Information Management

IBM

Informix Flexible Grid

Die Entwicklung von der HDR Replikation zum Flexible Grid

Gerd Kaluzinski

gerd.kaluzinski@de.ibm.com

Übersicht

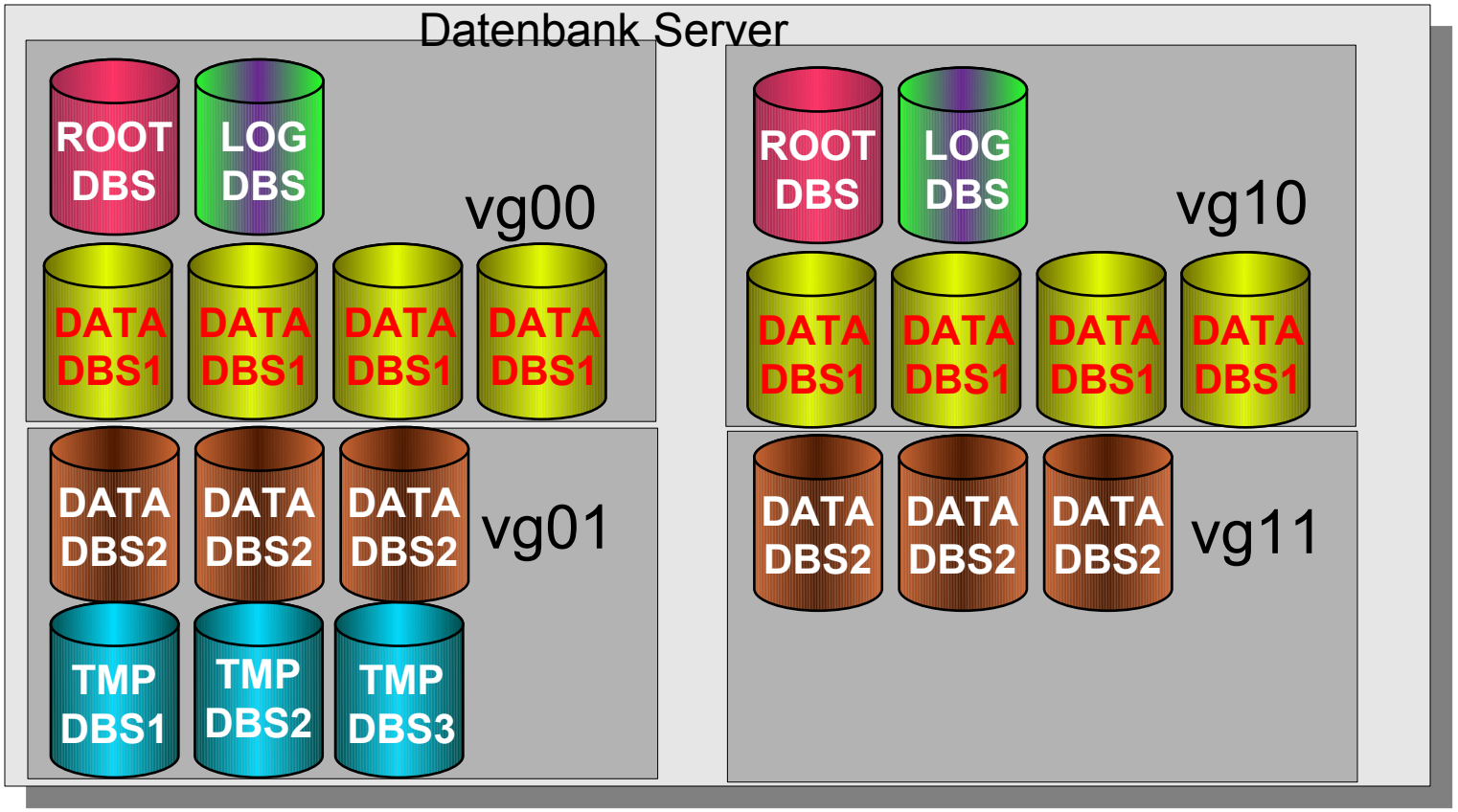
- Basis — Warum überhaupt Replikation ?
- Mirroring — Der Ursprung der Absicherung gegen Ausfälle
- HDR — Wenn ein Rechner allein nicht sicher genug ist
- RSS — Weit drunten im Bunker steht ein Backup ...
- SDS — Lastverteilung und Ausfallsicherheit
- ER/CDR — Unterschiedliche Versionen zur Absicherung
- Flexible Grid — Das kann die ER/CDR nun auch

Basis – Warum überhaupt Replikation ?

