

Hotsos Symposium 2006

End to End Performance Diagnosis in Oracle

Tanel Põder

<http://www.tanelpoder.com>

About Tanel Põder:

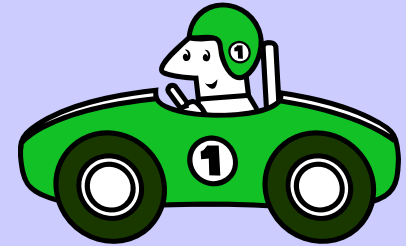
- Occupation: Consultant, researcher
- Area of expertise: *End-to-end* performance, scalability, availability
- Oracle experience: 8 years (+1 yr. sqlplus exp)
- Certification: OCM (2002), OCP (1999)
- Professional affiliations: OakTable Network

About the presentation:

- Is actually going to be useful to you!
- 70% fundamentals, 30% techniques
- Proposes a way for end to end performance diagnosis
- A way to prove that *THIS IS NOT A DATABASE PROBLEM!!!™* for 3rd party apps w. no source & support

Measuring from inside a car

- Driver concludes the mileage from counting the number of wheel rotations
- Driver concludes the mileage from counting the RPM multiplying it with the gear factor
- BCHR – Blindly Counting wHeel Rotations



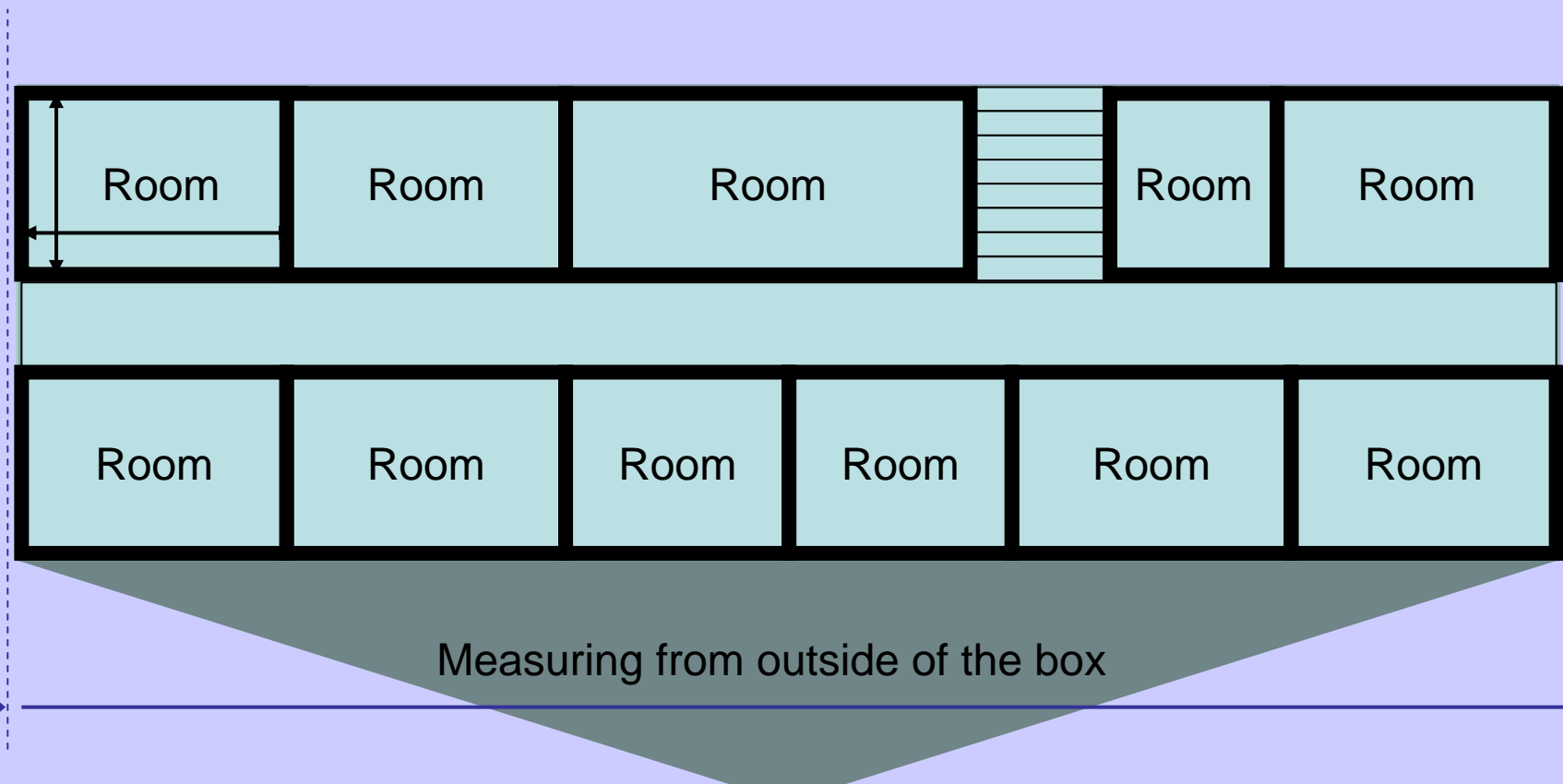
Factors affecting car drivers calculations

- Car tires are different sized than during initial tests
- Driver loses attention during critical periods, misses important RPM data
- Mileage meter is just an automated method of blindly counting wheel rotations

You can't measure size of the box being inside the box

Let say that one morning you wake up in a room...

