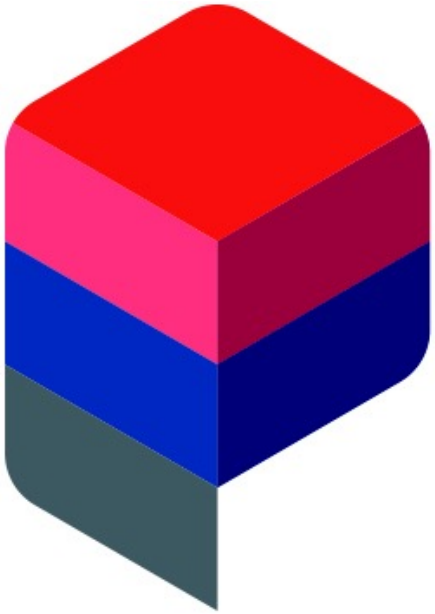


performing databases

Noch mehr laute Nachbarn und störende Kinder

Oracle Multitenant Performance II

Martin Klier

Performing Databases GmbH
Mitterteich / Germany

Speaker

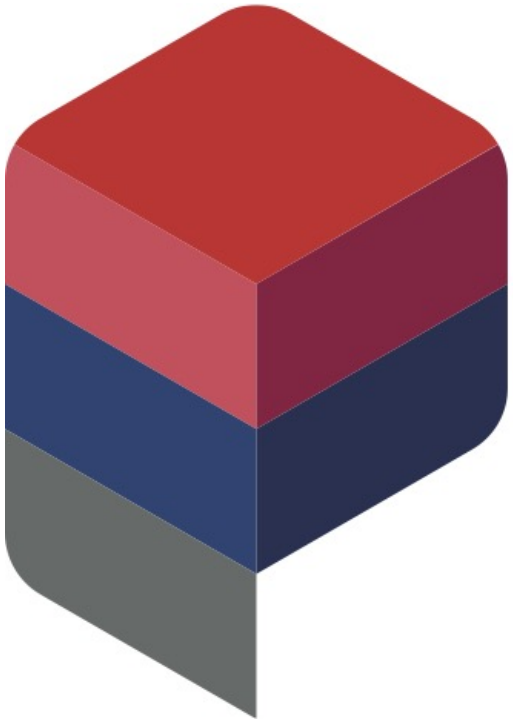
- Martin Klier
- Solution Architect and Database Expert
- My focus:
  - Performance + Tuning
  - Highly available systems
  - Cluster and Replication
- Linux since 1997
- Oracle Database since 2003

ORACLE ACE Director

Performing Databases

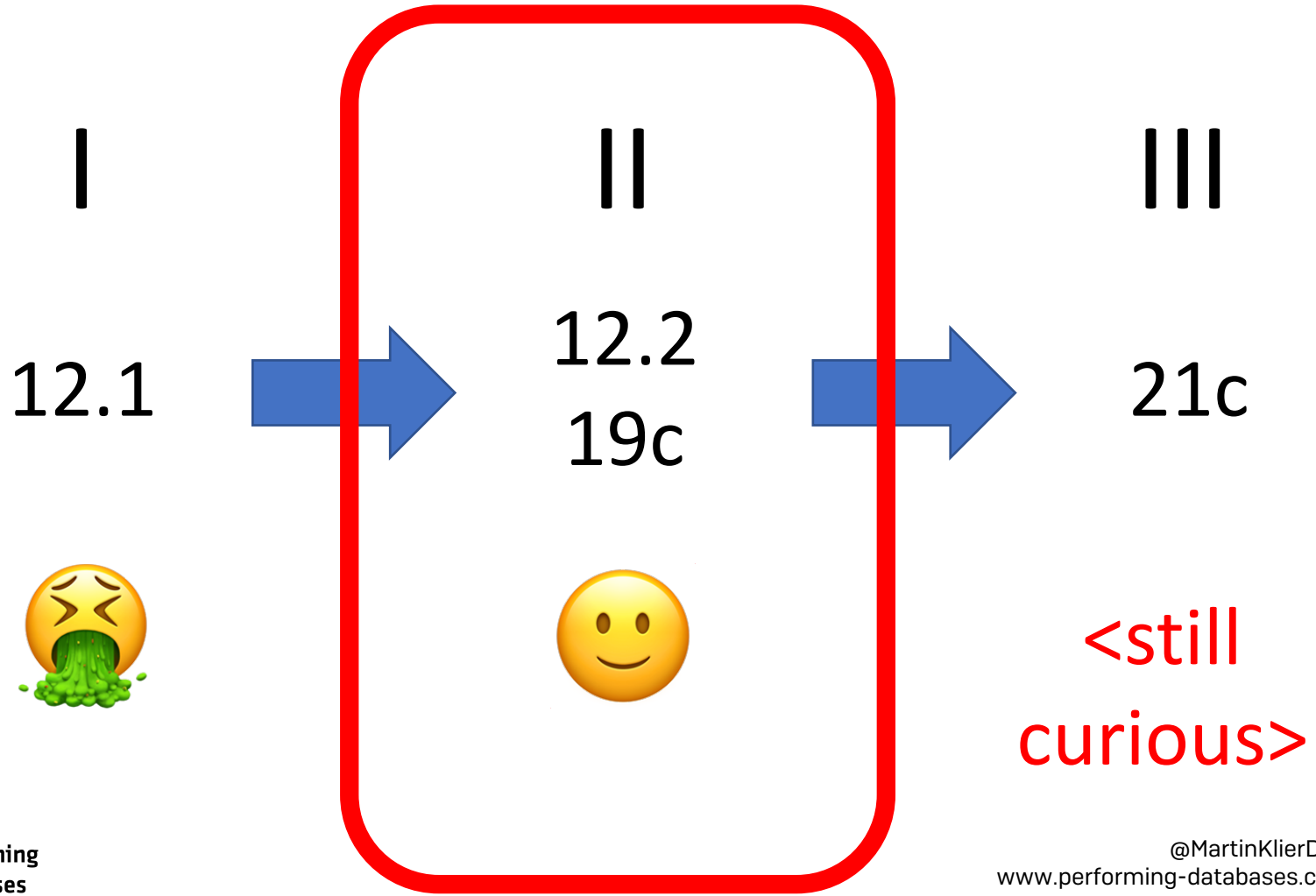
- Three Experts for Database technology
  - Concepts and Project Competence
  - Architecture- and System planning
  - Licensing
  - Implementation and Troubleshooting

- Get in touch
  - Performing Databases GmbH
    Wiesauer Strasse 27
    95666 Mitterteich // Germany
  - http://www.performing-databases.com
  - Twitter: @PerformingDB

