Managing Replication Conflicts

Ludwig Krispenz, Brussels, Oct 20th

redhat.

Agenda

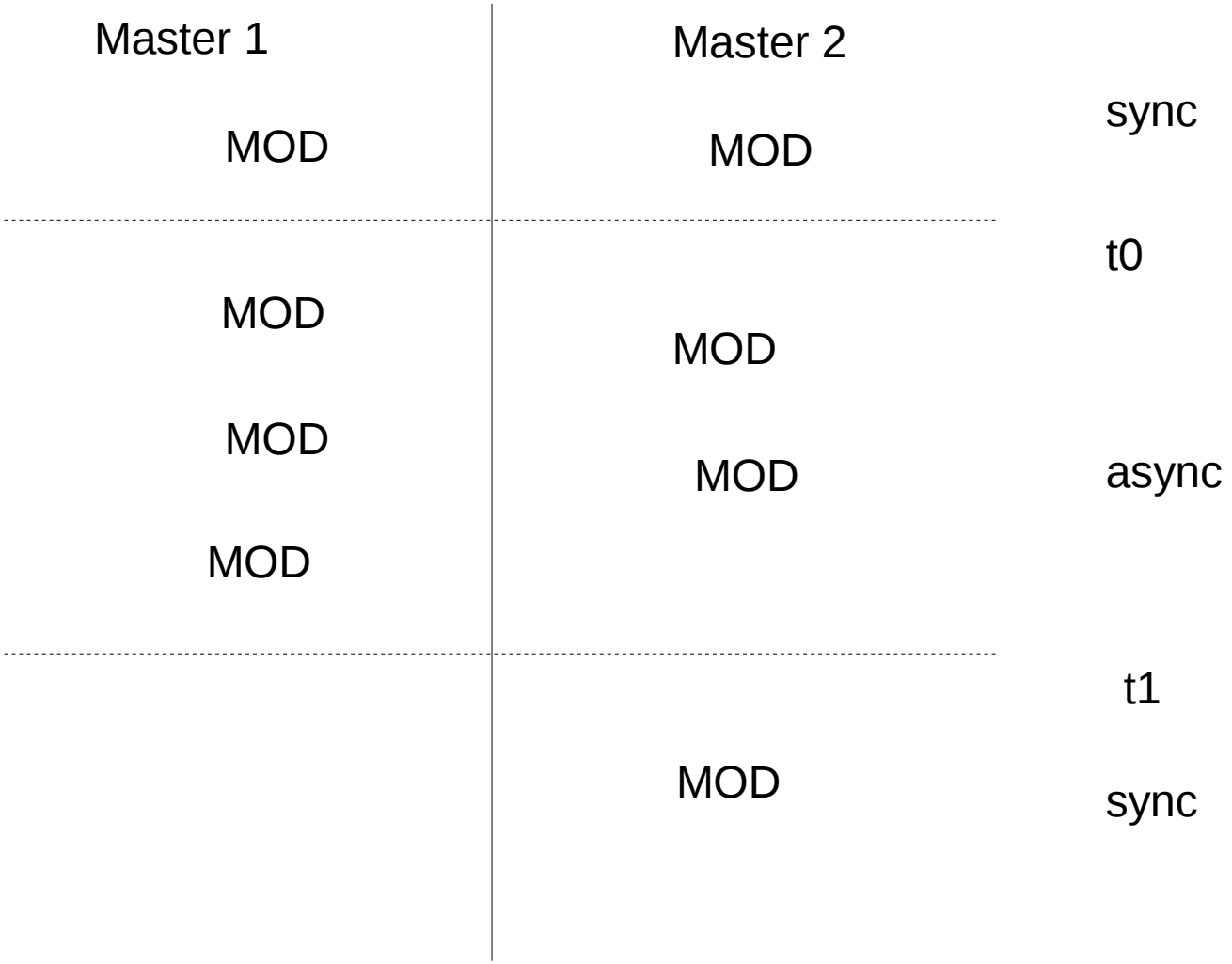
- Conflict scenarios
- Current solutions
- Problems
- New Conflict Resolution

