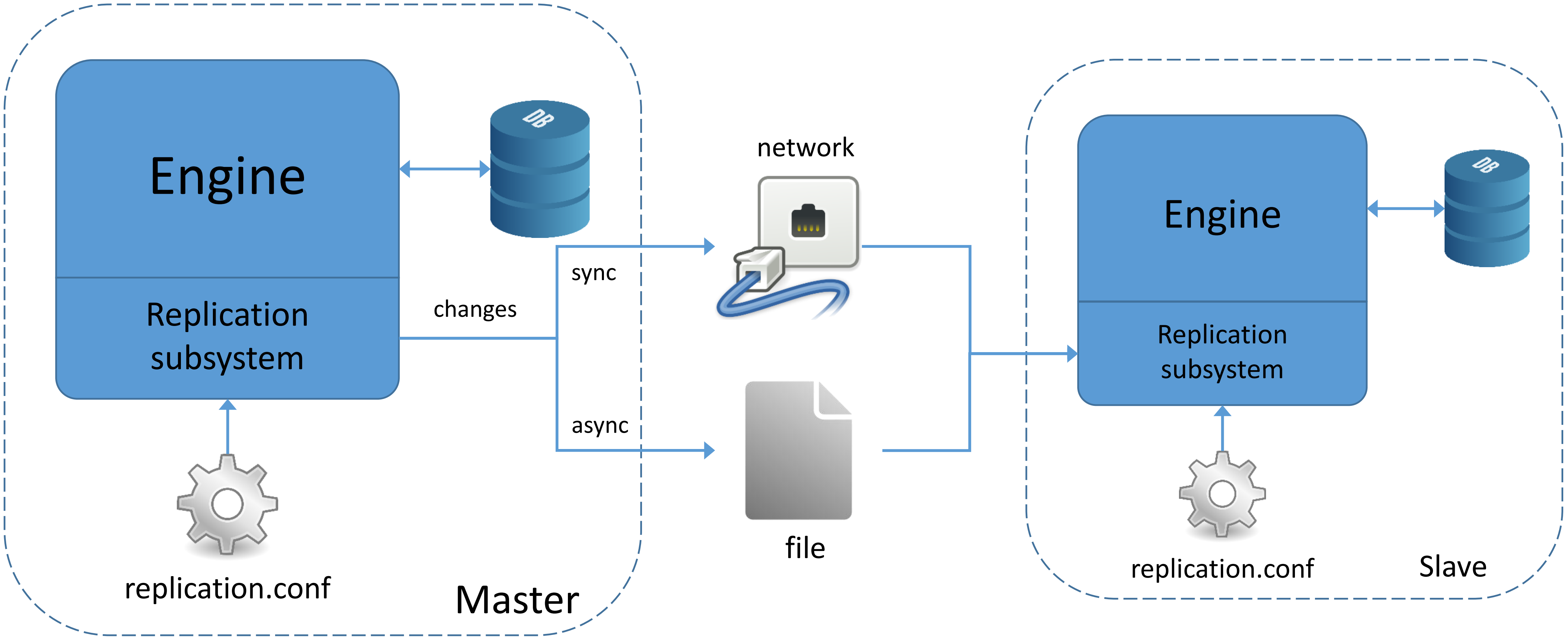
HA cluster based on Red Database engine-level replication

Dmitry Starodubov, head of department

RED SOFT


Engine
Replication subsystem
changes
sync
async
replication.conf
Master
network
file
Engine
Replication subsystem
replication.conf
Slave

REDSOFT

Replication events:

- Attachment start and finish
- Transaction start, commit and rollback
- Savepoint start, release and rollback
- Record insert, update and delete
- Generator change

Possible conflicts on the slave:

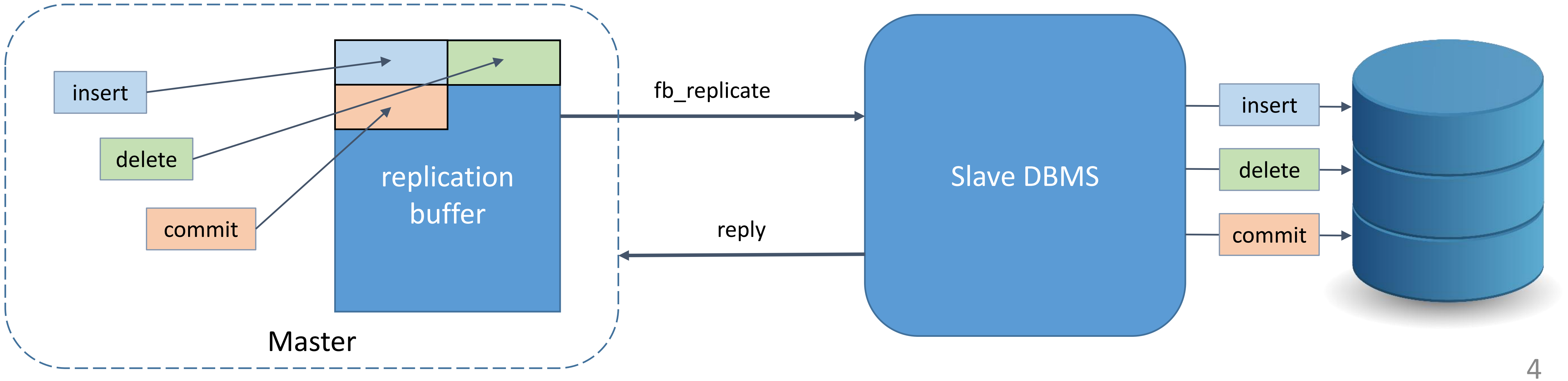
- Record is inserted and it already exists
- Record is updated or deleted and it’s absent
- Transaction is started and it already exists
- Transaction is finished and it’s absent

DML conflicts resolution (master priority):

- Inserted record exists – update it
- Updated record does not exist – insert it
- Deleted record does not exist – ignore it