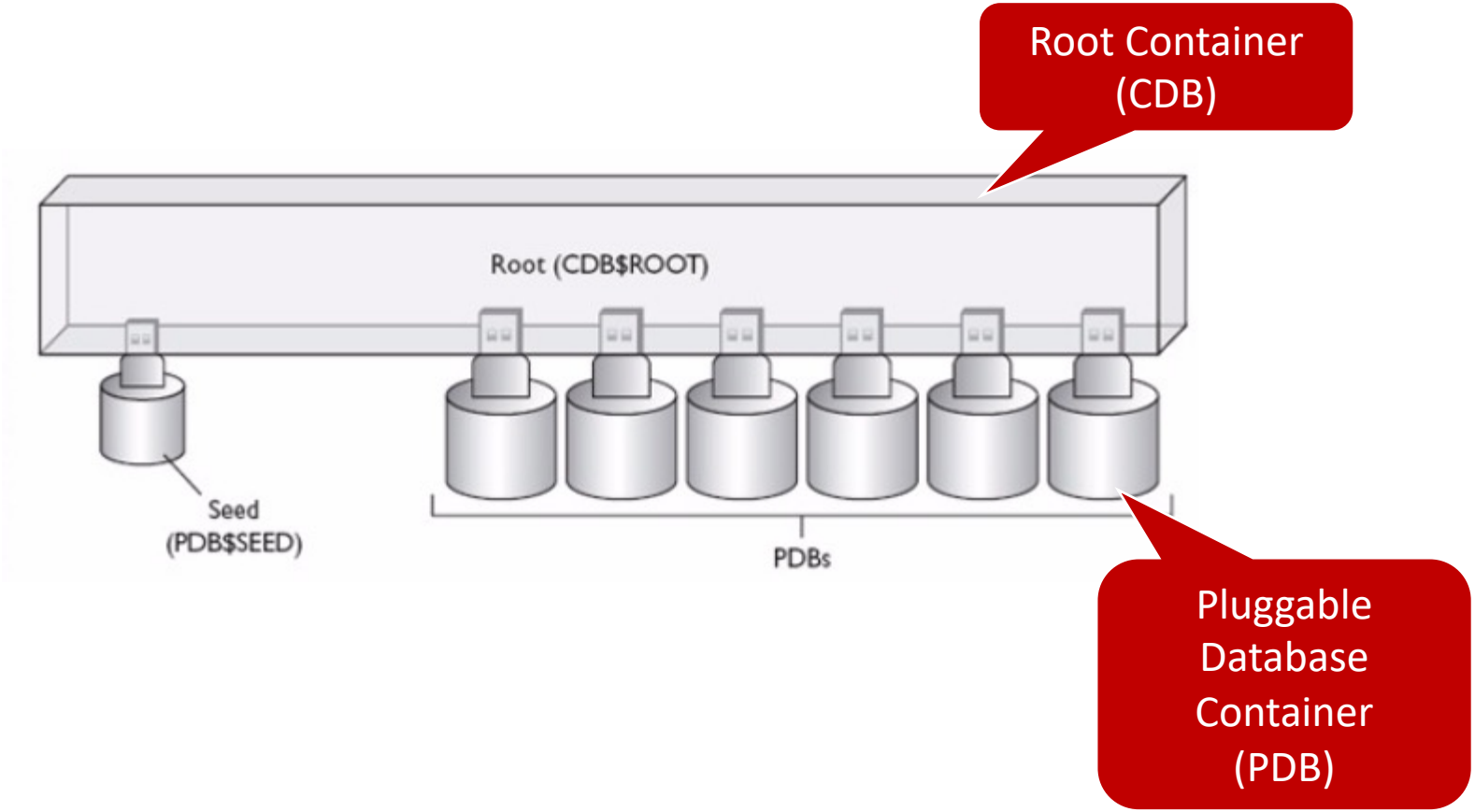


Oracle Multitenant „Pluggable Database“

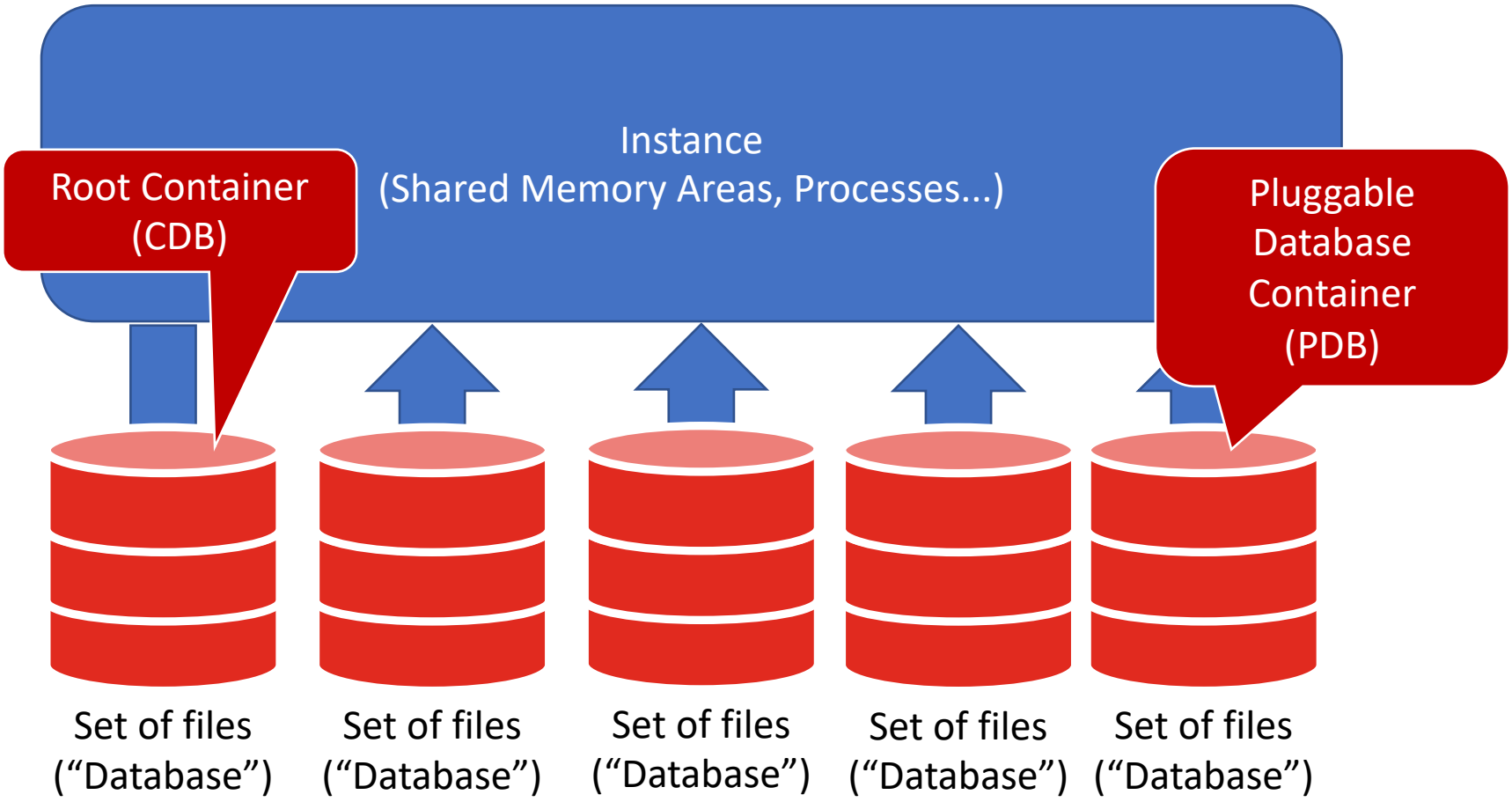
Generations



Terminology



Architecture (simplified)

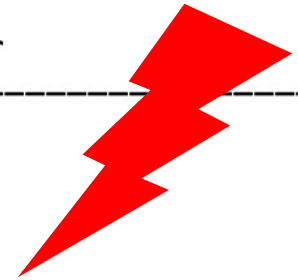


Check where you are!

```
select sys_context('Userenv','Con_name') "current container" from dual;
```

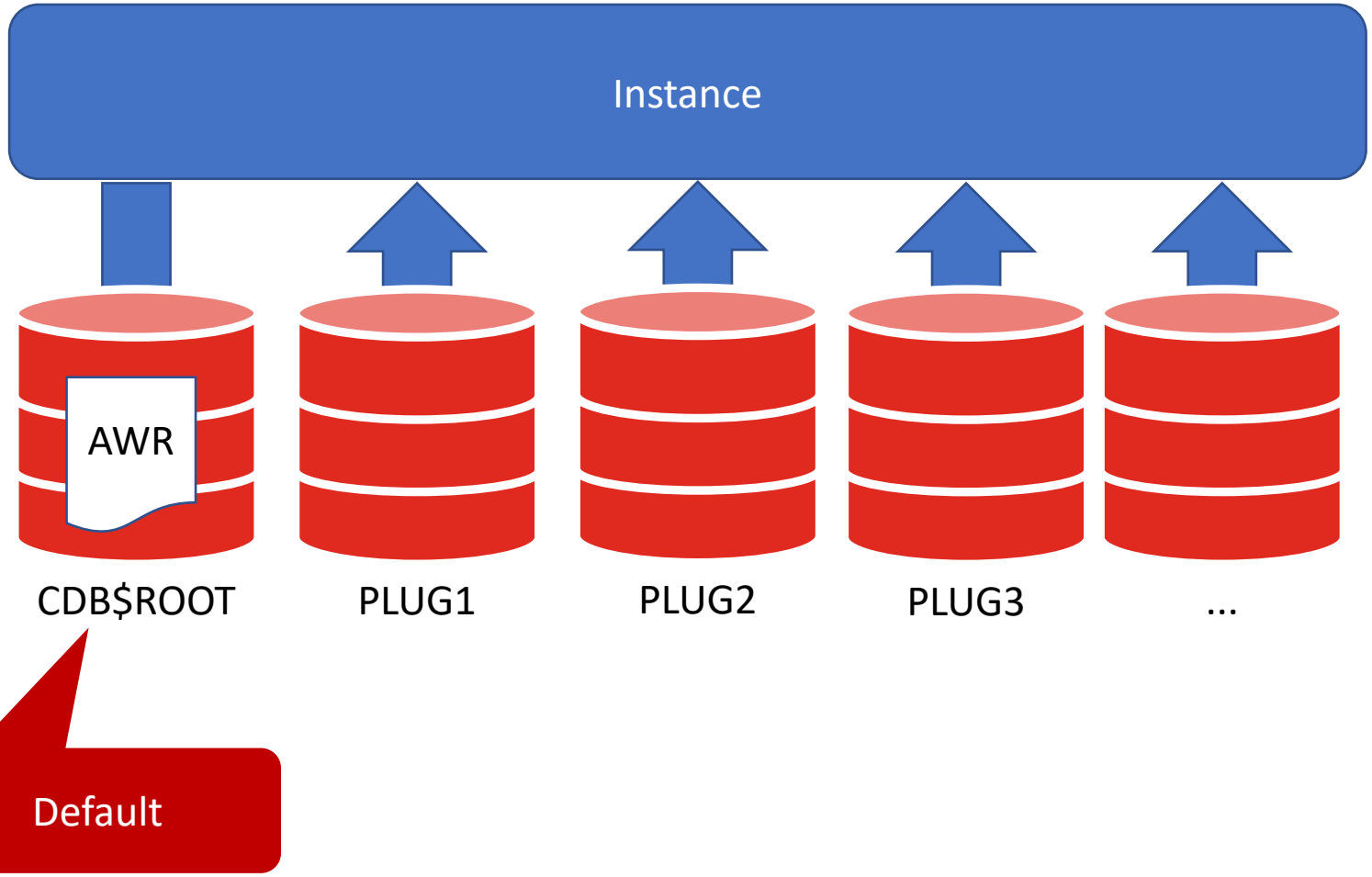
ORACLE_PDB_SID

```
[oracle@ora-bench1 ~]$ (PDB19MT) export ORACLE_PDB_SID=PLUGXXX
[oracle@ora-bench1 ~]$ (PDB19MT) sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Mon May 17 14:16:01 2021
Version 19.9.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.9.0.0.0

SQL> select sys_context('Userenv','Con_name') "current container" from dual;

current container
------------------------------------------------------------------------------
CDB$ROOT
```



