

# Workload Processor

This presentation shows the basic use case for the following solutions

- WLP – Workload Processor
- WLA – Workload Analytics
- WLT – Workload Test

# Workload Processor

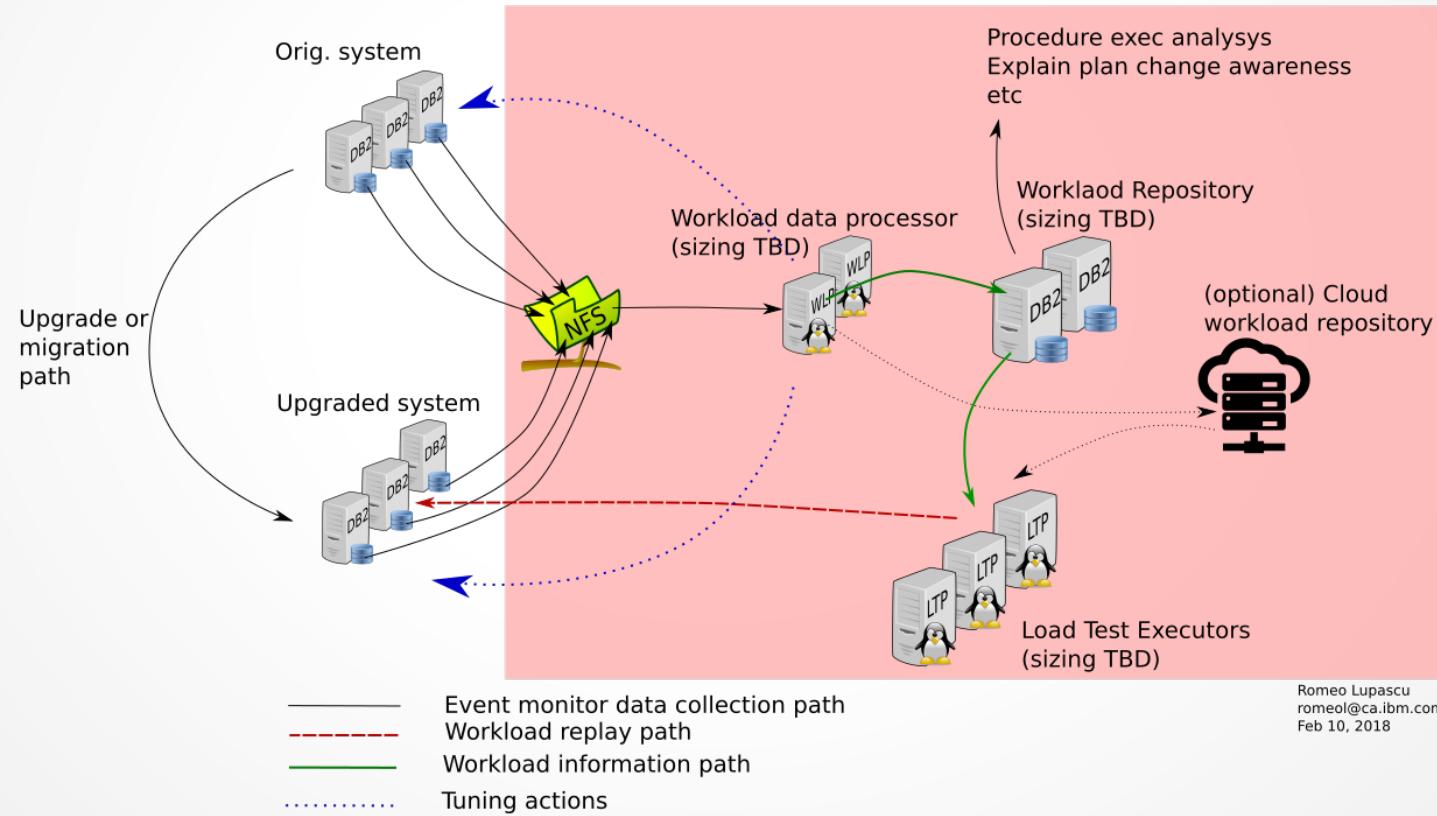
- Architecture
- Typical deployment use case
- Installing the WLPT package
- Basic connectivity configuration
- Preparing the database for workload capture
- Create a new load test descriptor
- Execute a load test
- Check test outcome
- Compare performance by using histograms