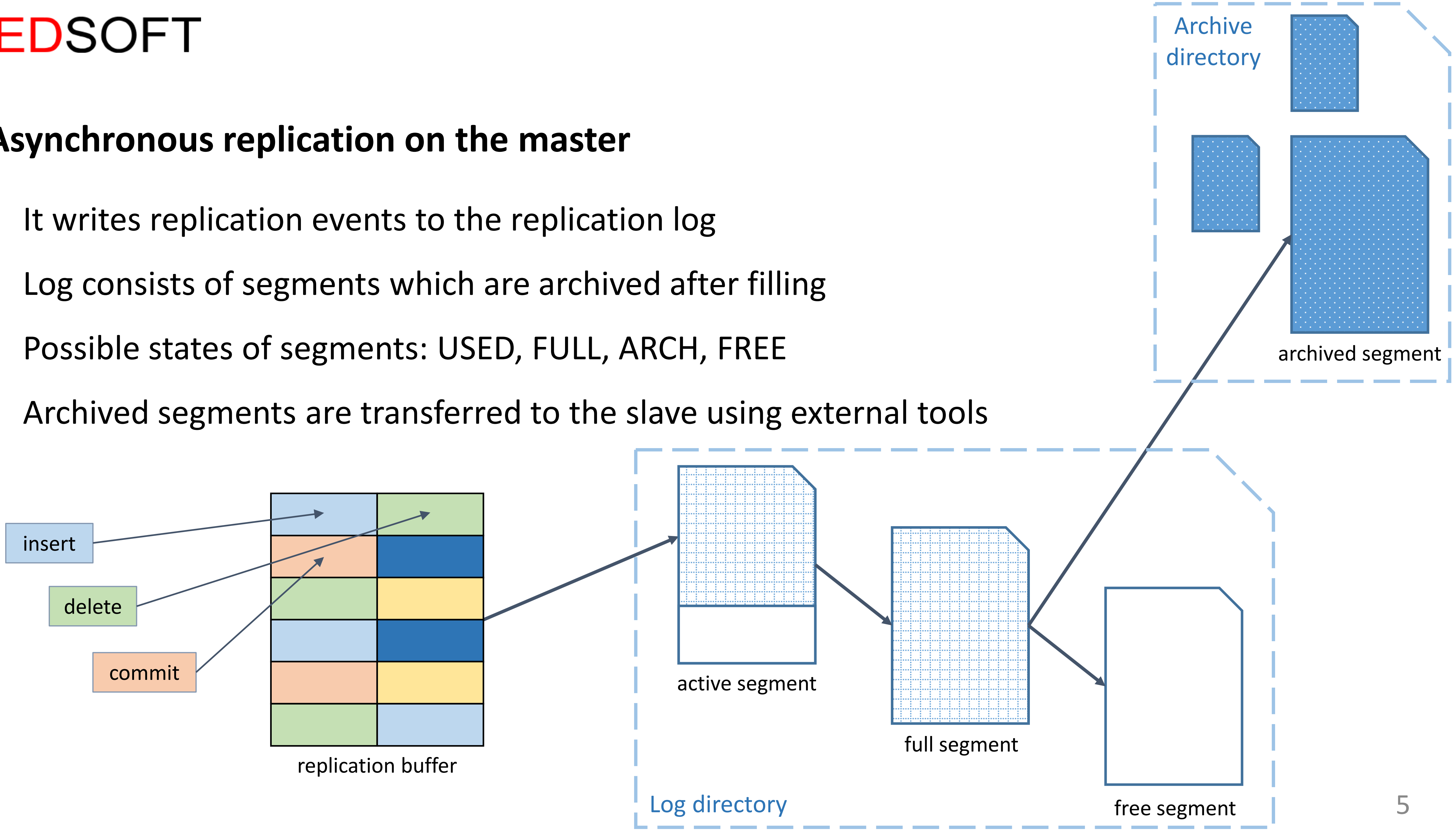
Synchronous replication

- Connects to the slave database over the network
- Sends buffers of replication events to the slave and waits for reply
- Replica is up to date at the cost of some delays
- Can be used to create high availability cluster



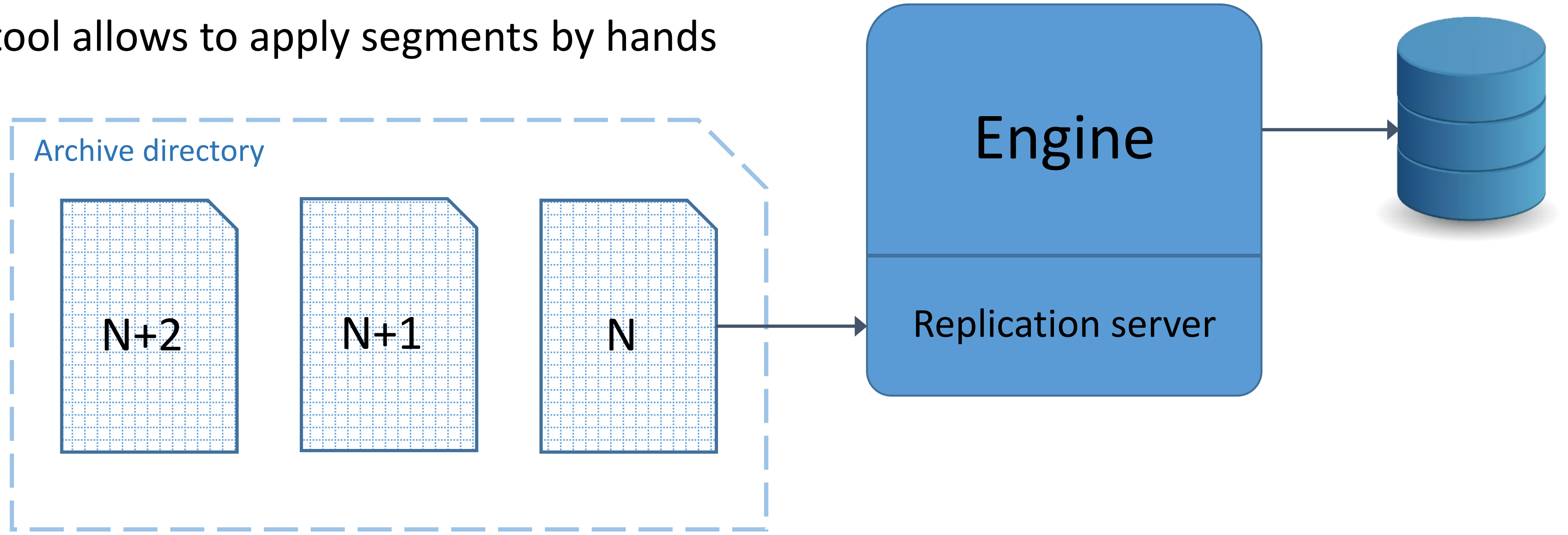
Asynchronous replication on the master

- It writes replication events to the replication log
- Log consists of segments which are archived after filling
- Possible states of segments: USED, FULL, ARCH, FREE
- Archived segments are transferred to the slave using external tools



