CERN IT Department

Online/offline replication via Oracle Streams

WLCG Collaboration Workshop
22-27 January 2007

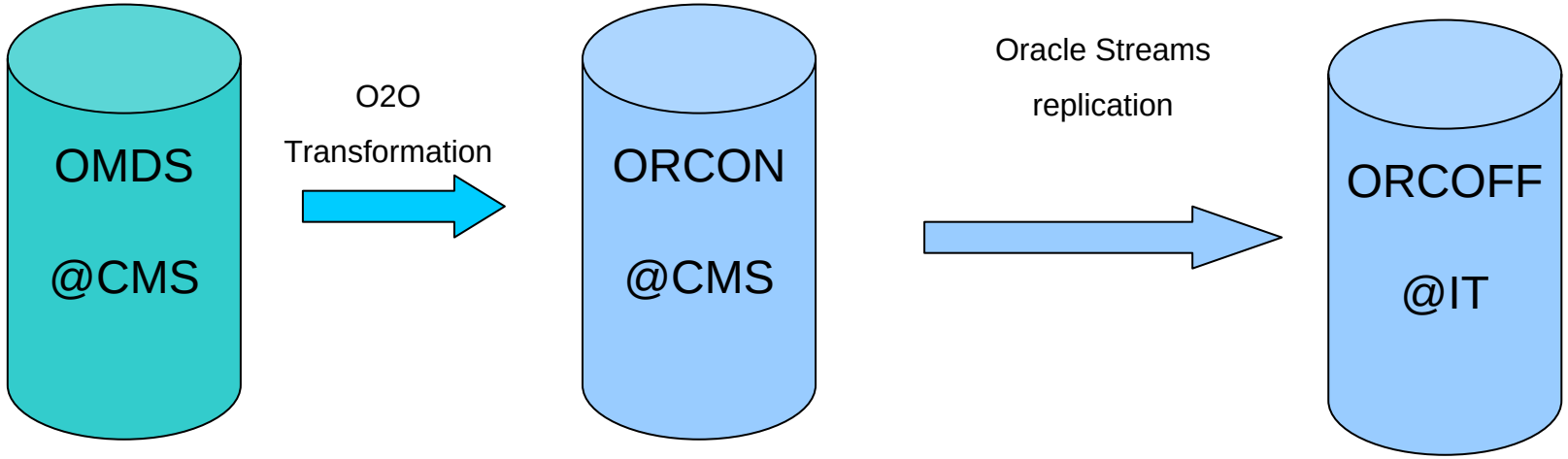
Marcin Bogusz
CERN, PH-CMG

Agenda

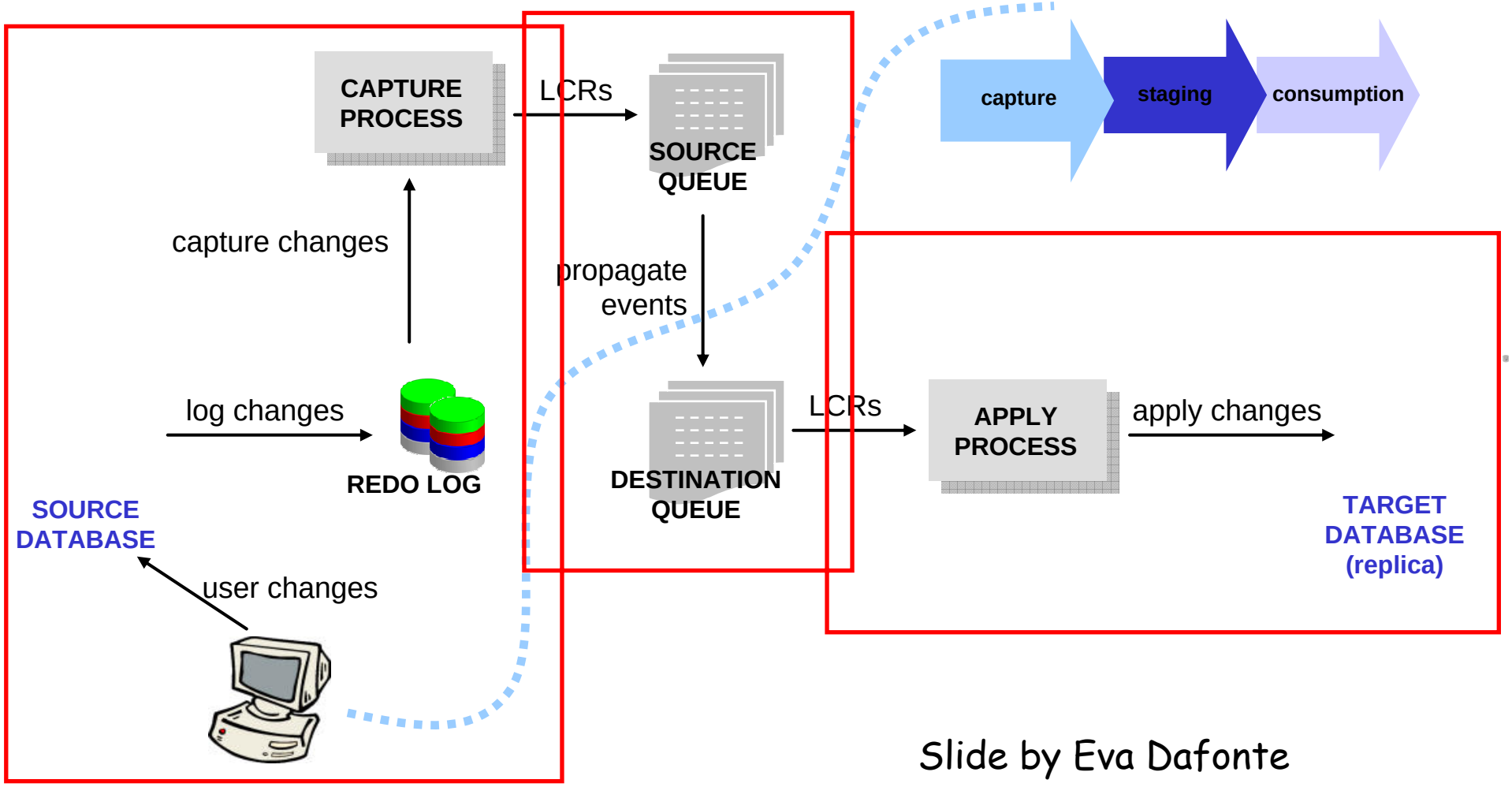
CERN IT Department

- Online/offline replication setup
- Replication tests
- Performance
- Questions

Setup


OMDS
@CMS
O2O
Transformation
ORCON
@CMS
Oracle Streams
replication
ORCOFF
@IT

STREAMS Architecture



Slide by Eva Dafonte

Test setup

CERN IT Department

- Replication between two dual node RACs
