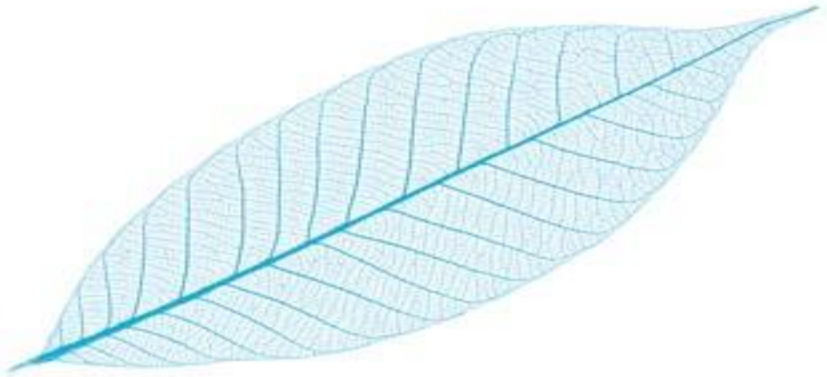
By Lasse Jenssen

Performance doesn't happen by accident

- Database Performance for DBAs and Developers (Java & others)



EVRY

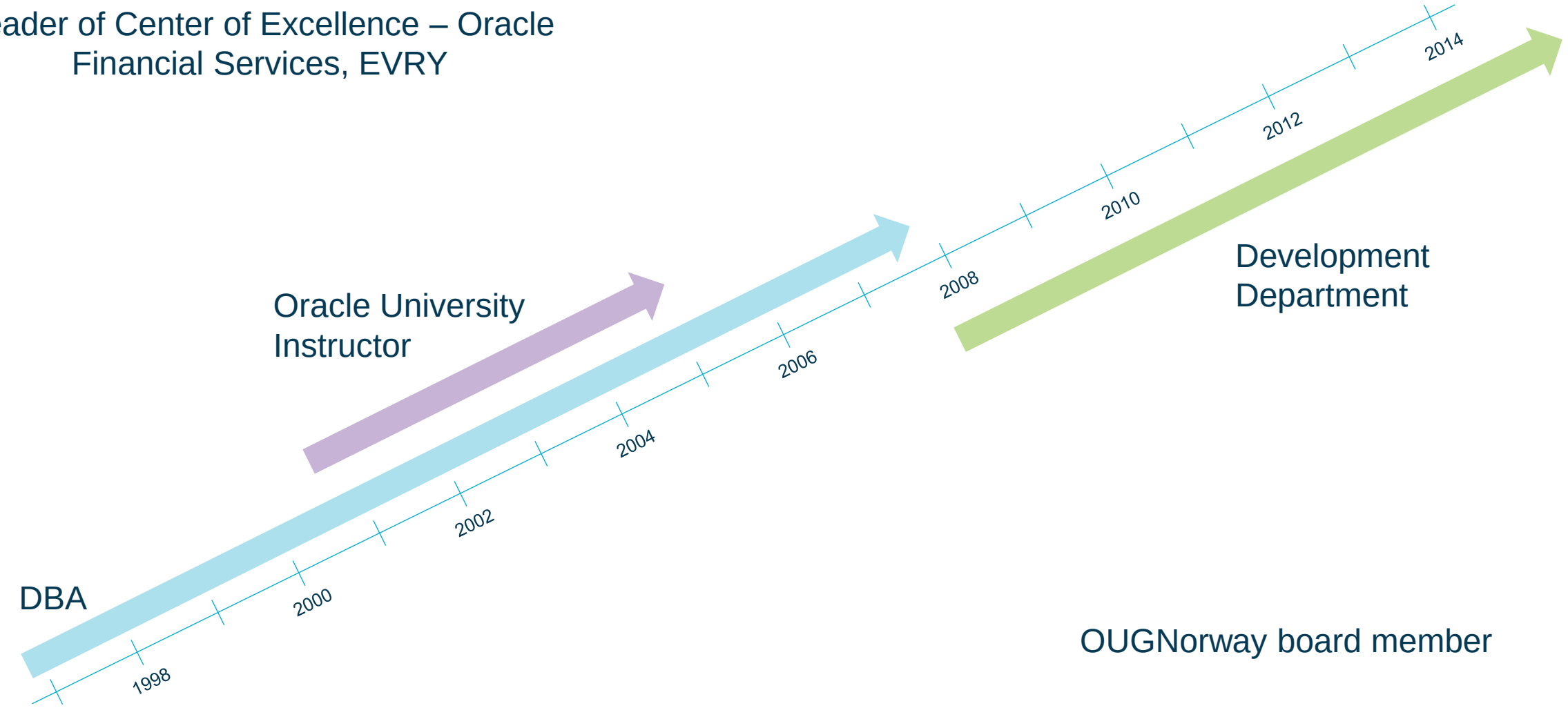
I am a proud member of an EMEA Oracle User Group

