin TO joe;         -- joe in “group” admin
GRANT wheel TO admin;       -- admin in “group” wheel
```

- “INHERIT” permette di eriditare i permessi dei ruoli di un utente
- cambio di ruolo (se posseduto): `SET ROLE` *`role_name`*`;`

Esercizi

- Create gli utenti “student-admin” e “segretario” come in precedenza
- Create due utenti “pciaccia” e “zack” con ruolo “docente”
  - il ruolo ha i permessi visti in precedenza per “docente”

DBMS per sysadm

High-availability

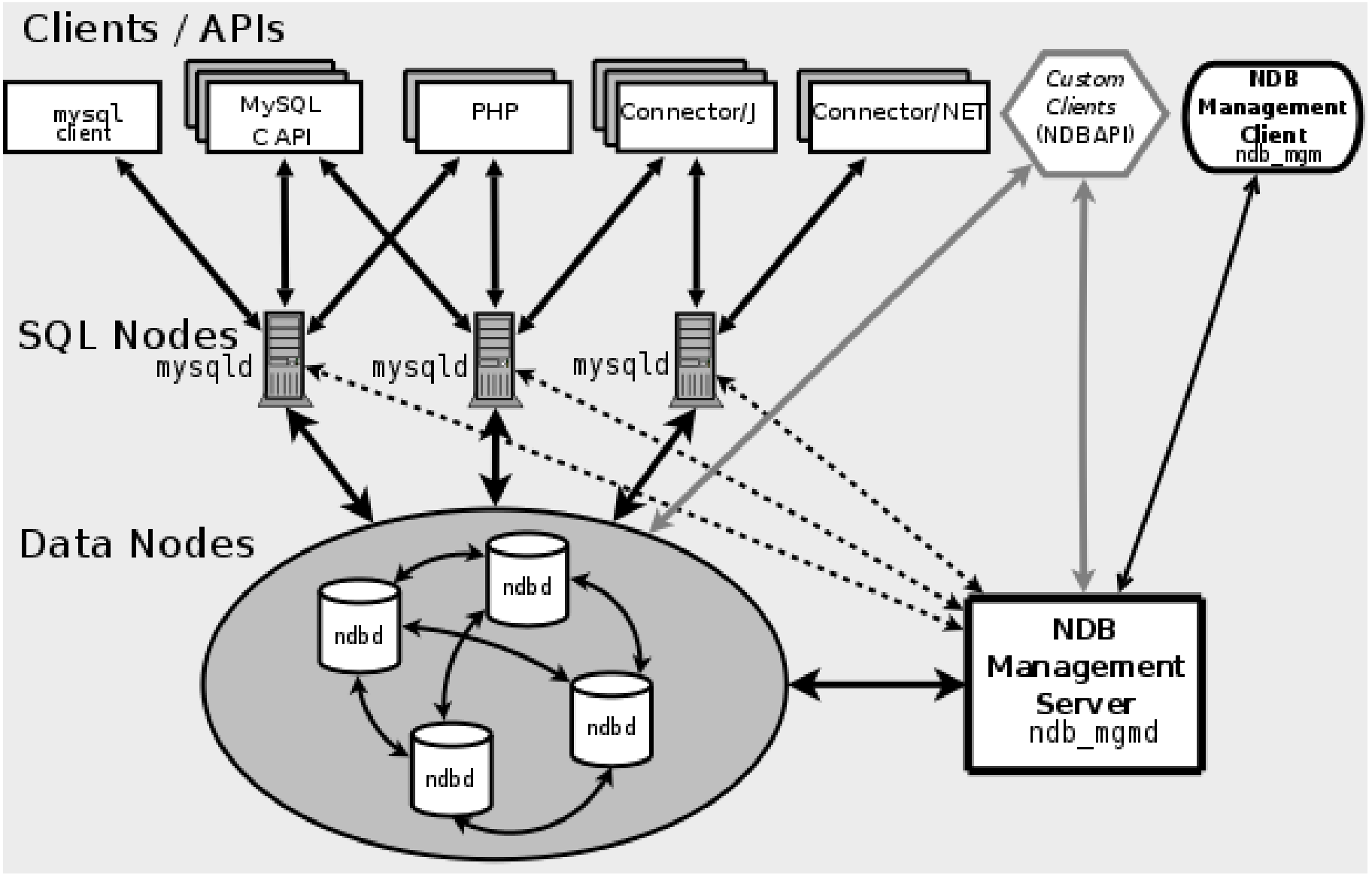
MySQL cluster

- MySQL cluster è un'architettura *high availability* basata su MySQL