- Capture process runs on instance #2 of d3r
- Apply process runs on instance #2 of test1

Test environment

CERN IT Department

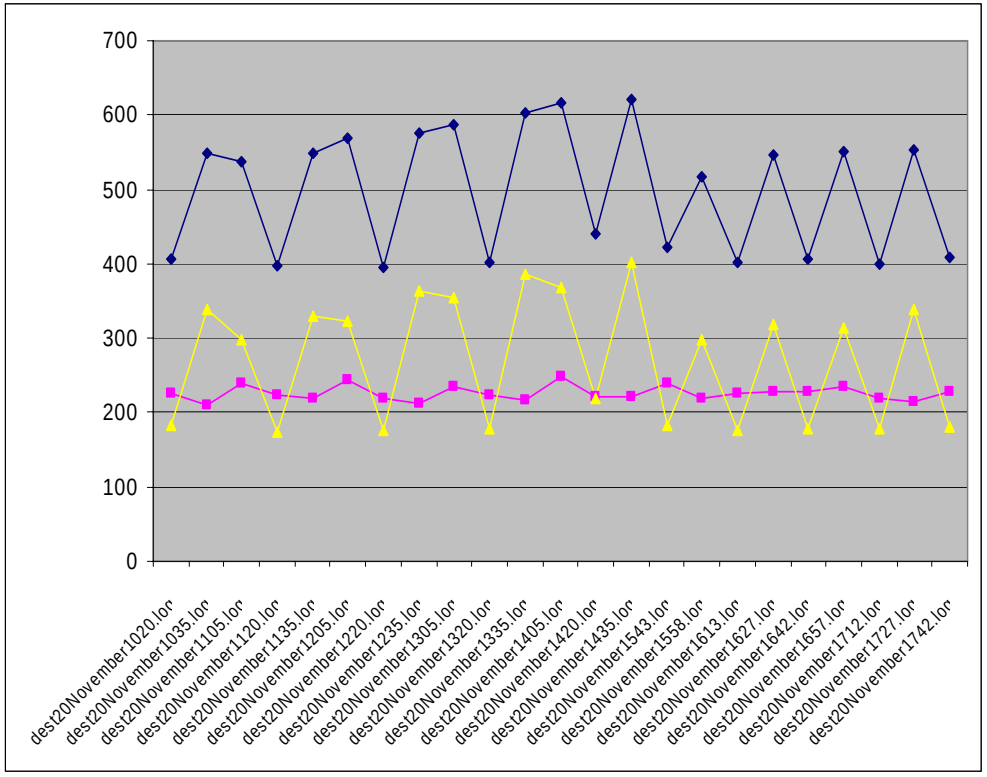
- Test replicates ecalpedestals objects
- 7 objects have to be inserted in a single transaction
- Each object is 1,63 MB in size, after object-to-relational mapping 61200 st_ecalpedestals_item rows
- Objects are generated by PL/SQL procedure provided by Ricky Egeland
- We have 428,400 row inserts in a single commit, which corresponds to the number of LCRs
- LCR size is row-dependant

