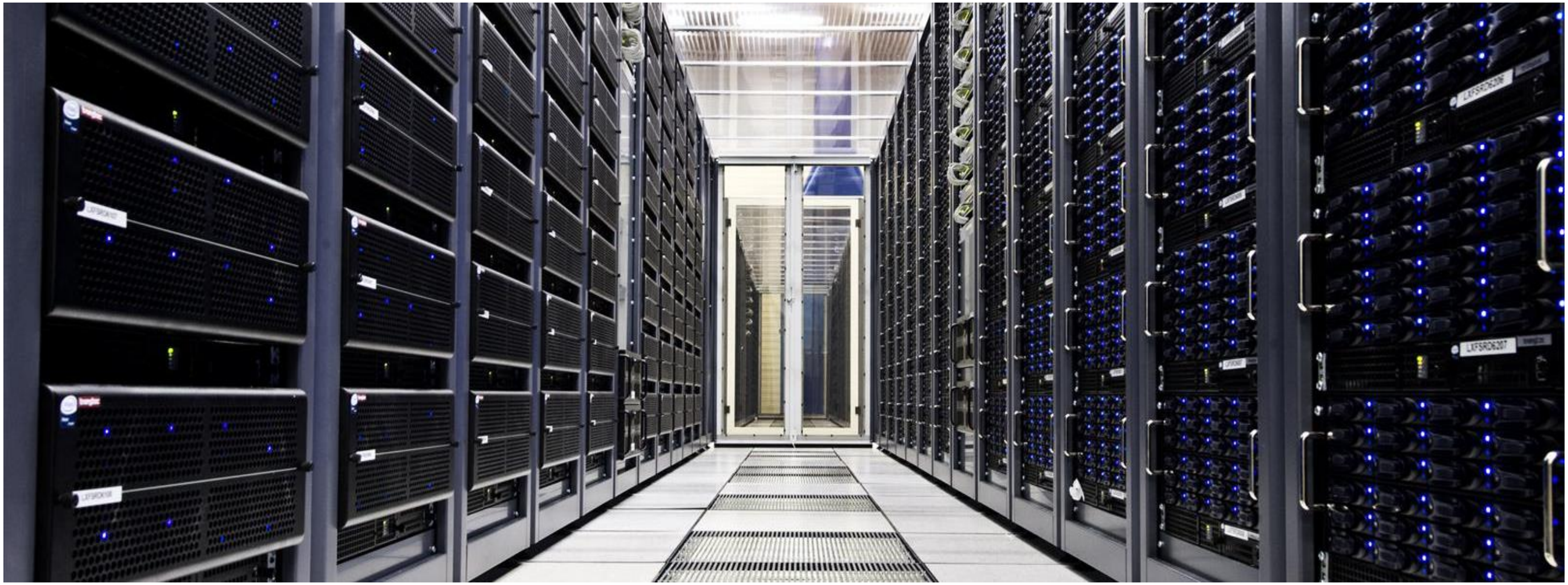
CERN

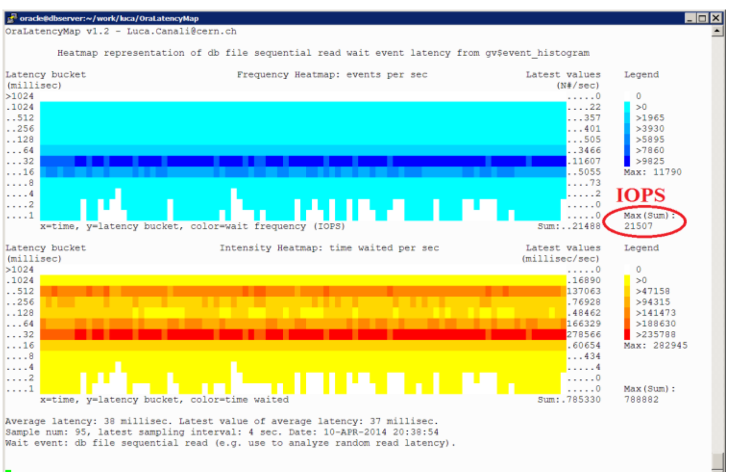
A Closer Look at CALIBRATE_IO

Luca Canali, CERN
Marcin Blaszczyk, CERN
UKOUG TECH14, Liverpool, December 2014



About Luca

- Senior DBA and team lead at CERN IT
  - Joined CERN in 2005
  - Working with Oracle RDBMS since 2000
- Passionate to learn and share knowledge, how to get most value from database technology
- @LucaCanaliDB - http://cern.ch/canali