Are you a member yet?

Meet us at
south upper lobby of Moscone South

www.iouc.org

Lasse Jenssen
Leader of Center of Excellence – Oracle
Financial Services, EVRY



OUGNorway board member

Timeline

Cary Millsap

"Thinking Clearly About Performance"
(method-r.com)

The Theory

Some general thoughts about Performance

A Method

How to work with Performance

The Implementation

Oracle End-To-End Metrics

The Learnings

From use in Financial Services, EVRY

The Theory

Some general thoughts about Performance

We traditionally measure performance in terms of

response time and throughput

= Time / Task

= Tasks / Time

Looking at Response Time we might define

FAST as (R <= SLA)

v$sysstat, v$sesstat, v$segstat, v$session_event, v$session_wait, v$sql, etc.

Statspack & AWR

In the same time DBA's usually work with

System or Session Level Metrics

Active Session History (ASH)

Quest Foglight & Performance Analyzer

PaymentCreate

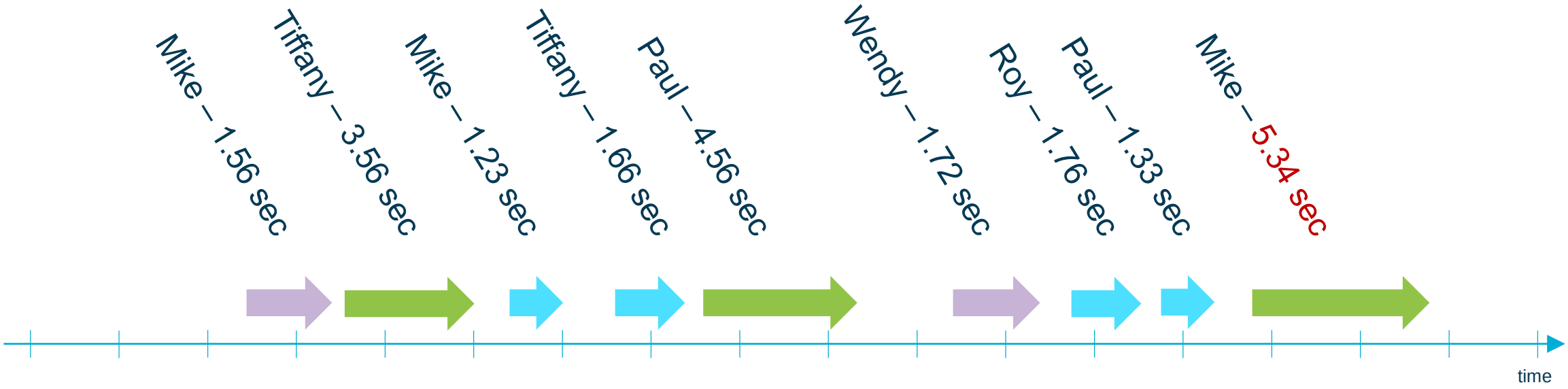
Fast (R <= 2 sec)

PaymentRead

Fast (R <= 2 sec)

PaymentHistoricRead

Fast (R <= 5 sec)



A session in a connection pool

Aggregated numbers hide important details

The reported values might or might not
be related to your performance problem

Some thoughts about the

performance analysis

We need a method that works all the time

You really need a plan

EVRY

We're having a performance issue in the … blab, blab, blab.

We think it could be the database. Could you please have a look?

Ok. Thanks.

What now?

I'm looking at my database monitoring screen right now.

Everything is green!

