Worldwide distribution of experimental physics data using Oracle Streams

Eva Dafonte Pérez
Database Administrator @CERN
Outline

• CERN and LHC Overview
• Oracle Streams Replication
• Replication Performance
• Optimizations: Downstream Capture, Split and Merge, Network, Rules and Flow Control
• Periodic Maintenance
• Lessons Learned
• Tips and Tricks
• Streams Bugs and Patches
• Scalable Resynchronization
• 3D Streams Monitor
• New 11g Streams Features
• Streams Setups Examples
• Summary
CERN and LHC

• European Organization for Nuclear Research
  – world’s largest centre for scientific research
  – founded in 1954
  – mission: finding out what the Universe is made of and how it works

• LHC, Large Hadron Collider
  – particle accelerator used to study the smallest known particles
  – 27 km ring, spans the border between Switzerland and France about 100 m underground
  – will recreate the conditions just after the Big Bang
The LHC Computing Challenge

- **Data volume**
  - high rate x large number of channels x 4 experiments
  - 15 PetaBytes of new data each year stored
  - much more data discarded during multi-level filtering before storage

- **Compute power**
  - event complexity x Nb. events x thousands users
  - 100 k of today's fastest CPUs

- **Worldwide analysis & funding**
  - computing funding locally in major regions & countries
  - efficient analysis everywhere
  - **GRID technology**
Distributed Service Architecture

- **T0** – autonomous reliable service
- **T1** – DB backbone
  - all data replicated
  - reliable service
- **T2** – local DB cache
  - subset data
  - only local service

**Online DB**
- autonomous reliable service

**Oracle Streams**
- http transfer
- Cross DB copy and MySQL/SQLite files

**Oracle DB cluster**
- MySQL/SQLite file
- Squid cache
- FroNTier server

**Read-only access at Tier-1/2 (at least initially)**
Oracle Streams Replication

- Technology for **sharing information** between databases
- Database changes captured from the redo-log and propagated asynchronously as Logical Change Records (LCRs)
Replication Performance

- The atomic unit is the change record: LCR
- LCRs can vary widely in size
  - Throughput is not a fixed measure

- Capture performance:
  - Read changes from the redo
    - from redo log buffer (memory - much faster)
    - from archive log files (disk)
  - Convert changes into LCRs
    - depends on the LCR size and number of columns
  - Enqueue the LCRs
    - concurrent access to the data structure can be costly
Replication Performance

- Propagation performance:
  - Browse LCRs
  - Transmit LCRs over the network
  - Remove LCRs from the queue
    - Done in separate process to avoid any impact

- Apply performance:
  - Browse LCRs
  - Execute LCRs
    - Manipulate the database is slower than the redo generation
    - Execute LCRs serially => apply cannot keep up with the redo generation rate
  - Remove LCRs from the queue
Downstream Capture

- **Downstream capture** to de-couple Tier 0 production databases from destination or network problems
  - source database availability is highest priority
- Optimizing **redo log retention** on downstream database to allow for sufficient re-synchronisation window
  - we use 5 days retention to avoid tape access
- Dump fresh copy of dictionary to redo periodically
- 10.2 Streams recommendations (metalink note 418755)
Split & Merge: Motivation
• High memory consumption
• LCRs spilled over to disk
  → Overall Streams performance impacted
• When memory exhausted
  → Overall Streams replication stopped
Split & Merge

- **Objective:** isolate replicas against each other
  - **Split**
    - (original) Streams setup for “good” sites
      - drop propagation job/s to “bad” site/s
        → spilled LCRs are removed from the capture queue
    - (new) Streams setup for “bad” site/s
      - new capture queue
      - clone capture process and propagation job/s
    - does not require any change on the destination site/s
  - **Merge**
    - move back the propagation job/s to the original setup
    - clean up additional Streams processes and queue
    - does not require any change on the destination site/s
Split & Merge: Details

**Split:**
- gather cloning information:
  - **capture:**
    - rule set name
    - `start_scn = last applied message scn @target`
    - `first_scn = previous dictionary build < start_scn`
  - **propagation:**
    - rule set name
    - target queue name and db link