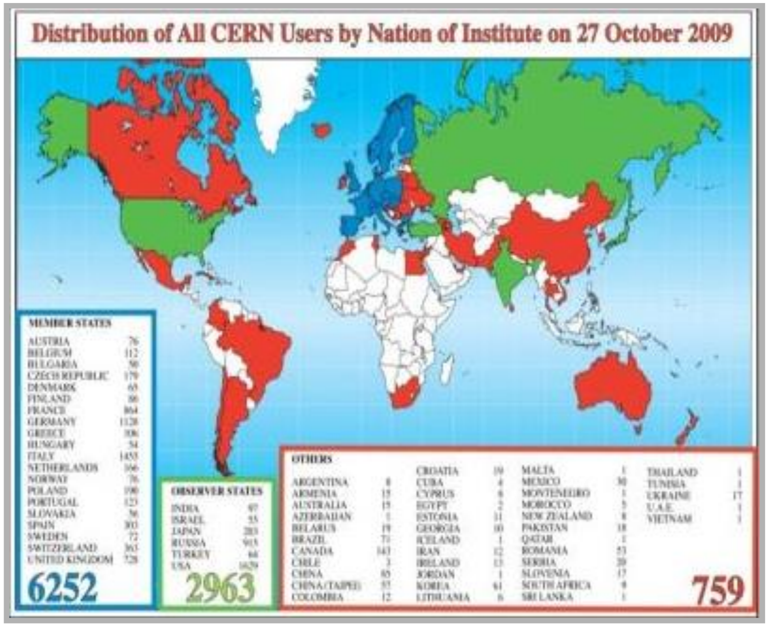


OakTable.net

About CERN

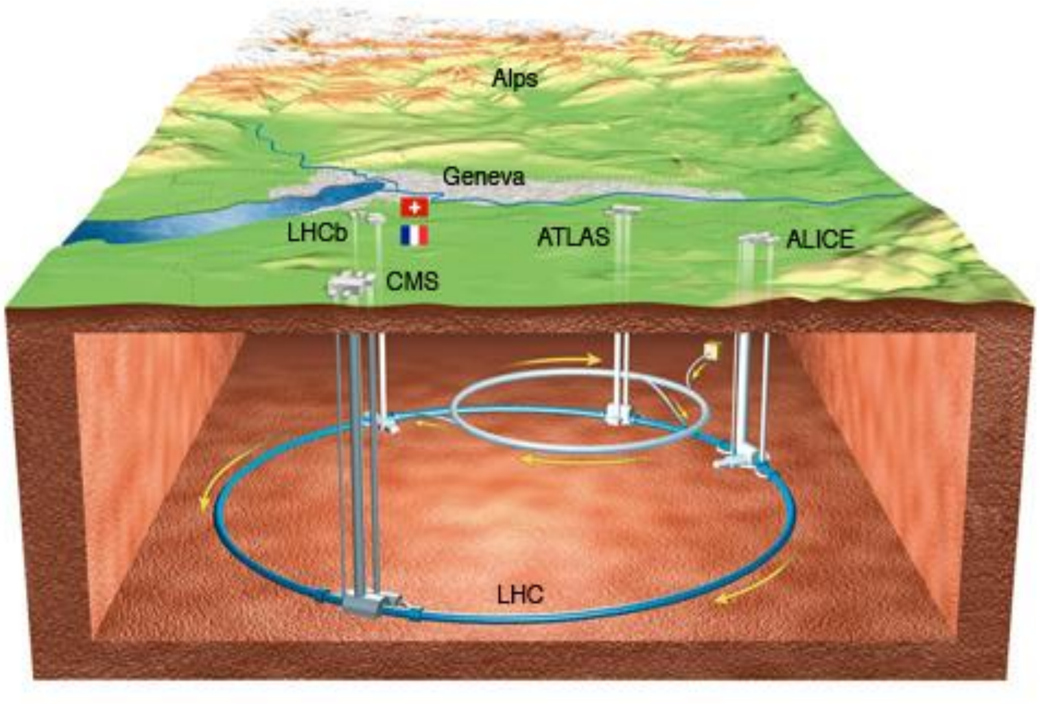
- CERN - European Laboratory for Particle Physics
- Founded in 1954 by 12 countries for fundamental physics research in a post-war Europe
- Today 21 member states + world-wide collaborations
  - About ~1000 MCHF yearly budget
  - 2’300 CERN personnel + 10’000 users from 110 countries

YEARS/ANS CERN



LHC is the World's Largest Particle Accelerator

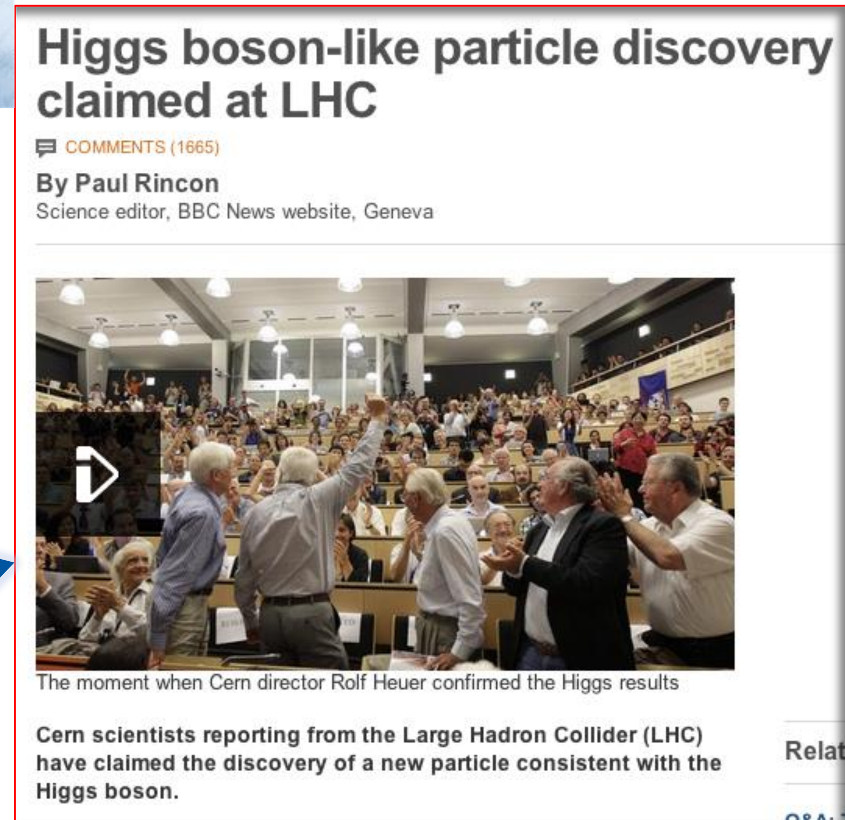
- LHC = Large Hadron Collider
  - 27km ring of superconducting magnets
  - Currently undergoing upgrades, restart in 2015



From particle to article..

How do you get from this

to this



Higgs boson-like particle discovery claimed at LHC

COMMENTS (1665)

By Paul Rincon
Science editor, BBC News website, Geneva

The moment when Cern director Rolf Heuer confirmed the Higgs results

Cern scientists reporting from the Large Hadron Collider (LHC) have claimed the discovery of a new particle consistent with the Higgs boson.

CERN Database Services

- ~100 Oracle databases, most of them RAC
  - Mostly NAS storage plus some SAN with ASM
  - ~500 TB of data files for production DBs in total
- Example of critical production DBs:
  - LHC logging database ~170 TB, expected growth up to ~70 TB/year
- But also as DBaaS, as single instances*
  - 160 MySQL CE (5.6, Dec 2014)
  - 16 PostgreSQL databases (version 9.2, Dec 2014)
  - 12 Oracle DBs (11g and 12c)

How I Became Interested in the Subject of Evaluating CALIBRATE_IO

Luca Canali @LucaCanaliDB · Mar 3

The question of CALIBRATE_IO vs. SLOB just came up at a Q&A at #HotSym14. I am for SLOB #justsaying

3 2

About Marcin

- ~9 years of working with Oracle DBs
  - 7+ of administering mission critical DB systems
- 5 years spent @CERN DB group
- Certified Oracle DBA & SQL Expert

ORACLE Certified Professional

- 3rd time speaking @UKOUG/TECH conference

- Privately
  - Fan of motorbikes and ….
    sport fishing☺

Testing Oracle I/O

- I/O is critical for databases
- Learn techniques for
  - Investigating
  - Testing
  - Troubleshooting
  - Performance optimization

- How to measure Oracle DB workload?