# Workload Processor

The big picture  
Architecture

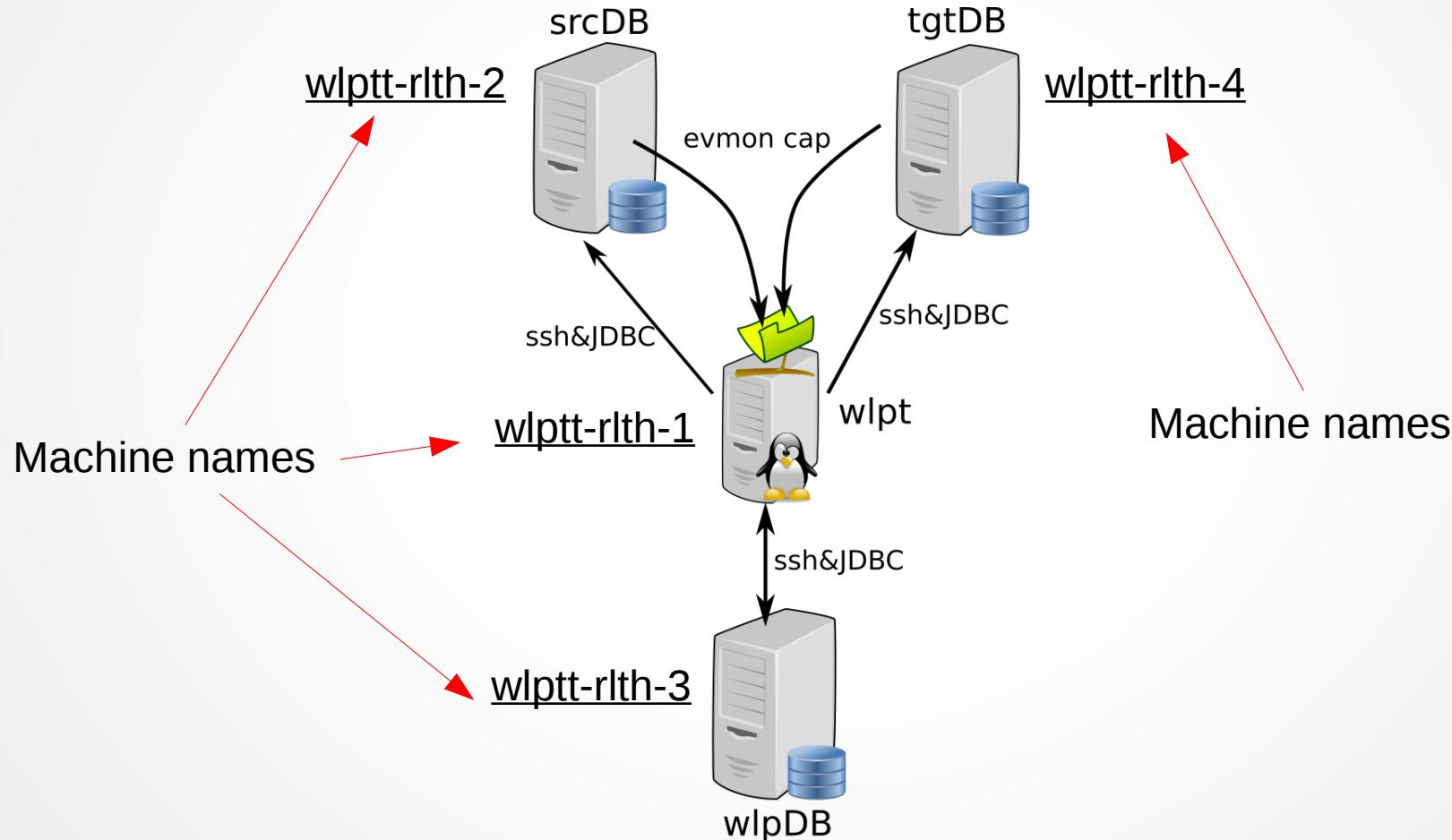
# Workload Processor

## Workload Processing and Reporting



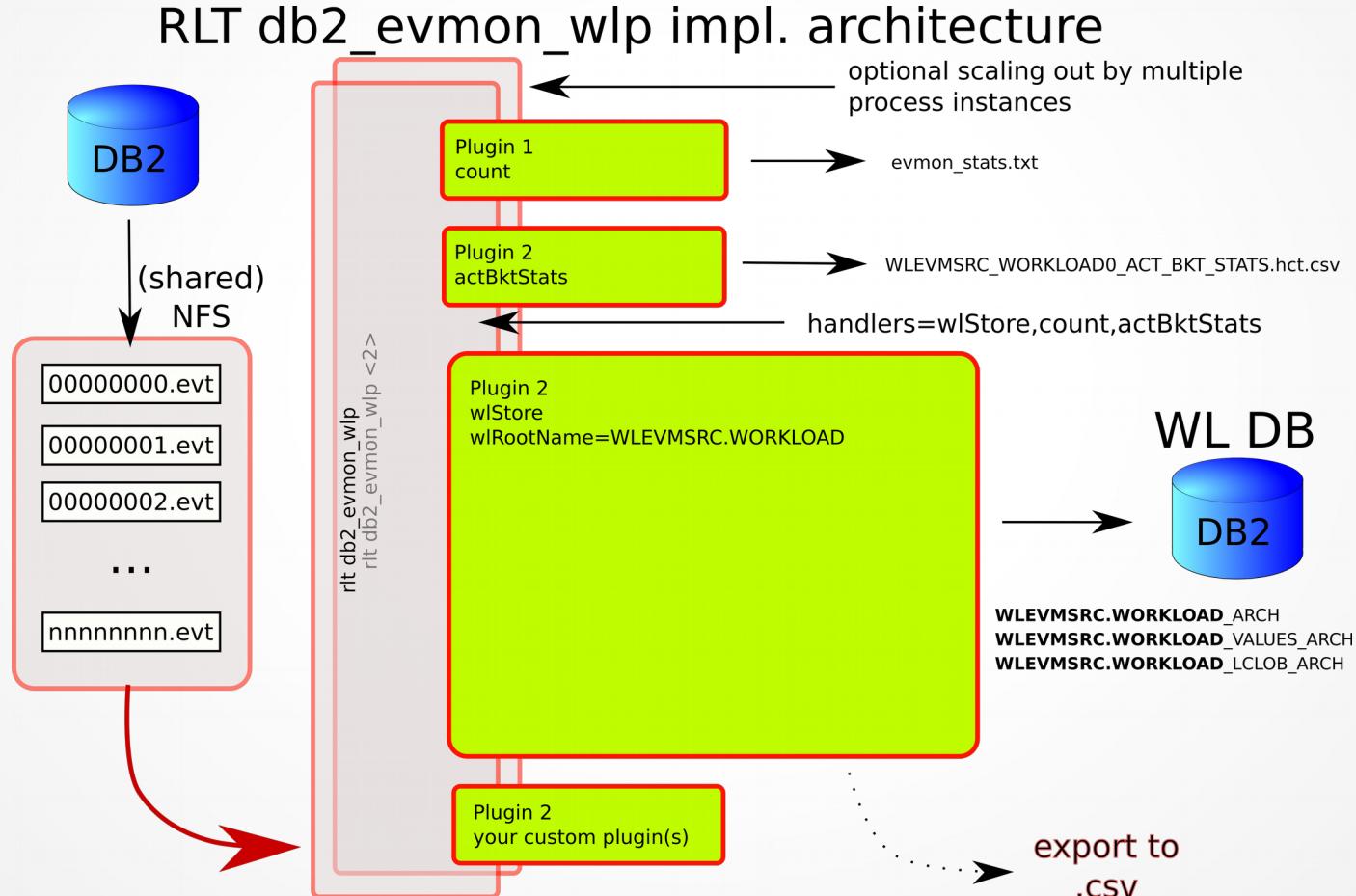
The Typical Use Case

# Workload Processor



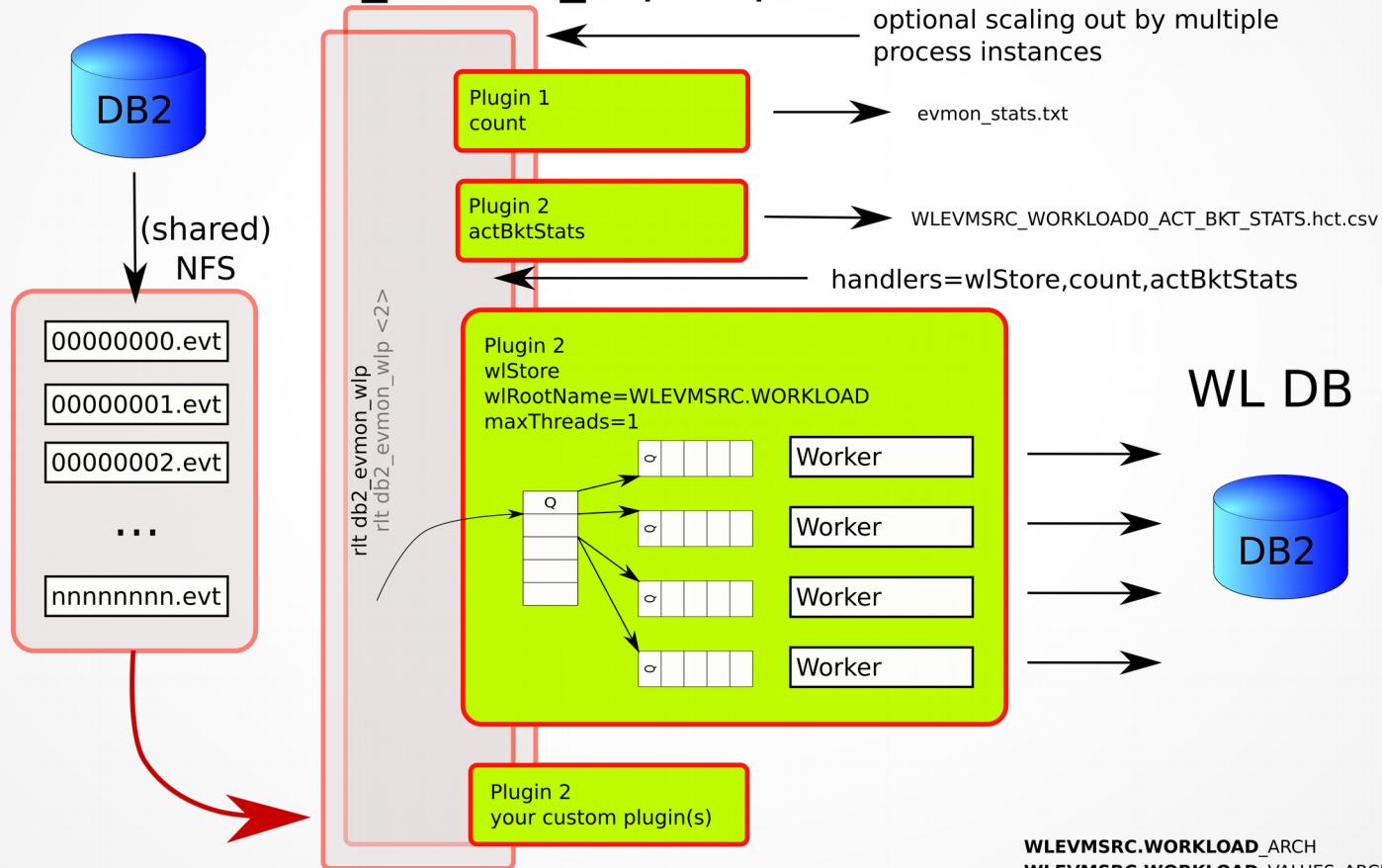
The DB2 event file parser architecture  
(command db2\_evmon\_wlp)