- Ausfallsicherheit
  - Plattenfehler
  - Memory Fehler
  - Logische Fehler
- Performance
  - Auswertung von Produktion trennen
  - Fester Datenbestand für Analyse
- Migration
  - Sicherheit durch zeitversetzte Migration

Mirroring – Der Ursprung der Replikation

- Ausfallsicherheit gegenüber Plattenfehlern
- Bessere Performance durch paralleles Lesen



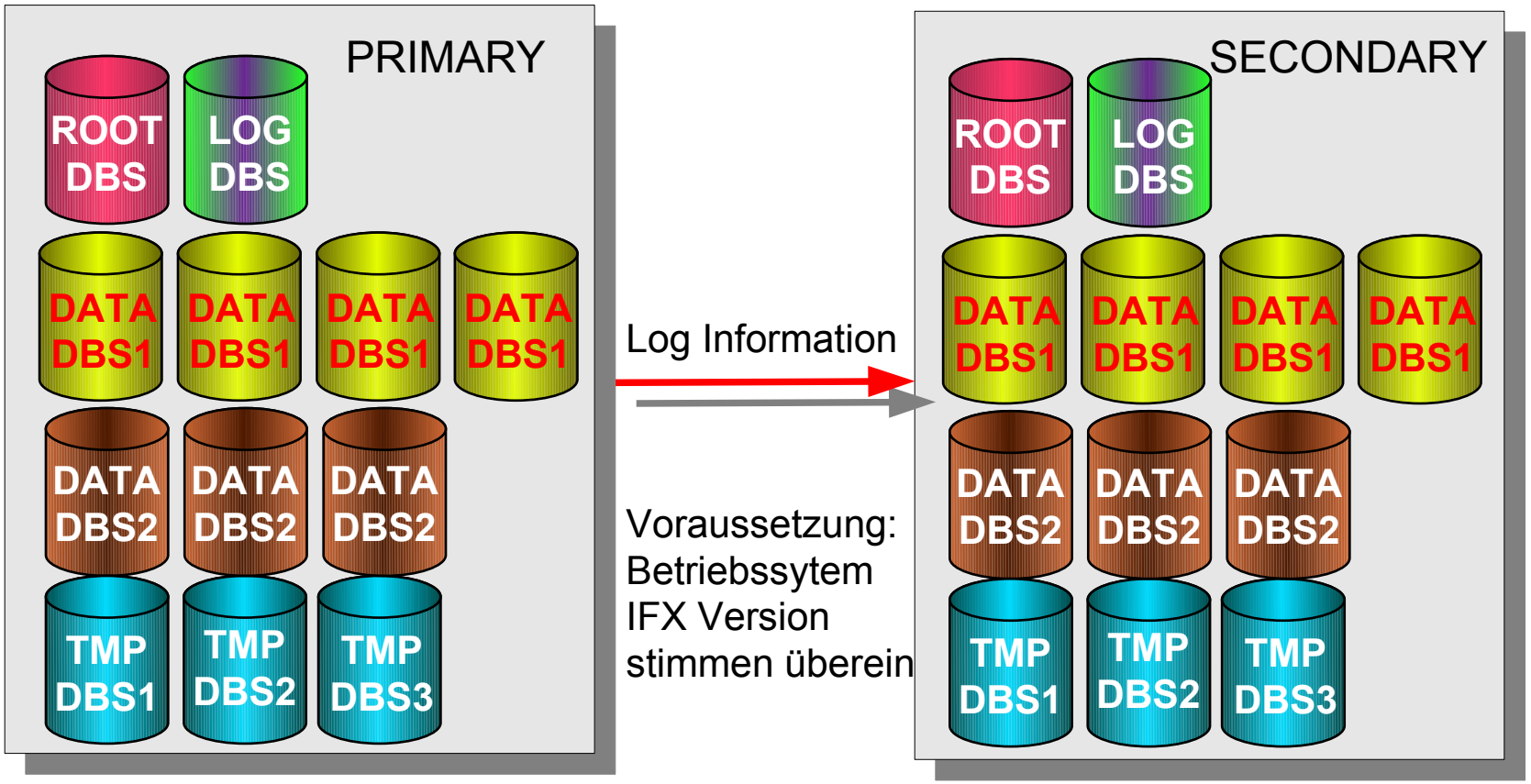
Mirroring – Auswirkung auf die Applikation ?

- Keine
  Die Applikation merkt nicht, ob die Datenbankinstanz Mirroring verwendet.

  DDL – Aufrufe wie “alter table” funktionieren unverändert.

HDR – Spiegelung einer gesamten Instanz

- Ausfallsicherheit gegenüber Serverausfall / RZ Ausfall
- Abspaltung der Auswertungen auf einen separaten Server



HDR – Auswirkung auf die Applikation ?

- Primary:

  Die Applikation merkt nicht, ob die Datenbankinstanz mittels HDR repliziert wird. DDL-Statements erlaubt.

- Secondary:

  Read-Only Applikationen können die Ressourcen nutzen.