Conflict scenarios: concurrent updates

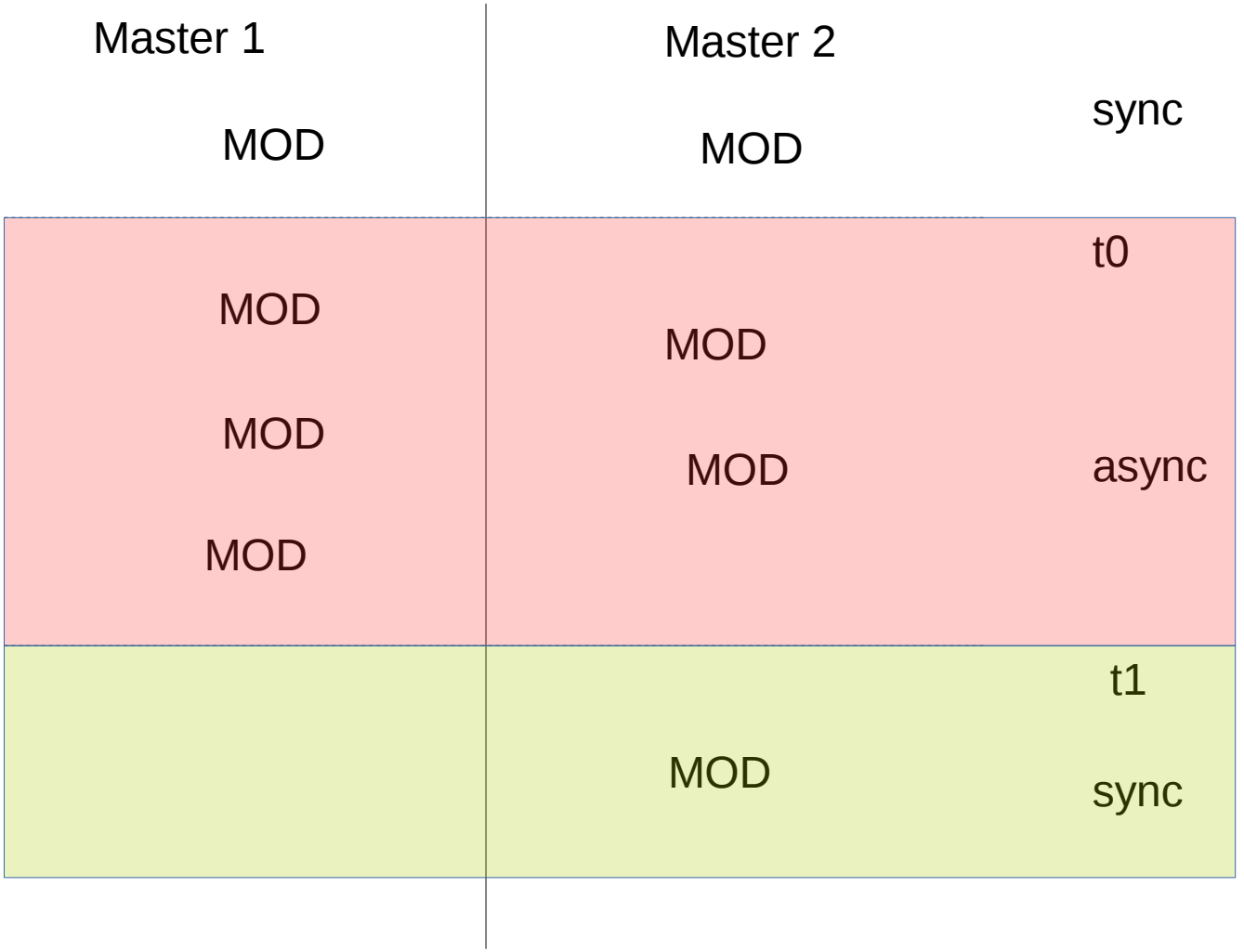
In a replication topology with loose consistency replication conflicts can occur if updates are applied concurrently* on different masters

** We talk about "performing operations at the same time" or "simultaneously" or "independently". This doesn't require the operations to be processed at exactly the same time, it means that operations on one server are performed before the operations on an other are received via replication. In the following examples we always assume that servers are in sync, operations are applied on severals servers before the next synchronization and then servers are synchronized again*

concurrency model (1)



concurrency model (2)