**Merge:**
- select the minimum required checkpoint scn between the 2 capture processes
- recover original propagation
TCP and Network Optimizations

- TCP and Network tuning
  - adjust system max TCP buffer (/etc/sysctl.conf)
  - parameters to reinforce the TCP tuning
    - DEFAULT_SDU_SIZE=32767
    - RECV_BUF_SIZE and SEND_BUF_SIZE
      - Optimal: 3 * Bandwidth Delay Product
- Reduce the Oracle Streams acknowledgements
  - alter system set events '26749 trace name context forever, level 2';
Streams Rules

- Used to control which information to share
- Rules on the capture side caused more overhead than on the propagation side
- Avoid Oracle Streams complex rules

Complex Rule

condition => '( SUBSTR(:ddl.get_object_name(),1,7) IN ("COMP200", "OFLP200", "CMCP200", "TMCP200", "TBDP200", "STRM200")
OR SUBSTR (:ddl.get_base_table_name(),1,7) IN ("COMP200", "OFLP200", "CMCP200", "TMCP200", "TBDP200", "STRM200") ) '

Simple Rule

condition => '((:ddl.get_object_name() >= "STRM200_A" and :ddl.get_object_name() <= "STRM200_Z") OR (:ddl.get_base_table_name() >= "STRM200_A" and :ddl.get_base_table_name() <= "STRM200_Z")
OR (:ddl.get_object_name() >= "OFLP200_A" and :ddl.get_object_name() <= "OFLP200_Z") OR (:ddl.get_base_table_name() >= "OFLP200_A" and :ddl.get_base_table_name() <= "OFLP200_Z"))

Avoid complex rules:
- LIKE
- Functions
- NOT
• Example: ATLAS Streams Replication
  - rules defined to filter tables by prefix
Flow Control

• By default, flow control kicks when the number of messages is larger than the threshold
  – Buffered publisher: 5000
  – Capture publisher: 15000

• Manipulate default behavior
• 10.2.0.3 + Patch 5093060 = 2 new events
  – 10867: controls threshold for any buffered message publisher
  – 10868: controls threshold for capture publisher
• 10.2.0.4 = 2 new hidden parameters
  – “_capture_publisher_flow_control_threshold”
  – “_buffered_publisher_flow_control_threshold”
• Example: ATLAS PVSS Streams Replication
Periodic Maintenance

• Dump fresh copy of Dictionary redo
  – reduces the amount of logs to be processed in case of additional process creation

• Reduce high watermark of AQ objects
  – maintain enqueue/dequeue performance
  – reduce QMON CPU usage
  – metalink note 267137.1

• Shrink Logminer checkpoint table
  – improves capture performance
  – metalink note 429599.1

• Review the list of specific Streams patches
  – metalink note 437838.1
Lessons Learned

- SQL bulk operations (at the source db)
  - may map to many elementary operations at the destination side
  - need to control source rates to avoid overloading
- Batch processing
  - minimize the performance impact using Streams tags
  - avoid changes being captured, then run same batch load on all destination
- System generated names
  - do not allow system generated names for constraints and indexes
  - modifications will fail at the replicated site
  - storage clauses also may cause some issues if the target sites are not identical
Lessons Learned

• Replication of “grant” operations
  – grants on views, PL/SQL procedures, functions and packages are NOT replicated
  – grantee must exist at all destinations

• Long transactions (non-frequent commits)
  – Total number of outstanding LCRs is too large
  – LCRs are in memory too long
    → LCRs are spilled over to disk
    → Apply performance is impacted
  – All LCRs in a single transaction must be applied by one apply server
    → Parallel servers cannot be used efficiently
  – Too many unbrowsed messages enables flow control
    → Streams processes are paused
Lessons Learned

- Example: CMS replication - Online to Offline (CERN)
  - single transaction mapping 428400 LCRs
Lessons Learned

- Example: CMS replication - Online to Offline (CERN)
  - Use BLOB objects: single transaction mapping 3600 LCRs
Lessons Learned