Traditionally our …

DBAs were monitoring the database system

OS resources were monitoring

resources on the database server …

and on the application server

Network resources were monitoring the network

Storage resources were monitoring the disks

The user does not care about our

Background Systems

We really needed to change the way we were

Measuring performance

Performance is NOT an attribute of a system.

Performance is an attribute of

each individual experience

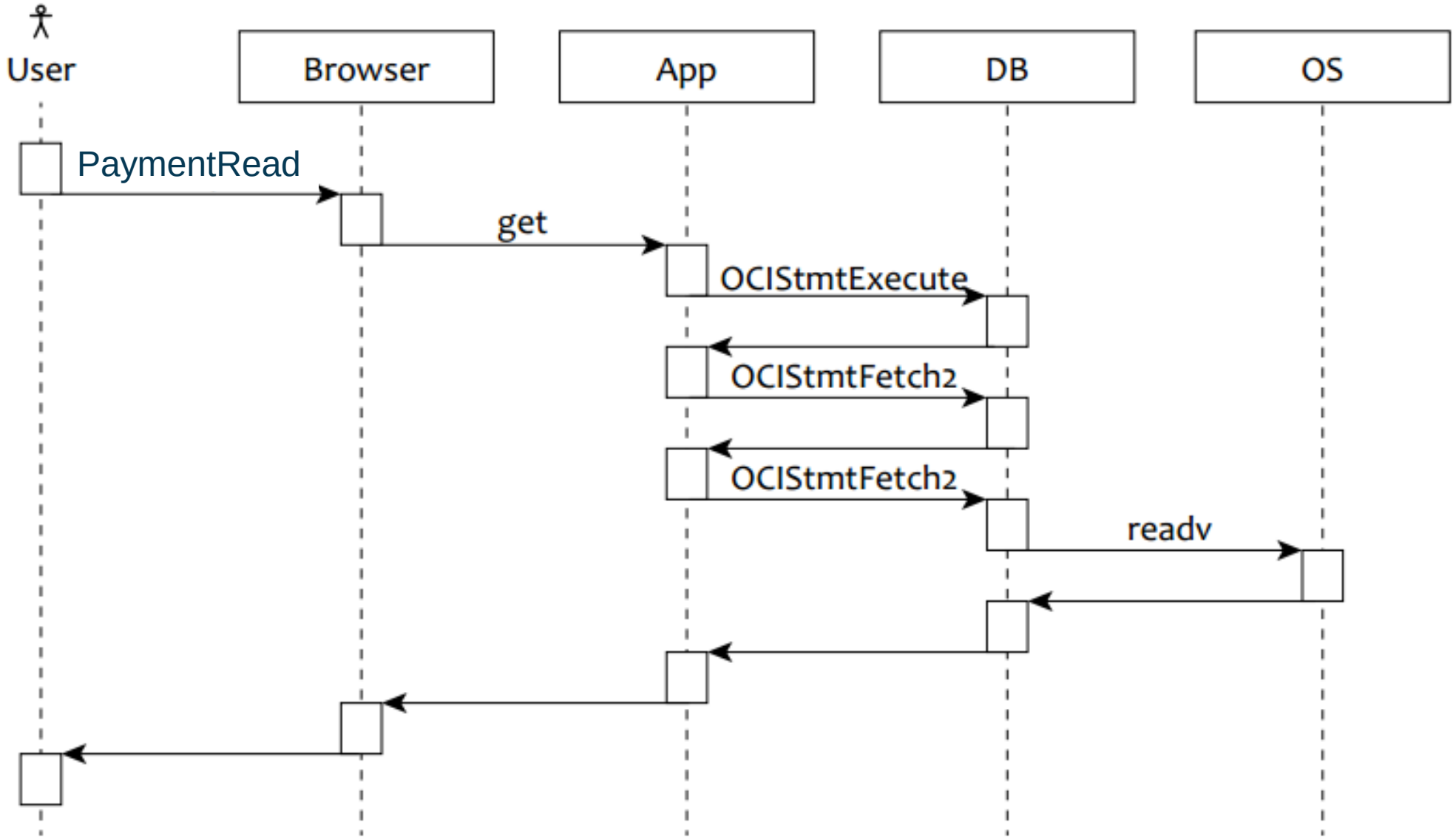
with a system

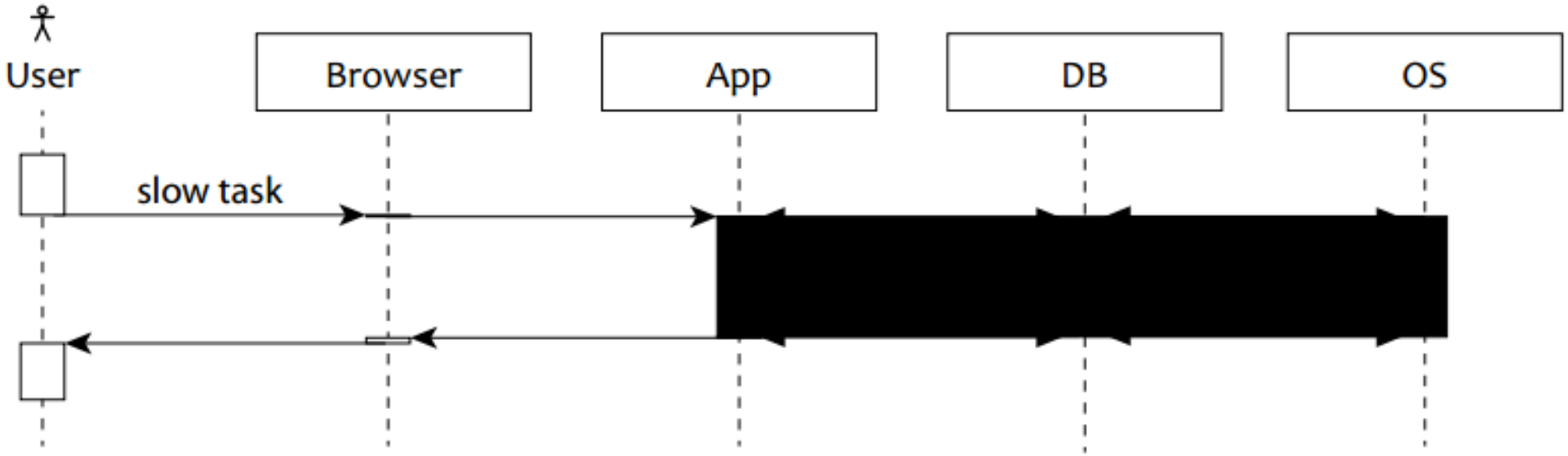
A Method

How should we work with Performance?

The “User Experience” consist of time spent along

The Code Path


User
Browser
App
DB
OS
PaymentRead
get
OCIStmtExecute
OCIStmtFetch2
OCIStmtFetch2
readv


User
Browser
App
DB
OS
slow task

“Performance is a feature”

We need

Instrumentation *telling us something about*

The Code Path in Oracle

Feature of every Oracle database since version 7

Available in both XE, SE & EE, and on ExaData

The closes thing to showing the code path in Oracle is

Oracle Extended SQL Trace

Logging what goes on in a session into a trace file

Showing us where time is actually spent

EVRY

```
…
=====================