Potential conflicts

Conflict resolution

Conflict scenarios: simple conflict

- Add cn=s2 on master1 and and master2 at the same time
- Result (in previous 389-ds):

```
dn: cn=s2+nsuniqueid=0bc0130a-0a2e11e7-b268a04a-
                                    bdab994e,cn=test_0,dc=example,dc=com
dn: cn=s2,cn=test_0,dc=example,dc=com
```

Conflict Scenarios: orphaned entries

Time	Master 1	Master 2
t1	Del P	
t2		Add child C of P

Conflict scenarios: composite conflicts

t0		
t1	Add E	
t2		Add E
t3	Del E	
t4		

t0		Add E
t1	Add E	
t2		
t3	Del E	
t4		

t0		
t1	Add E	
t2		
t3	Del E	
t4		Add E

Conflict scenarios: conflicts with children(1)

- Add cn=p2 on master1 and and master2 at the same time
- Add child cn=c2 on master1 and cn=c3 on master2

Time	Master 1	Master 2
t1	Add P2	
t2		Add P2
t3		Add Child C3 of P
t4	Add child C2 of P	

Conflict scenarios: conflicts with children(2)

- Result (in previous 389-ds)

```
dn: cn=p2,cn=test_4,dc=example,dc=com

dn: cn=p2+nsuniqueid=8f79ffb5-...-4dccfc06,cn=test_4,dc=example,dc=com

dn: cn=c3,cn=p2+nsuniqueid=8f79ffb5-…..-4dccfc06,cn=test_4,dc=example,dc=com

dn: cn=c2,cn=p2,cn=test_4,dc=example,dc=com
```

Current solutions: 389-ds

- Simple conflict:
  - If two entries with the same dn are added concurrently, one is transformed into a conflict entry
  - On all masters the same entry will become a conflict
- Orphaned entries:
  - Resurrect the deleted entry and keep as glue entry
- Other cases: apply operation on the replica to the entry with the specified nsuniqueid

Current solutions: other vendors

OpenDJ (forgerock), UnboundID: *they create conflict entries by adding a unique identifier to the dn of the conflict entry. UnboundID claims that conflict entries are invisible.*

https://ping.force.com/Support/PingIdentityArticle?id=kA340000000PMwBCAW

https://bugster.forgerock.org/jira/browse/OPENDJ-454

Active Directory: *Active directory also has a similar procedure to deal with naming conflicts, creating conflict entries with unified dn*

https://social.technet.microsoft.com/wiki/contents/articles/15435.active-directory-duplicate-object-name-resolution.aspx

OpenLDAP: no explicit handling but creates inconsistencies ins some of the ADD-DEL or ADD-MODRDN scenarios

Problems with current solution

- Confusion
- Inconsistencies
- Plugin messup

Problems: plugins - memberof

- Simple conflict and memberof
- Add group cn=g1 on master1 and and master2 at the same time
- Result

```
dn: cn=g1+nsuniqueid=68bbc90a-...-4dccfc06,cn=test_2,dc=example,dc=com
member: cn=m2_1,cn=test_2,dc=example,dc=com

dn: cn=g1,cn=test_2,dc=example,dc=com
member: cn=m2_1,cn=test_2,dc=example,dc=com

dn: cn=m2_1,cn=test_2,dc=example,dc=com
memberOf: cn=g1+nsuniqueid=68bbc90a-...-4dccfc06,cn=test_2,dc=example,dc=
memberOf: cn=g1,cn=test_2,dc=example,dc=com
```

Problems: plugins – managed entry

- Simple conflict and managed entry
- Add cn=user2 on master1 and and master2 at the same time
- Result

```
dn: cn=user2,ou=managed_groups,dc=example,dc=com
mepmanagedentry: cn=user2,ou=managed_groups,dc=example,dc=com

dn: cn=user2+nsuniqueid=8f79...-4dccfc06,ou=managed_groups,dc=example,dc
mepmanagedentry: cn=user2,ou=managed_groups,dc=example,dc=com

dn: nsuniqueid=8f79...4dccfc06+uid=user2,ou=managed_people,dc=example,dc
mepmanagedby: uid=user2,ou=managed_people,dc=example,dc=com

dn: uid=user2,ou=managed_people,dc=example,dc=com
mepmanagedby: uid=user2,ou=managed_people,dc=example,dc=com
```