Throughput, IOPS, Latency

- In the context of storage testing
- **IOPS** number of I/O operations per second
  - Random Read IOPS very important for OLTP
- **Latency** time to perform single operation
- **Throughput** amount of data moved in a given time
  - Depends on IOPS, Latency and type of data access
- Meaningful numbers - only if we know what happens on the storage level

Why Use Calibrate_IO to Test Storage?

- Simple & easy to run
  - Works out of the box
  - Integrated into DB engine
  - Do not require additional installation
  - Works on Active Data Guard

Calibrate_IO – Procedure

- User with SYSDBA privilege
- TIMED_STATISTICS=TRUE (STATISTICS_LEVEL=TYPICAL)
- Needs asynchronous I/O

```
DECLARE
  l_latency  PLS_INTEGER;
  l_iops     PLS_INTEGER;
  l_mbps     PLS_INTEGER;
BEGIN
   DBMS_RESOURCE_MANAGER.calibrate_io (num_physical_disks => &1,
                                       max_latency        => &2,
                                       max_iops           => l_iops,
                                       max_mbps           => l_mbps,
                                       actual_latency     => l_latency);
  DBMS_OUTPUT.put_line('Max IOPS = ' || l_iops);
  DBMS_OUTPUT.put_line('Max MBPS = ' || l_mbps);
  DBMS_OUTPUT.put_line('Latency  = ' || l_latency);
END;
/
```

Calibrate_IO – Input Parameters

- **num_physical_disks**
  - „*Approximate number of physical disks in the storage*"
  - Used to determine initial I/O load for calibration run
- **max_latency**
  - „*Maximum tolerable latency for database-block-sized IO requests*"
  - We used value of 100 miliseconds
- So Let's try!

Calibrate_IO – Workload

- We can use:
  - GV$SESSION
  - GV$ACTIVE_SESSION_HISTORY
- Drilling down „***Disk File I/O Calibration***” wait event

```
system@ORCL> @top   ← SCRIPT QUERYING FROM GVSESSION

 INST_ID    SID  SERIAL USERNAME     SQL_ID CALL_DT EVENT                     SERV_MOD_ACTION
-------- ------ ------- ----------- ------ ------- ------------------------- ---------------
→ 1st RAC NODE (…)
       1    387    1311 SYS  (CS0E)                26 Disk file I/O Calibration SYS$USERS
       1    390   14655 SYS  (CS09)                26 Disk file I/O Calibration SYS$USERS
       1   2658    7163 SYS  (CS08)                27 Disk file I/O Calibration SYS$USERS
       1   2657    8981 SYS  (CS0D)                26 Disk file I/O Calibration SYS$USERS
       1   1904     187 SYS  (CS0C)                26 Disk file I/O Calibration SYS$USERS
       1   1523     375 SYS  (CS0B)                26 Disk file I/O Calibration SYS$USERS
       1   1141      23 SYS  (CS0A)                26 Disk file I/O Calibration SYS$USERS
       1    765      29 SYS  (CS0F)                26 Disk file I/O Calibration SYS$USERS
→ 2nd RAC NODE (…)
       2      8    3253  (CS00)                    27 Disk file I/O Calibration SYS$USERS
       2   2277    3951  (CS06)                    27 Disk file I/O Calibration SYS$USERS
       2   1899      79  (CS05)                    27 Disk file I/O Calibration SYS$USERS
       2   1520   18449  (CS04)                    27 Disk file I/O Calibration SYS$USERS
       2    765   27311  (CS03)                    27 Disk file I/O Calibration SYS$USERS
       2    387   59069  (CS02)                    27 Disk file I/O Calibration SYS$USERS
       2     11   31477  (CS01)                    27 Disk file I/O Calibration SYS$USERS
       2   2654    8195  (CS07)                    27 Disk file I/O Calibration SYS$USERS
```

How to Measure I/O Workload?

- GV$IOFUNCMETRIC - available since 11g (11.1)
- Displays I/O statistics information, e.g:
- Throughput
  - Single block / Multiblock megabytes read/write per second
- IOPS
  - Single block / Multiblock read/write requests per second

```
system@ORCL> @iops

       INST_ID BEGIN_TIME                    IOPS
-------------- -------------------- --------------
             1 01-DEC-2014 11:19:00           3171         ← 1st RAC NODE
             2 01-DEC-2014 11:18:21           2402         ← 2nd RAC NODE


system@ORCL> @throughput

B_TIME               E_TIME               READ_TOT_MBPS WRITE_TOT_MBPS       TOT_MBPS
-------------------- -------------------- -------------- -------------- --------------
01-DEC-2014 11:20:21 01-DEC-2014 11:21:21            607            321            928
```

Calibrate_IO: 1st Phase

- Random reads from all nodes
- DB level: GV$IOFUNCMETRIC

```
system@ORCL> @iometric_details      ← SCRIPT QUERYING FROM GV$IOFUNCMETRIC

INST_ID BEGIN_TIME        FUNCTION RD_IOPS_SM RD_IOPS_LG RD_MBPS_SM RD_MBPS_LG WT_IOPS_SM WT_IOPS_LG WT_MBPS_SM WT_MBPS_LG
------- ----------------- -------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
      1 02-12-14 22:20:09 Others         1813          0         14          1         84          0          0          0
      2 02-12-14 22:21:31 Others         1856          0         15          0         21          0          0          0

BEGIN_TIME        END_TIME          READ_TOT_IOPS  READ_TOT_MBPS WRITE_TOT_IOPS WRITE_TOT_MBPS
----------------- ----------------- -------------- -------------- -------------- --------------
02-12-14 22:21:31 02-12-14 22:22:31           3763             31            215              0
```