Streams monitoring

- Tools
  - strmmon
  - Zbyszek's monitor
  - AWR
  - Logminer
  - Lemonweb
  - Oracle Enterprise manager
- Parameters
  - LCRs flow
  - Streams pools size
  - Redo generated on source and destination databases
  - Replication time

Performance

- Goal : to reduce replication time = increase throughput
- Seemed that replication time was dependant on interval between test runs
- Periodical time structure of the results
- Responsible for differences: KJC Wait events (AWR)
- Solution – 10.2.0.3 patch set



Performance 10.2.0.3

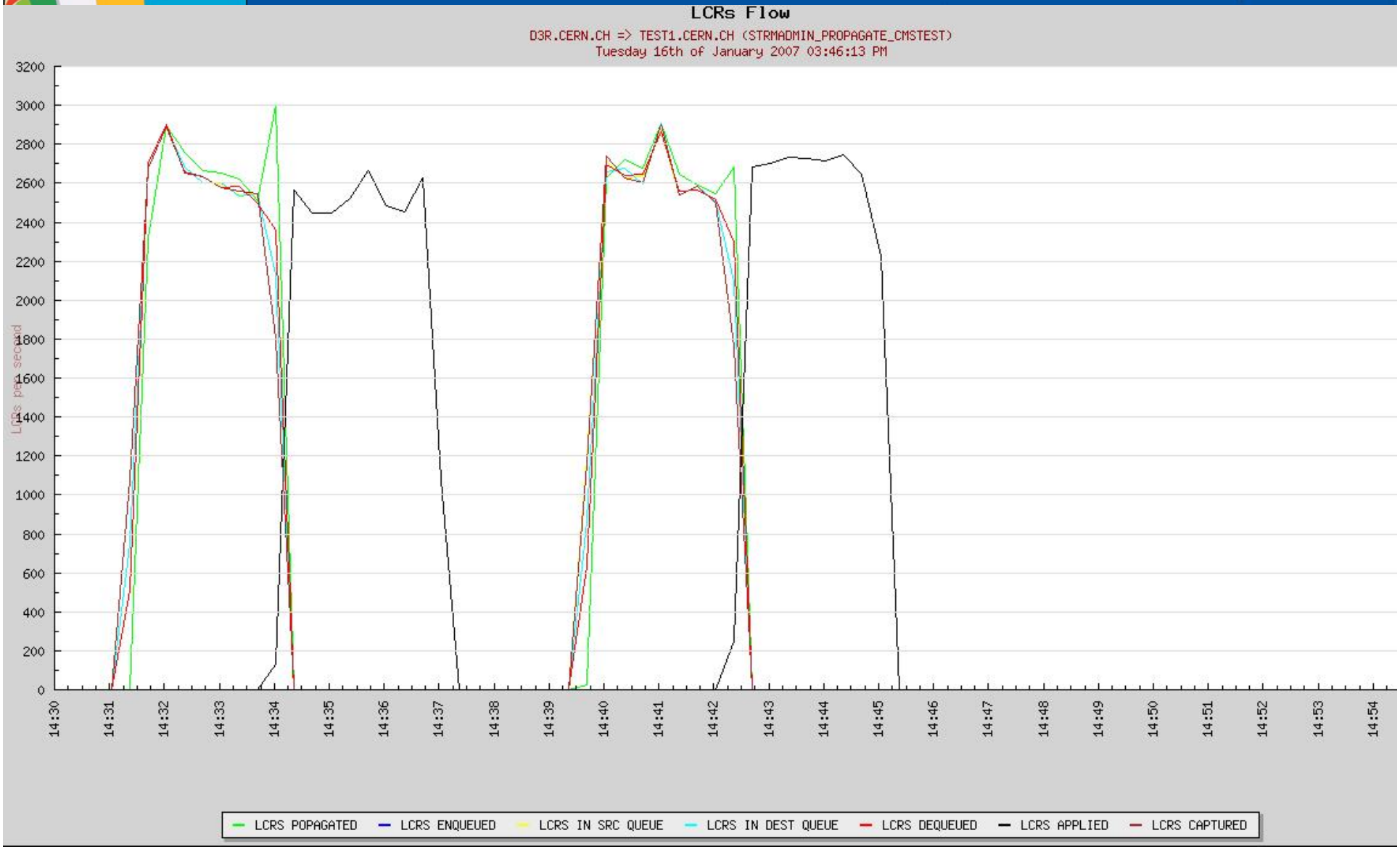
CERN IT Department

- Replication time – 300s
  - With 11.4 MB payload 0.038 MB/s
  - 1428 row inserts per second
  - Average LCR apply rate – 2700/s
- AWR report
  - Both CPUs utilized in ~50%
  - 120 seconds of Streams „AQ: enqueue blocked due to flow control” wait events
- Lemonweb
  - Very high I/O during test run (over 1 GB)