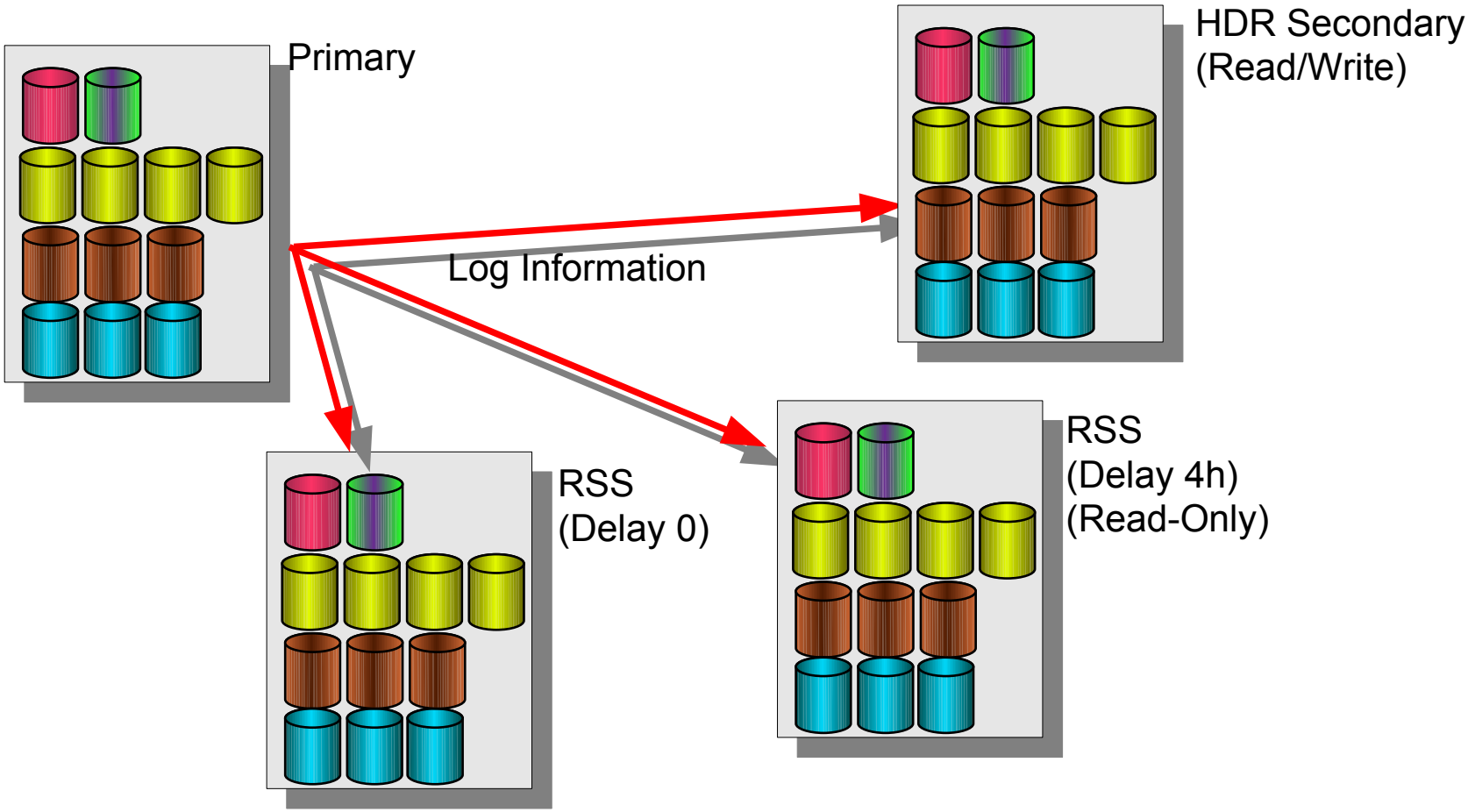
  Schreiben ist in lokale, temporäre Tabellen möglich

  Schreiben kann remote auf den Primary erfolgen, der dann die lokalen Tabellen füllt (nur für Ultimate Edition erlaubt).

