MySQL Replikering

IMT3003 kyrre.Begnum@ntnu.no

Risiko idag

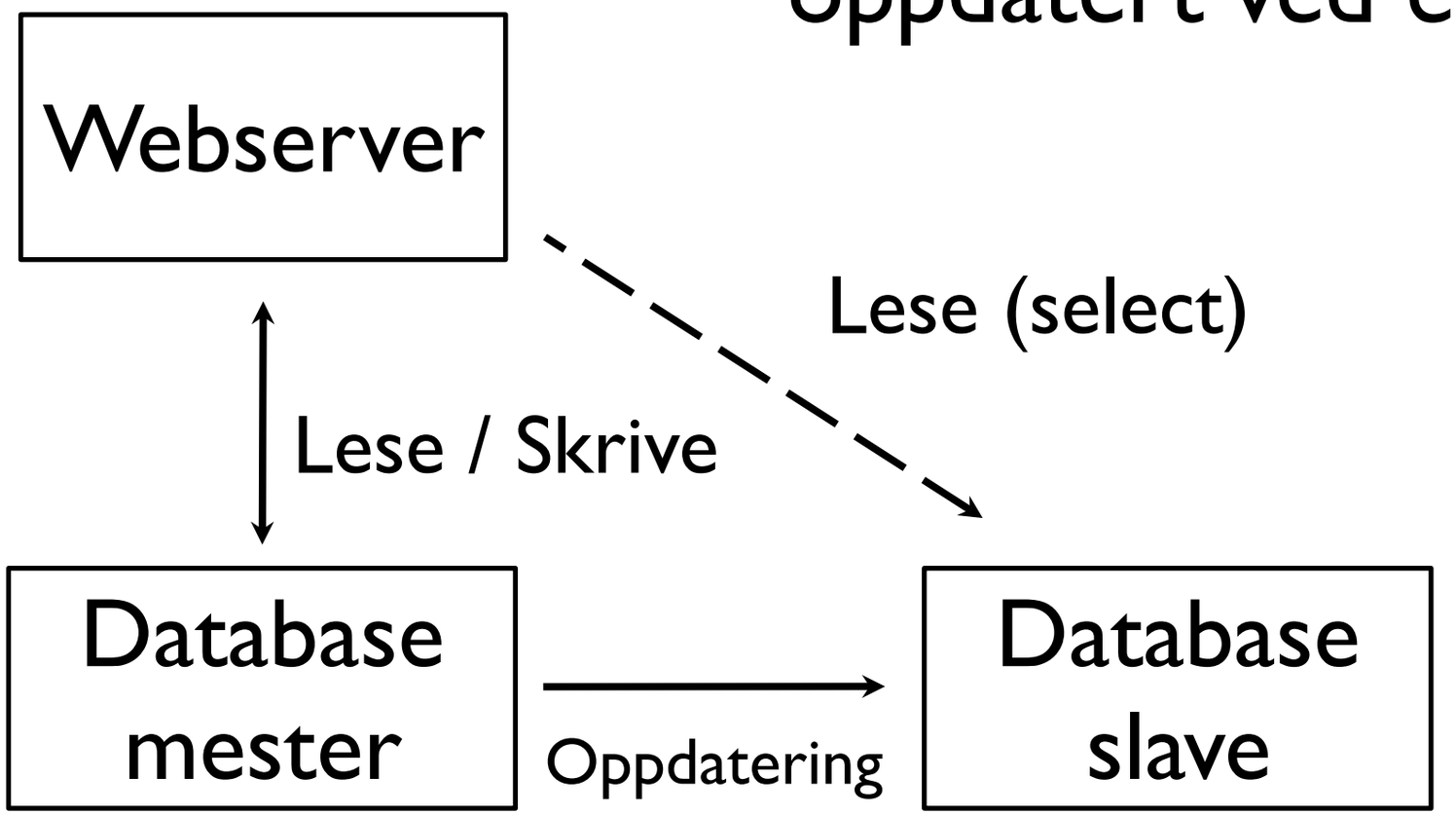
- db1 kan slite under press
- Lite minne kan føre til at mysql dør
- Mange søk tar veldig lang tid
- Når db1 går ned funker ikke websidene lenger

Mulig løsning

- En løsning ville vært å flytte noe av belastningen vekk fra db I
- Men hva kan flyttes ut og hva må beholdes?

Replikering

Det opprettes en eller flere kopier av dataene som blir oppdatert ved endringer.