- Feature:
  - Ridondanza (custom replica number)
  - Scalabilità (load sharing)
  - No single point of failure, *share-nothing* architecture
    - Scalabilità quasi-lineare
    - Commodity hardware
- Storicamente
  - Integrazione di un 3rd third party acquistando Alzato, by Ericsson
- http://dev.mysql.com/doc/refman/5.0/en/mysql-cluster.html

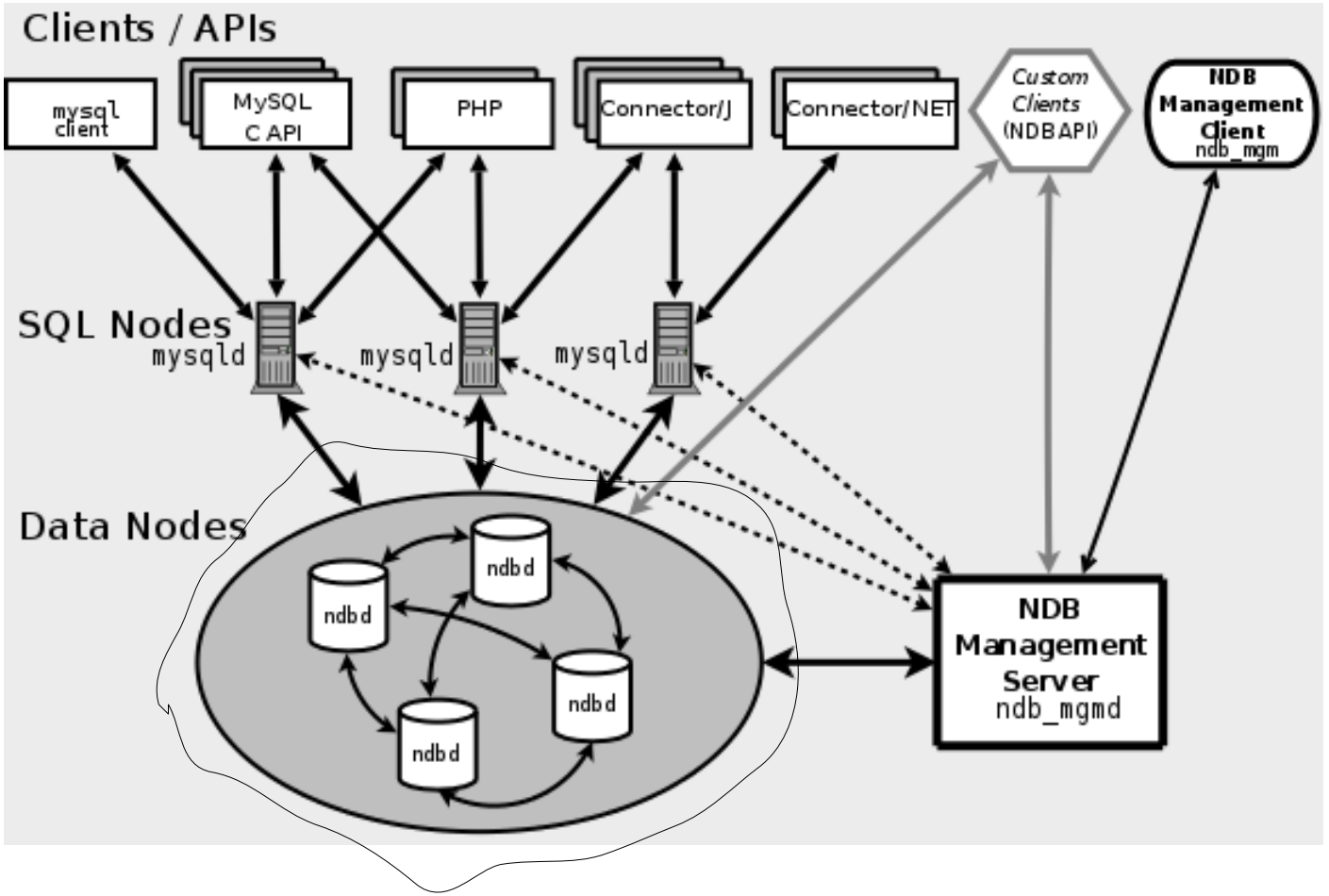
MySQL cluster – principi di base

- Clustering di database *in memory*
- Un cluster è composto da *nodi*
  - Esistono tipologie diverse di nodi
  - Ad ogni nodo corrisponde un processo (e.g. Mysqld)