- Apply oldest_message_number is not updated
  - caused by an old transaction not correctly removed from the apply spill table
  - dba_apply_spill_txn view in order to identify the transaction
  - set the apply parameter _IGNORE_TRANSACTION with the transaction id in the apply spill over queue
  - run purge_spill_txn procedure (metalink note 556183.1)

- Apply might degrade performance when applying transactions to tables > 10M rows
Tips and Tricks

• How to recover Streams if downstream database crashes
  – use source database as replacement
    • all archive logs are available
  – check the oldest message number applied at each of the destinations
  – select Streams dictionary SCN < min(oldest message numbers)
  – create the Streams queue and all the propagations
  – create capture process where
    • first_scn = dictionary SCN
    • start_scn = oldest_message_number

• Configure back the downstream database
  – build a new Streams dictionary
  – stop capture and wait until all LCRs are applied
  – repeat steps above
  – register the archive logs with the capture process
• Performing a **switchover** in a Streams environment
  
  – database hw migration with minimal downtime
  – completely transparent for destination databases
  – source database:
    
    • before the switchover: move forward first_scn
    • after the switchover: check that the archivelog files are registered with the capture process
      
      – otherwise, register them manually (from first_scn)
Streams Bugs and Patches

- Streams specific patches  metalink note 437838.1

- Bug 6452375
  - ORA-26687 in Streams from “drop table”
  - when two streams setups between same source and destination databases to replicate different schemas

- Bug 6402302
  - inconsistent capture/propagation/apply of DDLs in Streams
  - for example: “drop synonym” DDL is not captured/propagated or applied while create synonym is captured/propagated and applied
Streams Bugs and Patches

• ORA-00600: [KRVRDCBMDDLSQL1]
  – caused by rebuild index operation using parallel option
  – logminer corruption?
  – capture process could not be restarted at the current SCN
  – workaround proposed by Oracle: recreate capture using new dictionary after the index rebuild operation → data loss!!
  – complete re-instantiation of the Streams environment

• ORA-07445: exception encountered: core dump
  [kghufree()]+485
  – Oracle Database Change Notification cannot be used in a Streams environment
Scalable Resynchronization

- Target site out of the Streams recovery window
- Complete transfer of data (schemas and tables) using Oracle Data Pump might take too long
  - Example ATLAS Conditions data
- **Transportable Tablespaces**: move a set of tablespaces from one Oracle database to another
  - Export metadata of tablespace instead of data in tablespace
  - But tablespaces must be in **read-only** while the data is copied

*Example: ATLAS COOL test data - 67 GB
Export/Import
Transportable
Tablespaces*

Moving data using transportable tablespaces is much faster than Data Pump export/import
1. Create database links between databases
2. Create directories pointing to datafiles
3. Stop replication to site 5
4. Ensure tablespaces are read-only
5. Transfer the data files of each tablespace to the remote system
6. Import tablespaces metadata in the target
7. Make tablespaces read-write
8. Reconfigure Streams
Streams Monitoring

- Oracle Enterprise Manager
  - Streams monitoring enhancements on 10.2.0.5
- Oracle Streams STRMMON monitoring utility
- Streams configuration report and health check script

- Extended tool for Streams monitoring: 3D Streams Monitor tool @CERN
3D Streams Monitor

• Features:
  – Streams topology
  – Status of streams connections
  – Error notifications
  – Streams performance (latency, throughput, etc.)
  – Other resources related to the streams performance (streams pool memory, redo generation)

• Architecture:
  – “strmmon” daemon written in Python
  – End-user web application
    http://oms3d.cern.ch:4889/streams/main

• 3D monitoring and alerting integrated with WLCG procedures and tools
### ACTIVE STREAMS