# Workload Processor



# Workload Processor

## RLT db2\_evmon\_wlp impl. architecture



Installing RLT software

# Workload Processor

- Find your java executable, check type and version, recommended IBM or Oracle version 1.8 or higher
    - ]\$ java -version
  - Create a work folder for the rlt package and install it from the downloaded archive
    - ]\$ mkdir ~/rlt
    - ]\$ cd rlt
  - Verify the jar integrity and IBM certificate CN
    - ]\$ jarsigner -verify rlt\_db2wlpt\_<timestamp>.jar  
jar verified.
    - jarsigner -verify -certs -verbose rlt\_db2wlpt\_<timestamp>.jar | grep IBM  
...  
X.509, CN=International Business Machines Corporation, OU=IBM CCSS, O=International Business Machines Corporation, L=Armonk, ST>New York, C=US  
...
  - Install the software
    - ]\$ java -jar rlt\_db2wlpt\_<timestamp>.jar
- replace <timestamp> with an actual value example rlt\_db2wlpt\_2018\_11\_04\_20\_11\_32\_831.jar

# Workload Processor

```
[wlptu@wlptt-rlth-1 rlt]$ java -jar /software/rlt_db2wlpt_2018_11_09_14_53_29_650.jar
```

(C) Copyright IBM Corp. 2018

Licensed under the Apache License, Version 2.0 (the "License");  
you may not use this file except in compliance with the License.  
You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software  
distributed under the License is distributed on an "AS IS" BASIS,  
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
See the License for the specific language governing permissions and  
limitations under the License.

If you accept above conditions type YES+ENTER: YES

# Workload Processor

RLT Environment Main Properties	
Property Name	Property Value
Date	Fri Nov 09 11:57:17 PST 2018
Build timestamp	2018_11_09_14_22_53
Build ID	e1bf676e-5b52-4e0e-b42e-2d17513dd7bf
Instance Id	null
Installed in	/home/wlptu/bin/rlt_db2wlpt 2018_11_09_14_53_29_650
Java Home	/opt/ibm/db2/v11.1/java/jdk64/jre
RLT Home folder	/home/wlptu
RLT database folder	/home/wlptu/.rlt
RLT log config source	/home/wlptu/.rlt/rvt_log.properties
RLT log active levels	INFO WARN SEVERE
RLT Home folder	/home/wlptu
RLT_execHist.max_files	100
Kerberos5 config file	Default
JAAS config file	Default
User Home	/home/wlptu
User Name	wlptu
User Language	en
User Timezone	America/Los_Angeles
User Country	US
OS Name	Linux
OS Version	3.10.0-957.el7.x86_64
OS Architecture	amd64
OS Encoding	UTF-8
Memory max (MB)	512
Memory total (MB)	41
Memory free (MB)	11
Available processors	2
java.class.version	52.0
java.fullversion	JRE 1.8.0 IBM J9 2.8 Linux amd64-64 Compressed References 20 160106_284759 (JIT enabled, AOT enabled) J9VM - R28_20160106_1341_B284759 JIT - tr.r14.java_20151209_107110_02 GC - R28_20160106_1341_B284759_CMPRSS J9CL - 20160106_284759
java.runtime.version	pxa6480sr2fp10-20160108_01 (SR2 FP10)
java.specification.version	1.8
java.version	1.8.0
java.vm.specification.version	1.8
java.vm.version	2.8

The information table logged by the setup command and later on by using the command:  
]\${ rlt info }

Create or check your ~/env.properties

To get a commented template of env.properties  
use the command

```
]$ rlt env_help
```

# Workload Processor

Check RLT database alias configuration in ~/env.properties

Classic JDBC connection info example with encrypted password

```
srcdbAdm.connection.password=U/8IC03nRcFS42t6xQPXIItlTRusAeBox0QDJmxhL4xQ1Vpval2ljz5GebXj8rIti  
srcdbAdm.connection.password.crypt=yes  
srcdbAdm.connection.url=jdbc:db2://srcdb:50000/SAMPLE:\  
retrieveMessagesFromServerOnGetMessage=true;clientProgramName=RLTCmd;  
srcdbAdm.connection.user=db2inst1
```

Kerberos5 JDBC connection info example in password-less mode

SecurityMechanism=11 and KerberosServerPrincipal attributes are required

Both the user and password values must be   NONE  

```
krbTestPl.connection.url=jdbc:db2://lacy1.fyre.ibm.com:50000/sample:\  
retrieveMessagesFromServerOnGetMessage=true;clientProgramName=RLTCmd;\  
securityMechanism=11;KerberosServerPrincipal=lacy1.fyre.ibm.com@DB2KDB.FYRE.IBM.COM;  
krbTestPl.connection.user=_NONE_  
krbTestPl.connection.password=_NONE_
```

Linux shell integration

# Workload Processor

To execute rlt commands you need to use the ‘rlt’ main command then the rlt command name you wish to invoke.

For Linux OS-es you can avoid typing ‘rlt’ each time by creating command aliases.

RLT framework help you by providing a script to create aliases for each known rlt command and command profile (see next).

To create rlt aliases in your current login session simply “source” the content of the \$HOME/bin/rlalias script.

Once defined aliases also allows you to use the shell line auto-complete feature to discover command names.

Example:

```
[...]$ . rlalias  
[...]$ db_aliases  
... (command output)
```

Understanding command profiles

Any command in the RLT framework has valid default values for all its parameters. This means you can execute it by just using its name.

To keep things simple and for defining user services, command profiles can be used to define new default values for any/all of the command parameters and store them as a command profile.