  - Più nodi possono essere in esecuzionie su un singolo host
    - In generale: i nodi sono distribuiti su più host
    - I nodi comunicano via TCP
- ridondanza / scalabilità
  - Ogni nodo è replicabile
  - Nessun limite superiore alla ridondanza
- Le tabelle affidate al cluster utilizzano lo storage engine NDB (Network DataBase)

MySQL cluster – architettura

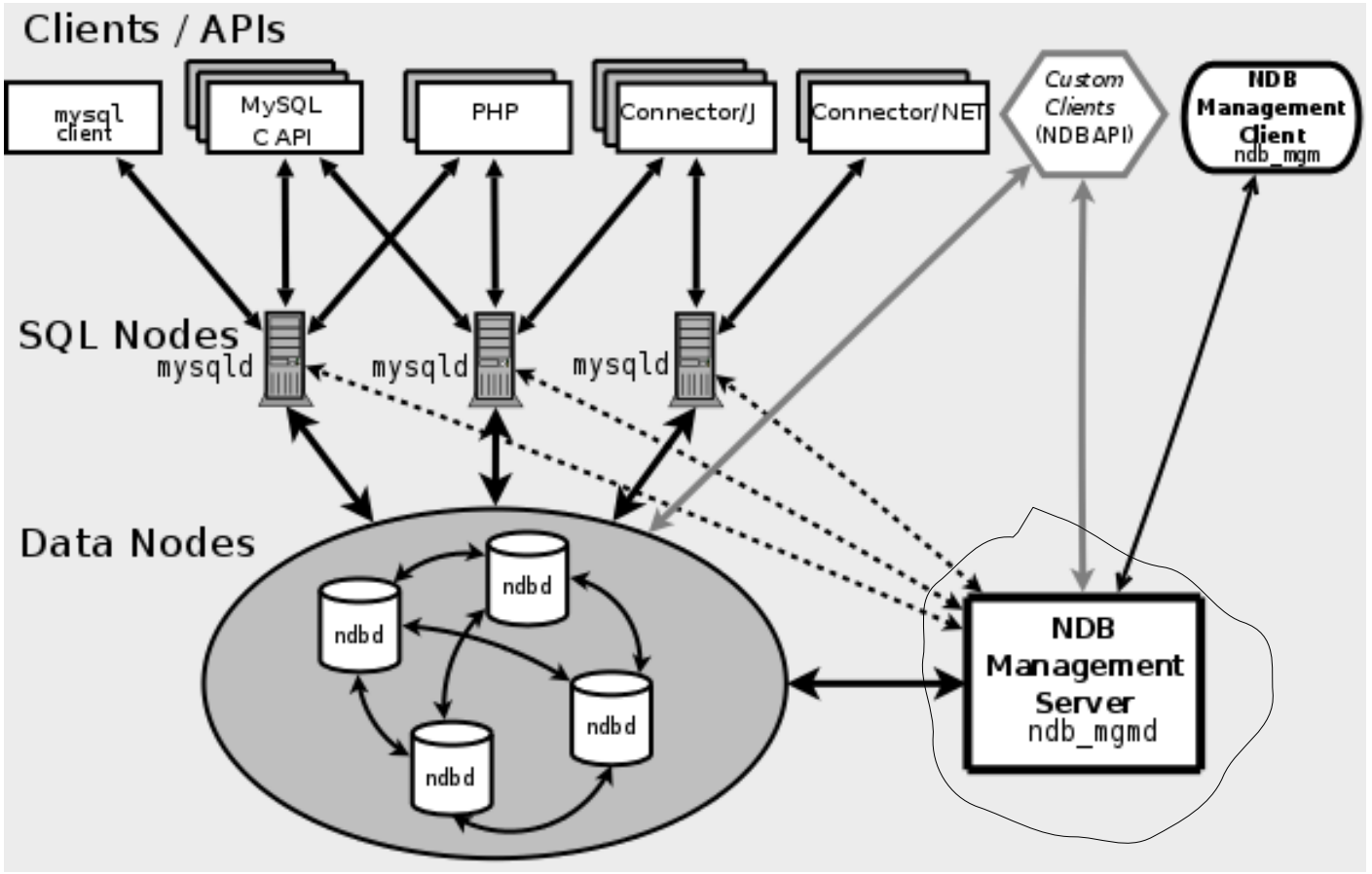


MySQL cluster – node types (1/3)



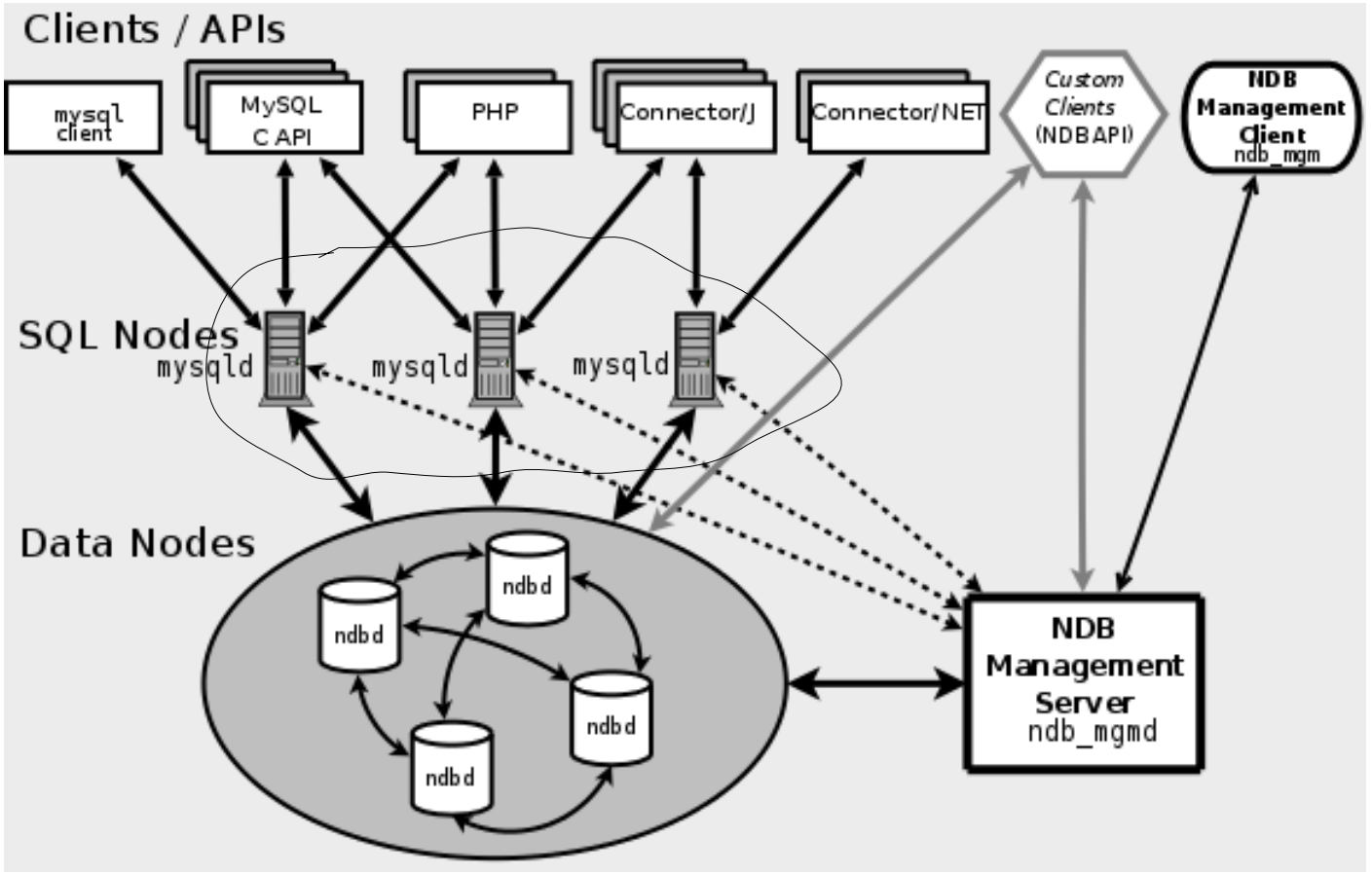
- Data node (processo ndbd)
  - Offrono storage per i dati
  - Più data node offrono ridondanza e data partitioning