PARSING IN CURSOR #4588810064 len=336 dep=1 uid=103 oct=3 lid=103 tim=12569729106521… sqlid='brm0gu1h11s02'
SELECT NVL(ACC.IBAN, … WHERE T.TRANSACTIONSID = :B1
END OF STMT
BINDS #4588810064:
 Bind#0
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  …
  value=46991
EXEC #4588810064:c=73,e=140,p=0,cr=0,cu=0,mis=0,r=0,dep=1,og=1,plh=239605389,tim=12569729106644
FETCH #4588810064:c=50,e=84,p=0,cr=12,cu=0,mis=0,r=1,dep=1,og=1,plh=239605389,tim=12569729106751
CLOSE #4588810064:c=1,e=2,dep=1,type=3,tim=12569729106787
=====================
BINDS #4575083120:
 Bind#0
  oacdty=122 mxl=4000(4000) mxlc=00 mal=00 scl=00 pre=00
  oacflg=00 fl2=206001 frm=00 csi=00 siz=4000 off=0
  value=Unhandled datatype (122) found in kxsbndinf
Bind#1
  oacdty=02 mxl=22(21) mxlc=00 mal=00 scl=00 pre=00
  oacflg=03 fl2=1206001 frm=00 csi=00 siz=24 off=0
  kxsbbbfp=110a64170  bln=22  avl=04  flg=05
  value=46991
EXEC #4575083120:c=55123,e=108613,p=0,cr=322,cu=0,mis=1,r=0,dep=1,og=1,plh=2003972165,tim=12569729228698
WAIT #4575083120: nam='db file sequential read' ela= 53137 file#=6 block#=528500 blocks=1 obj#=79460 tim=…
FETCH #45750831 20:c=350,e=53659,p=1,cr=27,cu=0,mis=0,r=1,dep=1,og=1,plh=2003972165,tim=12569729282431
…
```