Each command profile is stored in a property file

```
~/.rlt/RLT/<command-name>.<profile-name>.properties
```

List all profile names or a subset including a string

```
[... rlt]$ rlt rlt_profile_list
```

```
[... rlt]$ rlt rlt_profile_list .<profile-part-name>
```

# Workload Processor

Create a new command profile:

- 1) By using specific command named parameter 'save\_to\_profile'

Example:

```
[... rlt]$ rlt db_wlt_replay wlpdb uuid=712d610a-e33b-4de9-af0e-08256f4b6fc2  
save_to_profile=w
```

- 2) Direct edit or create (from another program) the command profile file

List the content of a profile (parameters and environment variables)

```
[... rlt]$ rlt rlt_profile_list <profile-name>
```

# Workload Processor

## Create a new command profile

Specifying commands environment parameters,  
more important JVM memory

Manually edit the command's profile and add RLT\_JAVA\_MEM\_OPTS

```
[... rlt]$ rlt rlt_profile_list db_wlt_replay.wlpdb
-----
Command profile definition
Profile:db_wlt_replay.wlpdb
Profile file:/home/<user>/.rlt/RLT/db_wlt_replay.wlpdb.properties
-----
srcAlias=wlpdb
cfgFQNR=WLPEXEC
uuid=773c9440-b334-4e47-8fa8-53873b4c9696
-----
[... rlt]$ vi /home/<user>/.rlt/RLT/db_wlt_replay.wlpdb.properties

# add 1 or 2 Gb (or more) for a server process
RLT_JAVA_MEM_OPTS="-Xms1024m -Xmx2048m"
```

Optional  
Verify the connectivity for all aliases

# Workload Processor

List all JDBC connection alias names by using the command

```
]$ rlt db_aliases
```

```
[wlptu@wlptt-rlth-1 rlt]$ rlt db_aliases
-----
In file:/home/wlptu/env.properties
-----
krbTestPl
srcdbAdm
srcdbTest
tgtdbAdm
tgtdbTest
wlpdb -> default
[wlptu@wlptt-rlth-1 rlt]$
```

Test one or a number of databases aliases connectivity by using the command

```
]$ rlt db_ping srcdbAdm,srcdbTest,tgtdbAdm,tgtdbTest,wlpdb
```

# Workload Processor

```
[wlptu@wlptt-rlth-1 rlt]$ rlt db_ping srcdbAdm,srcdbTest,tgtmdbAdm,tgtmdbTest,wlpdb
Trying to connect to database
URL:jdbc:db2://srcdb:50000/SAMPLE:retrieveMessagesFromServerOnGetMessage=true;clientProgramName=RLTCmd;
user:db2inst1
Database server:DB2/LINUXX8664 SQL10059
Connect (only) duration:637 ms
Trying to connect to database
URL:jdbc:db2://srcdb:50000/SAMPLE:retrieveMessagesFromServerOnGetMessage=true;clientProgramName=RLTCmd;
user:db2inst1
Database server:DB2/LINUXX8664 SQL10059
Connect (only) duration:521 ms
Trying to connect to database
URL:jdbc:db2://tgtmdb:50000/SAMPLE:retrieveMessagesFromServerOnGetMessage=true;clientProgramName=RLTCmd;
user:db2inst1
Database server:DB2/LINUXX8664 SQL11010
Connect (only) duration:561 ms
Trying to connect to database
URL:jdbc:db2://tgtmdb:50000/SAMPLE:retrieveMessagesFromServerOnGetMessage=true;clientProgramName=RLTCmd;
user:db2inst1
Database server:DB2/LINUXX8664 SQL11010
Connect (only) duration:460 ms
Trying to connect to database
URL:jdbc:db2://wlpdb:50000/WLPDB:retrieveMessagesFromServerOnGetMessage=true;clientProgramName=RLTCmd;
user:db2inst1
Database server:DB2/LINUXX8664 SQL11010
Connect (only) duration:422 ms
Active:srcdbAdm,srcdbTest,tgtmdbAdm,tgtmdbTest,wlpdb
Inactive:NONE
[wlptu@wlptt-rlth-1 rlt]$
```

# Workload Processor

Optional  
Check NFS shared storage

# Workload Processor

```
[wlptu@wlptt-rlth-1 rlt]$ cat /etc/exports  
/db2evmondata *(rw,root_squash)  
  
[wlptu@wlptt-rlth-1 rlt]$ tree /db2evmondata/  
/db2evmondata/  
|-- srcdb  
|   '-- sample  
|     '-- actwlevmon  
|       |-- 00000000.evt  
|       |-- 00000000.evt.lck  
|       '-- db2event.ctl  
|-- tgtdb  
|   '-- sample  
|     '-- actwlevmon  
|       |-- 00000000.evt  
|       |-- 00000000.evt.lck  
|       '-- db2event.ctl  
  
6 directories, 6 files  
[wlptu@wlptt-rlth-1 rlt]$
```

Shared folder nfs export

src DB2 sample DB capture

tgt DB2 sample DB capture

WLP host

On the machine where the wlpt software is installed (or a dedicated nfs server) create a folder called /db2evmondata end export it to the machines which hosts the DB2 instances you are going to capture event monitor data

# Workload Processor

The recommended (but not required) folder structure for storing event monitor data is:

```
/db2evmondata/<inst-name>/<db-name>/actevmon
```

inst-name: is the logical (you decide it) or host name of the DB2 instance

db-name: is the name of the DB2 database from where the event monitor data is collected

# Workload Processor

```
[db2inst1@wlptt-rlth-2 ~]$ tree /db2evmondata/  
/db2evmondata/  
└── srcdb  
    └── sample  
        └── actwlevmon  
            ├── 00000000.evt  
            └── 00000000.evt.lck  
            └── db2event.ctl  
└── tgtdb  
    └── sample  
        └── actwlevmon  
            ├── 00000000.evt  
            └── 00000000.evt.lck  
            └── db2event.ctl  
  
6 directories, 6 files  
[db2inst1@wlptt-rlth-2 ~]$
```

Host srcdb DB2 DB locally mounted NFS share

# Workload Processor

Slide:29

```
[db2inst1@wlptt-rlth-4 ~]$ tree /db2evmondata/
/db2evmondata/
├── srcdb
│   └── sample
│       └── actwlevmon
│           ├── 00000000.evt
│           ├── 00000000.evt.lck
│           └── db2event.ctl
└── tgtdb
    └── sample
        └── actwlevmon
            ├── 00000000.evt
            ├── 00000000.evt.lck
            └── db2event.ctl
6 directories, 6 files
[db2inst1@wlptt-rlth-4 ~]$
```

Host tgtdb DB2 DB locally mounted NFS share

Check the event monitor existence and state for source and target DB2 databases

# Workload Processor

Slide:31

```
[wlptu@wlptt-rlth-1 rlt]$ rlt db2_evmon_list srcdbAdm
```

```
Database host:port:srcdb:50000  
alias:srcdbAdm  
user:db2inst1
```

```
select m.evmonname,e.type,  
case when event_mon_state(m.evmonname)=1  
then 'true'  
else 'false'  
end as enabled,owner,autostart,target_type,target,e.filter from syscat.eventmonitors m,  
syscat.events e  
where e.evmonname=m.evmonname  
order by 1
```

EVMONNAME VARCHAR(128)	TYPE VARCHAR(128)	ENABLED VARCHAR(5)	OWNER VARCHAR(128)	AUTOSTART CHAR(1)	TARGET CHAR(1)	TYPE VARCHAR(762)	FILTER CLOB(65536)
DB2DETAILDEADLOCK WLPACTCAP	DETAILDEADLOCKS ACTIVITIES	true true	DB2INST1 DB2INST1	Y Y	F F	db2detaildeadlock /db2evmondata/srcdb/sample/actwlevmon	null null