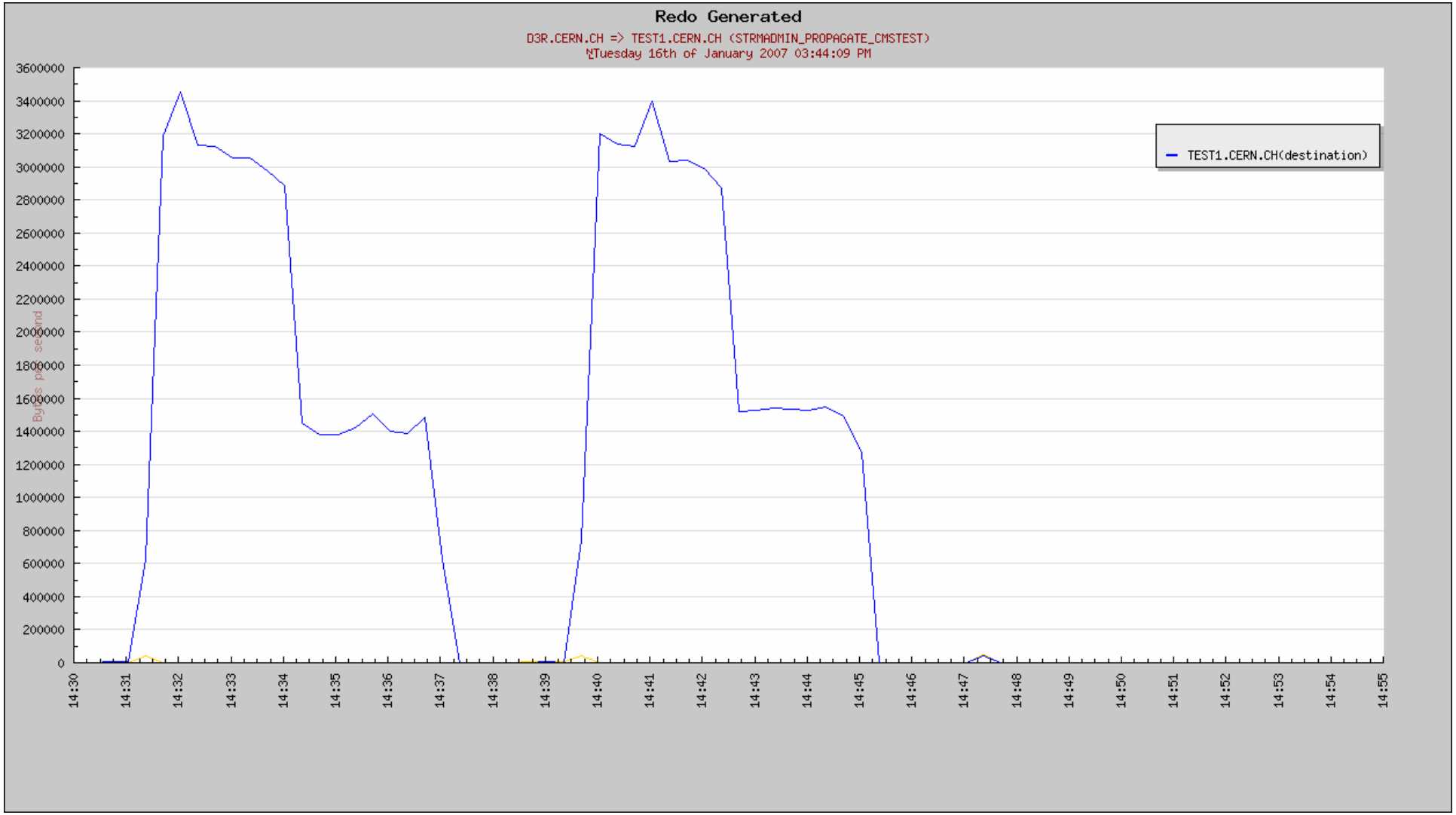
Redo

- High I/O turned out to be generated by Log Writer (LGWR) process
- Further investigation has shown that the redo on the destination site was 12 times larger than on the source database
- Log mining has shown that the redo is generated due to inserting to STREAMS$_APPLY_SPILL_MSGS_PART table
- Typical redo entry for st_ecalpedestals_item table was 75 bytes vs. 590 bytes for spill queue
- Redo volume on the destination is ~700 MB, both tables insert-related redo is ~300 MB so we still have 400MB unaccounted for

LCRs Flow


LCRs Flow
D3R.CERN.CH => TEST1.CERN.CH (STRMADMIN_PROPAGATE_CMSTEST)
Tuesday 16th of January 2007 03:46:13 PM
LCRs per second
3200
3000
2800
2600
2400
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
0
14:30
14:31
14:32
14:33
14:34
14:35
14:36
14:37
14:38
14:39
14:40
14:41
14:42
14:43
14:44
14:45
14:46
14:47
14:48
14:49
14:50
14:51
14:52
14:53
14:54
LCRS POPAGATED
LCRS ENQUEUED
LCRS IN SRC QUEUE
LCRS IN DEST QUEUE
LCRS DEQUEUED
LCRS APPLIED
LCRS CAPTURED