Traditional way to turn on trace by ...

```
SQL> alter session set even    0046 tr    me context forever, level 12';

******* run all of your    cessing here *

SQL> alter session set    ents '10046    name    ntext off';


SQL> execute dbms_moni    .sessi    race_enable(    ID, &&SERIAL, true, true);

******* the session in    running *****

SQL> execute dbms_monitor.s    trace_d    e(&&SID, &&SERIAL);
```

PaymentCreate

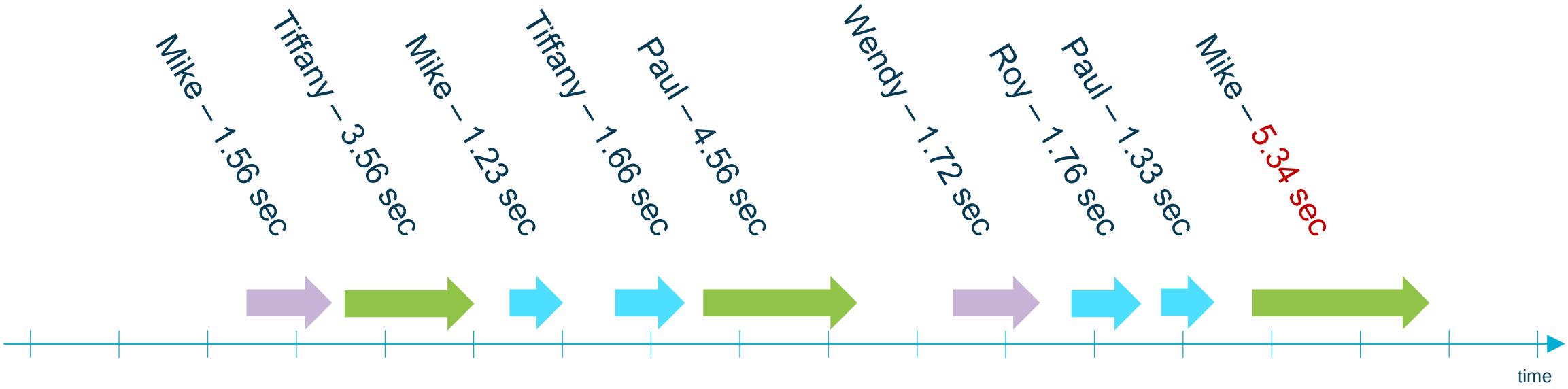
Fast (R <= 2 sec)

PaymentRead

Fast (R <= 2 sec)

PaymentHistoricRead

Fast (R <= 5 sec)



A session in a connection pool

1. Turn on tracing
2. Get the trace file
3. Analyze the trace

Trick #1: The answer to turning on trace (and a lot more) is

Oracle End-to-end Metrics

The specification

Oracle End-To-End Metrics

A whole lot more than just trace

A built-in framework from version 10g

Oracle End-To-End Metrics

Some tags “all” around in our Oracle instance

Each experience gets it’s own unique id.