Elapsed time:27 ms

Event mon. state

Capture folders

```
[wlptu@wlptt-rlth-1 rlt]$
```

# Workload Processor

```
[wlptu@wlptt-rlth-1 rlt]$ rlt db2_evmon_list tgtdbAdm
```

Database host:port:tgtdb:50000  
alias:tgtdbAdm  
user:db2inst1

```
select m.evmonname,e.type,
case when event_mon_state(m.evmonname)=1
then 'true'
else 'false'
end as enabled,owner,autostart,target_type,target,e.filter from syscat.eventmonitors m,
syscat.events e
where e.evmonname=m.evmonname
order by 1
```

EVMONNAME VARCHAR(128)	TYPE VARCHAR(128)	ENABLED VARCHAR(5)	OWNER VARCHAR(128)	AUTOSTART CHAR(1)	TARGET CHAR(1)	TYPE VARCHAR(762)	FILTER CLOB(65536)
DB2DETAILEDLOCK	DETAILDELOCKS	true	DB2INST1	Y	F	db2detaildeadlock	null
WLPACTCAP	ACTIVITIES	true	DB2INST1	Y	F	/db2evmondata/tgtdb/sample/actwlevmon	null

Elapsed time:34 ms

Event mon. state

Capture folders

```
[wlptu@wlptt-rlth-1 rlt]$
```

Create a new DB2 event monitor for activities with capture in files for source DB2 and target DB2 databases if one does not exists and enable workload activity capture

# Workload Processor

Slide:34

Create the event monitor if not exists (fails if one with same name exists)

```
]$ rlt db2_evmon_create_act_files srcdbAdm WLPACTCAP /db2evmondata/srcdb/sample/actwlevmon/  
maxFileSizePages=262144 simulate=no  
]$ rlt db2_evmon_enable srcdbAdm WLPACTCAP
```

Check the event monitor state

```
]$ rlt db2_evmon_list srcdbAdm
```

Activate the activity capture at workload level and create a restore DDL file in the current folder

```
]$ rlt db2_wl_act_cap_enable srcdbAdm includeActuals=no simulate=no
```

Check if the activity capture is set and values capture enabled

```
]$ rlt db2_wl_list srcdbAdm
```

Check the state of the RLT service(s) for workload replay

- The db\_wlt\_reply command can be used to execute workload tests by replying the activities described in the workload test definition
- The command is designed to be used as a continuously running process searching for new active test definitions to attach to and start execute statements as programmed
- To create a new RLT service first create a command profile then install it as a service; see the example below (check commands help for full parameter list)

```
[... rlt]$ rlt db_wlt_replay wlpdb save_to_profile=wlpdb
[... rlt]$ echo 'RLT_JAVA_MEM_OPTS="-Xms1024m -Xmx2048m"' >> $HOME/.rlt/RLT/db_wlt_replay.wlpdb.properties
[... rlt]$ rlt rlt_svc_install db_wlt_replay.wlpdb 5min
```

# Workload Processor

Service/command-profile names

Linux cron schedules

```
[wlptu@wlptt-rlth-1 rlt]$ rlt rlt_svc_list
```

Service	Suspended	State	StartTime	Schedule
db_wlt_replay.wlpdb	no	not running	Thu Mar 07 19:11:01 PST 2019	* * * * * /home/wlptu/bin/db_wlt_replay.wlpdb
rlt_rest_server_start.base	yes	suspended	Wed Dec 31 16:00:00 PST 1969	* * * * * /home/wlptu/bin/rlt_rest_server_start.base
rlt_svcc.main	yes	suspended	Wed Dec 31 16:00:00 PST 1969	* * * * * /home/wlptu/bin/rlt_svcc.main

```
[wlptu@wlptt-rlth-1 rlt]$
```

Service execution status

# Workload Processor

Slide:38

Listing all RLT Linux processes

```
[wlptu@wlptt-rlth-1 rlt]$ rlt ps  
wlptu 23481 1 38 19:22 pts/0 00:00:03 /opt/ibm/db2/v11.1/java/jdk64/jre/bin/java db_wlt_replay.wlpdb  
[wlptu@wlptt-rlth-1 rlt]$
```

Process PID

Process executable

Command profile names

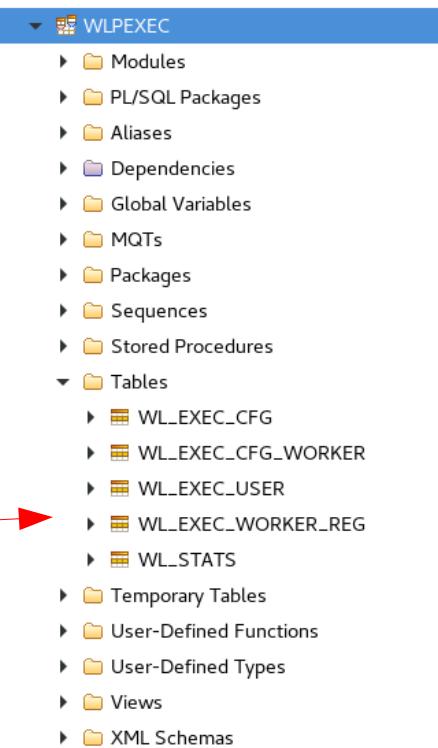
# Workload Processor

Ensure that your Load Test database objects are created

# Workload Processor

```
[wlptu@wlptt-rlth-1 rlt]$ rlt db wlt.a setup  
Creating table:WLPEXEC.WL_EXEC_CFG  
Creating table:WLPEXEC.WL_EXEC_CFG_WORKER  
Creating table:WLPEXEC.WL_EXEC_WORKER_REG  
Creating table:WLPEXEC.WL_EXEC_USER  
Creating table:WLPEXEC.WL_STATS  
[wlptu@wlptt-rlth-1 rlt]$
```

Creating (if not exists) database objects for load test definitions and results



Those tables are used to define execute and compare load tests by using the workload information captured by the WLP subsystem

Load test schema and objects  
(IBM DataStudio view)

# Workload Processor

Execute a simulated workload against the source database

Or wait for the regular workload to be executed by the applications

# Workload Processor

If you are testing your workload test env then you can simulate load by Executing the test workload insert 10 rows in 10 insert statements

```
[... rlt]$ rlt db_wlt_generate_seq_block_insert srcdbTest insertRowCount=10 batchSize=1
```

```
generated
[wlptu@wlptt-rlth-1 rlt]$ rlt db_wlt_generate_seq_block_insert srcdbTest insertRowCount=10 batchSize=1
Finalized block insert in table:RLT_TEST.TEST_TEST_TBL next PK:1010 inserted rowCount:10
[wlptu@wlptt-rlth-1 rlt]$
```

# Workload Processor

The screenshot shows the Workload Processor interface. On the left, a tree view of a database structure is displayed, with the 'Tables' node expanded to show 'TEST\_TEST\_TBL'. A red box highlights the connection entry 'rltwlp\_srcdb\_fyre\_db2inst1'. On the right, a table viewer titled 'RLT\_TEST.TEST\_TEST\_TBL' displays 10 rows of data. The columns are labeled 'PK [INTEGER]', 'DATA [VARCHAR(1024 OCTETS)]', 'TS [TIMESTAMP]', and 'BLOBDATA [BLOB]'. The data is as follows:

PK [INTEGER]	DATA [VARCHAR(1024 OCTETS)]	TS [TIMESTAMP]	BLOBDATA [BLOB]
1 0	D000000000000000000000000000000	2018-06-19 15:03:	
2 1	D000000000000000000000000000001	2018-06-19 15:03:	
3 2	D000000000000000000000000000002	2018-06-19 15:03:	
4 3	D000000000000000000000000000003	2018-06-19 15:03:	
5 4	D000000000000000000000000000004	2018-06-19 15:03:	
6 5	D000000000000000000000000000005	2018-06-19 15:03:	
7 6	D000000000000000000000000000006	2018-06-19 15:03:	
8 7	D000000000000000000000000000007	2018-06-19 15:03:	
9 8	D000000000000000000000000000008	2018-06-19 15:03:	
10 9	D000000000000000000000000000009	2018-06-19 15:03:	

A red arrow points from the text below to the table viewer.