Redo destination



Spilling

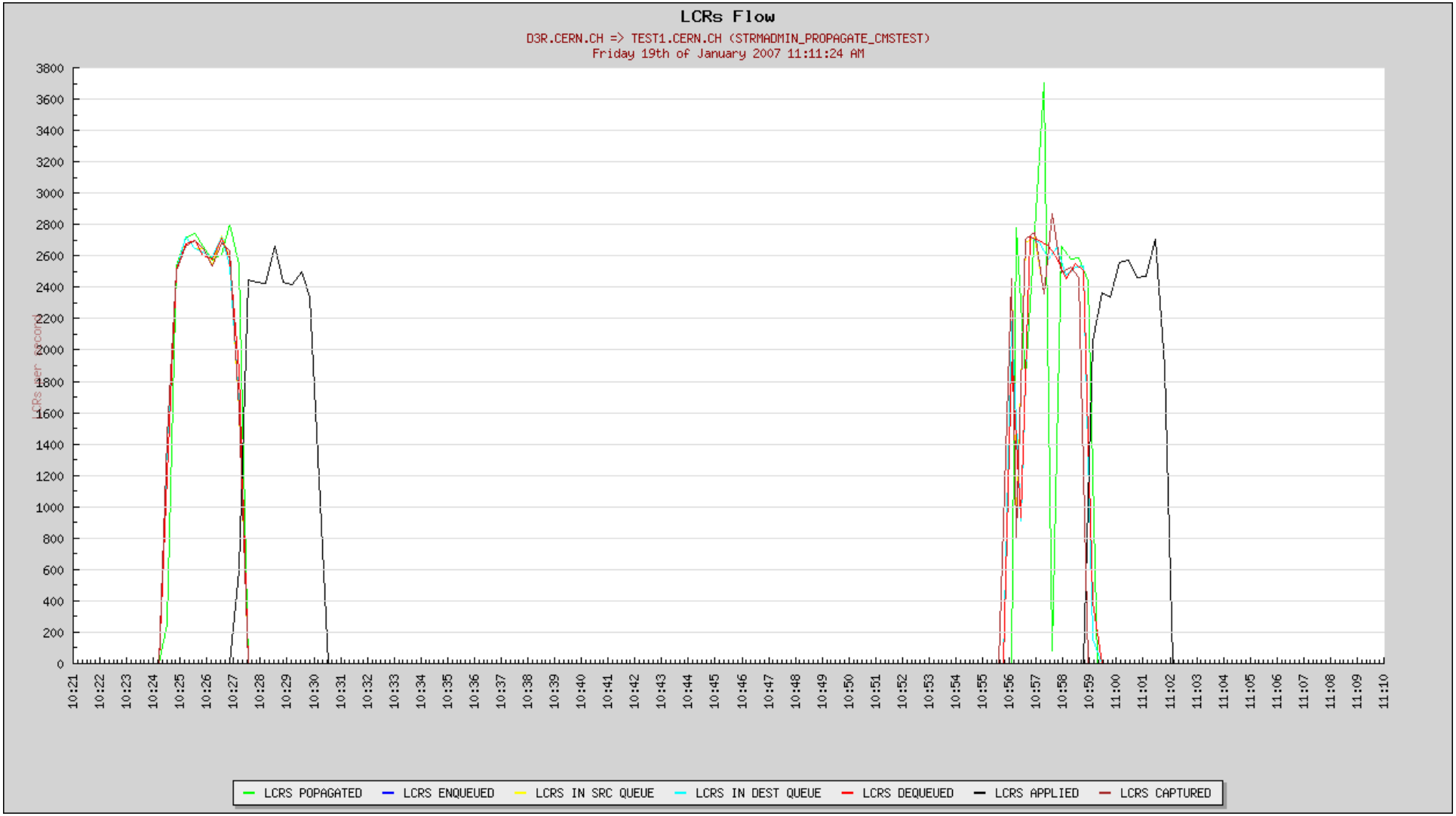
CERN IT Department

- Inserting data to STREAMS$_APPLY_SPILL_MSGS_PART table is related to the architecture of the apply process
- The messages are spilled to the disk from the memory
- Apply process' TXN_LCR_SPILL_THRESHOLD user modifiable parameter can change the spill threshold.
- Increasing the threshold over the number of LCRs in a transaction should disable the apply spilling
- However, increasing this parameter degrades the performance