- ..where you've never been before (details unimportant)
- And you feel an urge to find out how big is the whole house

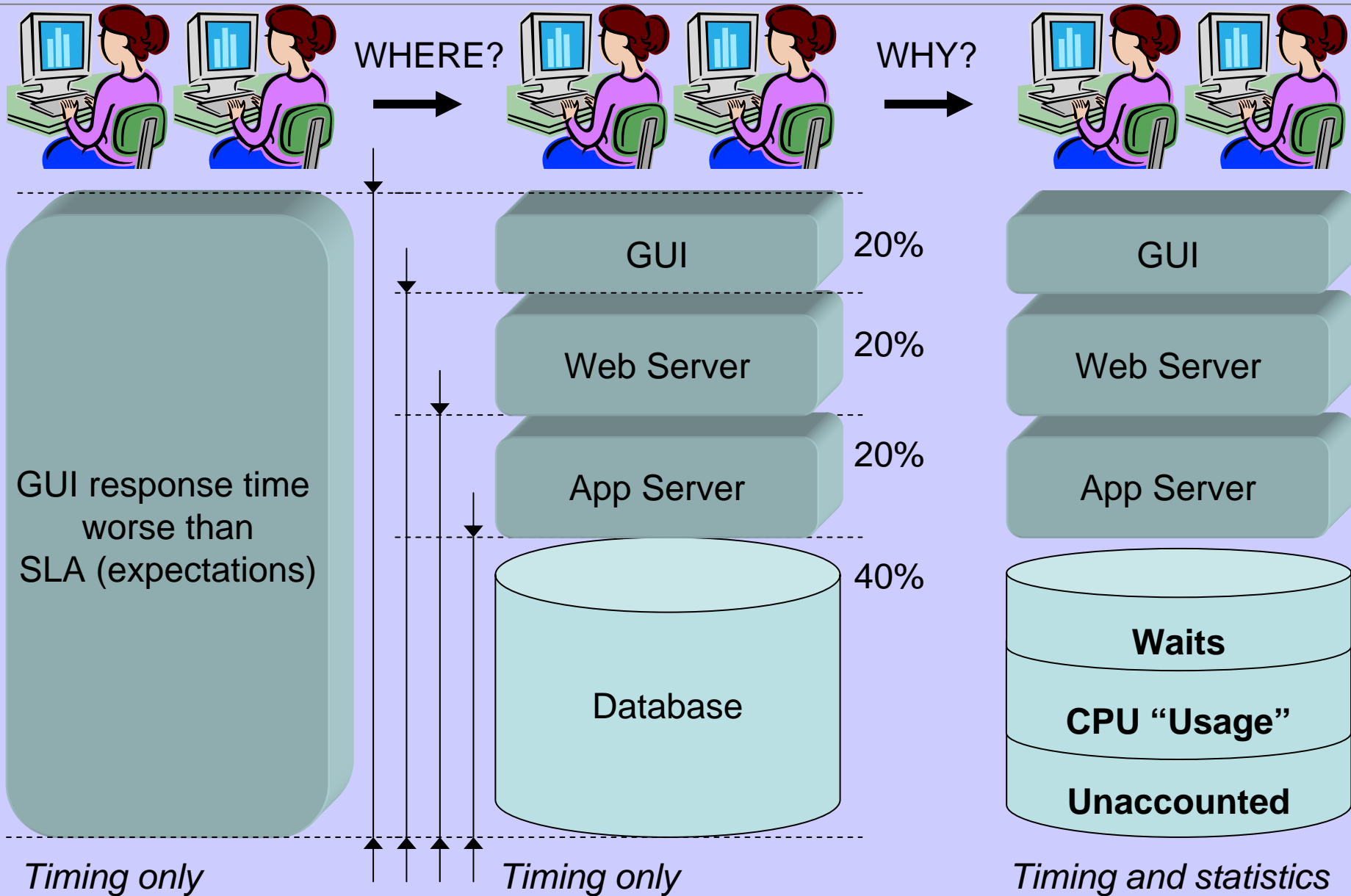


Instrumentation strategy (in my definition) consists of

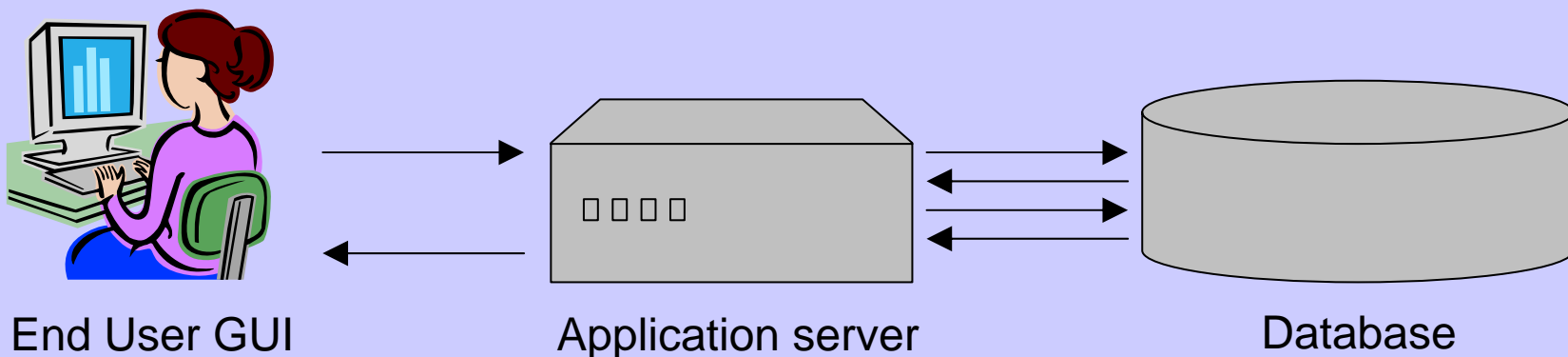
- Measuring strategy (development stage)
 - Where to install instrumentation points
 - When/how to enable instrumentation functions
- Data gathering strategy (deployment stage)
 - When and where to keep instrumentation data
 - Instrumentation data mapping and dependency association
- Analysis strategy (production, testing stage)
 - Reactive - process for drilldown/root cause analysis
 - Proactive – thresholds, exception reporting and data mining

! Instrumentation without *analysis strategy* is like a
• backup schedule without recovery strategy.