Connection : 9.30.56.207 - db2inst1 - rltwlp\_srcdb\_fyre\_db2inst1

The new created table and its row content. PK and data content can be used to verify the validity of the command

# Workload Processor

Load the event monitor workload data for the src db in the workload database

# Workload Processor

To load the captured workload in the workload database define a command profile if not already defined then execute the batch load (as many times as needed) by using the command profile

```
[... rlt]$ rlt db2_evmon_wlp src=/db2evmondata/srcdb/sample/actwlevmon/  
dbFolder=/db2evmondata/srcdb/sample/actwlevmon/ wlRootName=WLEVMSRC dbAlias=wlpdb  
save_to_profile=srcdb_actevm_batch
```

Executing the batch load by using the command profile defined above. By using rlt the command will be executed in background (equivalent to nohup rlt ... &) in order to handle longer time unassisted loads.

```
[... rlt]$ mkdir wlp_batch_load  
[... rlt]$ cd wlp_batch_load  
[... wlp_batch_load]$ rlta db2_evmon_wlp.srcdb_actevm_batch
```

# Workload Processor

Understanding the optimization=auto parameter  
for db2\_evmon\_wlp command

When optimization=auto the command will attempt to chose an ETL mode heuristically optimized based on how much data will be loaded.

- For a single event file ('nnnnnnnn.evt') with less than 100MB of raw data a single thread with SQL block insert will be used
- For a single (or more than one) event file with less than 500MB of raw data (in total) a multi-threaded model using SQL block inserts will be used
- For multiple event files where one or all raw size is larger than 500MB a multi-threaded using DB2 load is attempted. The tool will execute a small test DB2 load to confirm that loads are possible from the path specified in the dbFolder parameter. If the test fails the command will fall back to SQL block insert strategy for loading

# Workload Processor

Verify the status of your batch load by using the following command and files created by the command in its working directory (CWD)

- To check if the process is still running use:

```
[... rlt]$ rlt ps
```

And check if the command profile name is among the listed processes.

- Significant files in the command's CWD

```
[... rlt]$ cd wlp_batch_load  
[... wlp_batch_load]$ ls -ltr
```

# Workload Processor

Description of files produced by the command:

```
-rw-rw-r-- 1 wlptu wlptu      6 Dec 30 17:22 rlt.pid
-rw-rw-r-- 1 wlptu wlptu      5 Dec 30 17:22 rlt_24630.pid
-rw-rw-r-- 1 wlptu wlptu     36 Dec 30 17:22 process.uuid
-rw-rw-r-- 1 wlptu wlptu    766 Dec 30 17:22 WLEVMSRC_ACT_BKT_STATS.hct.csv
-rw-rw-r-- 1 wlptu wlptu   1282 Dec 30 17:22 DB2WLP_WLP_LOAD_STATS.hct.s000.csv
-rw-rw-r-- 1 wlptu wlptu   1291 Dec 30 17:22 StorageHandlerPerfInfo.log
-rw-rw-r-- 1 wlptu wlptu      59 Dec 30 17:22 rltEvmonProcessorLog.txt
-rw-rw-r-- 1 wlptu wlptu   1585 Dec 30 17:22 evmon_stats.txt
-rw----- 1 wlptu wlptu     228 Dec 30 17:22 nohup.out
-rw-rw-r-- 1 wlptu wlptu 3761271 Dec 30 17:22 rlt.0.0.log
```

# Workload Processor

Description of files produced by the command:

- rlt\*.pid – files contains the process pid and are created when the process is started
- process.uuid – unique identifier of this process, it is relevant only when multiple workload processors are used concurrently to load workload data, can be ignored in single process loads.
- nohup.out – command's stdout captured by nohup shell command inspect its content to have a quick look at the current execution state of the process
- - rlt.0.0.log (and rlt.<n>.<n>.log) – the commands log output (can be controlled via logging configuration)
- - rlt\_err\_<timestamp>.log – dedicated log file used to log exceptions only (exceptions are also logged as part of the main logger rlt.n.n.log files)

# Workload Processor

Description of files produced by the command:

- - StorageHandlerPerfInfo.log – a specialized fixed length log file logging information about the internal performance of the parse and load process for each internal thread in its queues. Used mainly for debugging tool performance issues, can be ignored in normal operations
- - rltEvmonProcessorLog.txt – command internal use; a fixed length log file tracking the current event monitor segment file processed and the binary (byte) position of the last entity processed. The file is essential for restarting processing if you stopped (killed) the workload processor command

# Workload Processor

Description of files produced by the command:

- evmon\_stats.txt – dynamic general statistics gathered for general parsing status process

```
[wlptu@wlptt-rlth-1 wlpcsv]$ cat evmon_stats.txt
-----
Name          Value
-----
Current seg name      /db2evmondata/srcdb/sample/actwlevmon/00000000.evt
Current seg length    678568
Current seg readPos   678568
Current seg read%     100.0
Stats.EventCount      462
Stats.EventSize.min   32
Stats.EventSize.avg   1460
Stats.EventSize.max   36609
Stats.EventPerSec.avg 0
Stats.TotalDataSizeGB 0.0
SQLM_ELM_EVENT_LOG_HEADER 1
SQLM_ELM_EVENT_ACTIVITY 23
SQLM_ELM_EVENT_ACTIVITYSTMT 23
SQLM_ELM_EVENT_DBHEADER 3
SQLM_ELM_EVENT_ACTIVITYVALS 409
SQLM_ELM_EVENT_START   3
```

# Workload Processor

Description of files produced by the command:

- WLEVMSRC\_ACT\_BKT\_STATS.hct.csv – generic statement execution times as a bucket (histogram) result. The last line in the file represents the most recent bucket (histogram) data set. The first row describes each bucket boundaries
- DB2WLP\_WLP\_LOAD\_STATS.hct.s000.csv – when DB2 load is used to load data each load operation will output one record of stats in this file. Check its content to verify how many rows were loaded.