MySQL cluster – node types (2/3)



- Management node (processo `ndb_mgmd`)
  - Storage della configurazione degli altri nodi
  - Operazioni di management: startup/shutdown, log, backup, …

MySQL cluster – node types (3/3)



- SQL node (processo `mysqld`)
  - Front-end per un cluster
  - Interaccia legacy di MySQL

MySQL cluster – altri componenti

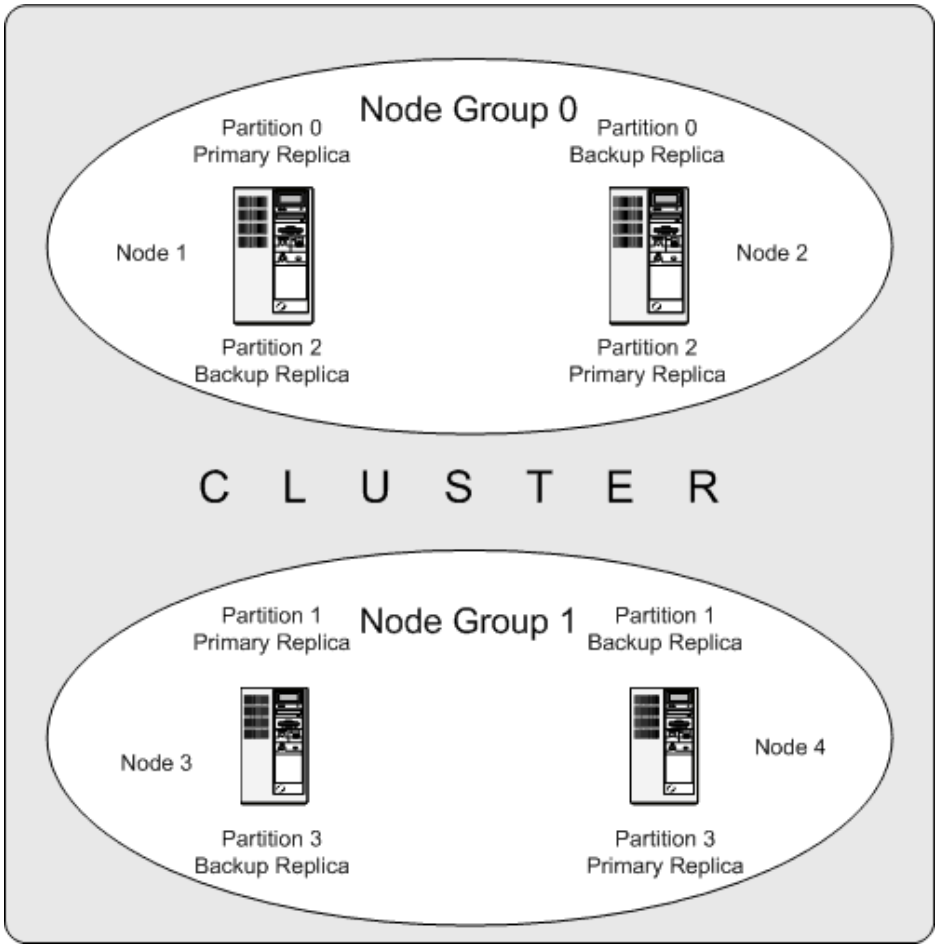
- `mysqld` è solo un esempio di *API node*
  - e.g.: altri programmi che usano la NDB API di libmysql
- Management clients: `ndb_mgm`
  - Interagiscono con `ndb_mgmd` per controllarlo

MySQL cluster – repliche, partizioni