<table>
<thead>
<tr>
<th>Stream</th>
<th>LCRs Cap</th>
<th>LCRs Enq</th>
<th>LCRs Prop</th>
<th>LCRs Deq</th>
<th>LCRs App</th>
<th>Cap Latency</th>
<th>Latency</th>
<th>Capture State</th>
<th>Propagation State</th>
<th>Apply State</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATL.DSC.CERN.CH</td>
<td>95134</td>
<td>1.05</td>
<td>1.05</td>
<td>0.0</td>
<td>0.0</td>
<td>23 sec</td>
<td>--</td>
<td>CAPTURING CHANGES</td>
<td>ENABLED</td>
<td>IDLE</td>
<td></td>
</tr>
</tbody>
</table>

#### SOURCE

**ATL.DSC.CERN.CH**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCN read</td>
<td>6083362646734</td>
<td>Read LCRs</td>
</tr>
<tr>
<td>LCRs captured</td>
<td>10491139 (54.27%)</td>
<td>Capture LCRs</td>
</tr>
<tr>
<td>SCN captured</td>
<td>6083362646734</td>
<td>Capture SCN</td>
</tr>
<tr>
<td>LCRs enqueued</td>
<td>613528 (25%)</td>
<td>Enqueue LCRs</td>
</tr>
<tr>
<td>SCN enqueued</td>
<td>6083362621274</td>
<td>Enqueue SCN</td>
</tr>
<tr>
<td>Capture Latency</td>
<td>39 sec</td>
<td>Capture Latency</td>
</tr>
</tbody>
</table>

#### QUEUE

**ATL.DSC.CERN.CH**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>STRM_QUEUE_ATL_APN</td>
<td>Queue Name</td>
</tr>
<tr>
<td>Id</td>
<td>179043</td>
<td>Queue Id</td>
</tr>
<tr>
<td>Outstanding Msg</td>
<td>96838833 (05/s)</td>
<td>Outstanding Msg</td>
</tr>
<tr>
<td>Cumulative Msg</td>
<td>158312 (08/s)</td>
<td>Cumulative Msg</td>
</tr>
</tbody>
</table>

#### APPLY

**ATL.DSC.CERN.CH**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>STRMAN_ADMIN_APPLY_ATLNL</td>
<td>Apply Name</td>
</tr>
<tr>
<td>LCRs Dequeued</td>
<td>4976455 (05/s)</td>
<td>LCRs Dequeued</td>
</tr>
<tr>
<td>Total LCRs Applied</td>
<td>48054206 (08/s)</td>
<td>Total LCRs Applied</td>
</tr>
<tr>
<td>SCN Dequeued</td>
<td>60833626467224</td>
<td>SCN Dequeued</td>
</tr>
<tr>
<td>Daqueue Latency</td>
<td>0 sec</td>
<td>Daqueue Latency</td>
</tr>
<tr>
<td>Transaction Received</td>
<td>528045</td>
<td>Transaction Received</td>
</tr>
<tr>
<td>Transaction Assigned</td>
<td>528045</td>
<td>Transaction Assigned</td>
</tr>
<tr>
<td>Transaction Applied</td>
<td>528045</td>
<td>Transaction Applied</td>
</tr>
<tr>
<td>State</td>
<td>IDLE</td>
<td>Apply State</td>
</tr>
<tr>
<td>HWM SGN</td>
<td>6083362621274</td>
<td>HWM SGN</td>
</tr>
<tr>
<td>HWL Apply Latency</td>
<td>20 sec</td>
<td>HWL Apply Latency</td>
</tr>
<tr>
<td>Error Time</td>
<td>15453 (08/s)</td>
<td>Error Time</td>
</tr>
</tbody>
</table>

---

### 3D Streams Monitor

- **Source**: ATL.DSC.CERN.CH
- **Destination**: OGMA.GRIDPPRL.AC.UK

---

**Note**: The screenshot contains visual diagrams and monitoring metrics for resource utilization, queue status, and apply operations in a 3D stream monitoring context.
### Streams report

#### ALICE streams activities between 27.10.2009 and 03.11.2009

- **Availability** | **Exceptions** | **Process** | **Applies** | **Queue** | **Undefined primary keys**