DDL-Aufrufe sind am Primary erlaubt.
RAW Tables und Truncate sind verboten

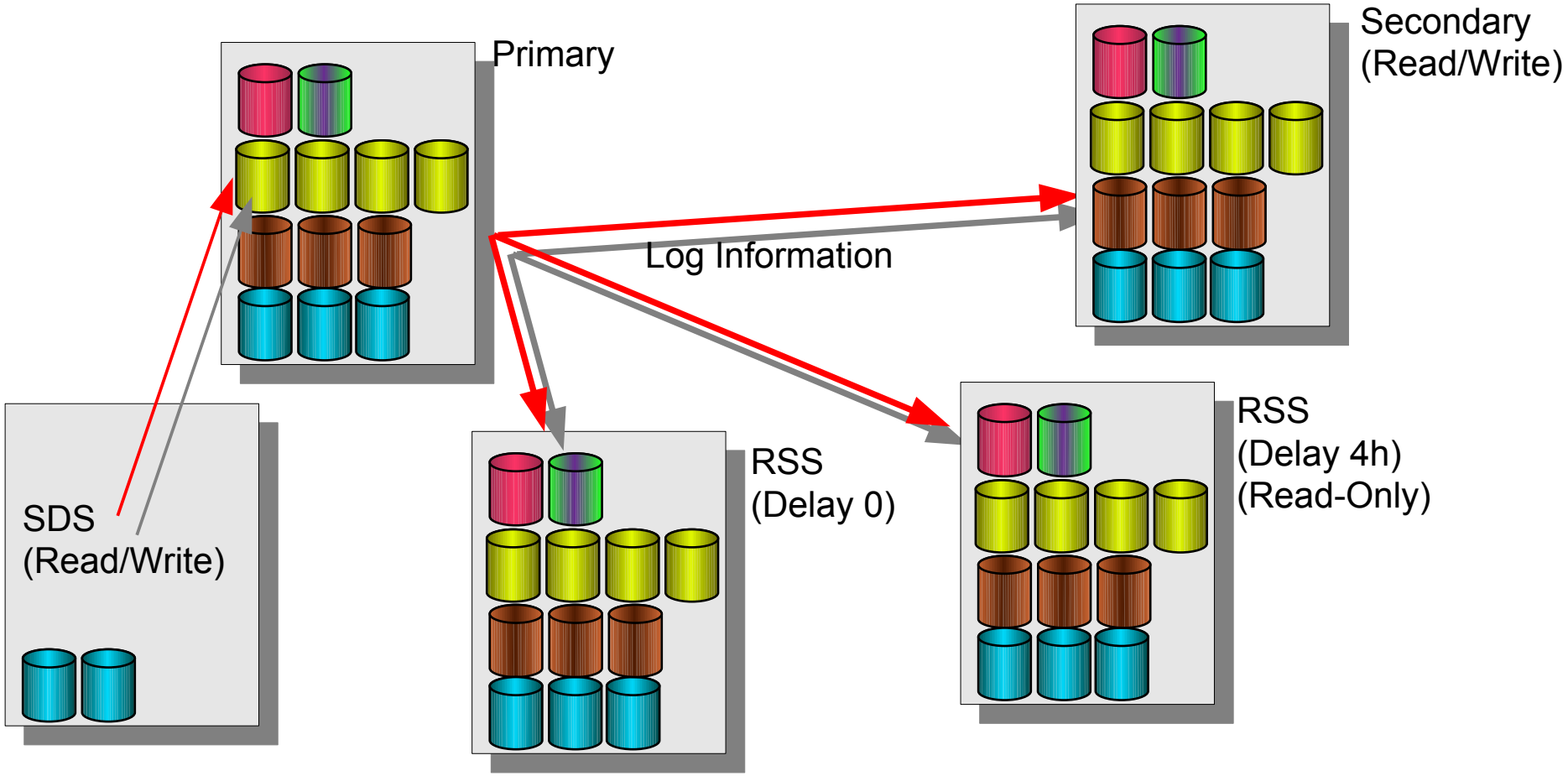
RSS – Spiegelung einer gesamten Instanz

- Ausfallsicherheit gegenüber Serverausfall / RZ Ausfall
- Delayed Apply / Read/Write am Secondary



