14.10 Replication features

Nagaraju Inturi

nagaraju.inturi@hcl.com

HCL PRODUCTS & PLATFORMS

Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Agenda

- Smart triggers
  - Demo
- Asynchronous post commit triggers
  - Demo
- Secondary server performance improvements

HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Value Proposition

- Selectively trigger events based on changes in server data
- Real time ‘push’ notifications help clients avoid polling the server
- Small data flow allows simple small clients to work with many triggered events at once

HCL PRODUCTS & PLATFORMS

Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Use case: Banking

- Bank accounts
  - I want to be alerted when an account balance drops below zero dollars
  - I don’t want to write SPL or install stored procedures
  - I want to be notified in my client application
  - I don’t want to poll the database for this information or re-query each time a balance changes from the client

HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Smart Trigger Bank Code

```
public class BankMonitor implements IfmxSmartTriggerCallback {
     public static void main(String[] args) throws SQLException {
          IfxSmartTrigger trigger = new IfxSmartTrigger(args[0]); // pass in JDBC URL to SYSADMIN database
          trigger.timeout(5).label("bank_alert");
          trigger.addTrigger("account", "informix", "bank",
                    "SELECT * FROM account WHERE balance < 0", new BankMonitor());
          trigger.watch(); //blocking call
     }

     @Override
     public void notify(String json) {
          System.out.println("Bank Account Ping!");
          if (json.contains("ifx_isTimeout")) {
                    System.out.println("-- No balance issues");
          }
          else {
                    System.out.println("-- Bank Account Alert detected!");
                    System.out.println("   " + json);
          }
     }
}
```


HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Example event data documents

- **Sample output for <u>Insert</u> operation:**

  {“operation”:"insert",”table”:"creditcardtxns",”owner”:"informix",”database”:"creditdb",”label”:"card txn alert",”txnid”:2250573177224,"operation_owner_id":200,"operation_session_id":5,”commit_time”:1488243530,”op_num”:1,”rowdata”:{“uid”:22,”cardid”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T10:35:10.000Z } }}

- **Sample output for <u>Update</u> operation:**

  {“opertion”:"update",table:"creditcardtxns",”owner”:"informix",”database”:"creditdb",”label”:"card txn alert",”txnid”:2250573308360, "operation_owner_id":200,"operation_session_id":5,”commit_time”:1488243832,”op_num”:1,”rowdata”:{uid:21,cardid:"7777-7777-7777-7777",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":"25-Jan-2017 16:15"} },”before_rowdata”:{“uid”:21,”cardid”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T10:35:10.000Z } }}

- **Sample output for <u>Delete</u> operation:**

  {“opertion”:"delete",”table”:"creditcardtxns",”owner”:"informix",”database”:"creditdb",”label”:"card txn alert",”txnid”:2250573287760, "operation_owner_id":200,"operation_session_id":5,”commit_time”:1488243797,”op_num”:1,”rowdata”:{“uid”:22,”cardid”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T13:35:06.000Z } }}

- **Sample output for multi row document when maxrecs input attribute set to greater than 1:**

  {[

  {“operation”:"Insert",”table”:"creditcardtxns",”owner”:"informix",”database”:"creditdb",”label”:"card txn alert", “txnid”:2250573309999, "operation_owner_id":200,"operation_session_id":5,”commit_time”:1487781325,”op_num”:1,”rowdata”:{uid:"7",”cardid”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T15:10:10.000Z } }},

