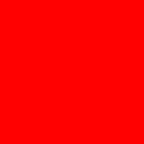




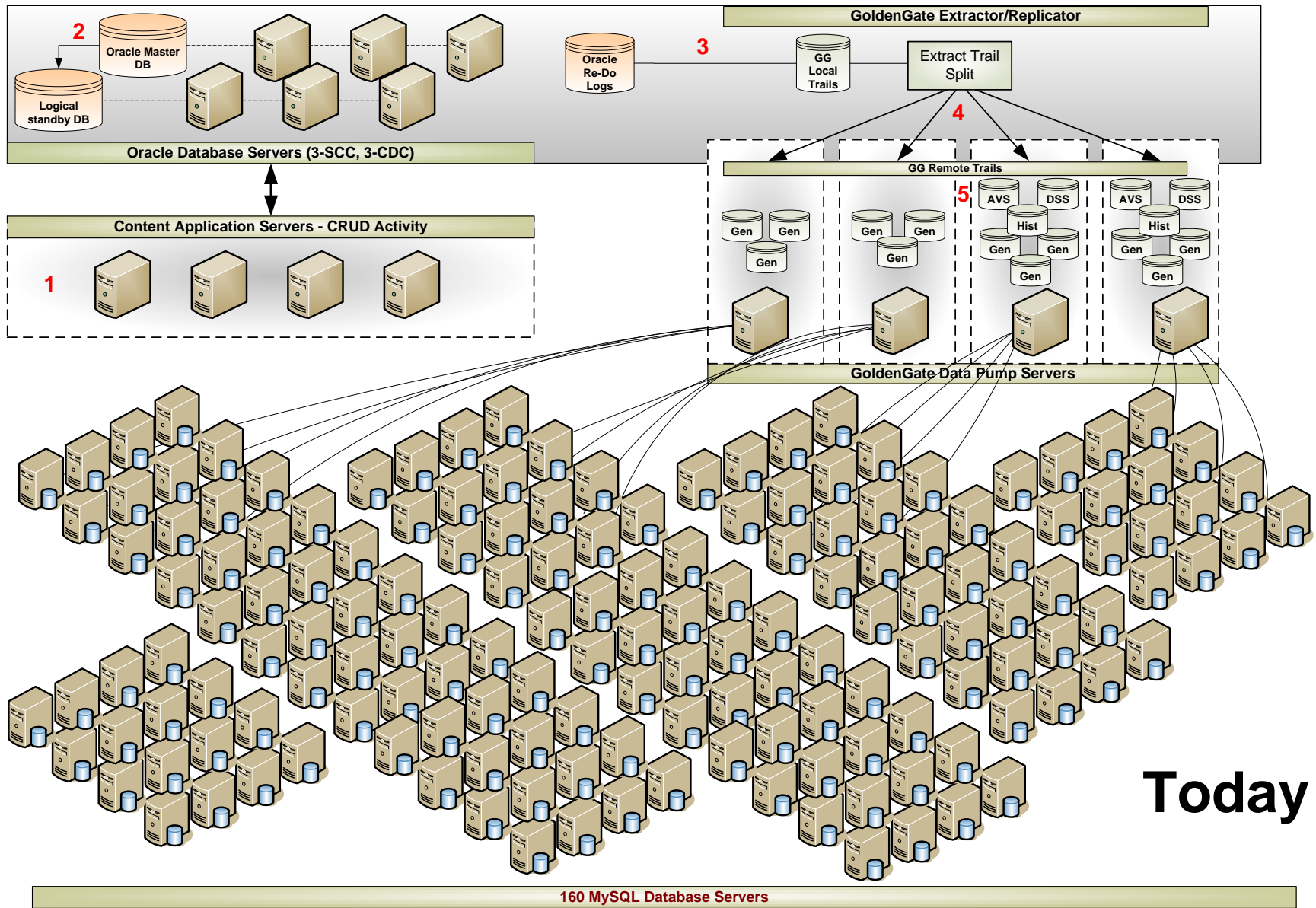
ORACLE
OPEN
WORLD

Your. Open. World.