Queue spilling

CERN IT Department

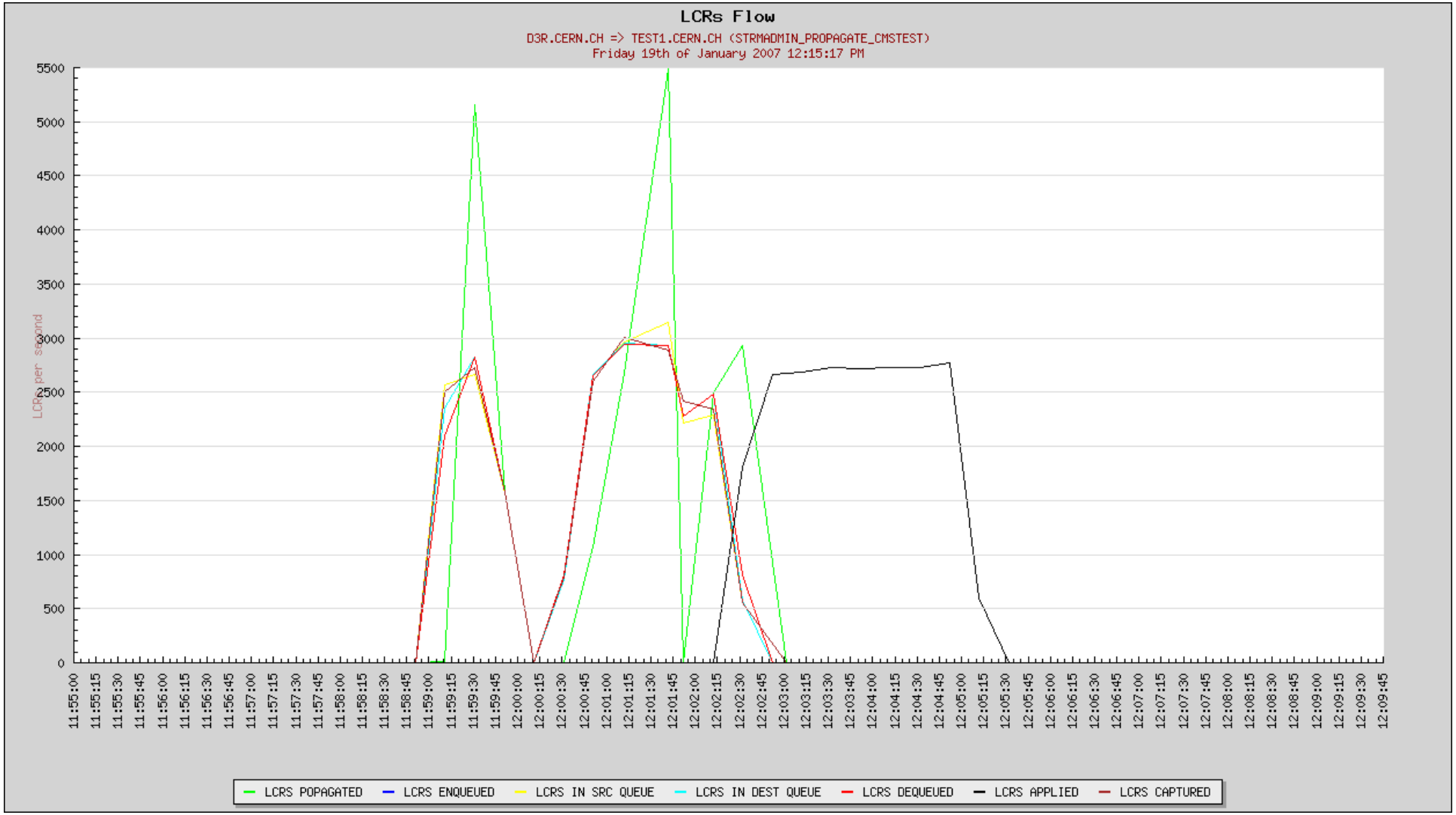
- Streams pool size is 1.3 GB
- Allocated streams pool size never exceeds 75%
- Spilling when messages enqueued for more than 5 minutes (not the case)
- Spilling when there's no memory available (maybe the case, but 1.3 GB!)
- Higher spill thresholds lead to „propagation gaps" and degrade performance, default threshold is 10,000
- While spill threshold set to values over 200K messages, queue spilling occurs – other than apply spilling