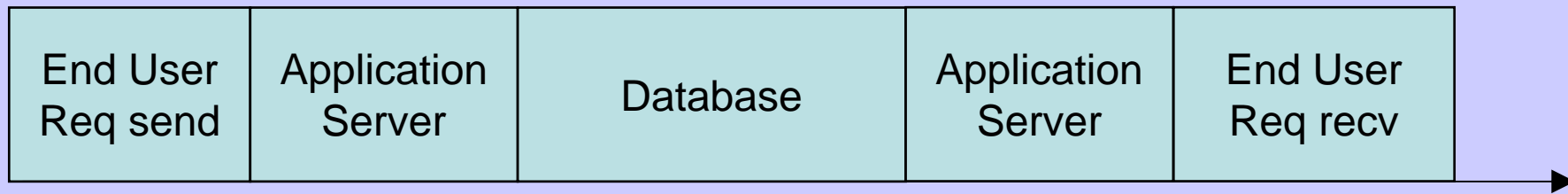
Two-step cyclic approach for drilldown to root cause



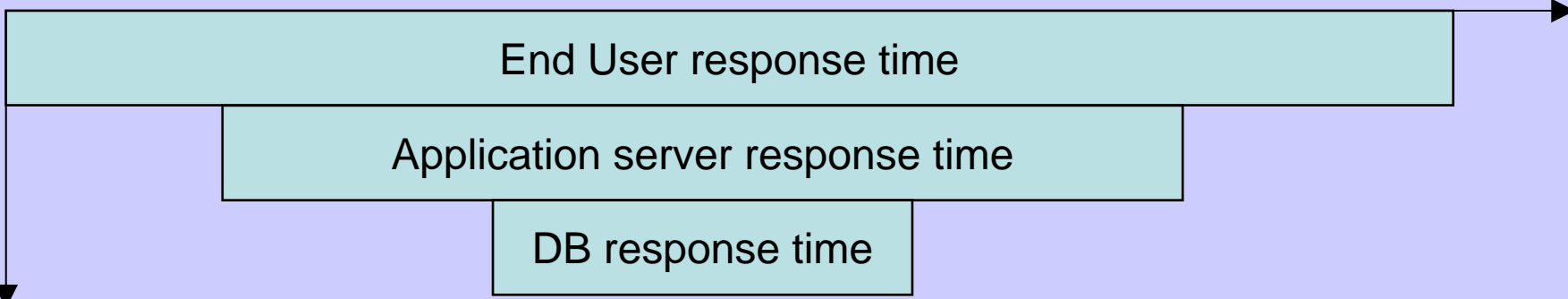
Measuring multitier and distributed information systems