**Scale out with TCO in check:
11g RAC vs MySql**

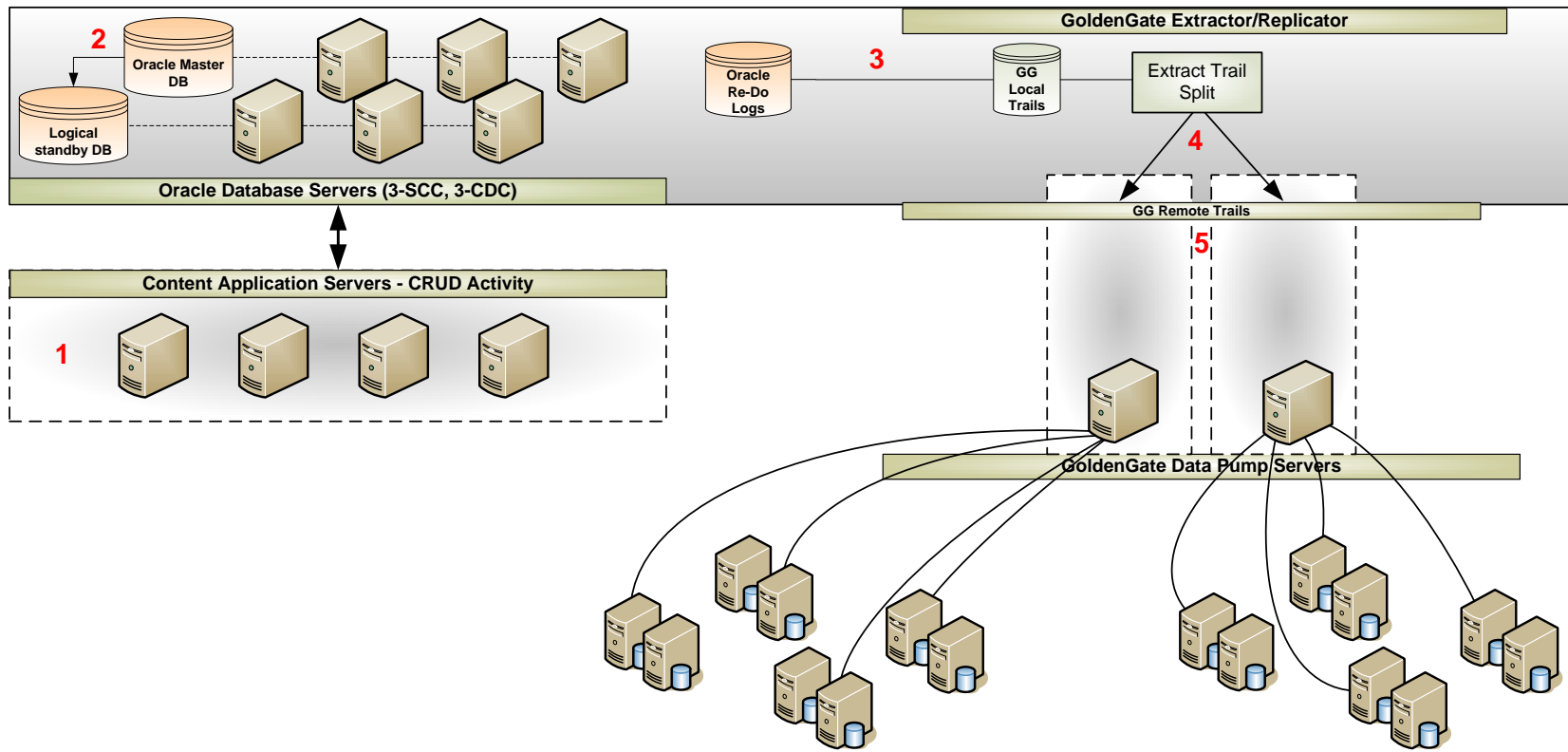


The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.