- OS level: `strace –p <PID>`

```
io_submit(47804644147200, 1, {{0x2b7a62e600b0, 0, 0, 0, 288}}) = 1
times({tms_utime=26, tms_stime=38, tms_cutime=0, tms_cstime=0}) = 1776996225
io_getevents(47804644147200, 1, 128, {{0x2b7a62e60320, 0x2b7a62e60320, 8192, 0}}, {600, 0}) = 1
(...)
io_submit(47804644147200, 1, {{0x2b7a62e611c0, 0, 0, 0, 286}}) = 1
times({tms_utime=26, tms_stime=38, tms_cutime=0, tms_cstime=0}) = 1776996226
io_getevents(47804644147200, 1, 128, {{0x2b7a62e600b0, 0x2b7a62e600b0, 8192, 0}}, {600, 0}) = 1
```

8KB reads

Calibrate_IO: 1st Phase

- Random reads from all nodes
- DB level: GV$IOFUNCMETRIC

```
system@ORCL> @iometric_details     ← SCRIPT QUERYING FROM GV$IOFUNCMETRIC

INST_ID BEGIN_TIME        FUNCTION RD_IOPS_SM RD_IOPS_LG RD_MBPS_SM RD_MBPS_LG WT_IOPS_SM WT_IOPS_LG WT_MBPS_SM WT_MBPS_LG
------- ----------------- -------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
      1 02-12-14 22:24:09 Others         2404          0         19          1         51          0          0          0
      2 02-12-14 22:23:31 Others         2568          0         20          0         21          0          0          0

BEGIN_TIME        END_TIME              READ_TOT_IOPS  READ_TOT_MBPS WRITE_TOT_IOPS WRITE_TOT_MBPS
----------------- ----------------- --------------- -------------- -------------- --------------
02-12-14 22:23:31 02-12-14 22:24:31            4972             40             72              0
```

- OS level: `strace –p <PID>`

```
io_submit(47804644147200, 1, {{0x2b7a62f6c140, 0, 0, 0, 275}}) = 1
times({tms_utime=130, tms_stime=166, tms_cutime=0, tms_cstime=0}) = 1777017121
io_submit(47804644147200, 1, {{0x2b7a62e627b0, 0, 0, 0, 278}}) = 1
times({tms_utime=130, tms_stime=166, tms_cutime=0, tms_cstime=0}) = 1777017121
io_submit(47804644147200, 1, {{0x2b7a62e90a40, 0, 0, 0, 281}}) = 1
times({tms_utime=130, tms_stime=166, tms_cutime=0, tms_cstime=0}) = 1777017121
(...)
io_getevents(47804644147200, 1, 128, {{0x2b7a62e781d0, 0x2b7a62e781d0, 8192, 0}, {0x2b7a62f78e00,
(...)
0x2b7a62ea6e20, 8192, 0}}, {600, 0}) = 128
```

ASYNCH IO

Calibrate_IO: 2nd Phase

- Sequential reads from all nodes
- DB level: GV$IOFUNCMETRIC

```
system@ORCL> @iometric_details     ← SCRIPT QUERYING FROM GV$IOFUNCMETRIC

INST_ID BEGIN_TIME        FUNCTION RD_IOPS_SM RD_IOPS_LG RD_MBPS_SM RD_MBPS_LG WT_IOPS_SM WT_IOPS_LG WT_MBPS_SM WT_MBPS_LG
------- ----------------- -------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
      1 02-12-14 23:00:09 Others            5          0          0        554          1         15          0         15
      2 02-12-14 23:00:31 Others            3          0          0        715         20          0          0          0

BEGIN_TIME        END_TIME          READ_TOT_IOPS READ_TOT_MBPS WRITE_TOT_IOPS WRITE_TOT_MBPS
----------------- ----------------- ------------- ------------- -------------- --------------
02-12-14 23:00:09 02-12-14 23:01:09             8          1268             36             15
```

- OS level: `strace –p <PID>`

```
io_submit(46921770770432, 1, {{0x2b22821be9b0, 0, 0, 0, 290}}) = 1
times(NULL)                             = 1831042508
(…)
io_submit(46921770770432, 1, {{0x2b2282259350, 0, 0, 0, 298}}) = 1
times(NULL)                             = 1831042509
io_getevents(46921770770432, 1, 128, {{0x2b22821e9050, 0x2b22821e9050, 1048576, 0}}, {600, 0}) = 1
times(NULL)                             = 1831042509
io_submit(46921770770432, 1, {{0x2b22821e9050, 0, 0, 0, 287}}) = 1
times(NULL)                             = 1831042509
io_getevents(46921770770432, 34, 128, {{0x2aacd396d310, 0x2aacd396d310, 1048576, 0},
 (...) , {600, 0}) = 34
```

1MB reads

Calibrate_IO: 3rd Phase

- Sequential reads from a single session
- DB level: GV$IOFUNCMETRIC

```
system@ORCL> @iometric_details     ← SCRIPT QUERYING FROM GV$IOFUNCMETRIC

INST_ID BEGIN_TIME        FUNCTION RD_IOPS_SM RD_IOPS_LG RD_MBPS_SM RD_MBPS_LG WT_IOPS_SM WT_IOPS_LG WT_MBPS_SM WT_MBPS_LG
------- ----------------- -------- ---------- ---------- ---------- ---------- ---------- ---------- ---------- ----------
      1 02-12-14 23:17:09 Others            8          0          0          0         81          0          0          0
      2 02-12-14 23:16:31 Others            5          0          0        328         11          0          0          0

BEGIN_TIME        END_TIME          READ_TOT_IOPS READ_TOT_MBPS WRITE_TOT_IOPS WRITE_TOT_MBPS
----------------- ----------------- ------------- ------------- -------------- --------------
02-12-14 23:16:31 02-12-14 23:17:32            14           328             92              0
```