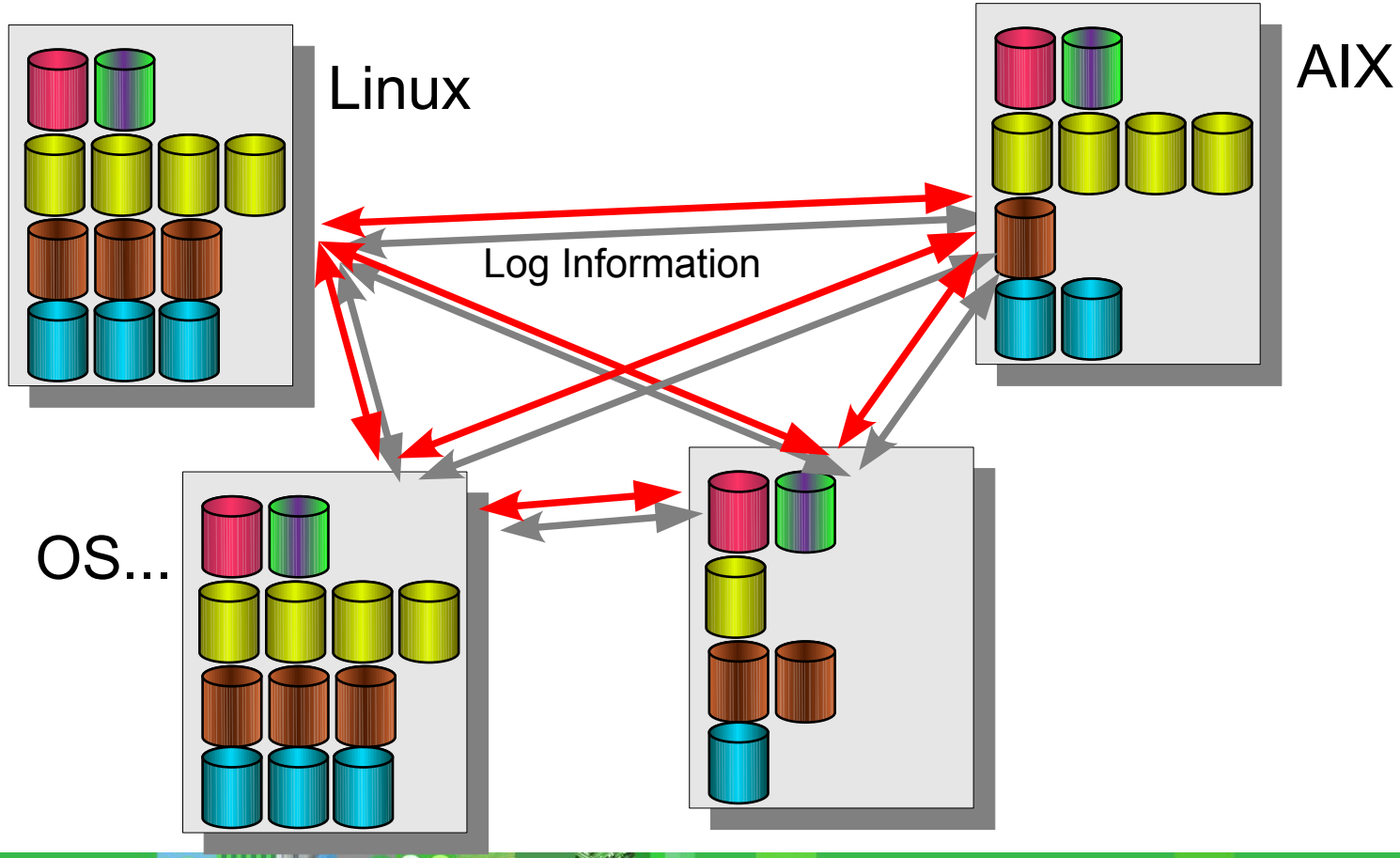
SDS – Weitere Instanz auf dem selben Datenbestand

- Ressourcen eines weiteren Servers für Applikationen auf dem selben Datenbestand



ER/CDR – Replikation je Tabelle mit “update anywhere”

- Ausfallsicherheit gegenüber Serverausfall / RZ Ausfall
- Unterschiedliche OS-Versionen und Informix Versionen möglich



ER/CDR – Auswirkung auf die Applikation ?

- Primary … bzw. Alle Teilnehmer der Replikation
  Die Applikation merkt bei DML-Statements nicht,
  ob die Datenbankinstanz mittels ER/CDR repliziert wird.

DDL-Aufrufe:

"Alter Table" nur bei Master Replicates erlaubt.
"Remaster" notwendig.

RAW Tables und Truncate sind verboten

Cool

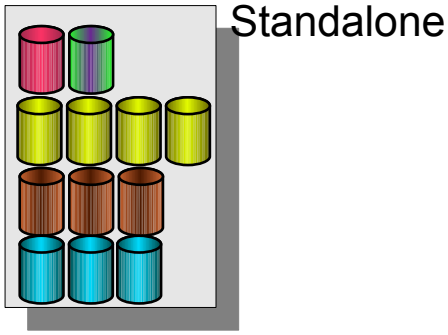
Rolling upgrade

Neu

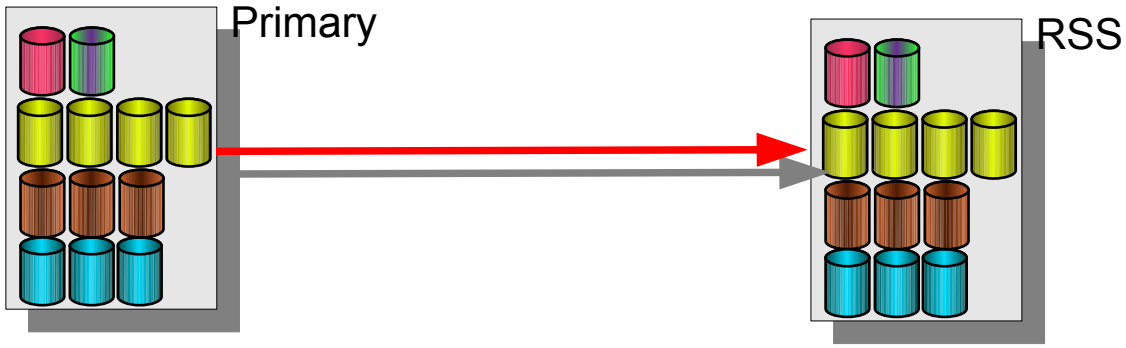
Rolling upgrade

- Die Idee:
  - Spiegelserver für die Zeit der Migration
  - Replikation der Instanz mittels ER/CDR
  - Aussetzen der Replikation um den 2. Server upzugraden
  - Synchronisierung der Replikation
  - Switch der Applikationen auf den 2. Server
  - Aussetzen der Replikation zum ersten Server
  - Upgrade des ersten Servers
  - Wiederaufnahme der Replikation
  - Switch der Applikationen auf den ersten Server
  - Abbau der Replikation