Automatic Workload Repository
AWR

AWR



SQL ordered by Gets DB/Inst: PDB19MT/PDB19MT Snaps: 1115-1116
-> Resources reported for PL/SQL code includes the resources used by all SQL
statements called by the code.
-> %Total - Buffer Gets as a percentage of Total Buffer Gets
-> %CPU - CPU Time as a percentage of Elapsed Time
-> %IO - User I/O Time as a percentage of Elapsed Time
-> Total Buffer Gets: 62,894,249
-> Captured SQL account for 115.7% of Total

Buffer Gets	Executions	Gets per Exec	%Total	Elapsed Time (s)	%CPU	%IO	SQL Id
64,362,163	491	131,083.8	102.3	61,868.8	.5	18.1	29qp10usqkqh0

Module: Sales Rep Query
PDB: PLUG1
SELECT TT.ORDER_TOTAL, TT.SALES_REP_ID, TT.ORDER_DATE, CUSTOMERS.CUST_FIRST_NAME
, CUSTOMERS.CUST_LAST_NAME FROM (SELECT ORDERS.ORDER_TOTAL, ORDERS.SALES_REP_ID,
ORDERS.ORDER_DATE, ORDERS.CUSTOMER_ID, RANK() OVER (ORDER BY ORDERS.ORDER_TOTAL
DESC) SAL_RANK FROM ORDERS WHERE ORDERS.SALES_REP_ID = :B1) TT, CUSTOMERS WHER

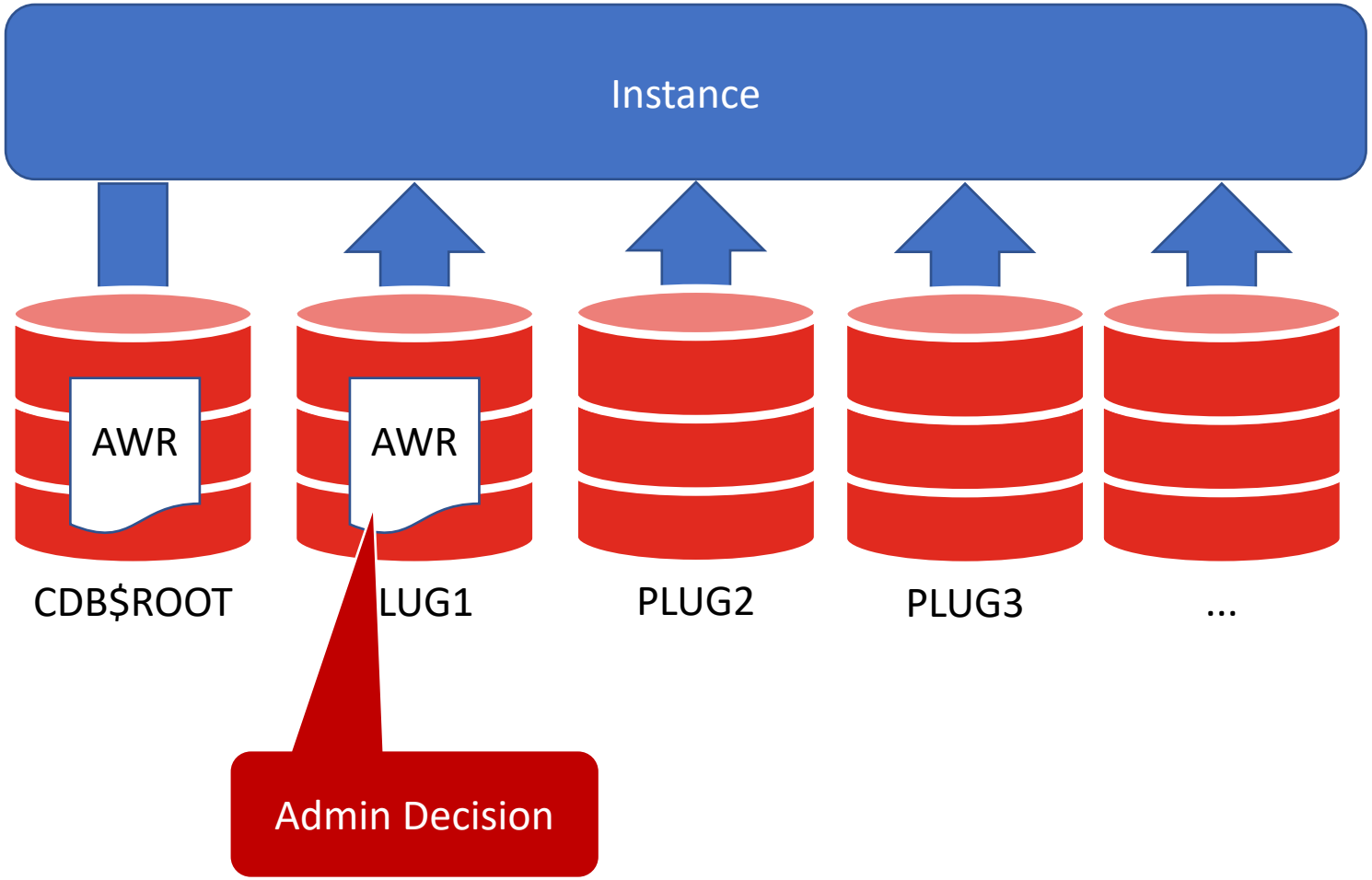
64,353,457	488	131,871.8	102.3	61,878.4	.5	18.1	cj9v3ynkm7uuy

Module: JDBC Thin Client
PDB: PLUG1
BEGIN :1 := orderentry.SalesRepsQuery(:2 ,:3 ,:4); END;

5,078,707	9,176	553.5	8.1	93,102.9	.1	16.1	0w2qpuc6u2zsp

Module: JDBC Thin Client
PDB: PLUG1
BEGIN :1 := orderentry.neworder(:2 ,:3 ,:4); END;

AWR



AWR in PDB

Switch to PDB – Check if you are there ;)

Manual Snapshots:

```
EXEC DBMS_WORKLOAD_REPOSITORY.create_snapshot;
```

Automatic Snapshots:

```
EXECUTE DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTINGS(
        interval => 10,     Default in PDB: 0 (=disabled)
        retention => 20160
);
```

AWR Report in PDB

@?/rdbms/admin/awrrpt

```
Specify the location of AWR Data
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
AWR_ROOT - Use AWR data from root (default)
AWR_PDB - Use AWR data from PDB
Enter value for awr_location: AWR_PDB

Location of AWR Data Specified: AWR_PDB
```

```
Current Instance
~~~~~~~~~~~~~~~~
DB Id          DB Name         Inst Num       Instance       Container Name
-------------- --------------- -------------- -------------- --------------
 3312395066     PDB19MT                       1 PDB19MT        PLUG1

Root DB Id      Container DB Id AWR DB Id
--------------- --------------- ---------------
   4269972545      3312395066      3312395066
```

```
Instances in this Workload Repository schema
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  DB Id      Inst Num   DB Name      Instance     Host
------------ ---------- ---------    ----------   ------
  3312395066     1      PDB19MT      PDB19MT      ora-bench1

Using 3312395066 for database Id
Using          1 for instance number
```

AWR Report in PDB

```
WORKLOAD REPOSITORY PDB report (PDB snapshots)

DB Name         DB Id    Unique Name DB Role          Edition Release     RAC CDB
------------ ---------- ----------- ----------------- ------- ----------- --- ---
PDB19MT       3312395066 PDB19MT     PRIMARY           EE      19.0.0.0.0 NO  YES

Instance     Inst Num Startup Time    User Name    System Data Visible
------------ -------- --------------- ------------ --------------------
PDB19MT             1 17-May-21 23:16 SYS          YES

Container DB Id  Container Name      Open Time
---------------- ---------- ---------------
     3312395066 PLUG1           17-May-21 23:16

Host Name        Platform                         CPUs Cores Sockets Memory(GB)
---------------- -------------------------------- ---- ----- ------- ----------
ora-bench1       Linux x86 64-bit                    8     8       1      94.31