Today

160 MySQL Database Servers



Tomorrow

8 Oracle 2 Node RAC Database Clusters - Blades

Objective

- Improve:
 - Capacity & Scalability
 - Operational Support
 - Performance Monitoring & Performance Management Features
- Reduce:
 - Hardware
 - Overall Complexity / Diversity
 - Cost
- Provide:
 - Effective growth through transparency
 - Real viable End-to-End Recovery

Success

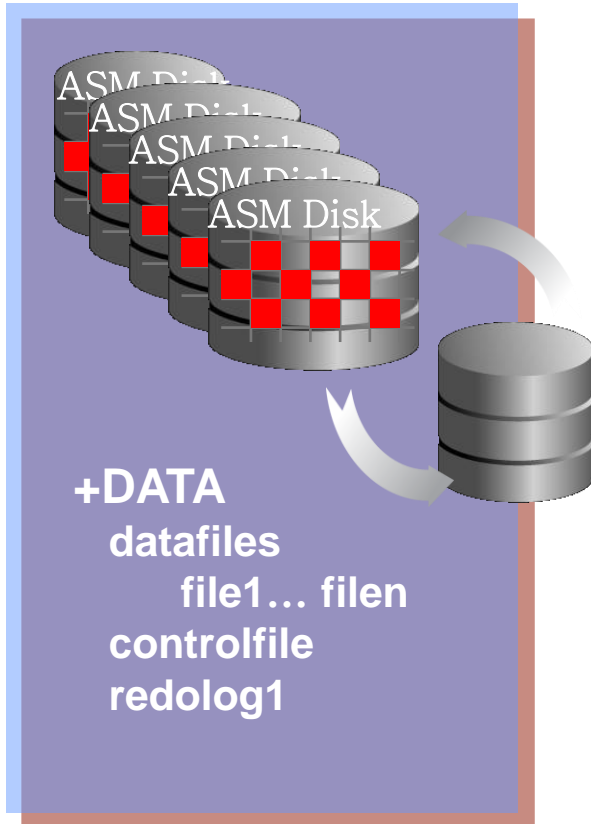
- Reduction in servers
- Reduction in processors
- Application to database server ratio improvements
- Simplified management
- Standard tooling
- Services approach
- Recovery time improvements