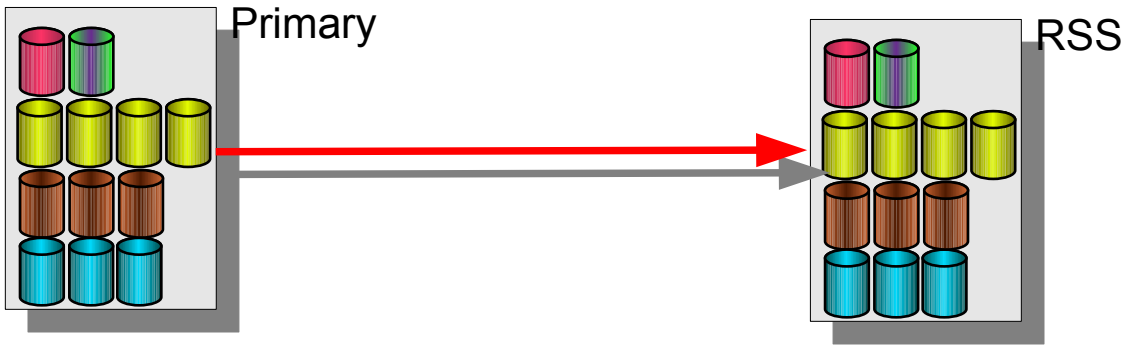
Rolling Upgrade



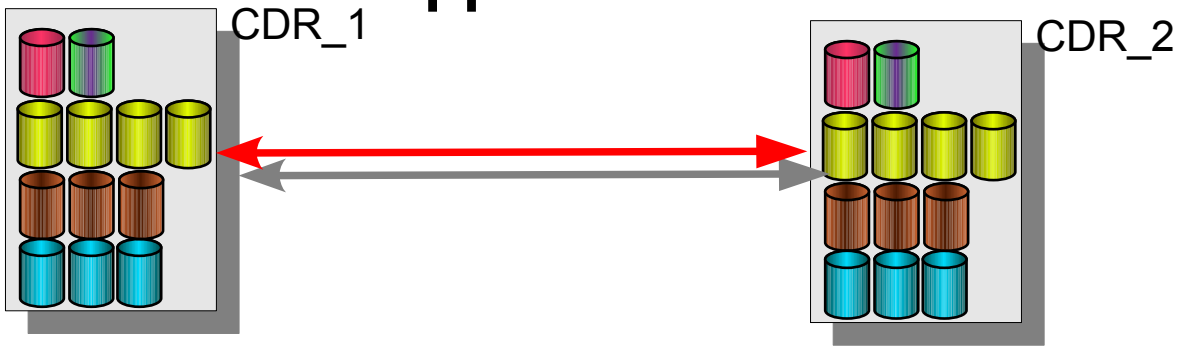
1. Schritt: HDR oder RSS Server aufbauen und Daten übertragen