- OS level: `strace –p <PID>`

```
io_submit(46921770770432, 1, {{0x2b22821be9b0, 0, 0, 0, 290}}) = 1
times(NULL)                          = 1831042508
(…)
io_submit(46921770770432, 1, {{0x2b2282259350, 0, 0, 0, 298}}) = 1
times(NULL)                          = 1831042509
io_getevents(46921770770432, 1, 128, {{0x2b22821e9050, 0x2b22821e9050, 1048576, 0}}, {600, 0}) = 1
times(NULL)                          = 1831042509
io_submit(46921770770432, 1, {{0x2b22821e9050, 0, 0, 0, 287}}) = 1
times(NULL)                          = 1831042509
io_getevents(46921770770432, 34, 128, {{0x2aacd396d310, 0x2aacd396d310, 1048576, 0}},
 (...) , {600, 0}) = 34
```

1MB reads

I/O Distribution Across Storage

- Calibrate_IO reads from all files in the database
- Load proportional to file size

```
system@ORCL> SELECT sysdate, file_no, filetype_name, sum(small_read_reqs),
sum(small_read_servicetime)
FROM GV$IOSTAT_FILE
WHERE filetype_name='Data File' GROUP BY file_no, filetype_name
ORDER BY file_no, filetype_name;

SYSDATE           FILE_NO FILETYPE_NAME SMALL_READ_REQS SMALL_READ_SERVICETIME
----------------- ------- ------------- --------------- ----------------------
02-12-14 23:57:55       1 Data File             1045986               33694991
02-12-14 23:57:55       2 Data File           789106250              409393779
(…)

system@ORCL> /
SYSDATE           FILE_NO FILETYPE_NAME SMALL_READ_REQS SMALL_READ_SERVICETIME
----------------- ------- ------------- --------------- ----------------------
02-12-14 23:58:21       1 Data File             1046037               33702193
02-12-14 23:58:21       2 Data File           789106352              409408018
(…)

system@ORCL> /
SYSDATE           FILE_NO FILETYPE_NAME SMALL_READ_REQS SMALL_READ_SERVICETIME
----------------- ------- ------------- --------------- ----------------------
02-12-14 23:59:28       1 Data File             1046199               33781055
02-12-14 23:59:28       2 Data File           789106607              409532304
(…)
02-12-14 23:59:28       8 Data File               70293                3234810
```

Calibrate_IO – Outputs

1. **MAX_IOPS**
   Maximum number of I/O requests that can be sustained
2. **MAX_MBPS**
   Maximum throughput of I/O that can be sustained
3. **LATENCY**
   Average latency of database-block-sized I/O
4. **MAX_PMBPS**
   Maximum throughput of large I/O requests that can be sustained by a single process
   max_pmbs avaliable only in DBA_RSRC_IO_CALIBRATE:

```
Max IOPS = 4997
Max MBPS = 1257
Latency  = 42

PL/SQL procedure
successfully
completed.
```

```
system@ORCL> select * from DBA_RSRC_IO_CALIBRATE;

START_TIME        END_TIME          MAX_IOPS MAX_MBPS MAX_PMBPS LATENCY NUM_PHYSICAL_DISKS
----------------- ----------------- -------- -------- --------- ------- ------------------
02-12-14 17:08:01 02-12-14 17:20:37     4997     1257       398      42                 50
```

- Automatic degree of parallelism uses MAX_PMBPS and MAX_MBPS

JBOD & ASM

46 SATA disks in JBOD, 2 node RAC on 11.2.0.4, Linux (RHEL)

```
system@ORCL> @calio 50 100
old   6:          DBMS_RESOURCE_MANAGER.calibrate_io (num_physical_disks => &&1,
new   6:          DBMS_RESOURCE_MANAGER.calibrate_io (num_physical_disks => 50,
old   7:                                               max_latency        => &&2,
new   7:                                               max_latency        => 100,
Max IOPS = 4778
Max MBPS = 1251
Latency  = 47

PL/SQL procedure successfully completed.

Elapsed: 00:09:54.25
```

- IOPS
  - Value seems correct (46 disks * ~100IOPS)
- Latency:
  - What does this number mean to us?
  - Different values reported for the same system and same input parameters
- Throughput:
  - Value too small but close to expected maximum
  - 2ports 4Gb/s per node, we expect ~1600MBPS for this system, which we confirmed by measurements taken with parallel query

NAS Storage

NAS with SSD Cache, 2 node RAC on 11.2.0.4, Linux (RHEL)

```
system@ORCL> @calio 60 100
old   6:          DBMS_RESOURCE_MANAGER.calibrate_io (num_physical_disks => &&1,
new   6:          DBMS_RESOURCE_MANAGER.calibrate_io (num_physical_disks => 60,
old   7:                                               max_latency        => &&2,
new   7:                                               max_latency        => 100,
Max IOPS = 4378
Max MBPS = 400
Latency  = 5

PL/SQL procedure successfully completed.

Elapsed: 00:10:44.90
```

- IOPS
  - Value seems to be correct (60 SATA disks, 7200rpm 60*~70 IOPS – we expect ~4200 IOPS)
- Latency:
  - What does this number mean to us?
  - Different values reported for the same system and same input parameters
- Throughput:
  - Seems too low (this system is able to reach 1 GBPS as measured with other methods)

Latency & Asynchronous I/O

- Let's rerun Calibrate_IO and measure latency
  - Using GV$SYSMETRIC:

```
system@ORCL> @sysmetric.sql ← SCRIPT QUERYING FROM GV$SYSMETRIC

Time+Delta     Metric                                          Total
-------------- ---------------------------------------------- ----------
12:04:59 /60s  Average Synchronous Single-Block Read Latency - Millisec   130

12:06:00 /60s  Average Synchronous Single-Block Read Latency - Millisec   167

12:06:21 /60s  Average Synchronous Single-Block Read Latency - Millisec   328

12:07:00 /60s  Average Synchronous Single-Block Read Latency - Millisec   588

12:08:00 /60s  Average Synchronous Single-Block Read Latency - Millisec   1981.1

12:09:00 /60s  Average Synchronous Single-Block Read Latency - Millisec   5001.5
```