Java: UUID.randomUUID().toString();

Trick #2: Introducing a

User Experience Id

067e6162-3b6f-4ae2-a171-2470b63dff00

PL/SQL: sys_guid()

.NET: Guid.NewGuid().toString();

MODULE=<Application Name>

Example: “PAY”

ACTION=<feature identifier>

Example: “PaymentHistoricRead”

Oracle Metric Tags

Tag	Database	JDBC
Module	64 bytes	48 bytes
Action	64 bytes	32 bytes
Client_identifier	64 bytes	64 bytes
ECID	64 bytes	64 bytes
Client_info	64 bytes	N/A

CLIENT_IDENTIFIER=<UUID>

Example: “067e6162-3b6f-4ae2-a171-2470b63dff00”

CLIENT_INFO=<Not used in EVRY>
ECID=<Not used in EVRY>

v$active_session_history

v$session

dba_hist_active_sess_history

v$sql_monitor

v$sql

Oracle Metric Tags

The MODULE, ACTION, CLIENT_IDENTIFIER, CLIENT_INFO, ECID

v$client_stats

v$serv_mod_act_stats

v$service_stats

And some more …

The MODULE, ACTION, CLIENT_IDENTIFIER, ECID & CLIENT_INFO

V$SESSION

```
SQL> select sid,serial#,s.service_name,s.module, s.action
     from v$session s where module in ('PWH','PAY');

       SID    SERIAL# SERVICE_NAME    MODULE     ACTION
---------- ---------- --------------- ---------- --------------------
        40      65001 pay_srv         PWH        DUEDATE_2251
        70      25965 pay_srv         PAY        PaymentRead
        99      16923 pay_srv         PAY        PaymentCreate
       459      46919 pay_srv         PAY        PaymentRead
       743      74592 pay_srv         PAY        PaymentHistoricRead

5 rows selected.
```

The MODULE, ACTION, CLIENT_IDENTIFIER

Trace file

```
*** 2014-09-24 12:25:11.332
*** SESSION ID:(56.4539) 2014-09-24 12:25:11.332
*** CLIENT ID:(067e6162-3b6f-4ae2-a171-2470b63dff00) 2014-09-24 12:25:11.332
*** SERVICE NAME:(pay_srv) 2014-09-24 12:25:11.332
*** MODULE NAME:(PAY) 2014-09-24 12:25:11.332
*** ACTION NAME:(PaymentHistoricRead) 2014-09-24 12:25:11.332
*** CONTAINER ID:(3) 2014-09-24 12:25:11.332


EXEC #140491268512760:c=0,e=55,p=0,cr=0,cu=0,mis=0,r=0,dep=0,og=0,…
WAIT #140491268512760: nam='SQL*Net message to client' ela= 1 driver id=…
WAIT #140491268512760: nam='SQL*Net message from client' ela= 137 driver …
=====================
…
```

The MODULE & ACTION

V$SQL

```
SQL> select sql_text from v$sql where module='PAY'
                                  and action='PaymentHistoricRead'

SQL_TEXT
----------------------------------------------------------------
SELECT TYPE_SETTLEMENT( SETTLEMENT.SETTLEMENTID, SETTLEMENT. …
SELECT TYPE_SETTLEMENT( SETTLEMENT.SETTLEMENTID, SETTLEMENT. …
BEGIN :1 := PWHDATA.PWH_READ_PKG.READ_HISTORICPAYMENT(:2 , :3 , :4 ); END;
SELECT TYPE_SETTLEMENTCHARGEINFO( NULL, NULL, AMOUNT, CURRENCY, …
SELECT TYPE_SETTLEMENTCHARGEINFO( NULL, NULL, AMOUNT, CURRENCY, …

5 rows selected.
```

The MODULE, ACTION, ECID

V$ACTIVE_SESSION_HISTORY

```
SQL> select sample_time, session_id, sql_id, module, action, event
     from v$active_session_history
     where ecid='067e6162-3b6f-4ae2-a171-2470b63dff00';

SAMPLE_TIME                    SID  SQL_ID        MODULE  ACTION       EVENT
------------------------------ ---- ------------- ------- ------------ ----------
24-SEP-14 02.27.20.269 PM        64 a9gr6bwxfadd3 PAY     PaymentRead  db_file..
24-SEP-14 02.27.19.259 PM        64 a9gr6bwxfadd3 PAY     PaymentRead  db_file..
24-SEP-14 02.27.18.259 PM        64 a9gr6bwxfadd3 PAY     PaymentRead
24-SEP-14 02.27.17.259 PM        64 a9gr6bwxfadd3 PAY     PaymentRead
```

How do we set the

End-To-End Metrics Tags?