Process flow in time



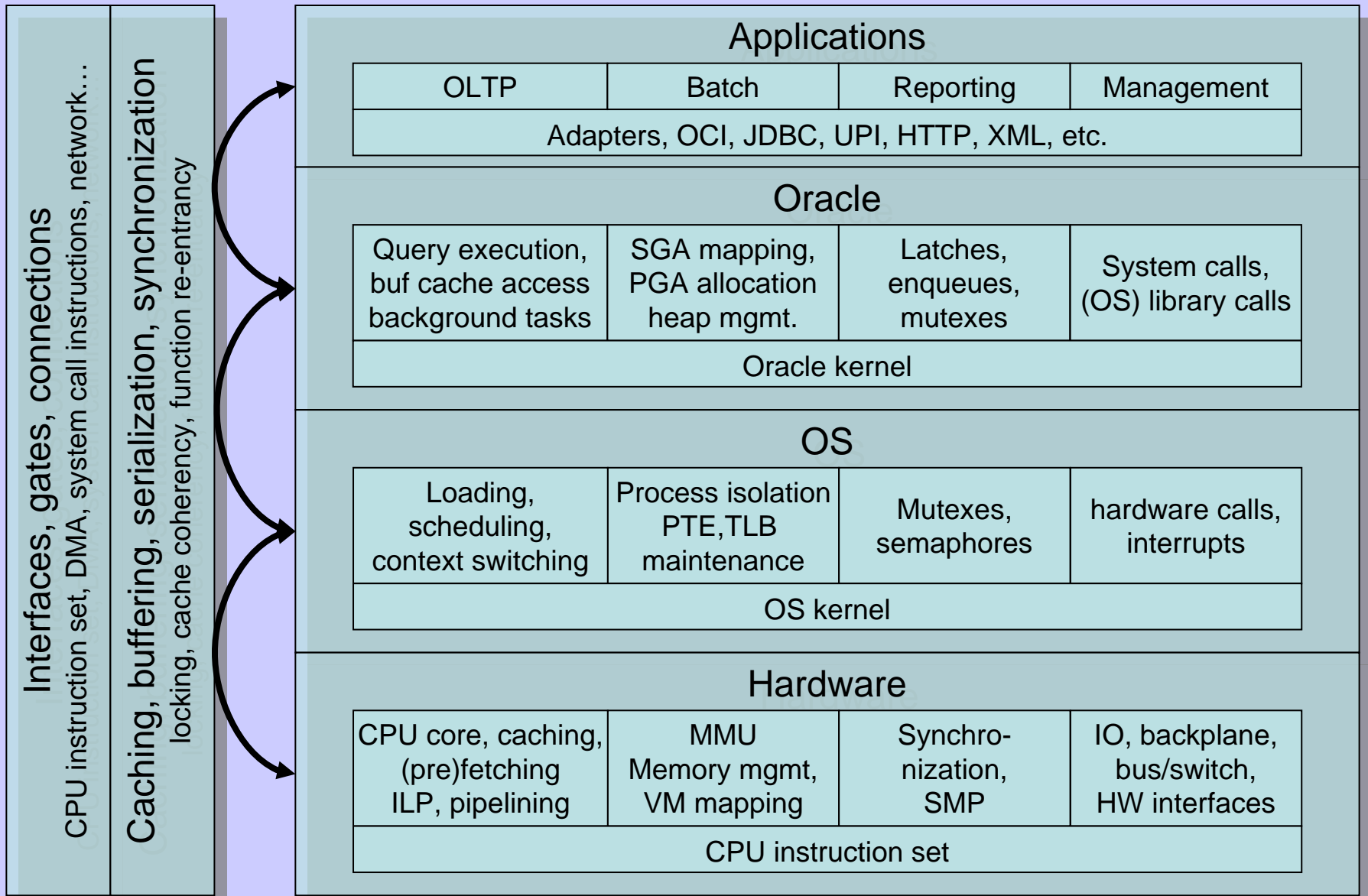
Response time



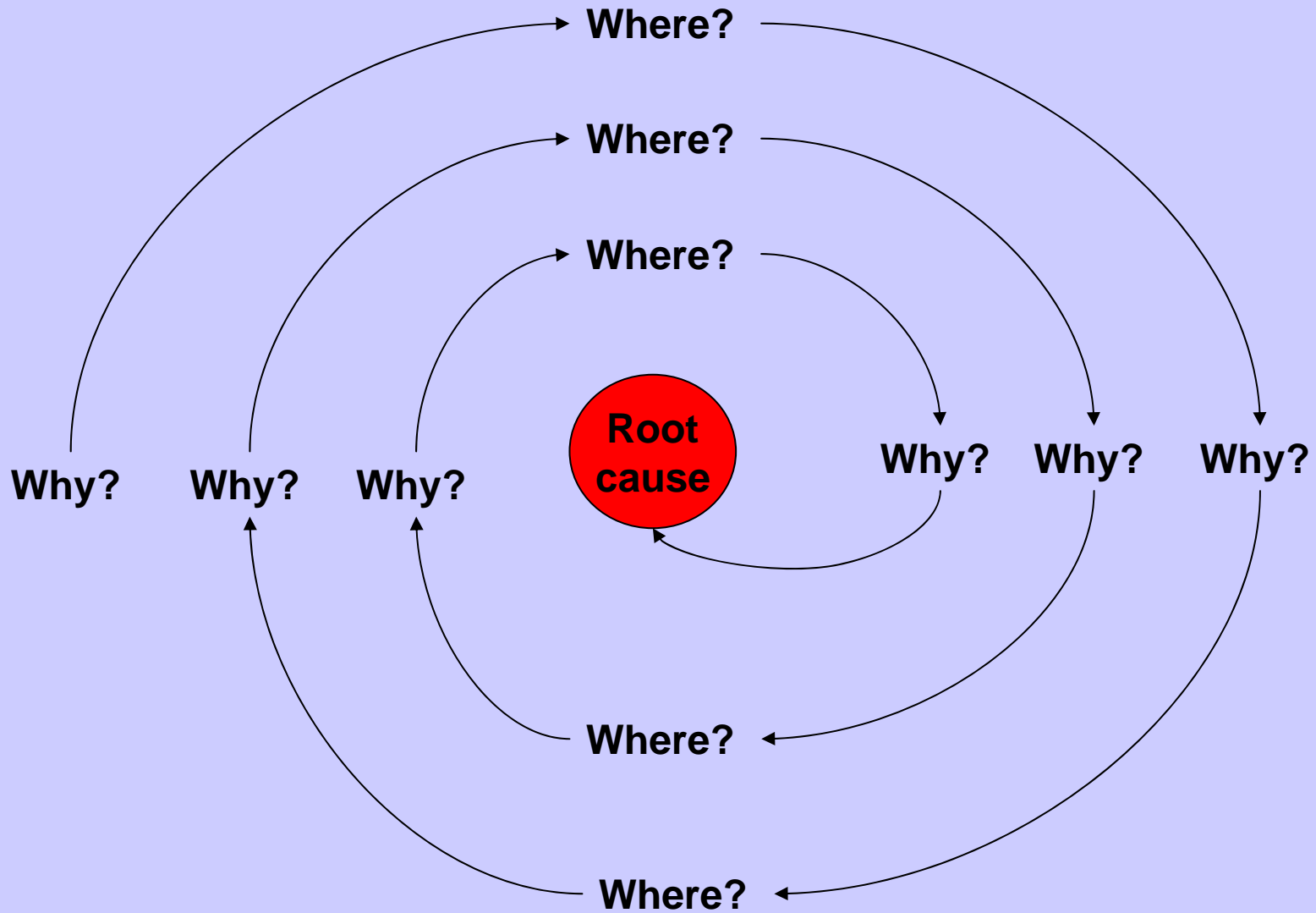
Modern information systems consist of many different components

- Client GUI, browser
- Internet, various ISPs, variable workloads, complex routing
- Internet-intranet gateway, firewalls, filters, internal routing
- Web servers, load balancers
- Application servers, queueing systems
- Databases (real databases and toy databases ;)
- Operating systems
- Hardware, servers
- Communication infrastructure
 - Transport networks, TCP/IP, Ethernet
 - Storage networks
 - Enterprise messaging

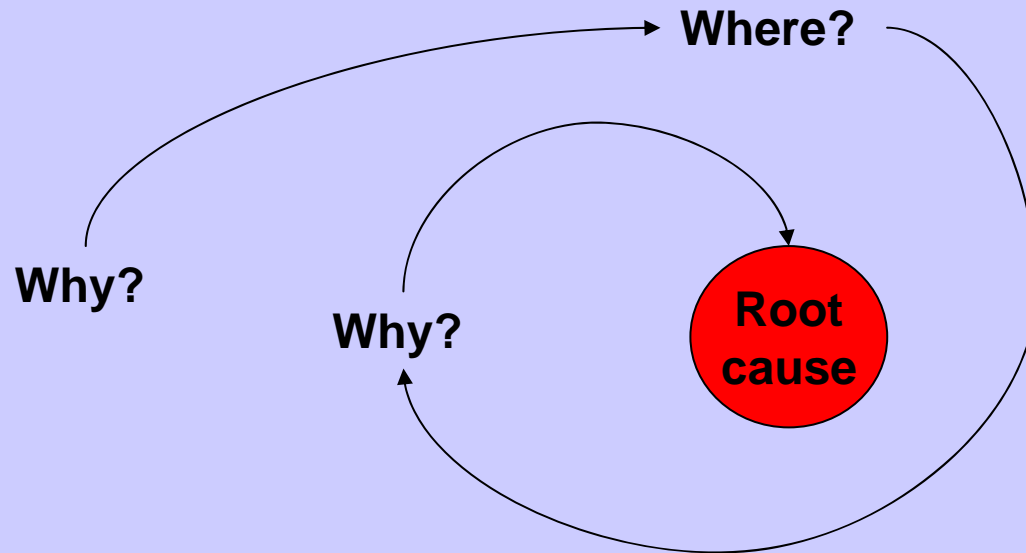
Path to understanding Oracle database performance