Asynchronous replication on the slave

- Segments have sequential numbers and are applied in order of creation
- Superclassic and super have separate thread for applying logs
- Classic needs special instance of superclassic with “-r” switch
- fblogmgr tool allows to apply segments by hands



Configuration options

- Configuration (replication.conf) is read at connection startup
- It consists of several sections:
  - **<database>** – contains the default parameters
  - **<database path_to_database>** – contains the parameters for the specified master
  - **<replica path_to_database>** – contains the parameters for the asynchronous slave
- Supported parameters:
  - **buffer_size** (**1MB**) – size of the buffer used to accumulate replication events
  - **disable_on_error** (**false**) – replication error causes the master to stop replication
  - **compress_records** (**false**) – replicated records are RLE-compressed before transmission

Configuration options

- **master_priority** (false) – conflicting records in the target database are modified to match records in the master database
- **include_filter** – SIMILAR-TO regular expression that defines what tables must be included into replication
- **exclude_filter** – pattern that defines what tables must be excluded from replication
- **exclude_without_pk** (false) - tables without unique index excluded from replication
- **alert_command** – program that is executed when the critical replication error happens
- **log_directory** – directory to store log files of asynchronous replication
- **log_file_prefix** – prefix for replication log file names

REDSOFT

Configuration options

- **log_segment_size** (16 MB) – maximum allowed size for a single replication segment
- **log_segment_count** (8) – maximum allowed number of full replication segments
- **log_archive_directory** – directory for the archived log files
- **log_archive_command** – program that is executed to archive full replication segment.

  Supported variables: $(logfilename), $(logpathname), $(archfilename), $(archpathname)

Example for Linux:

```
test ! -f $(archpathname) && gzip --fast -c $(logpathname) > $(archpathname)
```

Example for Windows:

```
copy $(logpathname) $(archpathname)
```

Configuration options

- **log_archive_timeout** (**0**) – timeout, in seconds, to wait when log_segment_count replication segments are marked as full and scheduled for archiving

- Parameters only for synchronous master
  - **replica_database** – connection string to the replica database for synchronous replication

  Format: `[<login>:<password>@]<database connection string>`

- Parameters only for asynchronous slave
  - **owner_auth** – authentication credentials for the database owner (<login>:<password>)
  - **log_directory** – directory for the log files to be applied
  - **master_database** – connection string to the master database

Configuration options

- **db_copy_command** – program that is executed to recreate the replica as a copy of the master database. Supported variables: $(masterdb), $(masteruser), $(masterpwd), $(replicadb), $(replicauser), $(replicapwd), $(guid)

Example for Windows:

```
del d:\db\repl.fdb &&
nbackup -u $(masteruser) -p $(masterpwd) -b 0 $(masterdb) d:\db\repl.fdb &&
nbackup -f d:\db\repl.fdb &&
gfix -user $(replicauser) -pas $(replicapwd) -replica $(guid) d:\db\repl.fdb &&
move /Y d:\db\repl.fdb $(replicadb)
```

Example for Linux:

```
export PATH=/opt/RedDatabase/bin:$PATH &&
rm -f /db/repl.fdb &&
nbackup -u $(masteruser) -p $(masterpwd) -b 0 $(masterdb) /db/repl.fdb &&
nbackup -f /db/repl.fdb &&
gfix -user $(replicauser) -pas $(replicapwd) -replica $(guid) /db/repl.fdb &&
mv -f /db/repl.fdb $(replicadb)
```

REDSOFT

Replication setup

- Finish all attachments to the master database
- Copy master database to the slave host
- Activate replication mode for the slave database: `gfix -replica {<master_GUID>}`
- Setup replication.conf:
  - for synchronous replication define section **<database path_to master>** with one or more parameters **replica_database**
  - for asynchronous replication on the master define section **<database path_to_master>** with parameters **log_directory**, **log_archive_directory** and **log_archive_command**
  - for asynchronous replication on the slave define section **<replica path_to_slave>** with parameters **owner_auth and log_directory**