Memory & I/O Optimization



Memory & I/O Optimization

ASM Disk Group



An integrated **storage manager** for
Oracle Database files

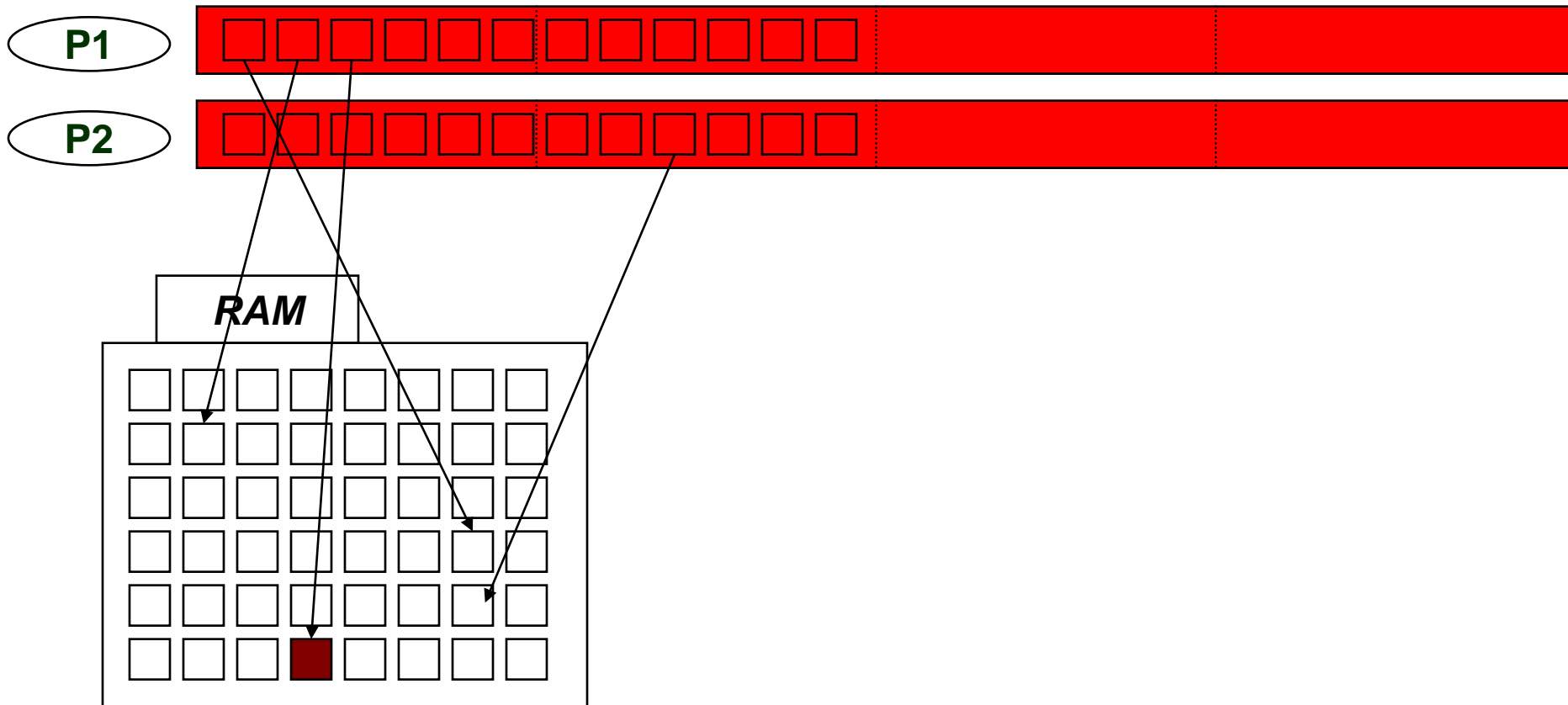
- High performance file system
- Integrated volume manager
- Fine or Coarse striping
- Flexible mirror protection
- Automatic data re-balancing
- Auto bad block detection & correction
- Highly scalable VLDB support
- Cluster support
- SQL, EM and ASMCMD management

Memory & I/O Optimization

- Automatic Storage management
 - Uses less buffers than file systems.
 - Some amount of memory used in shared pool for mapping
 - Using Direct I/O with ASM to by pass Oracle buffers for large scans and load operations.
 - Easy to manage and eliminates guess work

Memory & I/O Optimization

Process PageTable size can grow with no Hugepages



Memory & I/O Optimization

Calculate ProcessTable size with no HugePage.

1 OS block = 4K

For 42 GB of SGA we will need

45097156608 / 4092 = 11010048 memory map entries

Linux uses 4 to 8 bytes of process memory to map one OS block

Max size process pagetable can grow to access all buffers in SGA is **4 * 11010048 = 44MB.**

Memory & I/O Optimization

Calculate Process size with HugePages.

8 bytes can point to 2Mb of shared memory.

For 42 GB of SGA,

we will need $45097156608 / 2097152 = 21504$ memory map entries.

$8 \text{ bytes} * 21504 = 168\text{K}$.

Can support more users.

Faster access.

Memory & I/O Optimization

- HugePages on Linux reduces the size of per process memory pagetable
- Use to efficiently map virtual memory of a large SGA
- The application used shared servers to handle more than 4000 sessions per node.
- Observer cpu consumption reduced when prefetch increased from 10 to 69

Memory & I/O Optimization

- Future..