Only two questions in iterative performance diagnosis



Instrumentation for fast pinpointing



- The more instrumentation, the faster performance diagnosis could be – *potentially*
- More instrumentation, more performance overhead - *likely*
- Does few % of instrumentation overhead help to “reclaim” enough performance fast enough?
- Faster development (tuning), testing, capacity planning

Shortcomings of Oracle Wait Interface

Measures database performance from *inside the box*

- It isn't designed to know what happens outside the db

Measurement errors

- SQL*Net message from client

Level of detail

- Isn't always granular enough

All events are measured at the same level

- In order to get the database response time – lots of parsing and calculation has to be done

We need to continuously look at the bigger picture:

- Blind Cache Hit Ratios and such are luckily history
- We are concentrating on time based database tuning
- We are concentrating on session level timing instead of system level aggregated stats

However:

- Users don't care about database/session response time – they care about their *user interface response time*
- *End to end* response time can involve tens of components

And the number of components grows:

- Database isn't the usual suspect anymore
 - App servers, WAN access, ultrashort response times
- Timing only a single subcomponent of the whole system is not enough

Always start from end user response time

- Database response time reported in 10046 trace went from 50ms to 100ms!
- That's a 100% increase!
- But might be only 1% increase to the end to end response time
- Or might be only 10% increase in the real database response time – as clients see it

Always measure from outside the box

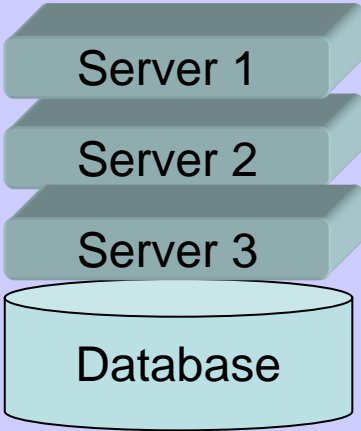
- You can combine this with in-the-box measurement to figure out delta – time spent invoking the component measured

Analyzing performance data

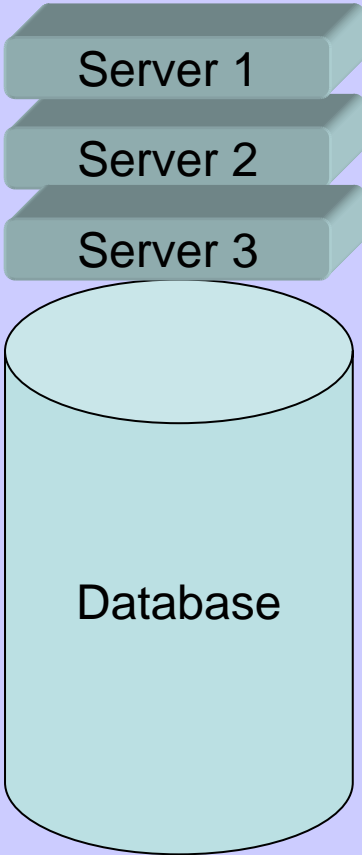
Response time



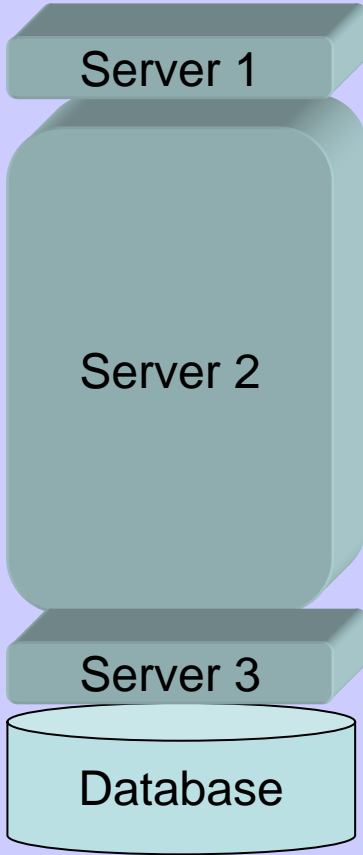
Baseline performance



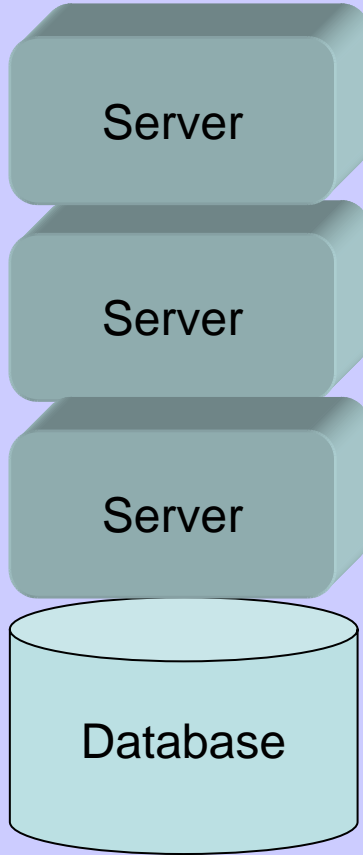
DB is obvious bottleneck



Server 2 is obvious bottleneck



Everybody takes more time



When timing method is exhausted...

- “Where” question is answered
- Even if the answer is “everywhere”

Continue with statistics

- Maybe number of concurrent users has increased
- Maybe amount of data transferred has increased

Baseline

- “Human” trial and error might get us there fast
- Cold following of methodology *will* get us there!
- Investing into a instrumentation strategy might be THE investment for your users and IT department