Fordeler

- Økt ytelse fordi man fordeler belastning
- Mer robusthet, siden slaven er oppe og kan brukes selv om mesteren er nede
- Backup kan kjøres fra en slave
- Kan gjøres i ettertid på en produksjonsserver (under visse forutsetninger)

Utfordringer

- Krever konstant synkronisering mellom mester og slave
  - Hvor langt bak er slaven?
- Mer komplisert oppsett
- All skriving må allikevel til ett sted
  - Kan bli en flaskehals
- Kan lett gjøre småfeil under oppsett

Replikering i MySQL

- En MySQL server kan settes opp til å bli enten en mester eller en slave
  - Mesteren er typisk hoveddatabasen og ofte den opprinnelige
  - Slaven kan kobles til en eksisterende database

Teknisk design

- Replikering henger tett sammen med binær-log funksjonaliteten i MySQL
- Mesteren dumper i praksis all informasjon som den også skriver til binær-log til slaven og den gjentar endringene lokalt
  - Det betyr at det er en konstant forbindelse mellom mester og slave
  - Dersom slaven faller fra, vil den huske siste posisjon i logfilen som den leste fra og få alle oppdateringene

Implementering

- MySQL bruker tre mysql tråder for hver replikering:
  - Slaven har en "IO" tråd, som kobler seg mot mesteren og ber om å få sent binlog oppdateringer
  - Mesteren lager en "Binlog Dump" tråd som sender oppdateringene
  - Slaven's IO-tråd skriver endringene til en relay-log. Denne blir lest av en "SQL" tråd som oppdaterer den lokale databasen
- Man kan sjekke tråder og deres status med kommandoen `SHOW PROCESSLIST\G`

MySQL mester

- Må gi REPLICATION SLAVE, REPLICATION CLIENT rettigheter til en (eller flere) MySQL brukere
- Vet ikke hvor mange slaver som finnes
- Vet ikke hvor synkronisert en slave er
- Lager én tråd/forbindelse per slave

MySQL slave

- Man kan sjekke tilstand med SHOW SLAVE STATUS\G
  - Denne informasjonen finnes i filene relay-log.info og master.info
    - Det er IO tråden som oppretter relay-log.** og vedlikeholder master.info
    - SQL-tråden vedlikeholder relay-log.info
    - Alle disse filene ligger under /var/lib/mysql (IKKE /var/log/mysql/)

Oppsett

- Oppsett av replikering har flere trinn:
  - Mesteren må gis en unik ID
  - Man tar live backup av databasen og noterer ned posisjon i binlog
  - Man oppretter en mysql bruker som har lov til å replikere
  - Slaven må gis en unik ID
  - Man initialiserer databasen på slaven vha. backup fra mester
  - På slaven settes posisjonen til mesterens binlog fil, samt påloggings info
  - Slave trådene startes

Mise en Place

- Nå kan det være lurt å tenkte på fremgangsmåten en gang ekstra før man setter igang
  - Opprett flere putty / ssh vinduer så man er inne på alle maskinene som er viktig
  - Man burde også gå ut av produksjon for å forhindre at databasen endres underveis
    - Ikke et absolutt krav...

På Mester, del 1

- I my.conf

```
server-id                = 1
# bruk det riktige navnet på bookface db:
binlog_do_db             = bf
```

- Restart server

```
service mysql restart
```

På Mester, del 2

- Kjør følgende i mysql (bruk riktig IP!):

```
grant REPLICATION SLAVE on *.* to 'repl'@'10.0.0.6' identified by "replicateme";
grant REPLICATION CLIENT on *.* to 'repl'@'10.0.0.6' identified by "replicateme";
flush privileges;
use bf;
flush tables with read lock;
show master status;
```

- Noter ned bin-log filen og posisjon!
- Ta backup av databasen fra shellet (eller kjør backup scriptet):

```
mysqldump -u root -p --opt bf > bf.sql
```

- Log inn i mysql igjen på db1 og fjern lås

```
unlock tables;
```

På Slave, del I

- I my.conf

```
server-id                       = 5
log_bin                 = /var/log/mysql/mysql-bin.log
binlog_do_db                    = bf
bind-address                    = 0.0.0.0
report-host                     = db2
```

- Restart mysql

På Slave, del 2

- Opprett databasen:
  `create database bf;`
- Hent dump fra db1 og set inn i databasen:
  `cat bf.sql | mysql -u root -p bf`
- Koble slaven til serveren (i mysql)

```
slave stop;
CHANGE MASTER TO MASTER_HOST='10.0.0.5',
MASTER_USER='repl', MASTER_PASSWORD='replicateme',
MASTER_LOG_FILE='mysql-bin.000???',
MASTER_LOG_POS=??;

slave start;
```

Sjekk resultat

- På begge maskinene:
  `show processlist\G`
- Må slaven:
  `show slave status\G`
- På mesteren:
  `show slave hosts\G`
- Først når vi går inn i produksjon og ser at antall brukere etc endrer seg kan vi se om det fungerer på ekte

Databaseklynger

- I en del tilfeller kan det være nyttig å ha flere maskiner koblet sammen som likeverdige medlemmer i en klynge
  - I en klynge (cluster) er alle "mestre" utad, men de har allikevel en intern sjef
  - Det er mest vanlig å ha et oddetall antall servere i en klynge, typisk 3 eller 5 eller kanskje mer.