              Snap Id      Snap Time      Sessions Curs/Sess
            --------- ------------------- -------- ---------
Begin Snap:         1 17-May-21 23:42:04        0       3.0
  End Snap:         2 17-May-21 23:47:47      475       2.4
   Elapsed:               5.70 (mins)
   DB Time:           6,760.25 (mins)
```

AWR Report in PDB

Load Profile	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	1,185.0	11.4	0.40	1.97
DB CPU(s):	3.2	0.0	0.00	0.01
Background CPU(s):	0.0	0.0	0.00	0.00
Redo size (bytes):	422,776.7	4,068.4		
Logical read (blocks):	293,080.3	2,820.3		
Block changes:	2,818.4	27.1		
Physical read (blocks):	3,785.3	36.4		
Physical write (blocks):	0.3	0.0		
Read IO requests:	708.5	6.8		
Write IO requests:	0.1	0.0		
Read IO (MB):	29.6	0.3		
Write IO (MB):	0.0	0.0		
IM scan rows:	0.0	0.0		
Session Logical Read IM:	0.0	0.0		
User calls:	600.8	5.8		
Parses (SQL):	303.9	2.9		
Hard parses (SQL):	0.6	0.0		
SQL Work Area (MB):	1.7	0.0		
Logons:	2.9	0.0		
User logons:	2.9	0.0		
Executes (SQL):	2,972.3	28.6		
Rollbacks:	0.6	0.0		
Transactions:	103.9			

No
Instance Efficiency Percentages

Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Avg Wait	% DB time	Wait Class
resmgr:cpu quantum	49,697	255K	5130.74ms	62.9	Schedule
db file sequential read	149,753	55.8K	372.33ms	13.7	User I/O
db file scattered read	72,122	33.9K	470.29ms	8.4	User I/O
log file sync	36,318	31.1K	854.97ms	7.7	Commit
buffer busy waits	5,023	14.1K	2816.99ms	3.5	Concurre

Scalability

Scaling Multitenant