Stateless protocols

Connection pooling

Queuing systems

Distributed computing

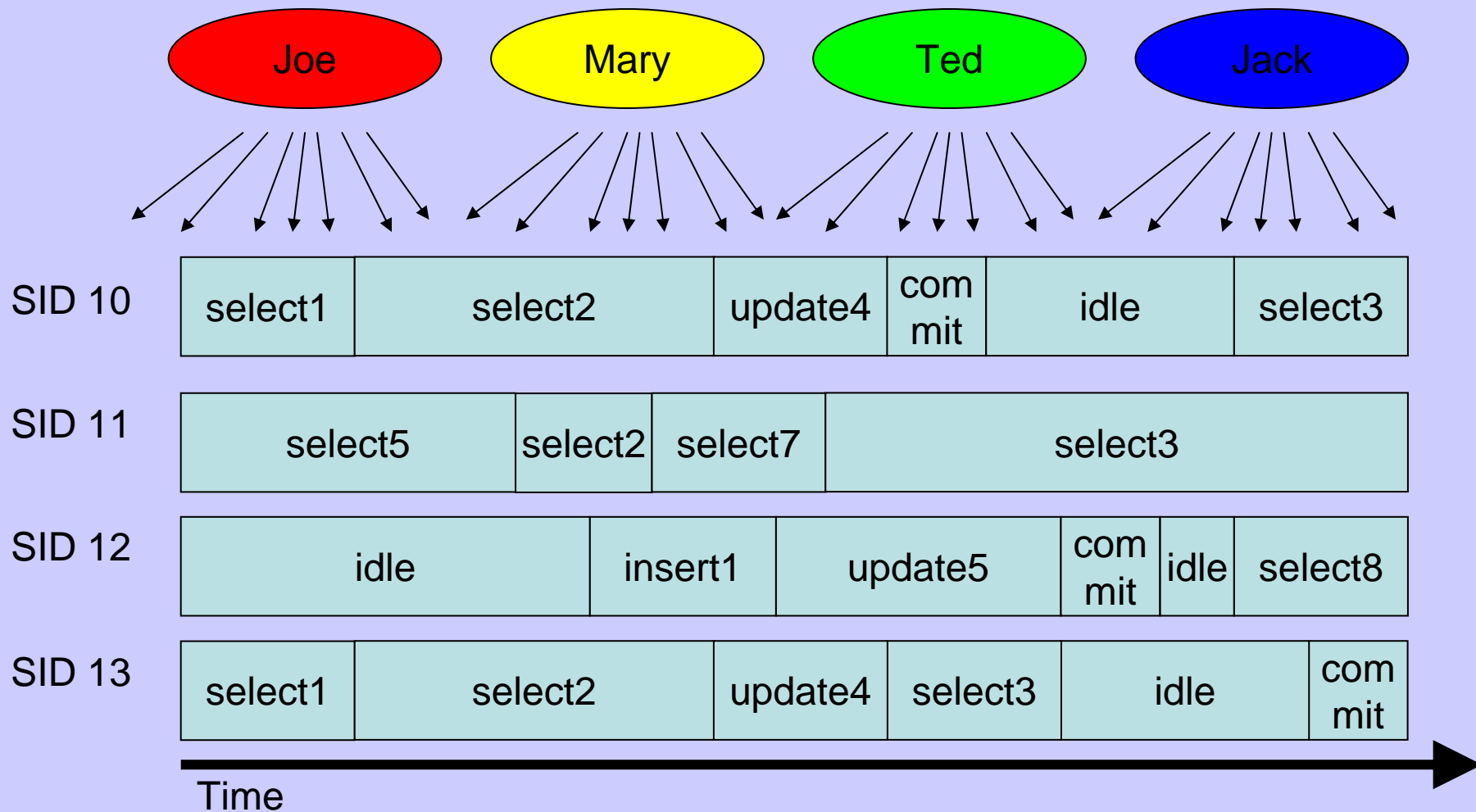
The usual dimensions for perf. instrumentation

- Time – when happened, how long duration
- Database session – which database session worked

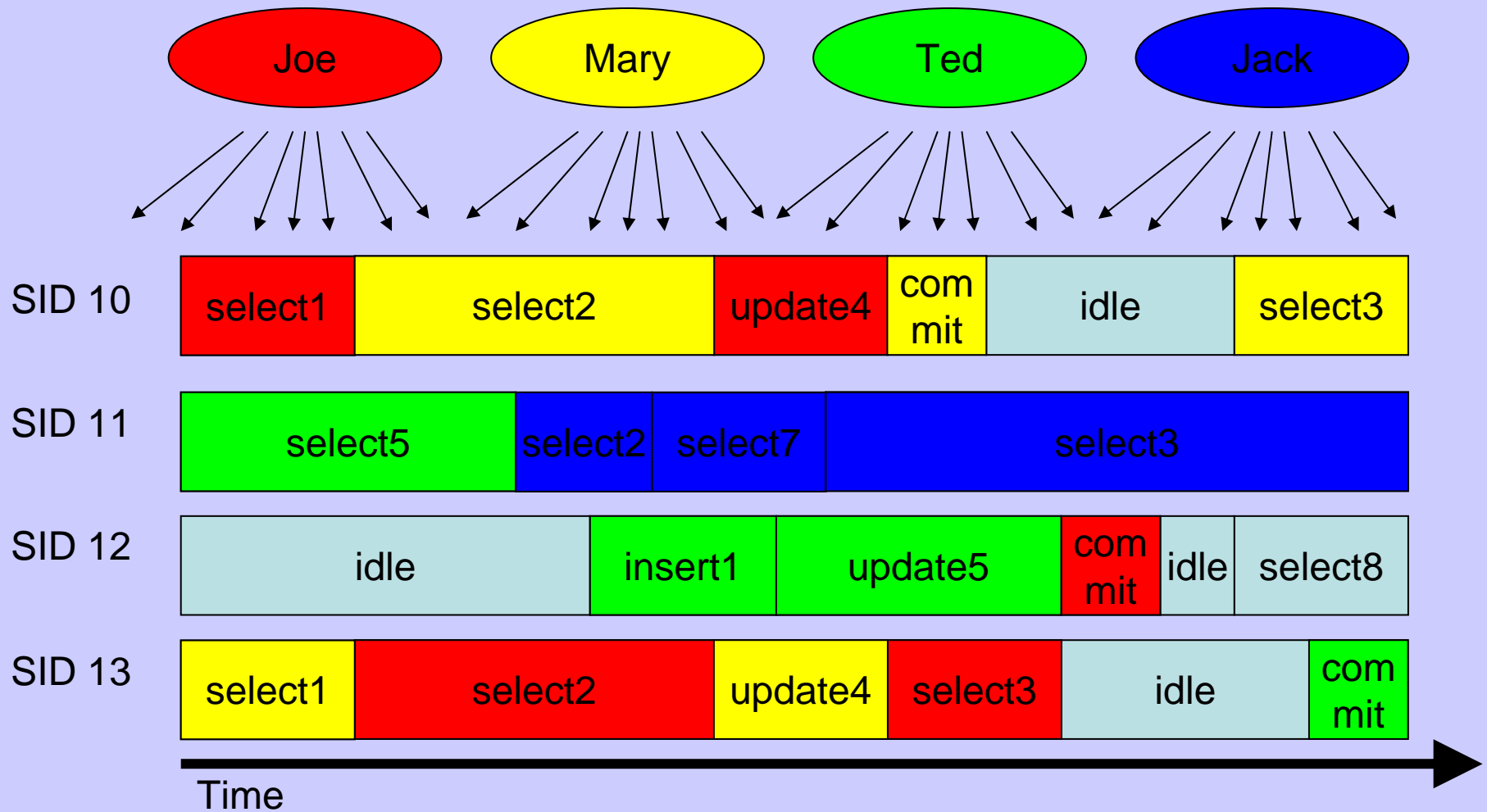
Need to have new dimensions for performance analysis