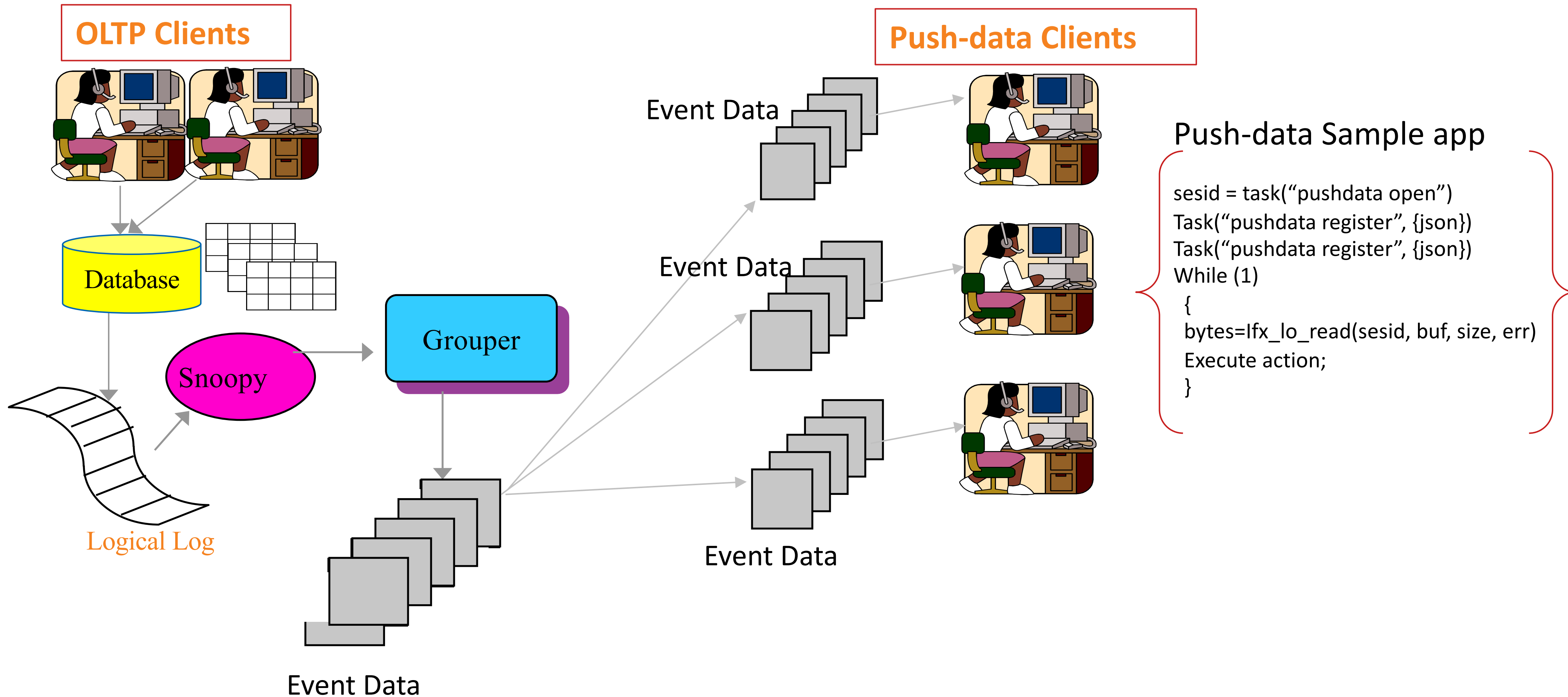
  {“operation”:"insert",table:"creditcardtxns",”owner”:"informix",”database”:"creditdb",”label”:"card txn alert",”txnid”:2250573177224, "operation_owner_id":200,"operation_session_id":5,”commit_time”:1488243530,”op_num”:1,”rowdata”:{“uid”:22,”cardid”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T16:20:10.000Z } }}

  ]}

HCL PRODUCTS & PLATFORMS

Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Architecture Diagram



Push-data Sample app

```
sesid = task("pushdata open")
Task("pushdata register", {json})
Task("pushdata register", {json})
While (1)
 {
 bytes=Ifx_lo_read(sesid, buf, size, err)
 Execute action;
 }
```


HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Demo!

HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Asynchronous post commit triggers – Replication to SPL routine – Streaming analytics

HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Use cases

- Realtime Streaming analytics on OLTP data
  - MIN,MAX,AVG,SUM for a group of records.
    - Example: Per store sales reports
  - Realtime leaderboard calculation for an online game
  - Build materialized views
- Data transformations
  - Add additional fields ---like store id– while replicating data to central server.
- Update external systems like graph database, Hadoop, Spark, Queuing services …
- “No Key” data replication support
  - Replicate data for tables that do not have primary key, unique index or ER key.