Dynamic Resident connection pooling (DRCP)

- Allows applications that cannot do middle-tier connection pooling to scale to tens of thousands of simultaneous connections
- Closely follows Dedicated Model. Removes overhead of dedicated server for short transactions
- pools “dedicated servers” which is the equivalent of a server foreground and a database session combined

Network Optimization



Network Optimization

- Prefetch sizing

Since we cannot predict type of sqls per minute, prefetch sizing was done by looking at DBA_HIST_SQLSTAT and trial runs

Found size of 60 to be optimal.

setPrefetchRowCount() in OCI client.

- Observed cpu consumption on shared server dispatchers reduce when prefetch increased from 10 to 60.
- average time for *sql*net mgs to client* was 0.9 milli second

Network Optimization

- Gigabit Inter node connection was sufficient.
 - Average internode traffic was ~4M per second.

Global Cache Load Profile

	Per Second	Per Transaction
Global Cache blocks received:	279.53	5,998.29
Global Cache blocks served:	289.23	6,206.35
GCS/GES messages received:	411.54	8,830.98
GCS/GES messages sent:	512.01	10,986.94
DBWR Fusion writes:	2.03	43.50
Estd Interconnect traffic (KB)	4,730.46	

OCI coding

- Eliminated third party library to reduce memory leaks
- Coded application to be multi threaded.
- Fixed code to eliminate any sequential access.

OCI Consistent Client Cache Benefits

- Frees application developers from building a shared per-process result cache shared by all sessions that is consistent.
- Extends server-side query caching to client side memory
 - leverages cheaper client-side memory
 - each application has it's working set cached locally
- Ensures better performance by eliminating round trips to the server
- Improves server scalability by saving server resources
- Transparently maintains cache consistency with server side changes
- Consistency mechanism works with RAC

OCI Consistent Client Cache

How does it work?

- Leverages a combination of unique Oracle technologies:
 - Oracle's snapshot based Read Consistency
 - Database Change Notification technology
- OCI Layer enhanced to lookup internal cache for all queries with `/*+ result_cache */`
- Consistency maintained by sending IN-BAND notifications on every roundtrip to server
 - in a relatively busy client, cache keeps sliding forward by catching up with the DB
 - in a relatively idle client, cache can trail behind DB no more than `CACHE_LAG` secs
- Query results cached in OCI client memory
 - per-process cache shared across multiple sessions/threads

OCI Consistent Client Cache Enabling

- Works with all OCI-based drivers
 - Including JDBC OCI, OCCl, ODP.Net, PHP, ODBC
- Activated with server or client parameter
 - Server
 - CLIENT_RESULT_CACHE_SIZE (default 0, cache disabled)
 - CLIENT_RESULT_CACHE_LAG (optional, 3000ms default)
 - Client (set in sqlnet.ora)
 - OCI_RESULT_CACHE_MAX_SIZE (optional)
 - OCI_RESULT_CACHE_MAX_RSET_SIZE (optional)
 - OCI_RESULT_CACHE_MAX_RSET_ROWS(optional)
- Applications explicitly tag queries with SQL hint
 - `select /*+ result_cache */ id, name from products;`
- Enable statement caching in Drivers/layers such as JDBC, ODP.Net etc or OCI Statement Caching can be used

OCI Consistent Client Cache Usage Guidelines

- Look for candidate queries in AWR
 - frequent queries in Top SQL by CPU/Elapsed time
 - identify candidate queries on read-only/read-mostly tables
 - sprinkle the `/*+ result_cache */` hint on such queries
 - validate by comparing performance with/without caching
- Monitor usage
 - `client_result_cache_stats`
- Control the “lag”
 - `client_result_cache_lag`

Database Optimization



CPU Optimization

- Over 20000 executions per second

Load Profile

	Per Second	Per Transaction	Per Exec	Per Call
DB Time(s):	9.8	26.9	0.00	0.00
DB CPU(s):	8.5	23.2	0.00	0.00
Redo size:	3,804.4	10,443.8		
Logical reads:	144,074.4	395,509.4		
Block changes:	41.2	113.1		
Physical reads:	152.6	419.0		
Physical writes:	3.3	9.1		
User calls:	58,953.0	161,836.4		
Parses:	22,670.7	62,234.9		
Hard parses:	1.3	3.6		
W/A MB processed:	31,909,396.3	87,596,878.3		
Logons:	0.6	1.7		
Executes:	22,671.9	62,238.4		