Element	Terms & Conditions
Instance: SGA	Structure shared, QoS possible
Instance: Library Cache	Structure shared, content isolated
Instance: Buffer Cache	Structure shared, content isolated
Instance: Processes	Shared
Root Container: Online Redo Facility	**Shared, but “labelled” per PDB**
Root Container: Control File	Shared
PDB Container: Data Files	Independent

Configure and Rule

For better isolation and individualization, you can set parameters on PDB level:

DB_CACHE_SIZE:	The minimum buffer cache size for the PDB.
SHARED_POOL_SIZE:	The minimum shared pool size for the PDB.
PGA_AGGREGATE_LIMIT:	The maximum PGA size for the PDB.
PGA_AGGREGATE_TARGET:	The target PGA size for the PDB.
SGA_MIN_SIZE:	The minimum SGA size for the PDB.
SGA_TARGET:	The maximum SGA size for the PDB.

Thanks, Tim! <3

https://oracle-base.com/articles/12c/multitenant-memory-resource-management-for-pdbs-12cr2

Library Cache

Library Cache

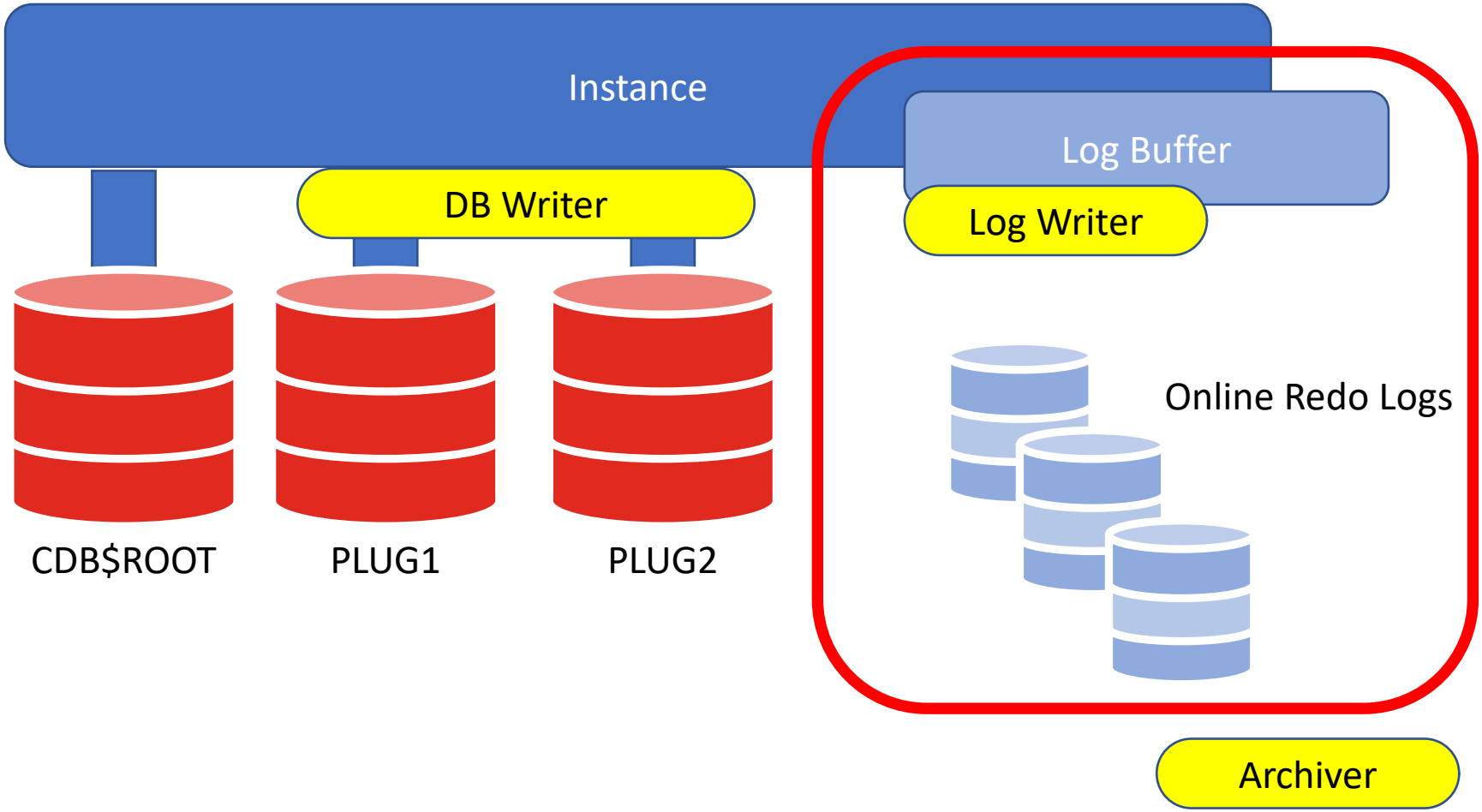
Conclusion:

- Regular use (Cursor Sharing) scales well
- Abuse (Library Cache Stress) does not

➔ Do code well, and it will scale
➔ Do code badly, and you will pay ☺

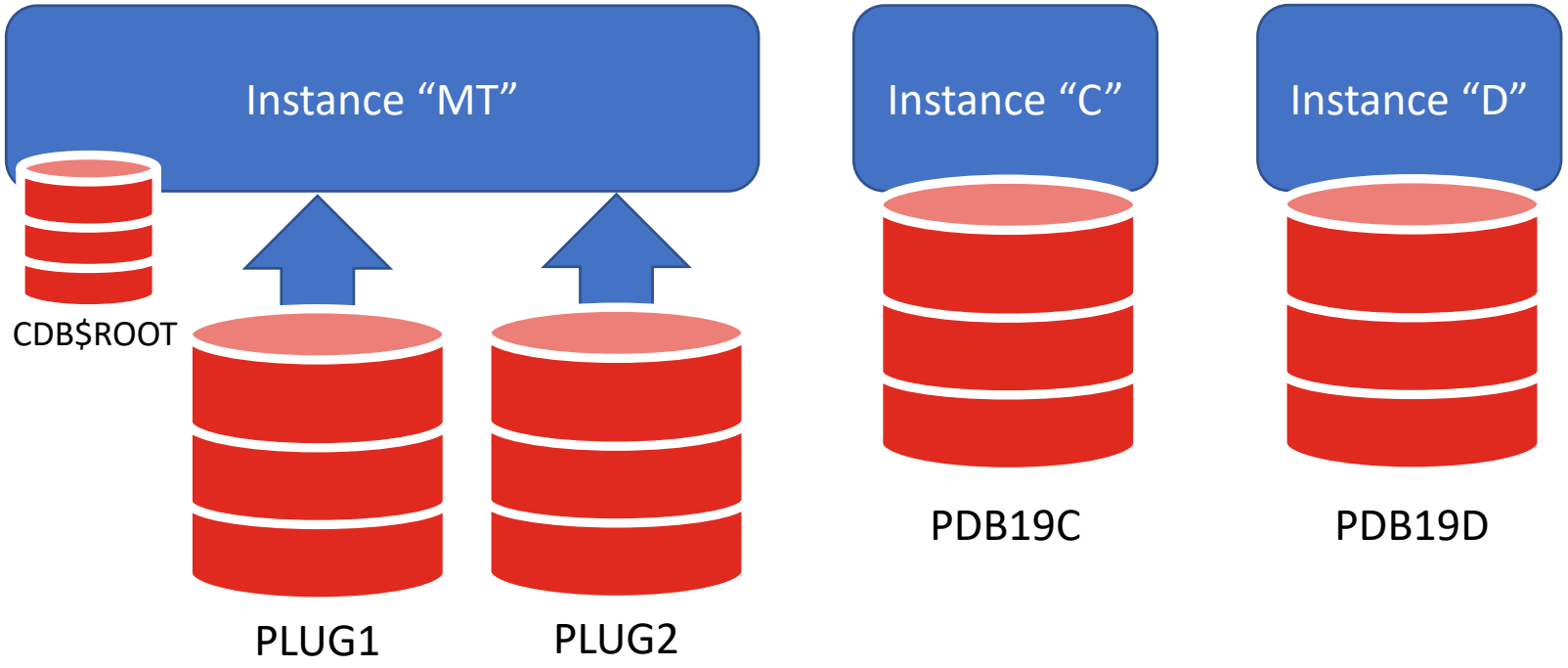
Shared Resources

Sharing Resources

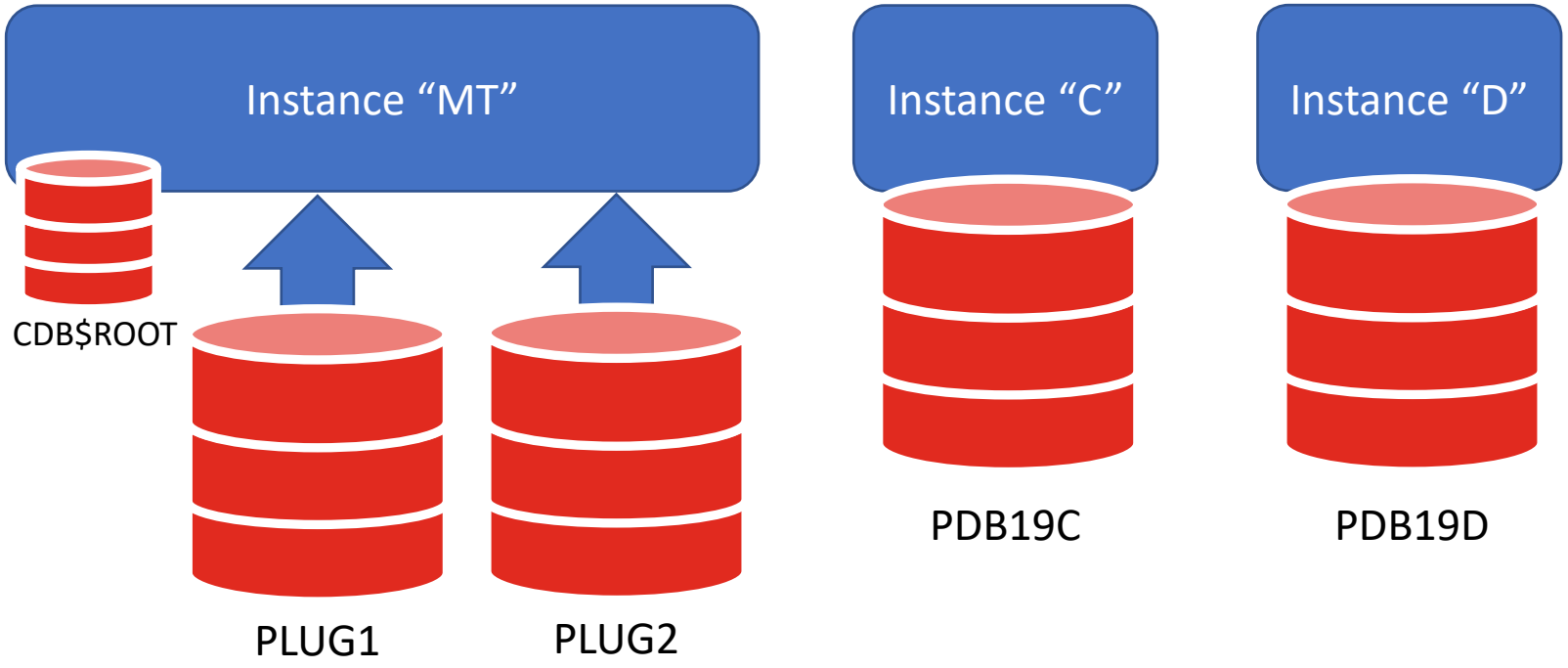


Redo Clash

Test Gear
Oracle 19c RU13 on Linux



Test Gear
Oracle 19c RU13 on Linux



Redo:
- 6 Groups
- 50MB each
- 2 Members

Redo:
- 3 Groups
- 50MB each
- 2 Members

Redo:
- 3 Groups
- 50MB each
- 2 Members

Test Gear – Mini_Loader

Load Generator “Mini_Loader”:

```
loop
    insert into table values (‘X’);
    commit;
end loop;
```

Baseline in my setup: 14.000 per second per instance
    maxing out 1 CPU

Test Gear – Maxi_Loader

Load Generator “Maxi_Loader”:

```
loop
    insert into table values (‘XXXXXXXXXXXXXX<1900x>’);
    insert into table values (‘XXXXXXXXXXXXXX<1900x>’);
    insert into table values (‘XXXXXXXXXXXXXX<1900x>’);
    insert into table values (‘XXXXXXXXXXXXXX<1900x>’);
    commit;
end loop;
```

Baseline in my setup: 6.000 per second per instance
maxing out 1 CPU

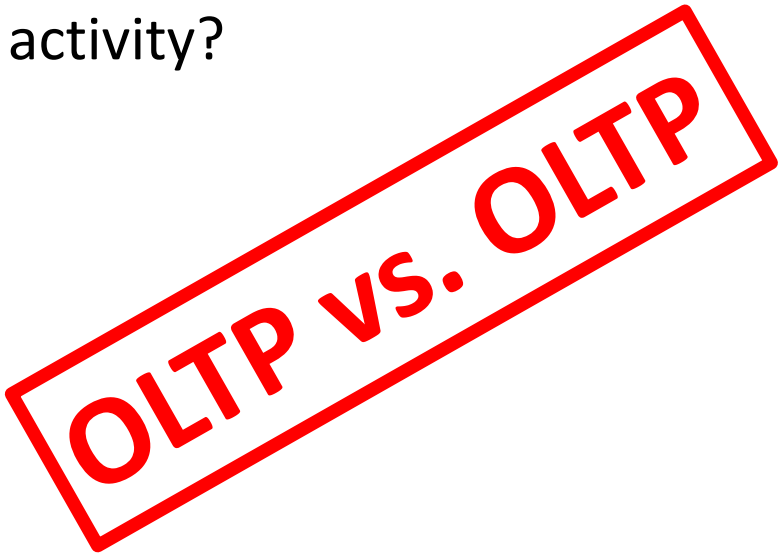
Redo Clash

Case 1

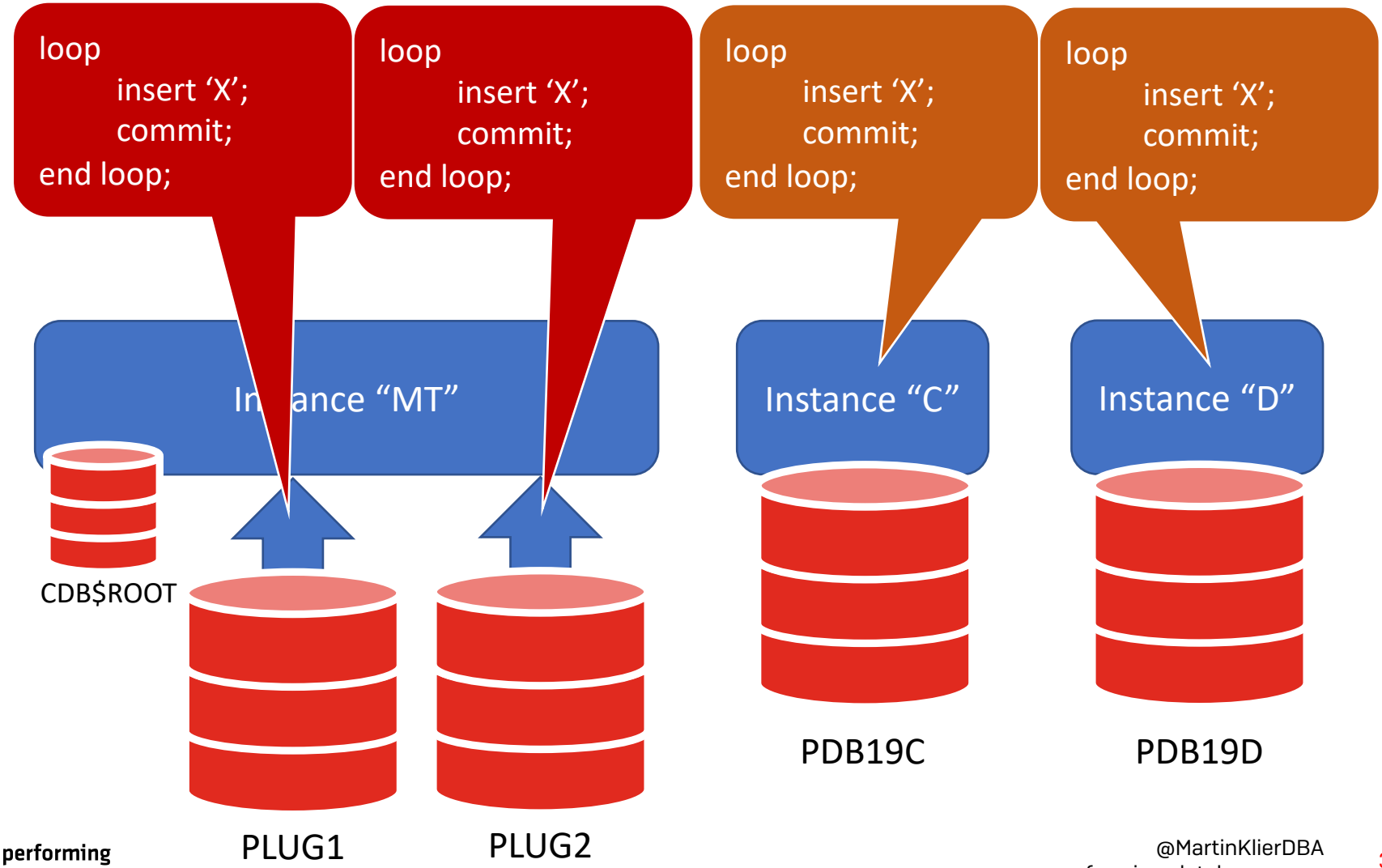
Case 1

Do classic databases influence each other's small and fast redo activity?

Do PDBs influence each other's small and fast redo activity?



Case 1


loop
insert 'X';
commit;
end loop;
loop
insert 'X';
commit;
end loop;
loop
insert 'X';
commit;
end loop;
loop
insert 'X';
commit;
end loop;
Instance "MT"
Instance "C"
Instance "D"
CDB$ROOT
PLUG1
PLUG2
PDB19C
PDB19D

Case 1 - Result

13-14k/s

PDB	PIT	WORKLOAD	ITERATIONS	RUNTIME_SE	IPS
PDB19C	17.11.2021-22:14:41	Mini_loader	4087999	30	13626.6633

PDB	PIT	WORKLOAD	ITERATIONS	RUNTIME_SE	IPS
PDB19D	17.11.2021-22:14:41	Mini_loader	4270999	30	14236.6633

PDB	PIT	WORKLOAD	ITERATIONS	RUNTIME_SE	IPS