#### Capture processes stats

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-10-27</td>
<td></td>
<td>cern.ch/streams/pool.captors</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-10-28</td>
<td></td>
<td>cern.ch/streams/pool.captors</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-10-29</td>
<td></td>
<td>cern.ch/streams/pool.captors</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-10-30</td>
<td></td>
<td>cern.ch/streams/pool.captors</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-10-31</td>
<td></td>
<td>cern.ch/streams/pool.captors</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-11-01</td>
<td></td>
<td>cern.ch/streams/pool.captors</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-11-02</td>
<td></td>
<td>cern.ch/streams/pool.captors</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
</tbody>
</table>

#### Propagation processes stats

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-10-27</td>
<td></td>
<td>cern.ch/streams/pool.propagators</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-10-28</td>
<td></td>
<td>cern.ch/streams/pool.propagators</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-10-29</td>
<td></td>
<td>cern.ch/streams/pool.propagators</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-10-30</td>
<td></td>
<td>cern.ch/streams/pool.propagators</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-10-31</td>
<td></td>
<td>cern.ch/streams/pool.propagators</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-11-01</td>
<td></td>
<td>cern.ch/streams/pool.propagators</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-11-02</td>
<td></td>
<td>cern.ch/streams/pool.propagators</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
</tbody>
</table>

#### Queue stats

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2009-10-27</td>
<td></td>
<td>cern.ch/streams/pool.queue_x</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-10-28</td>
<td></td>
<td>cern.ch/streams/pool.queue_x</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-10-29</td>
<td></td>
<td>cern.ch/streams/pool.queue_x</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-10-30</td>
<td></td>
<td>cern.ch/streams/pool.queue_x</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-10-31</td>
<td></td>
<td>cern.ch/streams/pool.queue_x</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-11-01</td>
<td></td>
<td>cern.ch/streams/pool.queue_x</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
<tr>
<td>2009-11-02</td>
<td></td>
<td>cern.ch/streams/pool.queue_x</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
<td>3.2 %</td>
</tr>
</tbody>
</table>
New Streams 11g Features

- Performance improvements:
  - reader process mines from the in-memory redo log buffers
    - minimizes disk I/O
    - reduces capture latency
  - direct communication between capture and apply processes: Combined Capture and Apply
    - improves LCR transmission throughput
    - reduces end-to-end replication latency
  - internal mechanism to execute change records and extensive caching
    - reduces CPU consumption
    - minimizes latch contention and other wait events
New 11g Streams Features

• Automatic Split and Merge
  – split a stream in cases where a replica is unavailable
  – merge into a single stream when replica catches up
  – procedures and sql script generation
  – automatic replication management based on thresholds

• Compare and Converge
  – compare objects across databases for inconsistency
  – resynchronize objects if required
  – table or column level synchronization
  – additional scripting for schema comparison
New 11g Streams Features

And more…

• Synchronous capture
• LCRs track through a Stream
• Topology and Performance Advisor
• New error messages for error handling
More Streams Setup Examples

- **ALICE**
  
  - ALIONR.CERN.CH(CERN) → PDBR.CERN.CH(CERN)
    - Capturing 32.57 LCRs/s
    - Propagating 32.52 LCRs/s
    - Applying 30.63 LCRs/s

- **CMS**
  
  - CMSNOR.CERN.CH(CERN) → CMSR.CERN.CH(CERN)
    - Capturing 132.66 LCRs/s
    - Propagating 132.4 LCRs/s
    - Applying 87.44 LCRs/s
  
  - CMSNOR.CERN.CH(CERN) → INT2R.CERN.CH(CERN)
    - Capturing 78.06 LCRs/s
    - Propagating 77.91 LCRs/s
    - Applying 67.49 LCRs/s
• LHCB
Summary

• The LCG Database Deployment Project (LCG 3D) has set up a world-wide distributed database infrastructure for LHC
  – some 33 RAC clusters = 636 CPU cores at CERN + several tens of nodes at 10 partner sites are in production now

• Large scale tests have validated that the experiment are implemented by the RAC & streams based set-up
  – backup & recovery tests have been performed to validate the operational procedures at all sites

• Monitoring of database & streams performance has been implemented building on grid control and strmmon tools
  – key to maintain and optimize any larger system