- Not relevant - we know that I/O is ASYNCH

Latency & Asynchronous I/O

- Let’s rerun Calibrate_IO and measure latency
  - Using GV$EVENT_HISTOGRAM:

```
sys@INTR:SQL> @ehm 60 Disk%Calibration ← SCRIPT QUERYING FROM GV$EVENT_HISTOGRAM

waiting for 60 sec (delta measurement interval = 60 sec)

Wait (ms)    N#           Event                   Last update time
----------- ----------- ----------------------- -----------------------------------
1             32968       Disk file I/O Calibration 04-DEC-14 04.22.17.057413 PM +01:00
2             25805       Disk file I/O Calibration 04-DEC-14 04.22.17.055425 PM +01:00
4             40757       Disk file I/O Calibration 04-DEC-14 04.22.17.057053 PM +01:00
8             48935       Disk file I/O Calibration 04-DEC-14 04.22.17.055730 PM +01:00
16            31128       Disk file I/O Calibration 04-DEC-14 04.22.17.054248 PM +01:00
32            6078        Disk file I/O Calibration 04-DEC-14 04.22.17.046247 PM +01:00
64            189         Disk file I/O Calibration 04-DEC-14 04.22.17.001269 PM +01:00
128           13          Disk file I/O Calibration 04-DEC-14 04.22.12.773398 PM +01:00
256           6           Disk file I/O Calibration 04-DEC-14 04.22.12.533209 PM +01:00
512           10          Disk file I/O Calibration 04-DEC-14 04.22.12.669338 PM +01:00

Avg_wait(ms) N#          Tot_wait(ms) Event
------------ ---------- ------------ -------------------
5.1          185258     947481.3     Disk file I/O Calibration
```

- Not relevant – event represents Random & Sequential IO

Calibrate_IO – Some Conclusions

- Easy to generate I/O workload
- Use Asynchronous I/O
- Describe I/O subsystem characteristics
  - **IOPS** –
    - For systems we tested value seems to be correct
  - **Latency** –
    - Average value seem to be incorrect and may vary (even for the same system and the same input parameters)
  - **Throughput** –
    - For JBOD & ASM is close to expected value (~80% of max)
    - For NAS storage seems to be too small (~50% of max)

IOPS and Latency Measurements Should Go Together

- Latency figures reveal details about the IOPS
  - I/O from cache or SSD, or HDD, high latency values..
- Latency drill down
  - With latency histograms
  - Heat map representation also very useful

Monitoring Latency - Snapshots

- Custom script: ehm.sql

```
primary:system@orclrac1> @ehm 60 db%sequential

waiting for 60 sec (delta measurement interval = 60 sec)

Wait (ms)    N#          Event                    Last update time
----------- ----------- ------------------------ -----------------------------------
1            12588       db file sequential read 20-NOV-13 04.52.02.549024 PM +02:00
2            638         db file sequential read 20-NOV-13 04.52.02.323209 PM +02:00
4            241         db file sequential read 20-NOV-13 04.52.00.017278 PM +02:00
8            1032        db file sequential read 20-NOV-13 04.52.02.407010 PM +02:00
16           6128        db file sequential read 20-NOV-13 04.52.02.520877 PM +02:00
32           3865        db file sequential read 20-NOV-13 04.52.02.526403 PM +02:00
64           622         db file sequential read 20-NOV-13 04.52.02.475484 PM +02:00
128          48          db file sequential read 20-NOV-13 04.52.02.454875 PM +02:00
256          2           db file sequential read 20-NOV-13 04.51.35.738163 PM +02:00
512          1           db file sequential read 20-NOV-13 04.51.54.617231 PM +02:00
1024         13          db file sequential read 20-NOV-13 04.52.01.560293 PM +02:00
2048         0           db file sequential read 20-NOV-13 03.19.40.350234 PM +02:00
4096         0           db file sequential read 15-NOV-13 02.25.22.371191 AM +02:00
8192         0           db file sequential read 31-OCT-13 01.01.10.757675 AM +02:00
16384        0           db file sequential read 28-OCT-13 11.51.50.122887 PM +02:00
32768        0           db file sequential read 11-OCT-13 12.42.21.599088 PM +02:00
65536        0           db file sequential read 11-OCT-13 12.42.21.601458 PM +02:00
131072       0           db file sequential read 11-OCT-13 12.42.21.606092 PM +02:00

Avg_wait(ms) N#          Tot_wait(ms) Event
------------ ---------- ------------ -------------------
8.5          25177       214095.1     db file sequential read
```

How Many IOPS a Modern System Can Sustain is Often Not Well Defined

- IOPS
  - HDD and SSD have different limit
  - SSDs much more performant than HDD for random IO
- Systems with HDD and SSD cache
  - Max N# of IOPS depends of how much of the workload is served by SSD and how much from HDD

I/O Workload Generation and Measurements with ORION

- ORION (Oracle I/O Numbers)
  - Calibrate_io is a sort of simplified ORION integrated in the engine DB
- Latest versions have latency histogram details
- ORION allows to run tests at variable load
  - Study how the system reacts from low load to saturation
  - Can run mixed workload (read + write)
  - Several other useful feature..

Example of ORION Produced Histogram

```
Latency Histogram for small IOs @ Small=35 and Large=0
        Latency:                        # of IOs (read)          # of IOs
(write)
      256 - 512         us:                 0                       0
      512 - 1024        us:                 2                       0
     1024 - 2048        us:                 2                       0
     2048 - 4096        us:                 71                      0
     4096 - 8192        us:                 2126                    0
     8192 - 16384       us:                 9572                    0
    16384 - 32768       us:                 6149                    0
    32768 - 65536       us:                 2829                    0
    65536 - 131072      us:                 890                     0
   131072 - 262144      us:                 581                     0
   262144 - 524288      us:                 538                     0
   524288 - 1048576     us:                 655                     0
  1048576 - 2097152     us:                 460                     0
  2097152 - 4194304     us:                 64                      0
  4194304 - 8388608     us:                 1                       0
```

Passive Benchmarking

- We trust the measuring tool to do all the work
- We run a given benchmarking tool and just collect the output
- For a complex system the results can be misleading
    - No understanding of why the system behaves in a certain way
    - Does not help in predicting the behaviour in the real-world scenarios

Active Benchmarking Provides Understanding

- Active benchmarking is about measuring the system while the benchmark workload runs
  - Use many measurement tools
  - Use standard tools
- This is a good methodology because it allows to
  - Understand why the system behaves in a certain way
  - Understand the limiting factors (bottlenecks)
  - Helps in predicting the systems behaviour
- Reference: Brendan Gregg's blog