LCRs Flow
D3R.CERN.CH => TEST1.CERN.CH (STRMADMIN_PROPAGATE_CMSTEST)
Friday 19th of January 2007 11:11:24 AM
LCRs per second
LCRS POPAGATED
LCRS ENQUEUED
LCRS IN SRC QUEUE
LCRS IN DEST QUEUE
LCRS DEQUEUED
LCRS APPLIED
LCRS CAPTURED

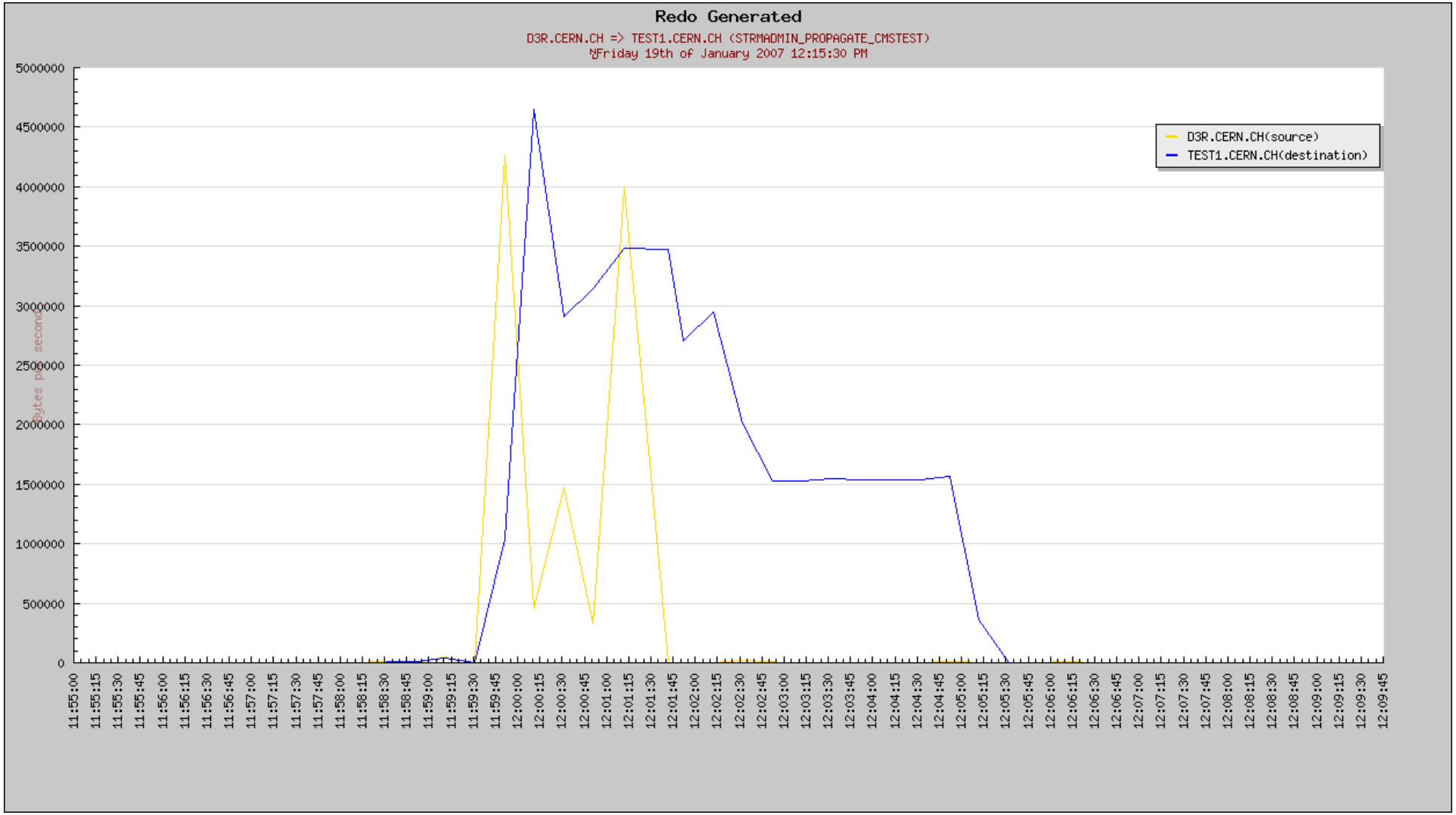
Strmmon - destination


```
2007-01-19 10:55:50 || test12-> | NET 4K 0 | PR01 0 | Q570584 0 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:55:52 || test12-> | NET 5K 0 | PR01 0 | Q570584 0 0 |  -  A003 0 0 19sec <100%I 0%F ->
2007-01-19 10:55:54 || test12-> | NET 5K 0 | PR01 0 | Q570584 0 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:55:56 || test12-> | NET 821K 0 | PR01 2526 | Q570584 2526 0 |  -  A003 0 0 19sec <100%I 0%F ->
2007-01-19 10:55:58 || test12-> | NET 1M 0 | PR01 3945 | Q570584 3940 0 |  -  A003 0 0 19sec <100%I 0%F ->
2007-01-19 10:56:00 || test12-> | NET 1M 0 | PR01 6005 | Q570584 6009 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:02 || test12-> | NET 356K 0 | PR01 1103 | Q570584 1099 0 |  -  A003 0 0 19sec <100%I 0%F ->
2007-01-19 10:56:04 || test12-> | NET 2M 0 | PR01 8197 | Q570584 8201 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:06 || test12-> | NET 316K 0 | PR01 1004 | Q570584 997 0 |  -  A003 0 0 19sec <100%I 0%F ->
2007-01-19 10:56:08 || test12-> | NET 1M 0 | PR01 4745 | Q570584 4753 0 |  -  A003 0 0 19sec <43%I 0%F ->
2007-01-19 10:56:10 || test12-> | NET 5K 0 | PR01 0 | Q570584 0 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:12 || test12-> | NET 5K 0 | PR01 0 | Q570584 0 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:14 || test12-> | NET 5K 0 | PR01 0 | Q570584 0 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:16 || test12-> | NET 5K 0 | PR01 0 | Q570584 0 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:18 || test12-> | NET 5K 0 | PR01 0 | Q570584 