- Tagging

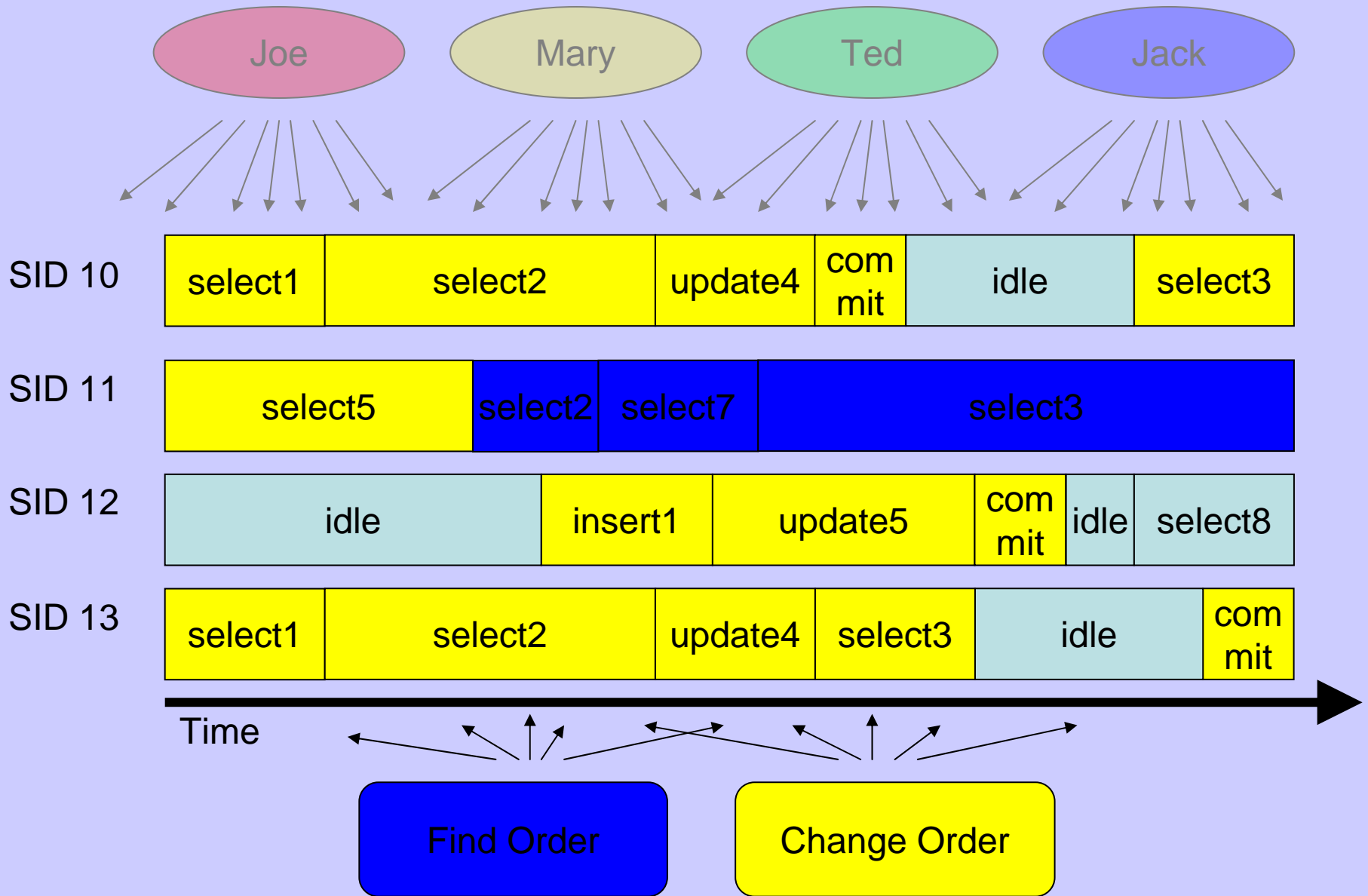
The need to fine grained instrumentation



End user oriented instrumentation



End user oriented instrumentation



Very easy in Oracle

- Fine grained instrumentation by:
 - CLIENT_ID (app. session ID, end user ID, request ID, etc..)
 - SERVICE
 - MODULE
 - ACTION

Setting tags

- DBMS_APPLICATION_INFO
 - SET_MODULE, SET_ACTION
- DBMS_SESSION.SET_IDENTIFIER
- OCIAttrSet() – piggybacking MODULE and ACTION with OCI call to save roundtrips
- Service based architecture, DBMS_SERVICE

Enabling selective tracing

- `DBMS_MONITOR.CLIENT_ID_TRACE_ENABLE()`
- `DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE()`
- Query `DBA_ENABLED_TRACES`

Enabling selective statistics gathering

- `DBMS_MONITOR.CLIENT_ID_STAT_ENABLE()`
- `DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE()`
- Query `DBA_ENABLED_AGGREGATIONS`

Query statistics from

- `V$CLIENT_STATS`
- `V$SERV_MOD_ACT_STATS`

Module and Action info is stored in parent cursor

- May get misleading data when using V\$SQL based views
- A shared child cursor will be using first parsing user's tags
- Even if the child cursor is accessing different objects

Solution

- Differentiate SQL statements by adding a comment
- `select /* MOD=ERP */ col1,col2 from tab where id = :x;`
- `select /* MOD=CRM */ col1,col2 from tab where id = :x;`

Transparent

- Instrumenting third party applications!