SLOB for Storage Testing

- SLOB – "Silly little Oracle benchmark"
  - Free tool, written and maintained by Kevin Closson
  - Version 2.2 just released, check it out!
- Several advantages
  - Runs the workload directly from the Oracle DB
    - Excellent to produce concurrent random I/O
  - Configurable load (concurrency)
    - Allows to ramp up from low load to saturation
  - Mixed workload (read+write) possible
  - It's becoming a standard: simplify sharing results

Active Benchmarking with SLOB

- Run SLOB workload and measure it with standard tools
- Focus moves from benchmarking to measuring
  - Know what to measure, know how to measure!
  - Use many tools at different layers and compare
- Use AWR reports
- Use real-time measurement
  - From V$ views
  - From ASM
  - From OS
  - From the storage instrumentation

Use GV$ Views on the DB Instance to Measure I/O

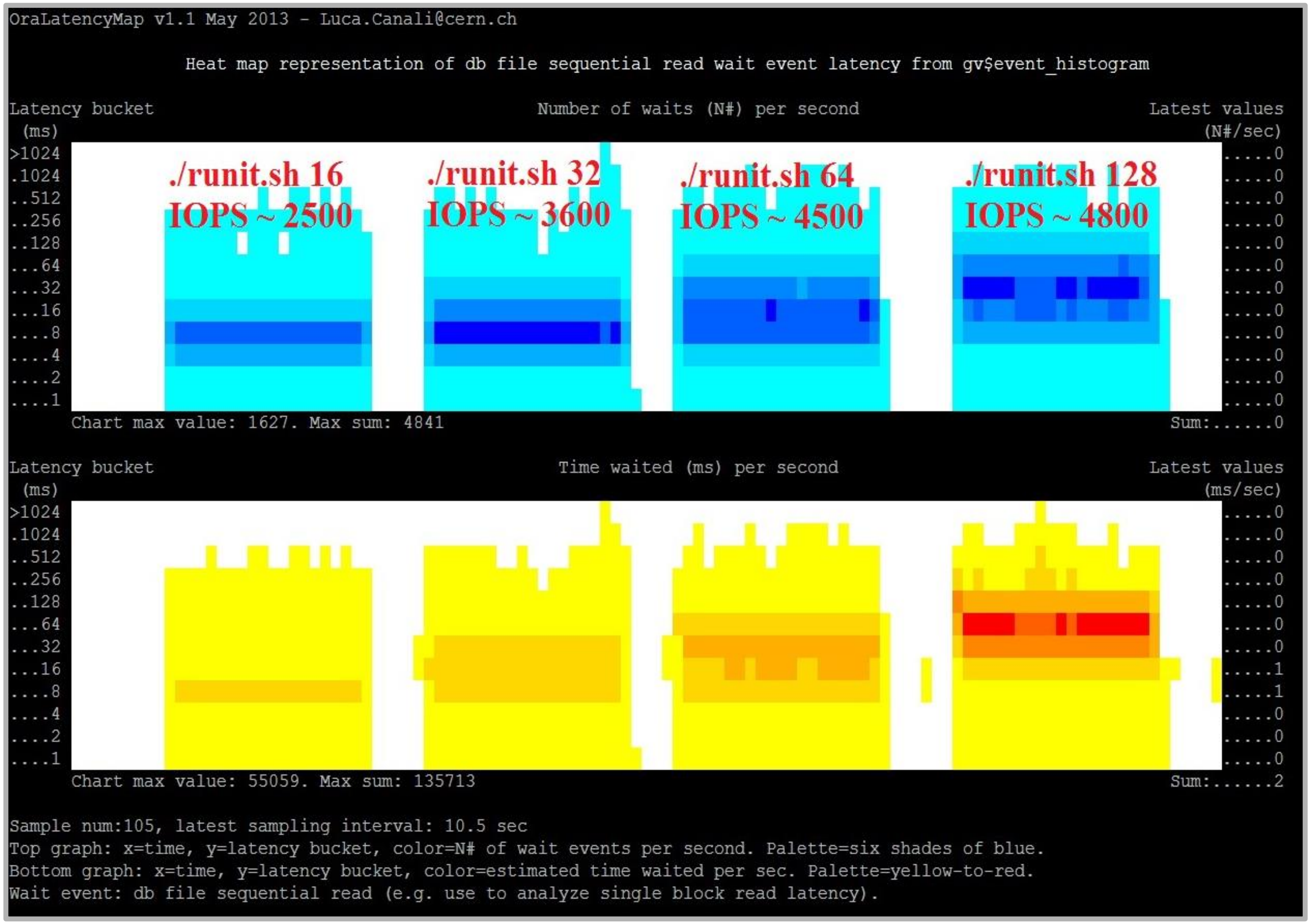
- Some of the interesting GV$ to use are:
  - GV$SYSMETRIC
  - GV$IOFUNCMETRIC
  - GV$IOSTAT_FILE (and rest of GV$IOSTAT_*)

Produce Latency Heatmaps from SLOB Tests

- Easy way to understand IOPS and latency using SLOB
  - Measure latency details using histograms of db file sequential read wait time from V$EVENT_HISTOGRAM
  - Plot data as latency heatmap (ex with OraLatencyMap)

- Note:
  - Oracle can execute SLOB workload with asynchronous I/O too, this shows as ‘db file parallel read’ wait event and does not represent a measurement of I/O latency
  - See notes on this slides to revert to ‘db file sequential read’ wait events