MariaDB og klynger

- Etter versjon 10 av MariaDB har støtte for klynger blitt sterkt forbedret
- MariaDB bruker Galera som synkroniseringsmotor mak kulissene
- Den lager også avtaler internt om hvilken ID som skal inkrementeres, dersom man velger AUTOINCRMENT i SQL koden
  - F.eks db1 vil lage id'er 1, 4, 7. db2 har lager 2, 5, 8 osv

Klynger Vs Replikering

- Klynger har generelt høyere krav til rask responstid mellom serverne, replikering har ingen slike krav
- Klynger kan øke tilgjengeligheten, ved at en node kan gå ned i en klynge og klienter kan gjøre endringer på andre servere. I et mester / slave oppsett vil man kun lese fra slavene hvis man mister en mester

Oppsett av databseklynge

- Mise en place:
  - Lag tre servere: db1, db2 og db3
  - Vi kommer til å bruke en eksisterende lastbalanserer til å fordele trafikken
  - De tre maskinene må ha satt opp respektive og egne IP adresser i /etc/hosts, det betyr at de må kunne pinge hverandre
  - De neste trinnene antar Ubuntu 14.04

Trinn 1: Sette opp pakkekilder

- Følgende kommandoer kjøres på alle tre serverne:

```
sudo apt-get install python-software-properties
apt-key adv --recv-keys --keyserver hkp://keyserver.ubuntu.com:80 CBCB082A1BB943DB
add-apt-repository 'deb [arch=amd64,i386] http://mariadb.biz.net.id/repo/10.1/ubuntu trusty main'
apt-get update
apt-get -y install mariadb-server
```

Trinn 2: Ekstra konfigurasjonsfil

- På alle tre servere lager man en ny konfigurasjonsfil:
  /etc/mysql/conf.d/cluster.cnf
- Filen er *litt* forskjellig på hver server stedet er vist med farger og ekstra kommentar

Trinn 2: Ekstra konfigurasjonsfil forts.

```
[mysqld]
# Cluster node configurations
wsrep_cluster_address="gcomm://db1,db2,db3"
# Make sure this is corresponds to hostname:
wsrep_node_address="db1"
innodb_buffer_pool_size=600M

# Mandatory settings to enable Galera
wsrep_on=ON
wsrep_provider=/usr/lib/galera/libgalera_smm.so
binlog_format=ROW
default-storage-engine=InnoDB
innodb_autoinc_lock_mode=2
innodb_doublewrite=1
query_cache_size=0
bind-address=0.0.0.0

# Galera synchronisation configuration
wsrep_sst_method=rsync
```

Trinn 3: Start db1 i oppstartsmodus

- db1 startes i en spesiell modus hvor andre servere kan melde seg på klyngen.
- Søreg først for at mysql er av:
  ```
  service mysql stop
  ps aux | grep mysql
  ```
- Start mysql i bootstrap modus
  ```
  service mysql bootstrap
  ```

Trinn 4: Start mysql på db2 og db3

- På db2 og db3 er vi nå klare til å koble oss til klyngen:
  service mysql stop
  service mysql start
- Noen ganger får man feilmeldinger her om en debianbruker som mangler. Ignorer dette.

Trinn 5: Start mysql på db1 i normal modus igjen

- Når klyngen er fulltallig, kan man skru av mysql på db1 og starte den på normalt vis igjen. De andre serverne vil holde klyngen oppe imens
  - På db1, stop tjenesten og vent til den faktisk er borte:

    ```
    service mysql stop
    ps aux | grep mysql
    ```
  - Start den igjen:

    ```
    service mysql start
    ```

Trinn 6: Sett opp bookface bruker

- Gå inn i mysql lokalt fra db1 og opprett en bruker til bookface som tidligere (sørg for riktige rettigheter)
- Det skal nå være mulig å koble seg på hvem som helst av de databasene fra en av webserverne

Trinn 7: Flytt over data

- Man kan nå flytte over bookface databasen fra den originale db1 til klyngen
- Kommandoen mysqldump kan være nyttig, men hva skal man ta med seg?

Trinn 8: Test ny databaseaddresse

- For å se om bookface fungerer som den skal, kan man midlertidig sette inn en IP addresse til f.eks nye db1 i config.php
- Dette skaper ikke redundans, men er en god test før man går videre

Trinn 9: Legg til lastbalansering av databasene i haproxy

- I haproxy.cfg på lastbalansereren kan man nå legge til følgende frontend og backend

```
frontend db
   bind 0.0.0.0:3306
   default_backend dbcluster


backend dbcluster
      mode tcp
      balance roundrobin
      server db1 DB1-IP:3306 check
      server db2 DB2-IP:3306 check
      server db3 DB3-IP:3306 check
```

Trinn 10: Restart haproxy og test

- HAproxy restarter man slik
  `service haproxy restart`
- Sjekk på lastbalansereren at porten 3306 lyttes til:
  `netstat -anltp`
- Man kan også teste med å bruke mysql kommandoen fra en av webservermaskinene
  `mysql -h lastbalanserer-ip -u bookfaceuser -p bookface`

Trinn 11: La bookface peke mot lastbalansereren

- Det siste trinnet er å endre database addressen i config.php
- Nå skal webserverne bruke alle tre databasene om hverandre
- Sjekk i status-siden til lastbalansereren at det er aktivitet på alle tre