Aspect oriented in a way

- Any OCI Call can be traced, regardless of application

Helps greatly proving which side of client application the time goes

Demo location:

- Install Oracle demos (from rdbms companion cd on 10g)
- `cd $ORACLE_HOME/rdbms/demo`
- `ociucb.mk, cdemoucbl.c`
- `export ORA_OCI_UCBPKG=ociucb`
- Run the application

Make sure you understand where the instrumentation points are installed

- sqlplus “set timing on” issue
- strace/truss can help understanding – as `gettimeofday()` calls are reported there

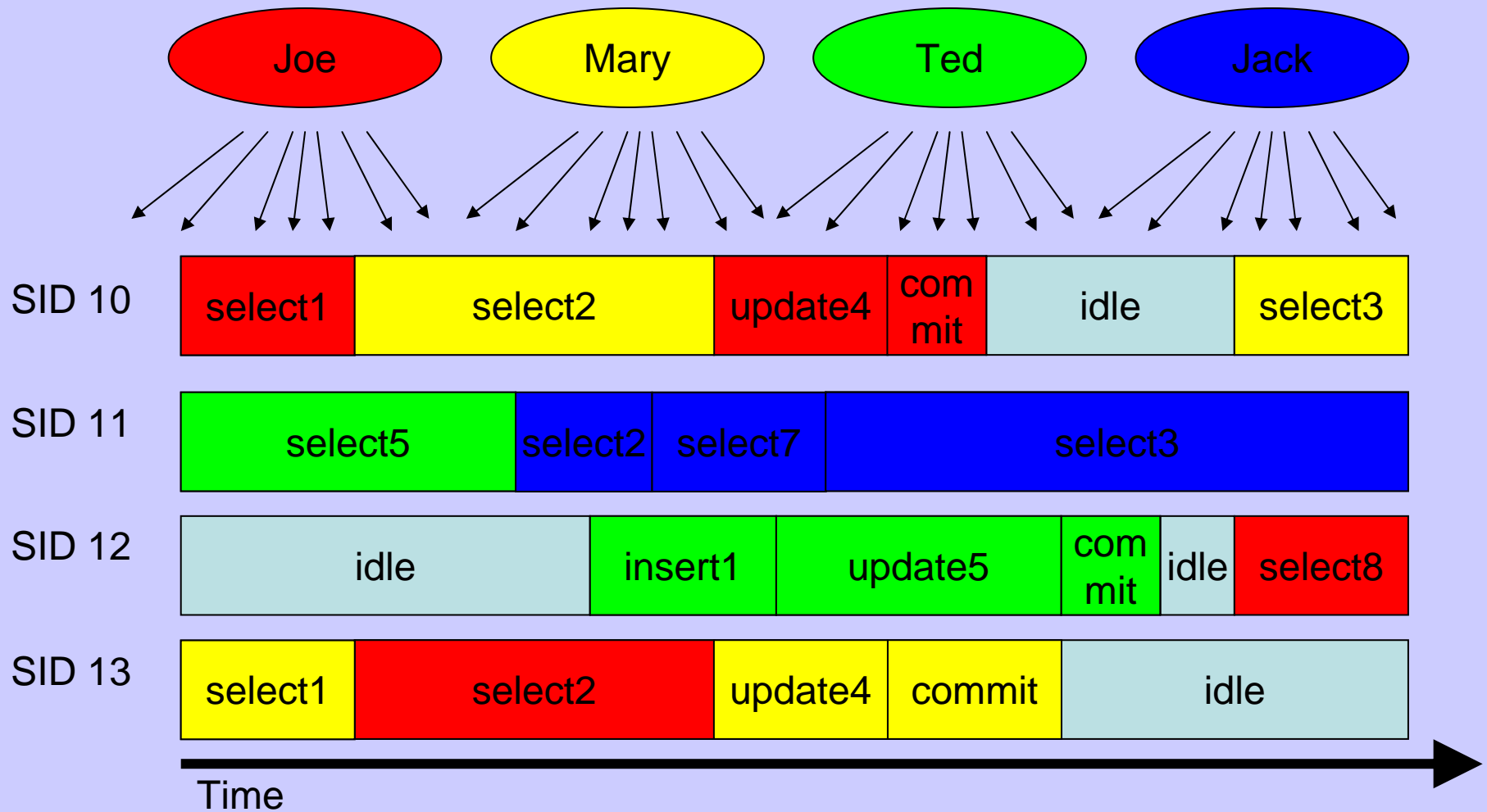
Measurement errors for CPU usage

- Should average out for longer operations
- But not for short operations

Inside-the-box measurement

- Doesn't see the whole picture
- Making assumptions & generalizations might be dangerous
 - SQL*Net message from client example

Importance of visualization



Importance of visualization

Which one is better to understand?

Session ID				
SID 10	Joe, select1 15%	Mary, select2 25%	Joe, update4 15%	Mary, commit 5%
SID 11	Ted, select5 30%	Jack, select2 12%	Jack, select7 15%	Jack, select3 40%
SID 12	idle, 30%	Ted, insert1 15%	Ted, update5 20%	Joe, commit 4%
SID 13	Mary, select1 15%	Joe, select2 30%	Mary, update4 15%	Joe, select3 13%

Real life systems are way more complex than current example

Summary

One hour just isn't enough for even scratching the surface of the problems what we have in the future

- Instrumentation is a *strategy*, not technique
- Keep looking at the big picture
- Forget about the aggregated performance data

Practical efficient end-to-end performance diagnosis is possible!

- Even for 3rd party applications using few creative solutions

DBAs will still be needed!!!

- Not for defragmenting tablespaces anymore though ;-)

Hotsos Symposium 2006

Thank You!

Tanel Põder

<http://www.tanelpoder.com>

tanel@tanelpoder.com