PLUG1	17.11.2021-22:19:58	Mini_loader	4247999	30	14159.9967

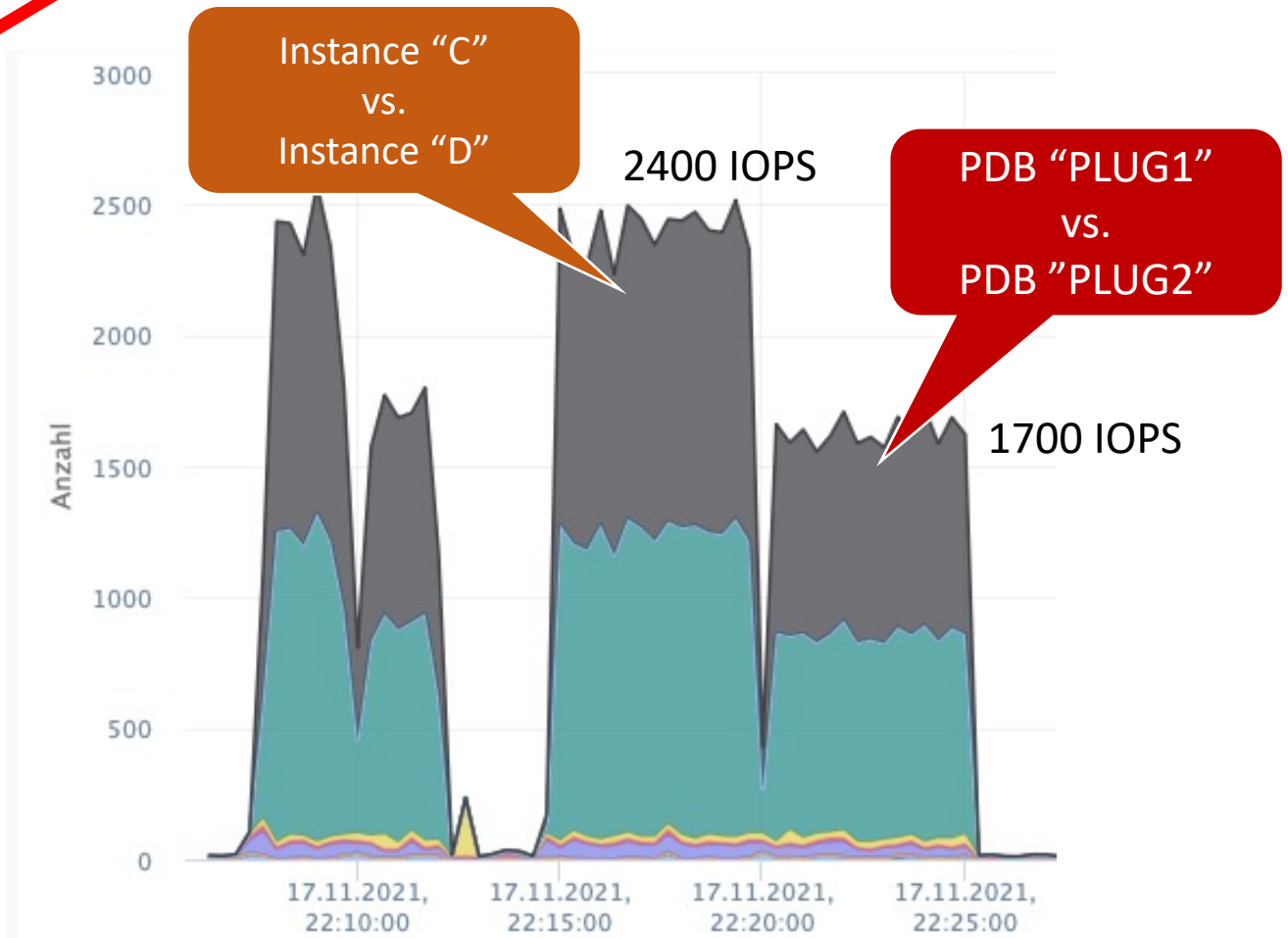
PDB	PIT	WORKLOAD	ITERATIONS	RUNTIME_SE	IPS
PLUG2	17.11.2021-22:19:58	Mini_loader	4379999	30	14551.4917

13-14k/s

Two Instances or two PDBs do not influence each other's small and fast Redo.

OLTP vs. OLTP

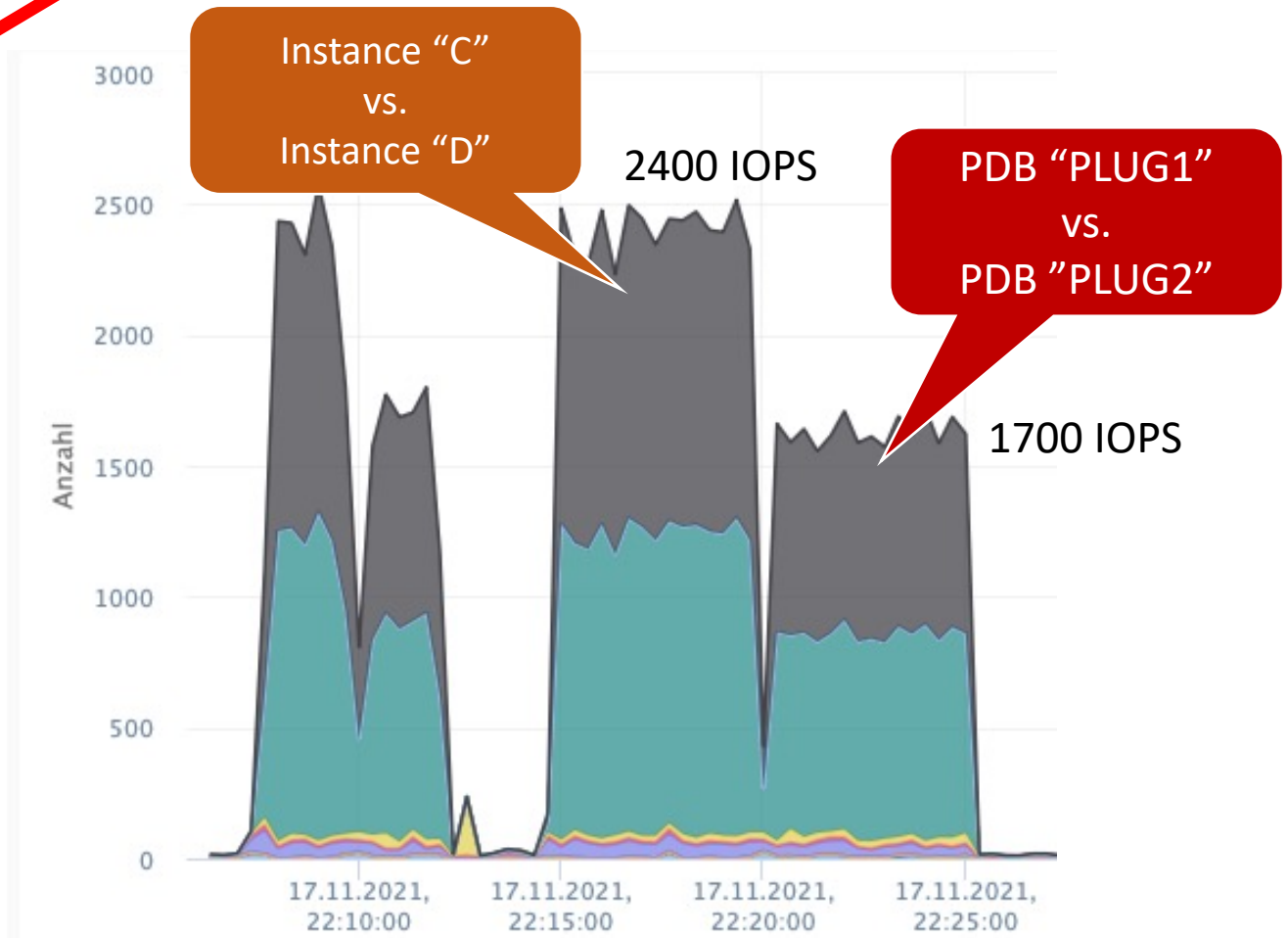
Case 1 - Result



... but two PDBs need 1/3 less IOPS for the same work!

OLTP vs. OLTP

Case 1 - Result



Two workloads in one “Classic” Instance show the same picture.

You see the overhead of “another instance”.

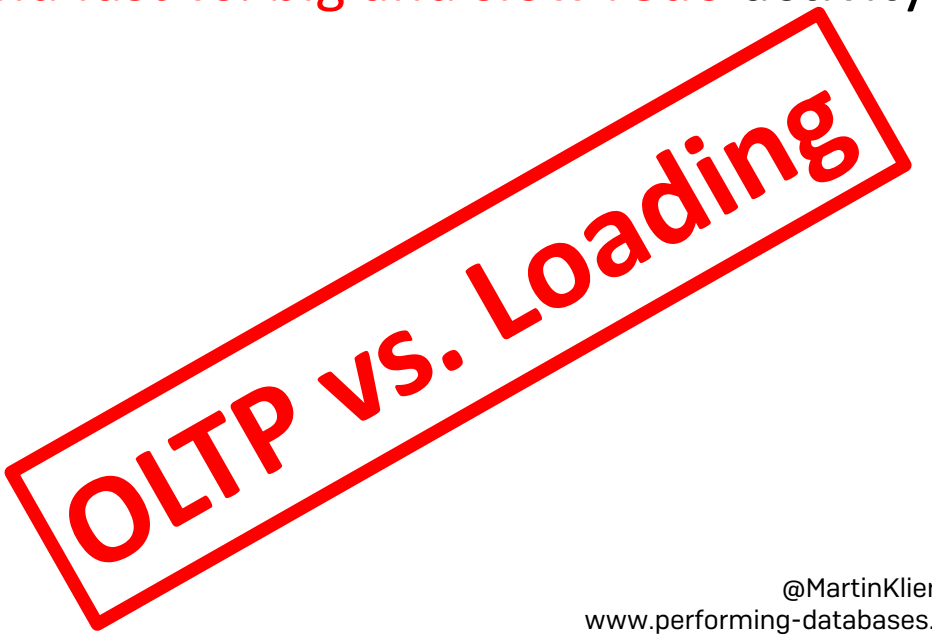
Redo Clash

Case 2

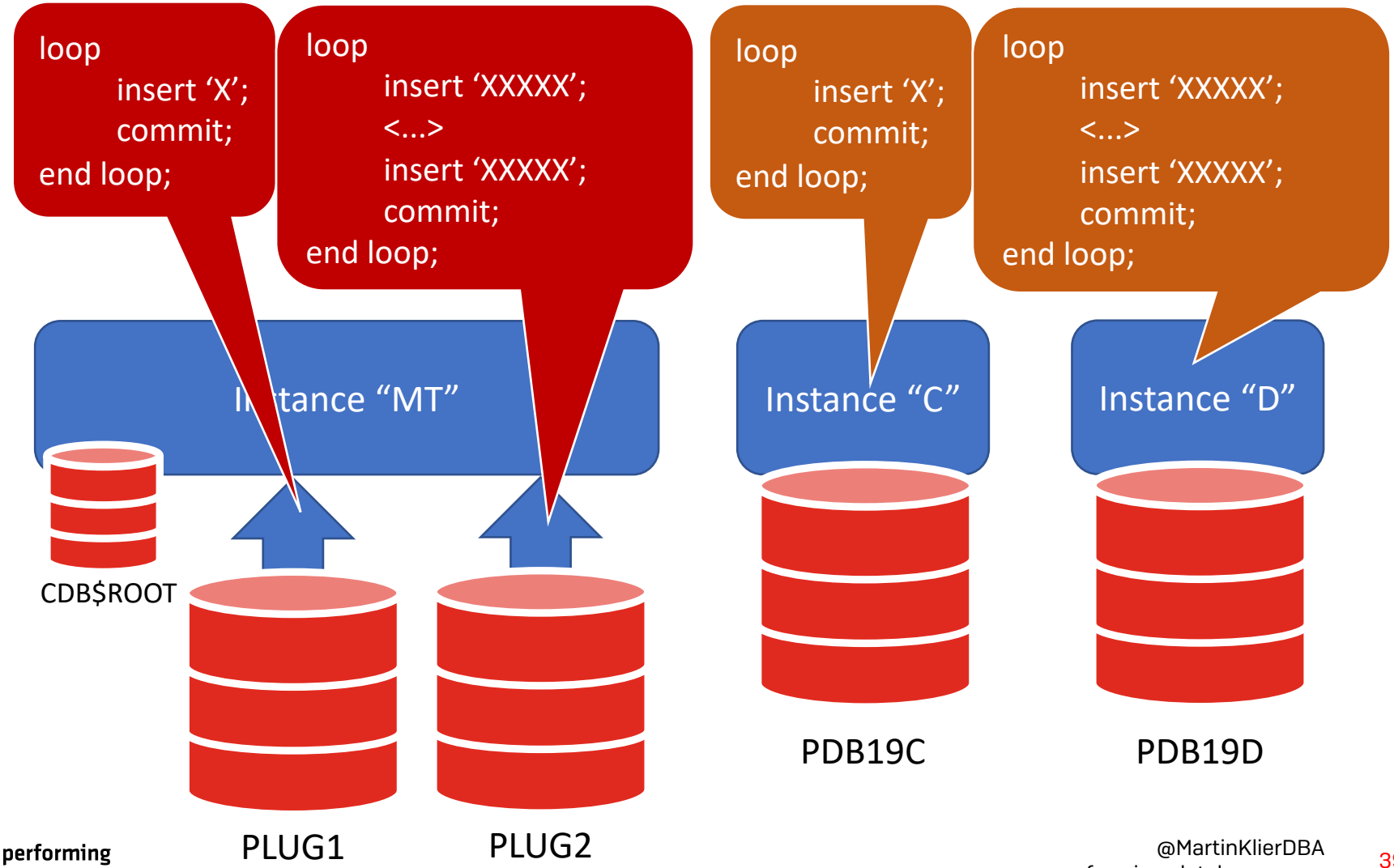
Case 2

Do classic databases influence each other
with small and fast vs. big and slow redo activity?

Do PDBs influence each other's
with small and fast vs. big and slow redo activity?



Case 2


loop
insert 'X';
commit;
end loop;
loop
insert 'XXXXX';
<...>
insert 'XXXXX';
commit;
end loop;
loop
insert 'X';
commit;
end loop;
loop
insert 'XXXXX';
<...>
insert 'XXXXX';
commit;
end loop;
Instance "MT"
Instance "C"
Instance "D"
CDB$ROOT
PLUG1
PLUG2
PDB19C
PDB19D

Case 2 - Result

PDB	PIT	WORKLOAD	ITERATIONS	RUNTIME_SE	IPS
PDB19C	17.11.2021-23:37:46	Mini_loader	1568999	10	14527.7685

PDB	PIT	WORKLOAD	ITERATIONS	RUNTIME_SE	IPS