SQL & PLSQL

The PLSQL implementation

```
exec dbms_application_info.set_module( module_name => 'PAY',
                                       action_name => 'PaymentHistoricRead');
exec dbms_session.set_identifier(client_id => '067e6162-3b6f-4ae2-…');

// The PaymentHistoricRead code

exec dbms_application_info.set_module( module_name => '',
                                       action_name => '');
exec dbms_session.set_identifier(client_id => '');
```

EVRY

Java ADF

The JDBC implementation

```
metrics = new String[OracleConnection.END_TO_END_STATE_INDEX_MAX];
metrics[OracleConnection.END_TO_END_MODULE_INDEX] = "PAY";
metrics[OracleConnection.END_TO_END_ACTION_INDEX] = "PaymentHistoricRead";
metrics[OracleConnection.END_TO_END_CLIENTID_INDEX] = "067e6162-3b6f-4ae2-a171-…";
((OracleConnection) connection).setEndToEndMetrics(metrics,(short)0);

// The PaymentHistoricRead code

metrics[OracleConnection.END_TO_END_MODULE_INDEX] = "";
metrics[OracleConnection.END_TO_END_ACTION_INDEX] = "";
metrics[OracleConnection.END_TO_END_CLIENTID_INDEX] = "";
((OracleConnection) connection).setEndToEndMetrics(metrics,(short)0);
```

EVRY

Oracle 12c & JDBC 4.1 Standard (ojdbc7)

The JDBC implementation

```
Properties p = new Properties();
p.put("OCSID.MODULE",          "PAY");
p.put("OCSID.ACTION",          "PaymentHistoricRead");
p.put("OCSID.CLIENTID",        "067e6162-3b6f-4ae2-a171-2470b63dff00");

connection.setClientInfo(p);

// The PaymentHistoricRead code

p.put("OCSID.MODULE",          " ");
p.put("OCSID.ACTION",          " ");
p.put("OCSID.CLIENTID",        " ");

connection.setClientInfo(p);
```

ODP.NET

The .NET implementation

```
conn.ModuleName = "PAY";
conn.ActionName = "PaymentHistoricRead";
conn.ClientId   = "067e6162-3b6f-4ae2-a171-2470b63dff00";

conn.ClientInfo = "<possible for additional information>";

// The ListPayment code

conn.ModuleName = "";
conn.ActionName = "";
conn.ClientId   = "";
conn.ClientInfo = "";
```

1. Turn on tracing
2. Get the trace file
3. Analyze the trace

Turning on trace ..

By using Metric tag values

```
exec dbms_monitor.serv_mod_act_trace_enable(service_name  => 'PAY_SRV',
                                            module_name   => 'PAY',
                                            action_name   => 'PaymentHistoricRead',
                                            waits         => true,
                                            binds         => true,
                                            plan_stat     => 'ALL_EXECUTIONS');
```

Warning!
BUT this works only if you code sets the metric tags in your application!!

```
SELECT * FROM dba_enabled_traces;
```

The “PaymentHistoricRead” service in our “PAY” application are experiencing response times over the SLA at 5 sec. Our logs tells us that 90% of this time is towards the database.

Excellent!

Ok. I see I have some sessions running with the module “PAY” and action tag “PaymentHistoricRead”. I’ll turn on trace on these features right away!

1. Turn on tracing
2. Get the trace file
3. Analyze the trace

Some facts …

The trace file is on the Oracle Database server, in the diagnostic_dest directory.

<diagnostic_dest>/diag/rdbms/<sid>/<instance_name>/trace

Your file is probably called ***dbname*_ora_*spid*_*id*.trc**
- where

dbname	is your *db_name* parameter value,
spid	is your session's *v$process.spid* value,
id	is your session's *tracefile_identifier* value.