REF Partitioning

Business Problem

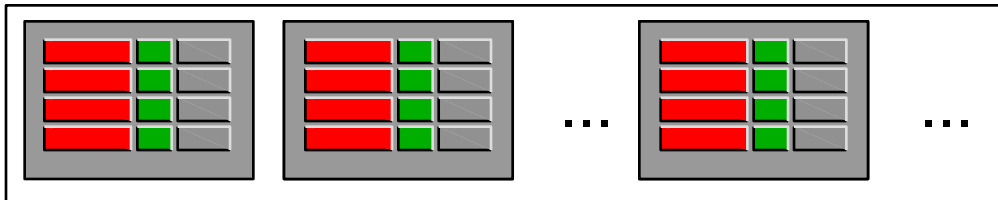
- Related tables benefit from same partitioning strategy
 - e.g. order – lineitem
- Redundant storage of the same information solves this problem
 - data overhead
 - maintenance overhead

Solution

- Oracle Database 11g introduces REF Partitioning
 - child table inherits the partitioning strategy of parent table through PK-FK relationship
 - intuitive modelling
- Enhanced Performance and Manageability

Before REF Partitioning

Table ORDERS



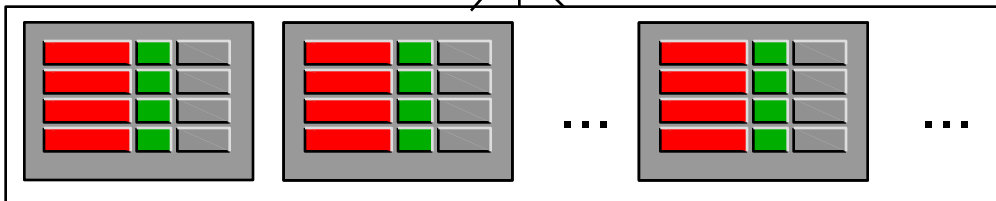
Jan 2006

Feb 2006

- RANGE(**order_date**)
- Primary key **order_id**