# Workload Processor

Inspect your workload database

# Workload Processor

List all workload database tables and views with minimal required workload column structure

```
[... rlt]$ rlt db_wlp_stats wlpdb  
WLEVMSRC.WORKLOAD  
[... rlt]$ rlt db_wlp_stats wlpdb rootName=WLEVMSRC.WORKLOAD save_to_profile=a
```

# Workload Processor

List general workload statistics for an workload with a given `rootName`

```
[... rlt]$ rlt db_wlp_stats.a reportType=info
```

```
...
```

FROMTIME TIMESTAMP(26,6)	TOTIME TIMESTAMP(26,6)	EVENTCOUNT INTEGER(10)
-----------------------------	---------------------------	---------------------------

2018-11-24 08:54:17.671	2018-12-30 17:18:31.429	4747330
-------------------------	-------------------------	---------

```
...
```

ACTIVITY_TYPE VARCHAR(64)	ACTIVITYCOUNT INTEGER(10)
------------------------------	------------------------------

CALL	11
DDL	92
OTHER	1
READ_DML	384
WRITE_DML	4746840

# Workload Processor

List general workload statistics for an workload with a given `rootName` partitioned by day

```
[... rlt]$ rlt db_wlp_stats.a reportType=info truncTimeInterval=day
```

```
...
-----
FROMTIME          TOTIME          EVENTCOUNT
TIMESTAMP(26,6)  TIMESTAMP(26,6)  INTEGER(10)
-----
2018-11-24 00:00:00.0 2018-11-24 00:00:00.0 4746807
2018-11-26 00:00:00.0 2018-11-26 00:00:00.0 1
2018-12-10 00:00:00.0 2018-12-10 00:00:00.0 24
2018-12-27 00:00:00.0 2018-12-27 00:00:00.0 8
2018-12-28 00:00:00.0 2018-12-28 00:00:00.0 465
2018-12-30 00:00:00.0 2018-12-30 00:00:00.0 23
...
...
```

# Workload Processor

TINTERVAL TIMESTAMP(26,6)	ACTIVITY_TYPE VARCHAR(64)	ACTIVITYCOUNT INTEGER(10)
2018-11-24 00:00:00.0	CALL	2
2018-11-24 00:00:00.0	DDL	8
2018-11-24 00:00:00.0	READ_DML	68
2018-11-24 00:00:00.0	WRITE_DML	4746729
2018-11-26 00:00:00.0	READ_DML	1
2018-12-10 00:00:00.0	READ_DML	24
2018-12-27 00:00:00.0	READ_DML	8
2018-12-28 00:00:00.0	CALL	8
2018-12-28 00:00:00.0	DDL	83
2018-12-28 00:00:00.0	READ_DML	273
2018-12-28 00:00:00.0	WRITE_DML	101
2018-12-30 00:00:00.0	CALL	1
2018-12-30 00:00:00.0	DDL	1
2018-12-30 00:00:00.0	OTHER	1
2018-12-30 00:00:00.0	READ_DML	10
2018-12-30 00:00:00.0	WRITE_DML	10

# Workload Processor

Compute workload statement statistics for an workload with a given `rootName`  
Aggregate by `session_auth_id,activity_type,sqlcode,sqlstate`

```
[... rlt]$ rlt db_wlp_stats.a reportType=wlp_stmts
```

SESSION_AUTH_ID	ACTIVITY_TYPE	SQLCODE	SQLSTATE	CNT	MIN_TIME	AVG_TIME	MAX_TIME
VARCHAR(128)	VARCHAR(64)	INTEGER(10)	CHAR(5)	INTEGER(10)	BIGINT(19)	BIGINT(19)	BIGINT(19)
DB2INST1	CALL	--null--	--null--	11	15	349	2420
DB2INST1	DDL	--null--	--null--	92	0	2741	124352
DB2INST1	OTHER	--null--	--null--	1	15	15	15
DB2INST1	READ_DML	--null--	--null--	384	0	12	1051
DB2INST1	WRITE_DML	--null--	--null--	4746840	0	0	456

# Workload Processor

Compute a bucket(histogram) on time intervals, workload statistics for an workload with a given rootName

```
[... rlt]$ rlt db_wlp_stats wlwpdb wlplibc rootNode=WLEVMSRC.WORKLOAD
```

RecNo	RowNum	Name	Value
integer	integer	varchar(256)	double
<hr/>			
0	0	VALUE_COUNT	4747330
0	1	under_10.00	4746810
0	2	10.00_110.00	416
0	3	110.00_210.00	56
0	4	210.00_310.00	21
0	5	310.00_410.00	18
0	6	410.00_510.00	2
0	7	510.00_610.00	0
0	8	610.00_710.00	0
0	9	710.00_810.00	0
0	10	810.00_910.00	2
0	11	over_1000.00	3
<hr/>			

# Workload Processor

Start a new load test definition

# Workload Processor

For ease of use create a command profile for db\_wlt command if none exists

```
[... rlt]$ rlt db_wlt alias=wlpdb tgtAlias=tgtDbTest wlRootFQN=WLEVMSRC.WORKLOAD  
wlTgtRootFQN=WLEVMTGT.WORKLOAD rootFQN=WLPEXEC save_to_profile=a
```

Create a new workload test definition

```
[... rlt]$ rlt db_wlt.a new test1 wlStartTime=2018-11-24 wlStopTime=2018-11-25 hashResult=yes
```

```
Creating table:WLPEXEC.WL_SESSION_TEST1  
Creating table:WLPEXEC.WL_SESSION_MAP_TEST1  
Created or updated record in:WLPEXEC.WL_EXEC_CFG cfg_name=test1 start_ts=2019-04-03 15:49:38.775  
Created or updated 1 records in:WLPEXEC.WL_EXEC_CFG_WORKER  
Created or updated 1 records in:WLPEXEC.WL_EXEC_USER
```

# Workload Processor

Check the test definition record

```
[... rlt]$ rlt db_wlt.a ls
```

RowNum	ColNum	Name	Value
Int	Int	Varchar(128)	Object
1	1	CFG_NAME	test1
1	2	START_TS	2019-07-23 14:35:23.409
1	3	START_TS_GMT	n
1	4	STATE	INIT
1	5	RQ_STATE	--null--
1	6	HASH_RESULT	n
1	7	INITIAL_STATE_UID	--null--
1	8	FINAL_STATE_UID	--null--
1	9	START_UID	--null--
1	10	START_APPL_ID	172.16.178.90.55780.190703232258
1	11	START_UOW_ID	70983
1	12	START_ACTIVITY_ID	1
1	13	START_TST	2019-07-03 16:26:12.398
1	14	STOP_UID	--null--
1	15	STOP_APPL_ID	172.16.178.90.55780.190703232258
1	16	STOP_UOW_ID	100007
1	17	STOP_ACTIVITY_ID	1
1	18	STOP_TST	2019-07-03 16:27:30.394
1	19	WL_SESSION_INFO	WLPEXEC.WL_SESSION_TEST1
1	20	WL_SESSION_USER_INFO	WLPEXEC.WL_EXEC_USER_TEST1
1	21	WL_SESSION_MAP_INFO	WLPEXEC.WL_SESSION_MAP_TEST1
1	22	WLP_STATS_BASELINE_KEY	--null--
1	23	WLP_STATS_BUCKET_CFG	--null--
1	24	WLP_STATS_REPLY_KEY	--null--
1	25	WLP_SRC_FRQN	WLEVMSRC.WORKLOAD_
1	26	WLP_FRQN	WLEVMTGT.WORKLOAD_
1	27	JDBC_URL	--null--
1	28	SLA_SLOW_PRC	20
1	29	SLA_FAST_PRC	20

Start the replay of the workload against the target database

# Workload Processor

Check if there are any active workload executors (db\_wlt\_replay services)

```
[... rlt] rlt db_wlt.a ps
```

WK_UUID VARCHAR(37)	HOST VARCHAR(128)	STATE VARCHAR(16)	LAST_PING_TS TIMESTAMP(26,6)	LAST_PING_SEC INTEGER(10)
773c9440-b334-4e47-8fa8-53873b4c9696	wlptt-rlth-1.fyre.ibm.com	SETUP	2019-04-06 13:59:29.990868	0

Start the load test execution using the tgtdbTest as database alias

```
[... rlt]$ rlt db_wlt.a start test1 tgtAlias=tgtdbTest
```

Monitor the execution state of the test (compact form)

```
[... rlt]$ rlt db_wlt.a exec_info test1
```

Monitor the execution state of the test (full form)

```
[... rlt]$ rlt db_wlt.a info test1
```