Sessions with DOP = *k* can create 2*k* + 1 trace files.

```
select value from v$diag_info where name='Diag Trace'
```

Oracle Utility: trcsess

Used to gather related trace data into one file

```
trcsess [output=<output file name >]   [session=<session ID>]
        [clientid=<clientid>]          [service=<service name>]
        [action=<action name>]         [module=<module name>]
        <trace file names>

Example:
Oracle> trcsess output=PaymentHistoricRead_20141002.trc
                module=PAY action=PaymentHistoricRead  *

Oracle> trcsess output=PaymentHistoricRead_20141002.trc
                clientid=067e6162-3b6f-4ae2-a171-2470b63dff00 *
```

TKPROF

TVD$XTAT

1. Turn on tracing
2. Get the trace file
3. Analyze the trace

MrProfiler
MrTools
(http://method-r.com)

Reading RAW trace file

Profile Report: pay_1_ora_16974264.html

METHOD R™ Profiler

License id 1000079 registered to lasse.jenssen@evry.com / EVRY AS.

Profiler version 5.2.8.11 built on Wed Jan 30 10:54:18 CST 2013.

Profiler XML created 2014-09-19T14:25:11.449832; kernel duration 0.084 seconds (1.31 MB/sec).

Total measured task response time:	$R = t_1 - t_0$ = 0.277924 seconds ≈ **0.278 seconds**
Task begin time:	t_0 = 1:34:47.015945 p.m. 19 September 2014 Friday
Task end time:	t_1 = 1:34:47.293869 p.m. 19 September 2014 Friday
Oracle release:	11.2.0.3.0
Oracle session id:	288.58427
Transactions:	1 commit, 0 rollback
Workload:	777 database buffer cache accesses
Errors and warnings:	0 parse errors, 0 other errors, 0 Profiler kernel warnings
Service/module/action:	pay_srv / PAY / PaymentHistoricRead
Client identifier:	067e6162-3b6f-4ae2-a171-2470b63dff00
Viewing characteristics:	screen | projection | print

Profile by Subroutine

e following table shows the decomposition of total task response time by internal subroutine.

	Subroutine	Duration		Cumulative duration		Call	Duration per call (seconds)			
		seconds	% *R*	seconds	% *R*	count	mean	min	skew	max
1.	**db file sequential read**	**0.129**	**46.3%**	0.129	46.3%	3	**0.042887**	0.021816		0.053708
2.	**unaccounted-for within dbcalls**	**0.065**	**23.5%**	0.194	69.8%	**330**	0.000198	-0.000109		0.032567
3.	**CPU service, EXEC calls**	**0.059**	**21.3%**	0.253	91.1%	**89**	0.000665	0.000000		0.034788
4.	CPU service, PARSE calls	0.008	2.9%	0.261	94.1%	16	0.000506	0.000226		0.000961
5.	SQL*Net message from client	0.007	2.7%	0.269	96.7%	6	0.001229	0.000000		0.001979
6.	CPU service, FETCH calls	0.004	1.3%	0.272	98.0%	182	0.000020	0.000002		0.000360
7.	unaccounted-for between dbcalls	0.003	1.1%	0.276	99.1%	15	0.000207	0.000018		0.001522
8.	row cache lock	0.002	0.6%	0.277	99.7%	6	0.000279	0.000005		0.000973
9.	rdbms ipc reply	0.000	0.2%	0.278	99.9%	4	0.000112	0.000049		0.000274
.0.	CPU service, unreported call(s)	0.000	0.0%	0.278	100.0%	1	0.000125	0.000125		0.000125
.1.	CPU service, CLOSE calls	0.000	0.0%	0.278	100.0%	86	0.000001	0.000000		0.000036
.2.	SQL*Net message to client	0.000	0.0%	0.278	100.0%	6	0.000002	0.000001		0.000003
.3.	Total	0.278	100.0%							

The Learnings

From the use in Financial Services, EVRY

Some thoughts about

Java Developers & ORM APIs

Most Java developers I know are very clever people, but tend to trust

ORM APIs like *Hibernate*

when it comes to writing code towards the database

EVRY

The way EVRY use the

Oracle Metric Tags

- A *Non-Functional requirement* for applications accessing our Oracle Databases

- Development
- Testing
- Maintenance
- Troubleshooting
- Focus towards “User Experience” – not systems

Summary

”Performance is a feature.”
“Oracle End-To-End Metrics is a feature.”

Thanks to …

Cary Millsap & Method-R
You – for spending 1 hour of your precious time

lasse.jenssen@evry.com
@lasjen

http://www.jcon.no/oracle