Example: Latency Heatmap of Random I/O at Increasing Load

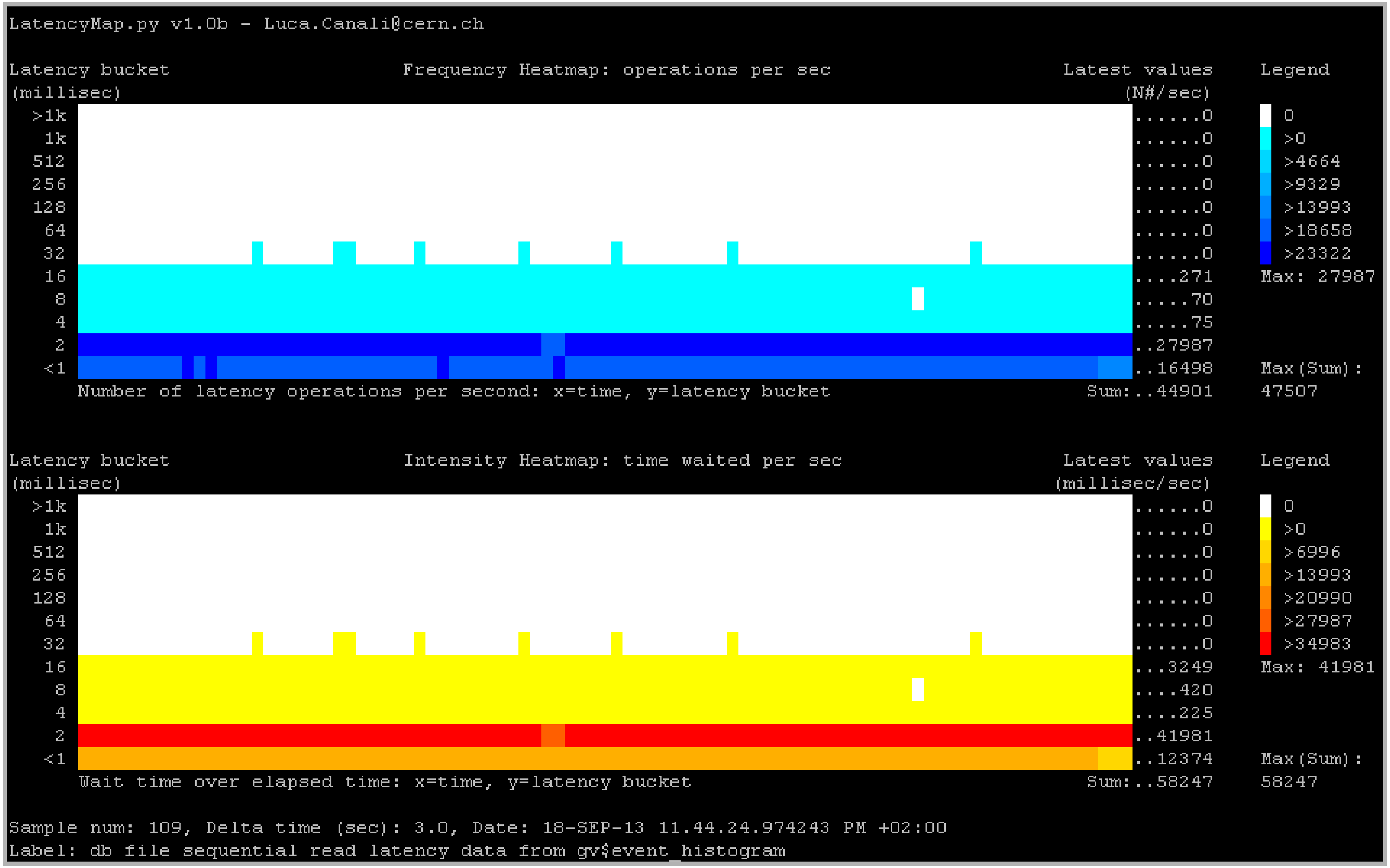


23 SAS disks JBOD & ASM

4 consecutive tests with increasing load

OraLatencyMap available at: http://canali.web.cern.ch/canali/resources.htm

Example: All I/Os from SSD cache
0.5 TB dataset, 100% in SSD, 56 sessions, random reads - NAS



ASM Examples

- Measure from ASM V$ metrics
  - GV$ASM_DISK_IOSTAT
  - GV$ASM_DISK_STAT
  - GV$ASM_DISKGROUP_STAT
- See also work of Bertrand Drouvot
  - asm_metrics.pl utility

Use OS Tools to Measure Activity on the I/O Subsystem

- Measure disk activity:
  - Use iostat, sar, collectl, dstat
- Advanced tools to measure latency
  - Oracle wait events do not measure correctly latency for asynchronous I/O
  - Measure directly from OS block device interface
  - Ftrace and SystemTap probes

  - See also talk “Modern Linux Tools for Oracle Troubleshooting”, in Hall 4A at 12:00

Another Way to Measure I/O from OS: Use Ftrace

- https://github.com/brendangregg/perf-tools

```
# ./iolatency 10
Tracing block I/O. Output every 10 seconds. Ctrl-C to to end.

  >=(ms) .. <(ms)   : I/O      |Distribution                          |
       0 -> 1       : 95       |##                                    |
       1 -> 2       : 74       |##                                    |
       2 -> 4       : 475      |#########                             |
       4 -> 8       : 2035     |######################################|
       8 -> 16      : 1245     |########################              |
      16 -> 32      : 37       |#                                     |
      32 -> 64      : 11       |#                                     |
      64 -> 128     : 7        |#                                     |
     128 -> 256     : 23       |#                                     |
     256 -> 512     : 10       |#                                     |
     512 -> 1024    : 4        |#                                     |
```

Oracle Wait Events for Asynchronous I/O Cannot Be Used to Study Latency

Example of how to measure I/O latency from the block I/O interface using SystemTap:

```
global latencyTimes, requestTime[10000]

probe ioblock_trace.request {
   requestTime[$bio] = gettimeofday_us()
}

probe ioblock.end {
   t = gettimeofday_us()
   s = requestTime[$bio]
   if (s > 0) {
       latencyTimes <<< (t-s)
       delete requestTime[$bio]
   }
}
```

Interpreting the Results

- How to translate IOPS and latency measurements into answers to questions that matter?
  - Will production workload be OK on this storage?
- Measure production workload and compare
  - use active benchmarking and workload measurement with standard tools
  - Use the same measurement tools against production
- Do application-specific stress tests too

Conclusions

- Storage is critical and complex, need to test!
- CALIBRATE_IO provides an easy way to generate I/O load
  - However the output misses critical details
- Methodology: use active benchmarking
  - Run an I/O workload generator
  - Measure key metrics with standard tools
- SLOB is a very good I/O workload generator
  - Great help for active benchmarking of storage

Acknowledgements and Contacts

- CERN Colleagues and in particular the Database Services Group
  - We have a blog http://db-blog.web.cern.ch

- Contacts:

Luca.Canali@cern.ch,

Marcin.Blaszczyk@cern.ch

* All scripts demonstrated in this presentation: http://canali.web.cern.ch/canali/resources.htm

www.cern.ch