HCL PRODUCTS & PLATFORMS

Copyright © 2017 HCL Products & Platforms | www.hcltech.com

How it works? (1)

- Solution is based on Enterprise Replication
- As part of transaction replay, instead of applying data to target table, user defined stored procedure gets fired for insert, update and delete operations.
- Works with loopback replication
  - Source and target participant can be defined on the same table-- acts likes a post commit asynchronous trigger
  - Source and target tables can be on the same server instance either in the same database or in different database
- Target table can be on a different Enterprise Replication server instance.
- Target table is used for parsing replicated row and extracting column values– data will not be applied to target table
- User can specify where clause filter to fire SPL on specific dataset!

HCL PRODUCTS & PLATFORMS

Copyright © 2017 HCL Products & Platforms | www.hcltech.com

How it works?(2)

- Data is staged in ER queues for asynchronous processing

- Source server id, transaction id, transaction commit time and operation type(I/U/D) are passed in as argument values to the SPL routine along with user data.

- For update operation, both before and after image of the column values are passed in as arguments to SPL routine.

- SPL routine execution can be configured to be invoked as user informix or table owner.

HCL PRODUCTS & PLATFORMS

Copyright © 2017 HCL Products & Platforms | www.hcltech.com

New options to ‘cdr define replicate’ command

- --splname=<spl routine name>
  - Stored procedure routine name to apply data to. SPL routine must exist at all participants
  - Input arguments: operation type, source id, txnid, before image of the row column list, after image of the row column list
- --jsonsplname=<spl routine name>
  - Stored procedure routine name to apply data to. SPL routine must exist at all participants
  - Input arguments: json document
  - --jsonsplname option expects input arguments to splname routine to be a json datatype. With json document as input to SPL routine, same SPL routine can be used for registering ‘replication to SPL’ replicate definition on multiple tables. For certain use cases --like queueing data to message queues -- this makes developer job a lot easier.
  - --jsonsplname option is mutually exclusive to –splname option.
- --cascaderepl=y|n enable cascade replication
  - Required if replication to SPL needs to be executed for the data applied through Enterprise Replication

HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

--splname option stored procedure argument list

- Optype char(1) – operation type. Values include
  - I – Insert
  - U – Update
  - D – Delete
- Soucre_id integer – Source server id. Same as group id.
- Committime integer – Transaction commit time.
- Txnid bigint – Transaction id.
- Before value column list.
- After value column list.
  - Note: Column list for SPL routine extracted from select statement projection list

HCL PRODUCTS & PLATFORMS

Copyright © 2017 HCL Products & Platforms | www.hcltech.com

--jsonsplname option SPL routine json argument

Attribute name	Description
operation	Operation type: Insert/Delete/Update
table	Table name
owner	Table owner
database	Database name
txnid	8 byte unique id. Higher order 4 bytes: commit work log id, lower order 4 bytes: commit work log position.
commit_time	Transaction commit time for the event data.
rowdata	Row data in JSON document format. Data is returned in column name as key and column data as value.
before_rowdata	Before row data for “update” operation.

Example document format :

{“operation”:"insert",”table”:"creditcardtxns",”owner”:"informix",”database”:"creditdb",”txnid”:2250573177224,”commit_time”:1488243530,rowdata”:{“uid”:22,”car did”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T10:35:10.000Z } }}

{“opertion”:"update",table:"creditcardtxns",”owner”:"informix",”database”:"creditdb”,”txnid”:2250573308360,”commit_time”:1488243832,”rowdata”:{uid:21,cardid: "7777-7777-7777-7777",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":"25-Jan-2017 16:15"} },”before_rowdata”:{“uid”:21,”cardid”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T10:35:10.000Z } }}

{“opertion”:"delete",”table”:"creditcardtxns",”owner”:"informix",”database”:"creditdb",”txnid”:2250573287760,”commit_time”:1488243797,rowdata”:{“uid”:22,”car did”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T13:35:06.000Z } }}