0 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:20 || test12-> | NET 5K 0 | PR01 0 | Q570584 0 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:22 || test12-> | NET 203K 0 | PR01 600 | Q570584 600 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:25 || test12-> | NET 884K 0 | PR01 2750 | Q570584 2750 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:27 || test12-> | NET 884K 0 | PR01 2750 | Q570584 2750 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:29 || test12-> | NET 868K 0 | PR01 2750 | Q570584 2750 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:31 || test12-> | NET 884K 0 | PR01 2750 | Q570584 2750 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:33 || test12-> | NET 884K 0 | PR01 2746 | Q570584 2741 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:35 || test12-> | NET 772K 0 | PR01 2510 | Q570584 2507 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:37 || test12-> | NET 882K 0 | PR01 2757 | Q570584 2757 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:39 || test12-> | NET 996K 0 | PR01 3046 | Q570584 3045 0 |  -  A003 0 0 19sec <0%I 0%F ->
2007-01-19 10:56:41 || test12-> | NET 756K 0 | PR01 2290 | Q570584 2298 0 |  -  A003 0 0 19sec <0%I 0%F ->
```

2 x spilling, 130K threshold



2 x spilling, 130K threshold



LCR performance

CERN IT Department

- Performance greatly depends on the number of LCRs in a transaction, rather than LCR size
- Test with BLOB objects instead of relational data
  - The same payload but binary data
  - Inserting 7 x 1.63 MB binary objects
  - Such a transaction produces ~3600 LCRs
  - Replication time ~30 seconds (factor 10 difference)
  - No queue spilling
  - Probably no apply spilling
  - Redo ratio source/destination is 45/66 (MB)

Questions?

Marcin Bogusz
CERN, PH-CMG

Thank you

Marcin Bogusz
CERN, PH-CMG