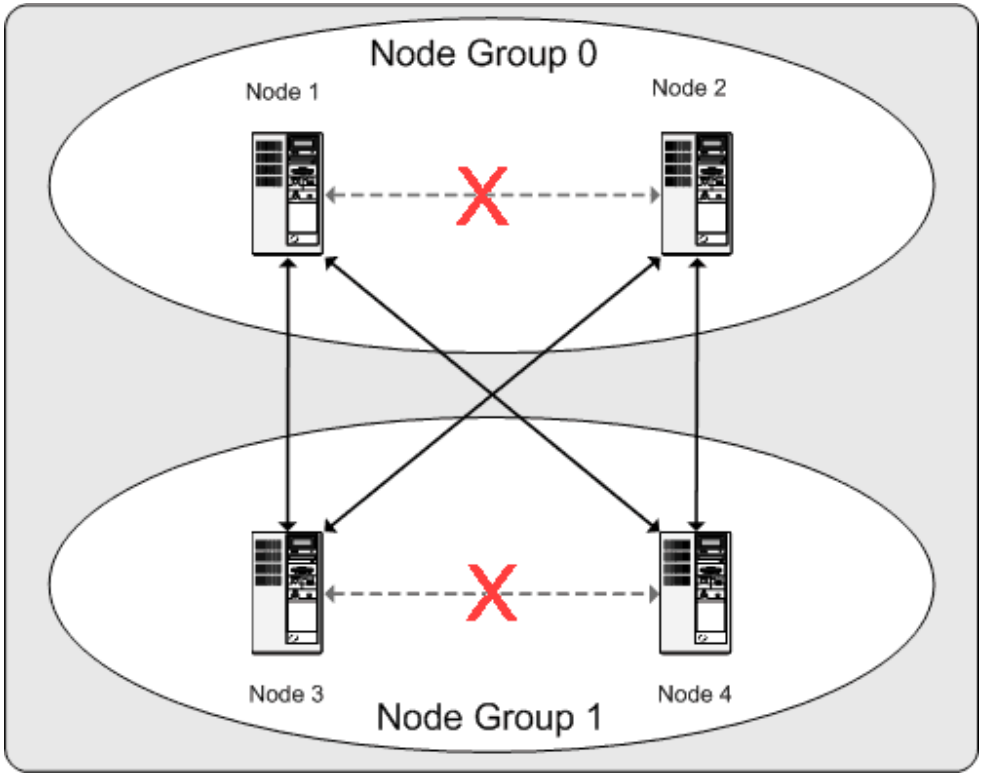
- Una volta decisa quantità è tipologia di nodi …
  - … il numero di data node è fissato (= n. di processi ndbd)
- Il numero di *repliche* (copie ridondante di dati) è un parametro di configurazione
  - `NumberOfReplicas` configuration key
- Ogni data node fa parte di un *node group*
  - n. node groups = n. data nodes / `NumberOfReplicas`
- Una *partizione* è una frazione dei dati del database
  - Ogni partizione è assegnata ad un data node
    - Un data node ha in generale più partizioni associate
  - La divisione in partizioni delle tuple è stabilita da una funzione di hash applicata alla chiave primaria

MySQL cluster – node groups

- Messy ? :-)



- Dati al sicuro fino a quando almeno un nodo per gruppo è up and running



MySQL cluster – note sistemistiche

- MySQL cluster è pensato per reti locali, dietro DMZ
  - Non scala come overhead di rete su WAN
    - Min 100 Mbps
  - Le comunicazioni inter-nodo non sono criptate
  - Switch/router specifici per i nodi del cluster giovano
- Ogni processo richiede connessioni TCP entranti
  - Problematico per le nostre virtualbox :-(

Intermezzo
Virtualbox – inter-host comm.