HCL PRODUCTS & PLATFORMS

Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Asynchronous post commit trigger support

- Define loopback replication server
- Create ‘replication to SPL’ type replicate with same “database and table” information for both source and target participants. Loopback server group name shall be specified with target participant definition:
- Example:
  - cdr define repl rep1 -C always -S row -M g_cdr_utm_nag_1 -A -R --splname=logger4repl2spl "test@g_mygroup:informix.t1" "select * from t1" "test@g_loopback:informix.t1" "select * from t1”
    - Note: g_mygroup is the local server ER group, and g_loopback is the pseudo ER server group.

HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Use case 1 – Build staging table for data changes (--jsonsplname example)

```
create database test with log;

create table t1 (c1 int , c2 int);

create table t2 (col1 int , col2 float);

create table staging (data json);

create procedure logger4repl2spl (data json)

   insert into staging values (data);

end procedure;

$ cdr define repl rep1 -C always -S row -M g_cdr_utm_nag_1 -A -R --jsonsplname=logger4repl2spl "test@g_mygroup:informix.t1" "select * from t1" "test@g_loopback:informix.t1" "select * from t1”

$ cdr define repl rep2 -C always -S row -M g_cdr_utm_nag_1 -A -R --jsonsplname=logger4repl2spl "test@g_mygroup:informix.t2" "select * from t2" "test@g_loopback:informix.t2" "select * from t2”

$ cdr start repl rep1

$ cdr start repl rep2
```

HCL PRODUCTS & PLATFORMS

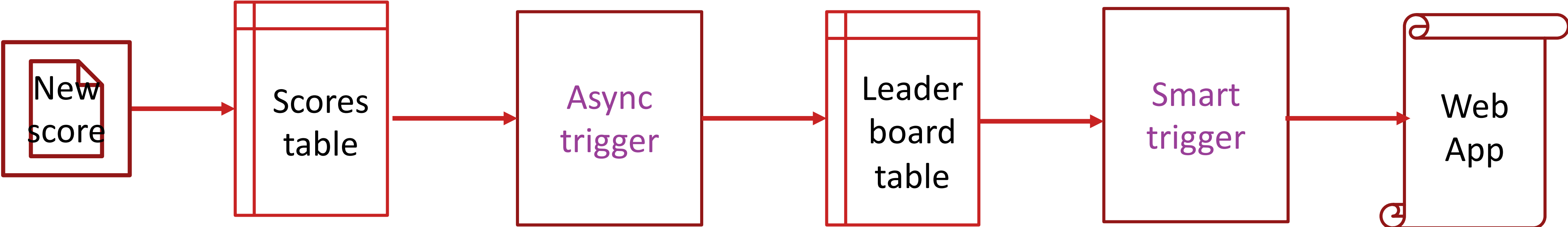
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Use case 2 – Realtime aggregation framework

```
drop database retaildb;

create database retaildb with log;

create table sales (customerid int, storeid int , bill_amount float);

create table sales_summary(storeid int , s_count int, s_sum float, s_avg float, s_min float, s_max float );

CREATE PROCEDURE store_agg(opType char(1), srcid integer, committime integer, txnid bigint, customerid_bef integer, storeid_bef int, bill_amount_bef float, customerid int, storeid_aft int , bill_amount float)

---- -----

----  ----

END PROCEDURE;

$ cdr define repl rep1 -C always -S row -M utm_group_1 -A -R --serial --splname=store_agg "retaildb@g_mygroup:informix.sales" "select * from sales" "retaildb@g_loopback:informix.sales" "select * from sales"

$ cdr start repl rep1
```


HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Use case 3: Leader board calculation



HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Use case 4 – Publish data to MQTT using Java or C UDR

```
create database mqtt with log;

create table customer (name char(128), id int);

execute procedure sqlj.install_jar('file:$INFORMIXDIR/jars/mqtt_trigger.jar', 'mqtt_trigger_jar',1);

$cdr define repl mqrepl -C always -S row -M g_informix -A -R --serial --jsonsplname=mqtt_put
"mqtt@g_informix:informix.customer" "select * from customer" "mqtt@g_lb:informix.customer" "select * from customer"

$ dbaccess mqtt -

  > insert into customer values(”Bill", 1);

$ mosquitto_sub -t 'test/topic' -v

test/topic
{"operation":"insert","table":"customer","owner":"informix","database":"mqtt","txnid":21475983636,"commit_time":1550879224,"rowdata":{"name":”Bill","id":1 }}
```


HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Comparing Async triggers, Smart triggers and CDC

Async triggers	Smart triggers	CDC
Async trigger logic written in SPL, C and Java UDRs.	Trigger logic written in application and middleware service.	Data processing done in client application
Designed for real-time streaming analytics on OLTP data	Designed for event processing and business logic exception handling	Designed for Data streaming/replication
Can register where clause	Can register where clause	No where clause support
Data in SQL or JSON format	Data in JSON format	Byte stream
Push technology	Push technology	Push technology
Only committed transactions sent for async trigger execution	Only committed transactions sent to Smart Trigger analysis	All records returned to the user including rollbacked operations
QoS: At least once delivery	QoS: At most once delivery	QoS: At least once delivery. CDC can read old log files

HCL PRODUCTS & PLATFORMS

Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Demo

HCL PRODUCTS & PLATFORMS

Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Log replay performance

HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Log replay performance

- More than 500% improvement in log replay performance for RSS, SDS and HDR secondary servers.
- Similar improvement noticed for crash recovery performance

HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

What's changed ?

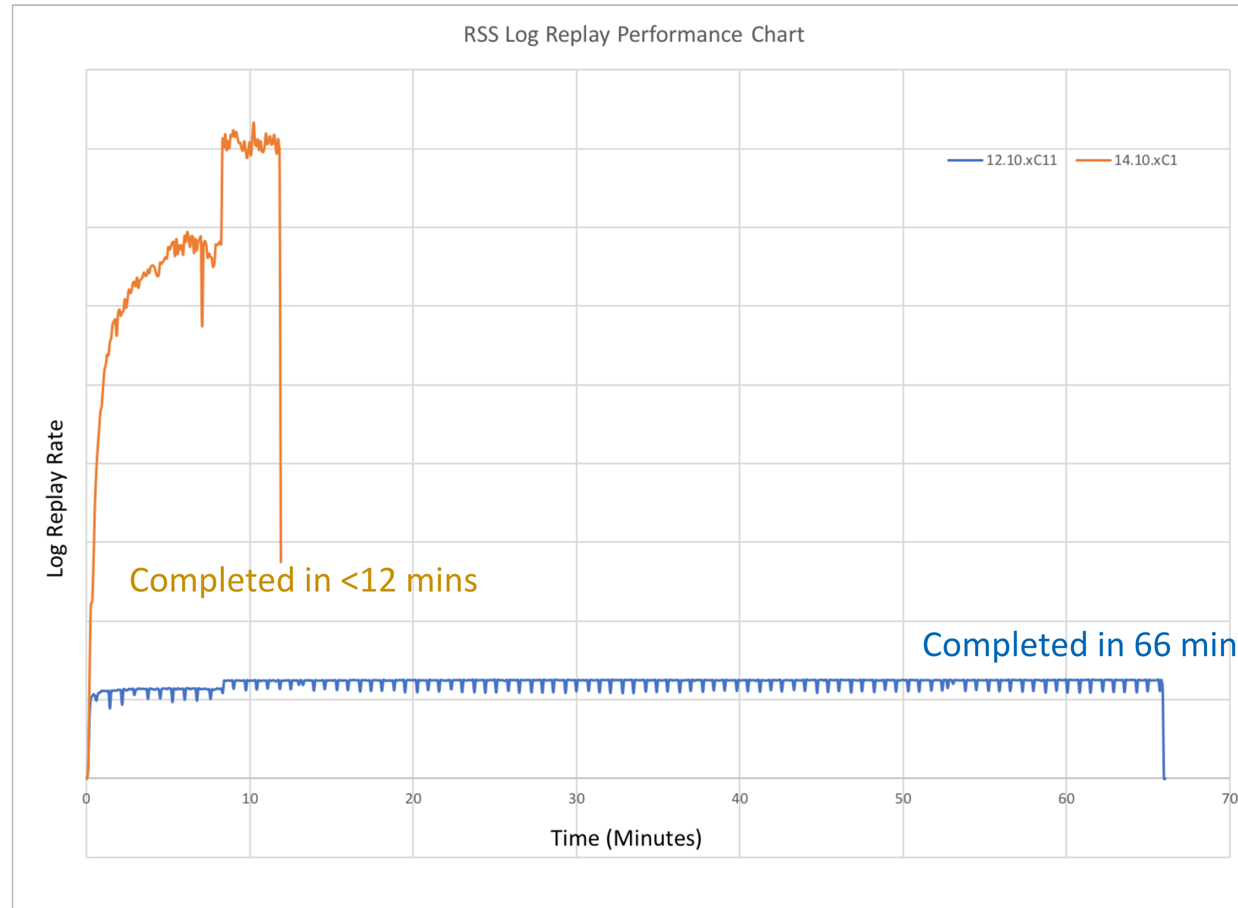
- Improvement to log replay performance
  - Minimized latch and thread communication overhead
  - Improvements to Read-ahead functionality at secondary server
  - Non-blocking checkpoints at RS secondary server