- Redundant storage of order_date
- Redundant maintenance

Table LINEITEMS



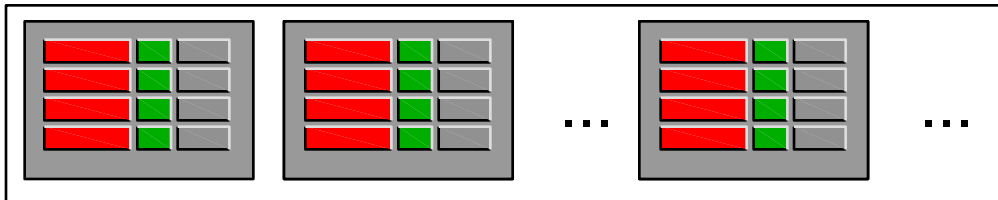
Jan 2006

Feb 2006

- RANGE(**order_date**)
- Foreign key **order_id**

REF Partitioning

Table ORDERS



Jan 2006

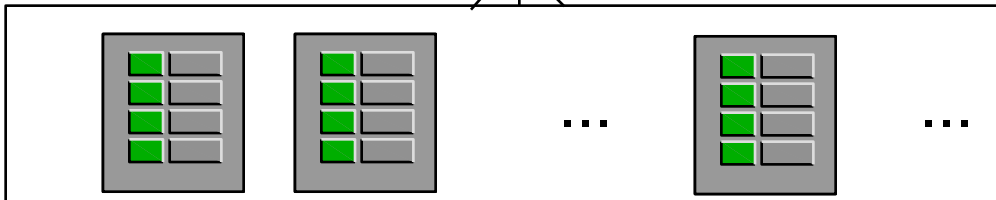
Feb 2006

- RANGE(**order_date**)
- Primary key **order_id**

PARTITION BY REFERENCE

- Partitioning key inherited through PK-FK relationship

Table LINEITEMS

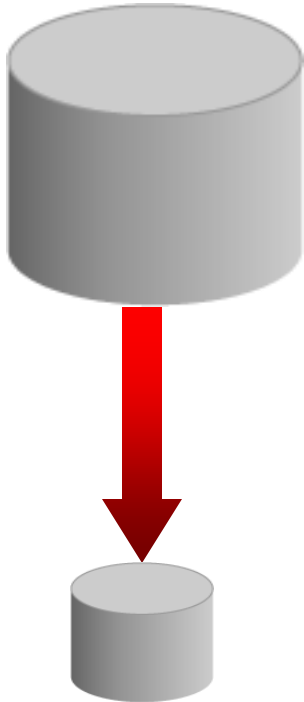


Jan 2006

Feb 2006

- RANGE(**order_date**)
- Foreign key **order_id**

Compression for Mainstream

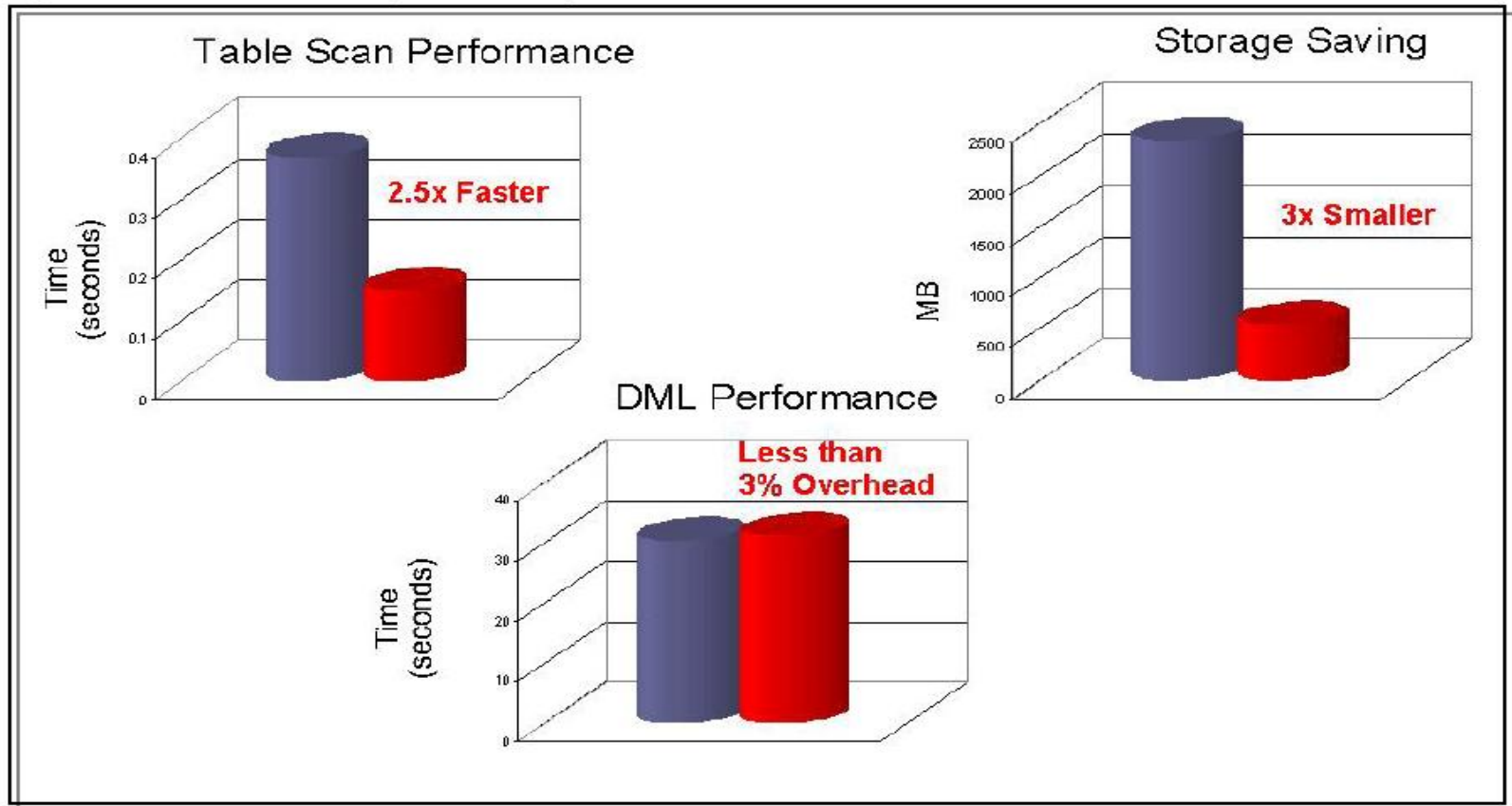