PDB19D	17.11.2021-23:37:47	Maxi_loader	683996	10	6392.48598

PDB	PIT	WORKLOAD	ITERATIONS	RUNTIME_SE	IPS
PLUG1	17.11.2021-23:43:28	Mini_loader	1326999	26	5103.84231

PDB	PIT	WORKLOAD	ITERATIONS	RUNTIME_SE	IPS
PLUG2	17.11.2021-23:43:28	Maxi_loader	1575996	26	5992.38023

14k/s

6k/s

5k/s

5k/s

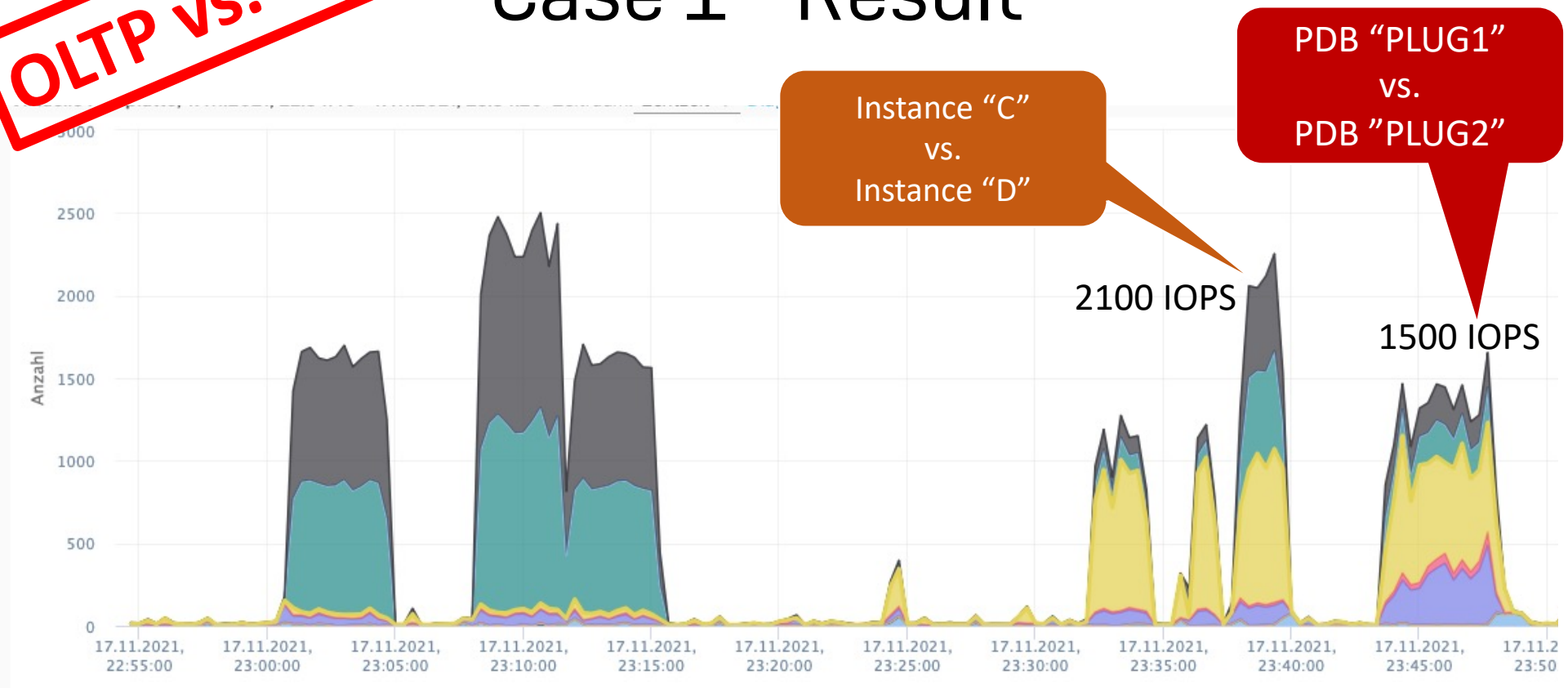
Two Instances – no influence big vs. small Redo

Two PDBs – negative impact on small Redo

(Same in one Instance, again...)

OLTP vs. Loading

Case 1 - Result



The “big-load” PDB throttles down
the “small-and-fast” PDB’s commit rate.

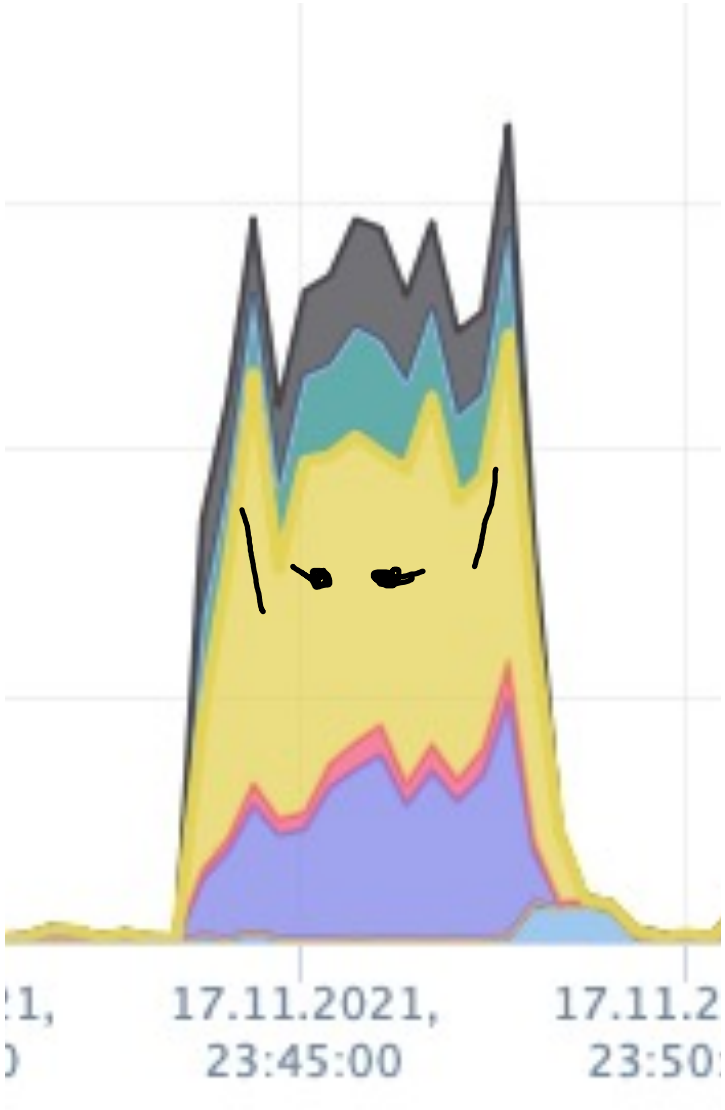
Conclusion

Case 1:
Two small-and-fast workloads in PDBs
go well together in Multitenant
more efficiently than having two “Classic” instances.

Case 2:
Small/fast Redo
does not love sharing a Multitenant DB
with a Big/slow Redo workload.

Know your needs!

Boo!



Karl Arao, wherever you are:
We salute you! ;)

Meet & Greet

martin.klier@performing-db.com
www.performing-databases.com

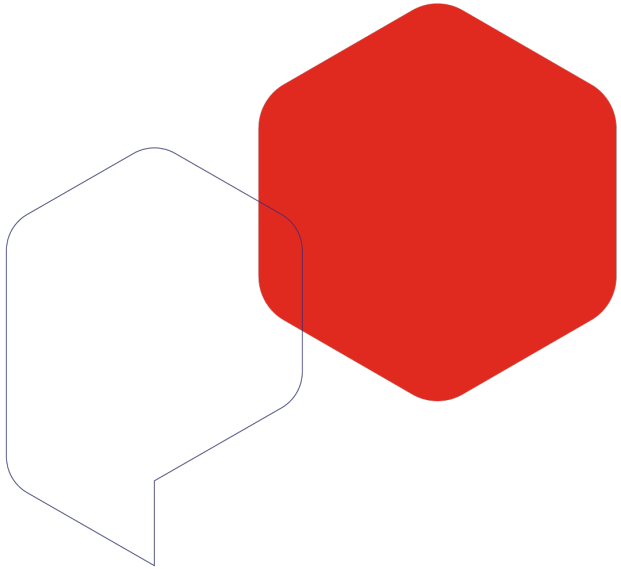
Many national // international events

DOAG
Deutsche ORACLE-Anwendergruppe e.V.

IT-Tage
2021

Efficiency is

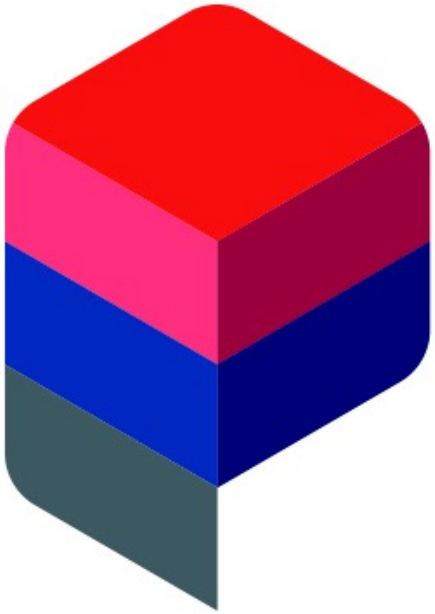
to stay in touch!



martin.klier@performing-db.com

http://www.performing-databases.com

performing databases



performing databases