- Benefits:
  - Near zero latency for replication between Primary and Secondary servers (RSS, SDS and HDR).
    - This enables customers to offload applications from primary server to one or more secondary servers.
    - This also helps customers to meet recovery point objective in case of disaster scenarios.
  - Impropved crash recovery performance
    - This helps customers improve on recovery time object as crash recovery time is 5 times better than before!
  - Improved log restore performance
    - This helps customers improve on recovery time object in scenarios where server needs to be restored from backup.

HCL PRODUCTS & PLATFORMS

Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Performance comparison chart for 8 minute workload at primary



Copyright © 2017 HCL Products & Platforms | www.hcltech.com

onstat changes (1)

- Added replication latency, and log replay rate to ‘onstat –g laq’
  - Note: Log replay rate is only available with “-r” option.

- Example:

```
Secondary Apply Queue:     Total Buffers:12 Size:2048K Free Buffers:0
Log Recovery Queue:        Total Buffers:12 Size:20480K Free Buffers:0
Log Page Queue:            Total Buffers:512 Size:4K Free Buffers:1
Log Record Queue:          Total Buffers:1000 Size:16K Free Buffers:1

Transaction Latency: 1 seconds
Apply rate: 347887.41 recs/sec
```


HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

New onconfig parameters(1)

- SEC_APPLY_POLLTIME
  - In micro seconds, how long apply thread should poll for new work before yielding.
  - Recommended value for smaller systems (between 1 to 8 CPUVPs): 0
  - Recommended value for larger systems(>= 16 CPUVPS): 1000
  - Recommended to move poll threads to NETVP
- SEC_LOGREC_MAXBUFS
  - Configure number of log buffers to be used for replaying log records at secondary server. Each log buffer of size 16KB.
  - Recommended value : <= 1000 buffers
- RSS_NONBLOCKING_CKPT
  - 1 - Enable non-blocking checkpoint at RS secondary server.

HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

New onconfig parameters(2)

- SEC_DR_BUFS
  - Number of DR buffers to use for replication.
  - Applicable to both HDR primary and HDR/RSS/SDS secondary servers
  - Buffer size same as LOGBUFF size
  - Supported values : >=12

HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

OFF_RECOVERY_THREADS and LTAPEBLK recommendation

- Original recommendation:
  - 4 times CPUVPs
- New recommendation
  - Not more than 23 threads!
  - 5, 7, 11 or 23.
- LTAPEBLK
  - Recommended value: 20480 (20MB)

HCL PRODUCTS & PLATFORMS
Copyright © 2017 HCL Products & Platforms | www.hcltech.com

Questions ?

© 2014 IBM Corporation