- Networking di virtualbox fino ad ora (NAT)
  - NAT (Network Address Translation) software
    - I **pacchetti in uscita** dalla macchina guest vengono tradotti in pacchetti prodotti dal software host (il processo virtualbox)
    - Le **tabelle di NAT** vengono mantenute dal software host
    - I **pacchetti in entrata** al software host vengono tradotti e consegnati a processi della macchina guest
- Come fare comunicare macchine guest (in esecuzione su host differrenti) tra loro?
  - Virtualbox offre varie possibilità …
    - http://download.virtualbox.org/virtualbox/2.1.4/UserManual.pdf, capitolo 6
  - Per i nostri scopi è sufficiente il port-forwarding

Intermezzo
Virtualbox – port forwarding

- Il software host, così come fa NAT, può fare port-forwarding
  - virtualbox fa bind di porte sulla macchina host
  - Tabella di forwarding: porte host → porte guest
  - I pacchetti ricevuti sulle porte host sono consegnate sulle porte guest nell'ambiente emulato
- Cookbook
  - (command line interface)

```
VBoxManage setextradata "Linux Guest"
  "VBoxInternal/Devices/pcnet/0/LUN#0/Config/guestssh/Protocol" TCP
VBoxManage setextradata "Linux Guest"
  "VBoxInternal/Devices/pcnet/0/LUN#0/Config/guestssh/GuestPort" 22
VBoxManage setextradata "Linux Guest"
  "VBoxInternal/Devices/pcnet/0/LUN#0/Config/guestssh/HostPort" 2222
```

- More info http://download.virtualbox.org/virtualbox/2.1.4/UserManual.pdf §6.4.1

MySQL cluster – howto

- Data and SQL node
  - Necessitano di sapere dove trovare il (o i) management node
  - In `/etc/mysql/my.cnf`
    - Parametro `ndb-connectstring`
- SQL node
  - Deve essere configurato per abilitare NDB storage engine
  - In `/etc/mysql/my.cnf`
    - Parametro `ndbcluster`

```
[mysqld]
ndbcluster                      # run NDB storage engine
ndb-connectstring=192.168.0.10  # location of management server

[mysql_cluster]
ndb-connectstring=192.168.0.10  # location of management server
```

MySQL cluster – howto (cont.)

- Management node

```
# Options affecting ndbd processes on all data nodes:
[ndbd default]
NoOfReplicas=2    # Number of replicas
DataMemory=80M    # How much memory to allocate for data storage
IndexMemory=18M   # How much memory to allocate for index storage

# TCP/IP options:
[tcp default]
portnumber=2202   # This the default; however, you can use any

# Management process options:
[ndb_mgmd]
hostname=192.168.0.10           # Hostname or IP address of MGM node
datadir=/var/lib/mysql-cluster  # Directory for MGM node log files
```

MySQL cluster – howto (cont.)

- Management node

```
# Options for data node "A":
[ndbd]
hostname=192.168.0.30           # Hostname or IP address
datadir=/usr/local/mysql/data   # Directory for this data node's data
files


# Options for data node "B":
[ndbd]
hostname=192.168.0.40           # Hostname or IP address
datadir=/usr/local/mysql/data   # Directory for this data node's data
files


# SQL node options:
[mysqld]
hostname=192.168.0.20           # Hostname or IP address
```

Esercizi

- Create un MySQL cluster di 4 nodi
  - 2 data node
  - 1 management node
  - 1 SQL node
  - Ogni nodo sia in esecuzione su una macchina guest diversa
  - … ogni macchina guest su una macchina host diversa
- … lavorate a gruppi