Problems: plugins

- Simple conflict and managed entry
- Delete managing conflict entry
- Result

```
dn: cn=user2+nsuniqueid=8f79f...-4dccfc06,ou=managed_groups,dc=example,dc
mepmanagedby: uid=user2,ou=managed_people,dc=example,dc=com

dn: uid=user2,ou=managed_people,dc=example,dc=com
mepmanagedentry: cn=user2,ou=managed_groups,dc=example,dc=com
```

Solution

- Define conflict resolution rules
  - Consistency rule
  - Correctness rule
  - Transparency rule
  - Best effort rule(s)
- Define conflict representation and necessary artifacts to resolve conflicts
- Apply rules

Conflict resolution rules(1)

consistency rule

After processing a given set of operations on all servers, the contents of the database on all servers has to be identical

Conflict resolution rules(2)

correctness rule

The result processing a given set of operations should be the same as if the set of operations were applied on a single server in the order they were originally received

Conflict resolution rules(3)

transparency rule

Operations which are rejected in the update resolution process have to be invisible to normal client operations .

Conflict resolution rules(4)

Best effort rules:

Unfortunately the correctness rule cannot be applied to all combinations of operations without extremely overhead in processing and maintaining replication state information.

Example:

Time	Master 1	Master 2
t1	Modrdn E → X	
t2		Mod E

Conflict resolution rules(5)

Best effort rules:

If the entry a modify operation has to be applied does not exist, attempt to apply the operation to the entry with the same nsuniqeuid as the entry where the operation was first applied to (sticky rule).

Solution (1)

- Hide conflict entries
  - Add objectclass ldapsubentry
- Track presence of entries by artefacts
  - Tombstone: represents a deleted entry
  - Cenotaph: represents a renamed entry
  - Track time an entry with a specific dn did exist

```
dn: cenotaphID=c6be880c-afe011e7-b42ae714-dd13d87b+cn=ax9,cn=test_1,dc=example,dc=co
objectClass: extensibleobject
objectClass: nstombstone
objectClass: top
cenotaphfrom: 59e05f35000000010000
cenotaphto: 59e05f37000000020000
nstombstonecsn: 59e05f37000000020000
cenotaphid: c6be880c-afe011e7-b42ae714-dd13d87b
cn: ax9
```

Solution (2)

- Enhance update resolution to get consistent results: example ADD (1)

In a replicated ADD we can have

Existing entry with nsuniqueid to be added: ex_nsuid_e
Existing entry with dn of entry to be added: ex_dn_e
Entry to be added: add_e
csn of the ADD operation: opcsn

Step 1: check if ex_nsuid_e exists
⇒ entry was already added, NOOP

Solution (3)

- Enhance update resolution to get consistent results: example ADD(2)

Step2: Check if conflicting cenotaphs or tombstones for DN(add_e) exist.

We need to check this even before checking id ex_dn_e exists. An existing ex_dn_e could have been added after an entry with the same dn has been deleted or renamed, this has to be checked first.

Case1: fromCSN < opcsn < toCSN

⇒ add_e has to become a conflict

Case2: a tombstone with opcsn < fromCSN < toCSN exists

⇒ resurrect the tombstone as conflict and turn Add_e directly into a tombstone

Case3: same as case2 for a cenotaph

Solution (3)

- Enhance update resolution to get consistent results: example ADD(3)

Step3: if ex_dn_e exists, check which is the newer
Case 1: add_e is newer
==> new entry becomes conflict
Case 2: ex_dn_e is newer
==> add add_e as valid entry
==> existing entry becomes conflict
==> move children of ex_dn_e to add_e

Step 4: ex_dn_e does not exist
⇒ we can add the entry,
but have to verify that also the parent exists or handle it: resolve parent

Iceberg Diagram for replication conflicts