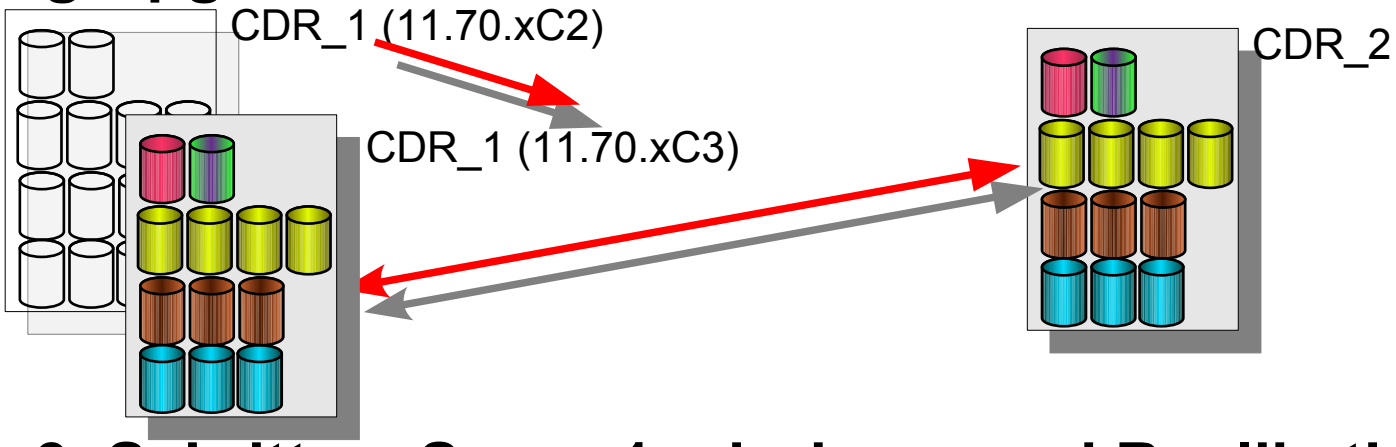
Rolling Upgrade



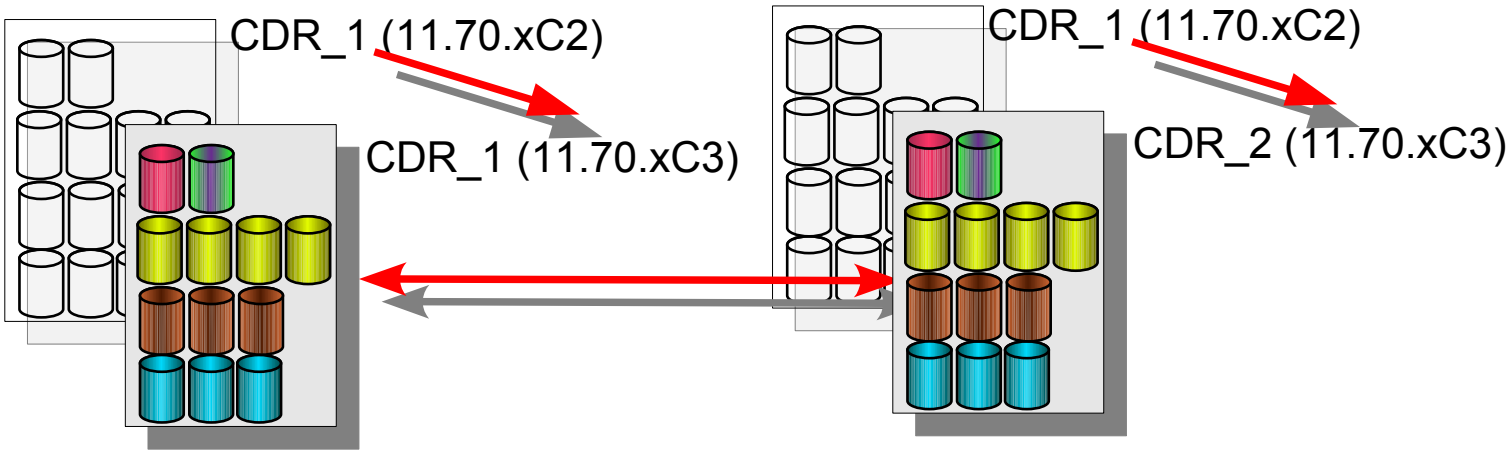
2. Schritt: CDR Server aufbauen, Replicates je Tabelle erstellen
Applikation läuft auf zweitem Server



Rolling Upgrade



3. Schritt: Server1 migrieren und Replikation wieder starten. CDR beibehalten ?



Rolling upgrade

- Die Realisierung (HDR aufsetzen):
  - Aufsetzen einer HDR mittels Script:

    . /home/informix/set_server1; ontape -s -L 0 -t STDIO | rsh server2 ". /home/informix/set_server2; ontape -p -t STDIO"

    rsh server2 ". /home/informix/TEST/set_server2; onmode -d secondary server1"

    . /home/informix/TEST/set_server1; onmode -d primary server2

Rolling upgrade

- Die Realisierung (HDR in CDR umwandeln):
  - Automatische Konvertierung einer HDR zur CDR testen:

cdr check sec2er server2

The version of the server you are using has full ERKEY support.
WARNING: CDR_SERIAL value on server1 can cause collisions.
WARNING: Database space logdbs is becoming full.
WARNING: Using the same values for CDR_SERIAL can cause collisions.

Secondary conversion to ER is possible.

Rolling upgrade

- Die Realisierung (HDR in CDR umwandeln):
  - Automatische Konvertierung einer HDR zur CDR testen mit Auflistung der Aktionen:

cdr check sec2er -p server2

...

Secondary conversion to ER is possible.

```
cdr define serv -c server1_rep -I server1_rep
dbaccess - - <<EOF
database mx42;
alter table 'informix'.warehouses add ERKEY;
EOF

cdr define repl --connect=server1_rep sec2er_1_1302602697_call_type      \
--master=test1_rep --conflict=always --scope=row                         \
"mx42@server1_rep:'informix'.call_type"                                  \
"select * from 'informix'.call_type"

cdr start repl --connect=server1_rep sec2er_1_1302602697_call_type
```

...