REDSOFT

Example of asynchronous replication setup

- On the master:

```
<database d:\db\r_master.fdb>
      log_directory d:\db\rlog
      log_segment_size 163840
      buffer_size 4096
      log_segment_count 6
      log_archive_directory d:\db\rlog_a
      log_archive_command "copy $(logpathname) $(archpathname)"
</database>
```

- On the slave:

```
<replica d:\db\r_slave.fdb>
      owner_auth sysdba:masterkey
      log_directory d:\db\rlog_a
</replica>
```

Information about replication status

- GSTAT –H output:

```
gstat -h d:\db\R_SLAVE.FDB
Database header page information:
…
        Attributes                    force write, replica

    Variable header data:
        Database GUID:  {6C81FDE1-9978-417C-11BD-FFA63E5AA6A0}
        Replication master GUID:        {6C81FDE1-9978-417C-11BD-FFA63E5AA6A0}
```

REDSOFT

Information in MON$REPLICATION table

MON$TYPE	replication mode: 1 – sync master, 2 – async master, 3 – slave
MON$CONNECTION_STRING	replica connection string or replication log directory
MON$ACTIVE	1 – replication is active, 0 – disabled, N – sequential number for async master
MON$LAST_MODIFIED	last activity time of replication: sending or receiving packet, writing to log
MON$WAITFLUSH_COUNT	count of replication buffer flushes
MON$WAITFLUSH_TIME	total time of replication buffer flushes
MON$WAITFLUSH_TRANSFER	total size of replication buffer flushes
MON$BACKGROUND_COUNT	count of replication buffer flushes in background
MON$BACKGROUND_TIME	total time of replication buffer flushes in background
MON$BACKGROUND_TRANSFER	total size of replication buffer flushes in background

REDSOFT

Command line tools: fblogmgr

Usage:

-D[atabase] <database>	: database name
-U[ser] <username>	: user name
-P[assword] <password>	: password
-A[rchive] <sequence>	: archive specified segment
-A[rchive] all	: archive all segments
-F[orce]	: force archiving half-full segments
-C[onfig]	: print logging configuration
-S[egments]	: enumerate log segments
-Z	: report version

```
fblogmgr.exe -d d:\db\R_MASTER.FDB -u sysdba -p masterkey -s

Log status:
   Current sequence: 23
   Last modified: 2016-09-29 12:10:59
   Total log size: 0 bytes in 2 segments
   Free segments: 2, full segments: 0, archived segments: 0

Available log segments:
   File name: r_master.fdb.log-000
   Sequence: 23
   State: free
   Size in use: 0 bytes

   File name: r_master.fdb.log-001
   Sequence: 20
   State: free
   Size in use: 0 bytes
```

Command line tools: fbreplmgr

Usage: <command> [<options>] <replica>

Commands:

-A[pply]	: apply logs to replica
-C[reate]	: create/copy from master
-S[tatus]	: report status

Options:

-U[ser] <username>	: user name
-P[assword] <password>	: password
-V[erbose]	: verbose output
-Z	: report version

```
fbreplmgr.exe -u sysdba -p masterkey -s d:\db\R_SLAVE.FDB
Status for replica d:\db\R_SLAVE.FDB:
        Master database: d:\db\r_master.fdb
        Master GUID: {6C81FDE1-9978-417C-11BD-FFA63E5AA6A0}
        Archive directory: d:\DB\rlog_a\
        Control file: d:\DB\rlog_a\{6C81FDE1-9978-417C-11BD-
FFA63E5AA6A0}
        Current segment: 24
        Oldest segment: absent
        Total segments in the queue: 0
```

REDSOFT

Command line tools: fbrepldiff

Usage:

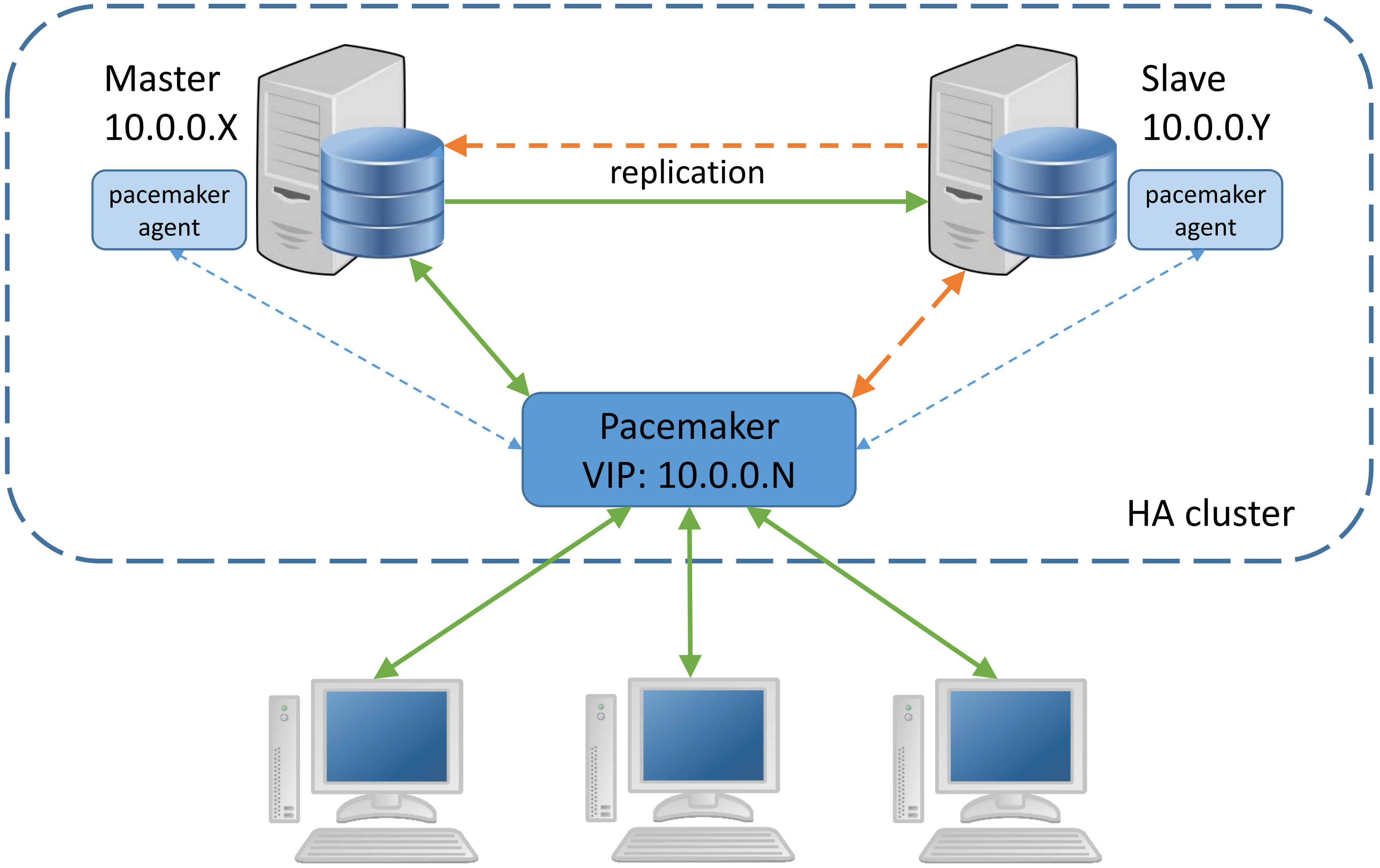
```
-D[atabase] <database>   : master database name
-R[eplica] <database>    : replica database name
-U[ser] <username>       : user name
-P[assword] <password>   : password
-C[onfig] <config>       : configuration file
-M[etadata]     : compare only metadata
-S[ynchronize]  : wait until master and slave synchronize
-T[imeout]      : timeout to wait for synchronization
-V[erbose]      : verbose output
-Z              : report version
```

Comparison:

- Read all tables falling under configuration filters and having PK
- If list of tables does not match - ERROR
- If primary keys for table do not match – ERROR
- Read all records in the tables with the same names sorted by PK
- If types of record fields do not match – ERROR
- If record lengths do not match – ERROR
- If fields content does not match – ERROR
- Read all BLOB record fields
- If BLOB lengths do not match – ERROR
- If BLOBs content do not match – ERROR
- If all the above checks were finished without error then return OK