# Workload Processor

[... rlt]\$ rlt db_wlt.a exec_info test1	
<hr/>	
Name	Value
Test id	test1
Test state	active
DB Server time	2019-04-06 14:17:22.83
Requested start time	2019-04-06 08:29:56.29
Start time delta	-5 hrs, 47 min, 26 sec, 540 ms
Estimated test exec time	2 hrs, 34 min, 46 sec, 306 ms
Required replay process count	1
Required replay thread count	9
Attached replay process count	1
Wk[773c9440-b334-4e47-8fa8-53873b4c9696]	Total:4690535 Exec success:2611575 Exec with err:1395888
Exec start time	2019-04-06 13:25:12.4
Total activities executed	4007463
Total activities errors	1395888
Estimated test remaining time	1 hr, 42 min, 34 sec, 58 ms
<hr/>	

# Workload Processor

```
[... rtl]$ rlt db_wlt.a info test1
```

RowNum	ColNum	Name	Value
Int	Int	Varchar(128)	Object
1	1	CFG_NAME	test2
1	2	PROCESS_ID	0
1	3	AUTH_ID	DB2INST1
1	4	WK_UUID	773c9440-b334-4e47-8fa8-53873b4c9696
1	5	STATE	EXEC
1	6	STMT_TOTAL_COUNT	4690535
1	7	STMT_EXEC_COUNT	2718281
1	8	STMT_FAIL_COUNT	1395888
1	9	TEST_START_TS	2019-04-06 13:25:12.4
1	10	TEST_END_TS	2019-08-17 11:04:28.252

# Workload Processor

```
[... rtl]$ rlt db_wlt.a info test1
```

RowNum	ColNum	Name	Value
Int	Int	Varchar(128)	Object
1	1	CFG_NAME	test2
1	2	START_TS	2019-04-06 08:29:56.29
1	3	START_TS_GMT	n
1	4	STATE	active
1	5	RQ_STATE	__null__
1	6	INITIAL_STATE_UID	__null__
1	7	FINAL_STATE_UID	__null__
1	8	START_UID	__null__
1	9	START_APPL_ID	172.16.178.90.59926.181124175039
1	10	START_UOW_ID	2
1	11	START_ACTIVITY_ID	1
1	12	START_TST	2018-11-24 09:50:40.438
1	13	STOP_UID	__null__
1	14	STOP_APPL_ID	172.16.178.90.59986.181124192152
1	15	STOP_UOW_ID	1000003
1	16	STOP_ACTIVITY_ID	1
1	17	STOP_TST	2018-11-24 12:25:26.744
1	18	WL_SESSION_INFO	WLPEXEC.WL_SESSION_TEST2
1	19	WL_SESSION_USER_INFO	WLPEXEC.WL_EXEC_USER_TEST2
1	20	WL_SESSION_MAP_INFO	WLPEXEC.WL_SESSION_MAP_TEST2
1	21	WLP_STATS_BASELINE_KEY	__null__
1	22	WLP_STATS_BUCKET_CFG	__null__
1	23	WLP_STATS_REPLY_KEY	__null__
1	24	WLP_SRC_FRQN	WLEVMSRC.WORKLOAD_
1	25	WLP_FRQN	WLEVMTGT.WORKLOAD_
1	26	JDBC_URL	jdbc:db2://wlptt-rlth-4.fyre.ibm.com:50000/SAMPLE:retrieveMessagesFromServerOnGetMessage=true;clientProgramName=RLTCmd;

# Workload Processor

```
[... rtl]$ rlt db_wlt.a info test1 # check if the workload test is done
```

RowNum	ColNum	Name	Value
Int	Int	Varchar(128)	Object
1	1	CFG_NAME	test2
1	2	START_TS	2019-04-06 08:29:56.29
1	3	START_TS_GMT	n
1	4	STATE	done
1	5	RQ_STATE	--null--
1	6	INITIAL_STATE_UID	--null--
1	7	FINAL_STATE_UID	--null--
1	8	START_UID	--null--
1	9	START_APPL_ID	172.16.178.90.59926.181124175039
1	10	START_UOW_ID	2
1	11	START_ACTIVITY_ID	1
1	12	START_TST	2018-11-24 09:50:40.438
1	13	STOP_UID	--null--
1	14	STOP_APPL_ID	172.16.178.90.59986.181124192152
1	15	STOP_UOW_ID	1000003
1	16	STOP_ACTIVITY_ID	1
1	17	STOP_TST	2018-11-24 12:25:26.744
1	18	WL_SESSION_INFO	WLPEXEC.WL_SESSION_TEST2
1	19	WL_SESSION_USER_INFO	WLPEXEC.WL_EXEC_USER_TEST2
1	20	WL_SESSION_MAP_INFO	WLPEXEC.WL_SESSION_MAP_TEST2
1	21	WLP_STATS_BASELINE_KEY	--null--
1	22	WLP_STATS_BUCKET_CFG	--null--
1	23	WLP_STATS_REPLY_KEY	--null--
1	24	WLP_SRC_FRQN	WLEVMSRC.WORKLOAD_
1	25	WLP_FRQN	WLEVMTGT.WORKLOAD_
1	26	JDBC_URL	jdbc:db2://wlptt-rlth-4.fyre.ibm.com:50000/SAMPLE:retrieveMessagesFromServerOnGetMessage=true;clientProgramName=RLTCmd;

# Workload Processor

After the workload test is done check the collected statistics for the test case

TBD

# Workload Processor

Load the event monitor data from the target database in the workload database  
(once the load test is finished)

# Workload Processor

To load the captured workload in the workload database define a command profile if not already defined then execute the batch load (as many times as needed) by using the command profile

```
[... rlt]$ rlt db2_evmon_wlp src=/db2evmondata/tgt/sample/actwlevmon/  
dbFolder=/db2evmondata/tgt/sample/actwlevmon/ schema=WLEVMTGT dbAlias=wlpdb  
save_to_profile=tgtdb_actevm_batch
```

Executing the batch load by using the command profile defined above. By using rlt the command will be executed in background (equivalent to nohup rlt ... &) in order to handle longer time unassisted loads.

```
[... rlt]$ mkdir wlp_tgt_batch_load  
[... rlt]$ cd wlp_tgt_batch_load  
[... wlp_tgt_batch_load]$ rlt a db2_evmon_wlp.tgtdb_actevm_batch
```

Use same method to monitor the load process and detect end of processing as in the case of command profile `db2_evmon_wlp.srcdb_actevm_batch`

# Workload Processor

Check the histogram (bucket) results collected by the actBktStats plugin in the db2\_evmon\_wlp command

# Workload Processor

Check the WLEVMTGT\_ACT\_BKT\_STATS.hct.csv file in the PWD of the workload ETL command (db2\_evmon\_wlp.tgtdb\_actevm\_batch).

```
[... wlp]$ cat  
TS-MS,<10MS,10MS-20MS,20MS-30MS,30MS-40MS,40MS-50MS,50MS-60MS,60MS-70MS,...  
Timestamp,int,int,int,int,int,int,int,int,int,int,int,int,int,int,int,...  
1550002126288,407,10,2,2,2,0,1,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,...
```

First column contain the timestamp when the row was logged (use the last row as final stats). Second column is the count of statements executed in less than 10ms, next column (and so on) are bucket

# Workload Processor

Run analytic reports against the workload data captured from the target database

# Workload Processor

TBD

# Workload Processor

Q & A