Rolling upgrade

- Die Realisierung (HDR in CDR umwandeln):
  - Automatische Konvertierung einer HDR zur CDR ausführen:

cdr start sec2er server2

The version of the server you are using has full ERKEY support.
WARNING: CDR_SERIAL value on test1 can cause collisions.
WARNING: Database space logdbs is becoming full.
WARNING: Using the same values for CDR_SERIAL can cause collisions.

Secondary conversion to ER is possible.

….

Changed LOG_INDEX_BUILDS=1 in onconfig file.

Rolling upgrade

- Die Realisierung (HDR in CDR umwandeln):
  - Ausgaben im online.log:

On Primary:

'syscdr' database built successfully.
'sysha' database built successfully.

RSS server2 added
RSS Server server2 - state is now connected

CDR shutdown complete
RSS Server server2 - state is now disconnected
Internal stopping and restarting of ER was successful

ER clone: Successfully created new ER node with id 2 ('server2_rep')
CDR: Re-connected to server, id 2, name <server2_rep>

Rolling upgrade

- Die Realisierung (HDR in CDR umwandeln):
  - Ausgaben im online.log:

On Secondary:

DR: new type = RSS
DR: RSS secondary server operational
ER clone: Starting split from primary server 'server1'.
ER clone: Joining this server into the ER domain.
ER clone: Determining Grid participation.
ER clone: Adding this server to ER replicates.
ER clone: Waiting for ER control queues to drain at the progenitor server.
ER clone: Restarting ER queues at progenitor server and peers.
ER clone: Waiting for open transactions to complete at the progenitor server.
ER clone: Waiting for LSN to advance to 63:0x0x3f90dc
ER clone: LSN successfully advanced to 63:0x0x3f90dc
ER clone: Converting this server to standard mode.
...

Rolling upgrade

- Die Realisierung (HDR in CDR umwandeln):
  - Ausgaben im online.log:

On Secondary:

...
Shutting down SDS/RSS threads
SMX thread is exiting
DR: Turned off on secondary server
DR: new type = standard
ER clone: Adjusting Enterprise Replication parameters.
On-Line Mode
ER clone: Starting Enterprise Replication.
ER checkpoint started
ER clone: Successfully completed split from primary server 'server1'

Cool

Flexible Grid

Neu

Flexible Grid – Was ist das ?

- CDR ohne PRIMARY KEYS ?
  (“alter table <tabname> add ERKEY”)
- DDL-Befehle auf allen Servern ?
- DML-Befehle auf allen Servern ?
- Automatische Replikation für neue Tabellen ?

- Voraussetzung: CDR ist eingerichtet

Flexible Grid – Einrichtung

- cdr define grid — Erstellen eins Grid
- cdr delete grid — Löschen eines Grid
- cdr change grid — Ändern der Struktur eines Grid (add/delete)
- cdr enable grid — Aktivieren eines Grid
- cdr disable grid — Deaktivieren eines Grid
- cdr list grid — Anzeige eines Grid

Flexible Grid – Funktionen

ifx_grid_connect()	Zum Grid verbinden
ifx_grid_disconnect()	Verbindung zum Grid trennen
ifx_grid_execute()	Ausführen von Befehlen im Grid
ifx_grid_procedure()	Ausführen eines Procedure im Grid
ifx_grid_function()	Ausführen einer Funktion im Grid
ifx_grid_redo()	Erneuter Aufruf eines Gridbefehls
ifx_grid_purge()	Löschen bisheriger Grid Aufrufe
ifx_set_erstate()	Aktivieren/Deaktivieren der Replikation
ifx_get_erstate()	Abfragen des Replikation Status

Flexible Grid – Beispiel

Erstellen eines Grid auf bestehenden CDR-Servern:

```
cdr define grid test_grid test1_rep test2_rep …
      # alternativ --all (statt Liste der Server)
```

Aktivieren des Grid und Freischaltung für User

```
cdr enable grid -g test_grid -n test1_rep -n
  test2_rep -n test3_rep -u informix -u kalu
```

Flexible Grid – Beispiel

Ausführung eines DDL-Befehls auf allen Servern des Grid:

```
execute procedure ifx_grid_connect('test_grid', 1);
create table kalu_tab (f1 int, f2 char(42));
execute procedure ifx_grid_disconnect();
```

Ausfühung eines SQL-Befehls auf allen Servern:

```
execute procedure ifx_grid_execute("test_grid",
  "insert into kalu_tab values
             (dbinfo('sessionid'),DBSERVERNAME)");
```

Flexible Grid – Beispiel

Ausführungsstatus von Grid Befehlen abfragen:

```
cdr list grid -v
Node:test1_rep Stmtid:1 User:informix
  Database:sysmaster 2011-05-23 15:27:54
create table kalu_tab (f1 int, f2 char(42))
ACK test1_rep 2011-05-23 15:27:54
ACK test2_rep 2011-05-23 15:27:54
ACK test3_rep 2011-05-23 15:27:54
```

Fragen ?

Beispiele demnächst
im INFORMIX Newsletter

ifmxnews@de.ibm.com


Thank
You