REDSOFT

High availability cluster based on replication and pacemaker



REDSOFT

Example of replication configuration

- Enabled both sync and async replication
  - Synchronous replication is used to create the HA cluster
  - Asynchronous logs are transferred to the head department that uses replica for OLAP queries

```
<database /cluster/db/ncore-fssp-YY.fdb>
        replica_database 10.XX.4.11:ncore-fssp
        disable_on_error false
        compress_records false
        master_priority true
        exclude_without_pk true
        log_directory /cluster/journal/replication
        log_file_prefix ncore-fssp-YY
        log_segment_size 104857600 # 100MB
        log_segment_count 4
        log_archive_directory /cluster/journal/archive
        log_archive_command "/usr/bin/lzop -1 $(logpathname) -o $(archpathname).tmp &&
                             mv $(archpathname).tmp $(archpathname).lzo"
        log_archive_timeout 0
</database>
```

Example of cluster setup

- Create cluster that includes two nodes

```
# pcs cluster setup --name RDB_cluster server-edo rbdXX
```

- Create resource for xinetd service

```
# pcs resource create GDS_DB ocf:heartbeat:Xinetd service=firebird
# pcs resource clone GDS_DB
```

- Create resource for database

```
# pcs resource create RDB ocf:redsoft:RDB \
  db_file=/cluster/db/ncore-fssp-YY.fdb \
  db_alias=ncore-fssp \
  exclude_without_pk=true \
  log_directory=/cluster/journal/replication \
  log_file_prefix=ncore-fssp-YY \
  log_segment_size=104857600 \
  log_segment_count=4 \
  log_archive_directory=/cluster/journal/archive \
  log_archive_command='"/usr/bin/lzop -1 $(logpathname) -o $(archpathname).tmp && \
                      mv $(archpathname).tmp $(archpathname).lzo“' \
  log_archive_timeout=0 –disable
```

Example of cluster setup

- Configure database resource as master/slave:

```
# pcs resource master RDB-master RDB master-max=1 master-node-max=1 clone-max=2 clone-
node-max=1 notify=true
```

- Assign master location to the specific host “server-edo”

```
# pcs constraint location RDB-master prefers server-edo=100
```

- Enable resource

```
# pcs resource enable RDB
```

- Setup resource checking interval

```
# pcs resource op add RDB monitor interval=30s role=Master
```

- Create virtual IP

```
# pcs resource create RDB-IP ocf:heartbeat:IPaddr2 ip="10.XX.4.243" nic="eth0" cidr_netmask="24‟
```

- Assign virtual IP with the master host

```
# pcs constraint colocation add RDB-IP with master RDB-master INFINITY
```

- Specify services start order

```
# pcs constraint order start GDS_DB-clone then start RDB-master
```

REDSOFT

Example of cluster status

- Show node attributes

```
# crm_mon -A1
Last updated: Tue Oct  4 10:33:25 2016
Last change: Sat Oct  1 22:13:22 2016
Stack: cman
Current DC: rbd46 - partition with quorum
Version: 1.1.11-97629de
2 Nodes configured
45 Resources configured


Online: [ server-edo rbd46 ]
...
 Clone Set: GDS_DB-clone [GDS_DB]
     Started: [ server-edo rbd46 ]
 Master/Slave Set: RDB-master [RDB]
     Masters: [ server-edo ]
     Slaves: [ rbd46 ]
 RDB-IP  (ocf::heartbeat:IPaddr2):   Started server-edo
...
Node Attributes:
* Node server-edo:
    + master-RDB                        : 20
* Node rbd46:
    + master-RDB                        : 10
```

- Connect to virtual IP (current master)

```
# ./isql -u sysdba -p ***** 10.46.4.243:ncore-fssp
Database:  10.46.4.243:ncore-fssp, User: sysdba
SQL> select * from mon$replication;

MON$TYPE   MON$CONNECTION_STRING               MON$ACTIVE
MON$LAST_MODIFIED                           MON$WAITFLUSH_COUNT
MON$WAITFLUSH_TIME                       MON$WAITFLUSH_TRANSFER
MON$BACKGROUND_COUNT                        MON$BACKGROUND_TIME
MON$BACKGROUND_TRANSFER
=======================================================
1          sysdba:*****@10.46.4.11/3050:ncore-fssp    1
2016-10-04 10:54:25.8240                              1
0                                                     5
1                                                    15
3367
```