- Oracle 9i compresses table data only during bulk load
 - Data warehousing, ILM
- Table compression is now viable for all applications
 - Compress during random updates
- Typical compression ratio of **2x to 3x**
- Database directly reads compressed data bypassing decompression overhead
- Strategy - Compress 10 largest tables in a database
 - Reduce table data by half, increase CPU usage 5%
- Savings cascades into test, dev, standby, mirrors, archiving, backup, etc.

CPU Optimization

- 11g Data Compression

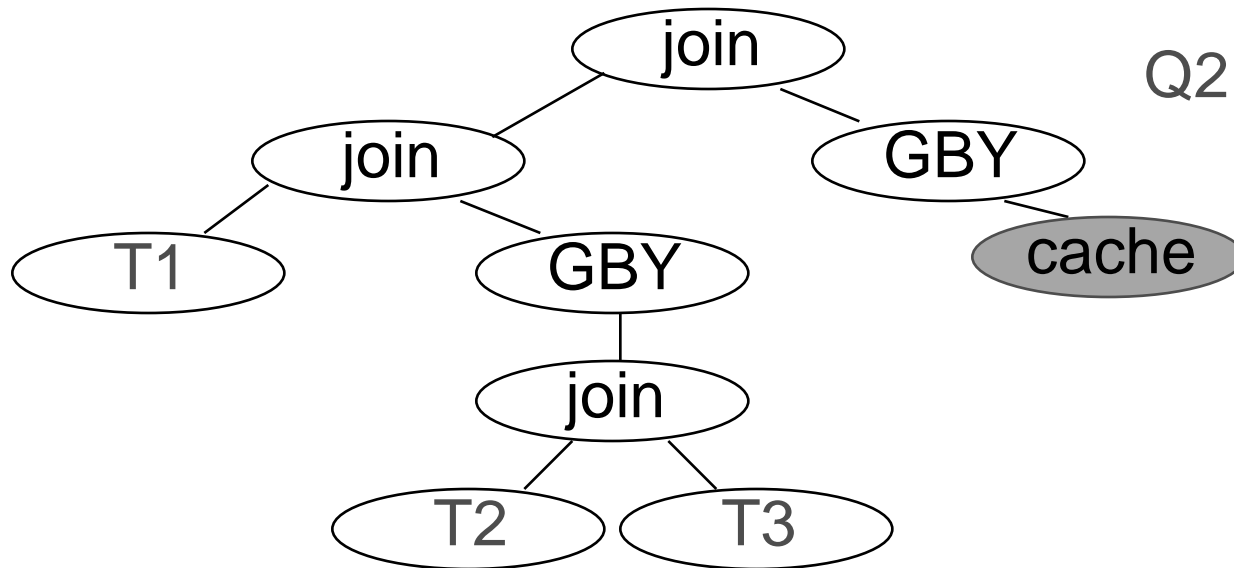
Figure 3. Performance Impact of Compression



Server Results Cache

taking the buffer cache to the next level

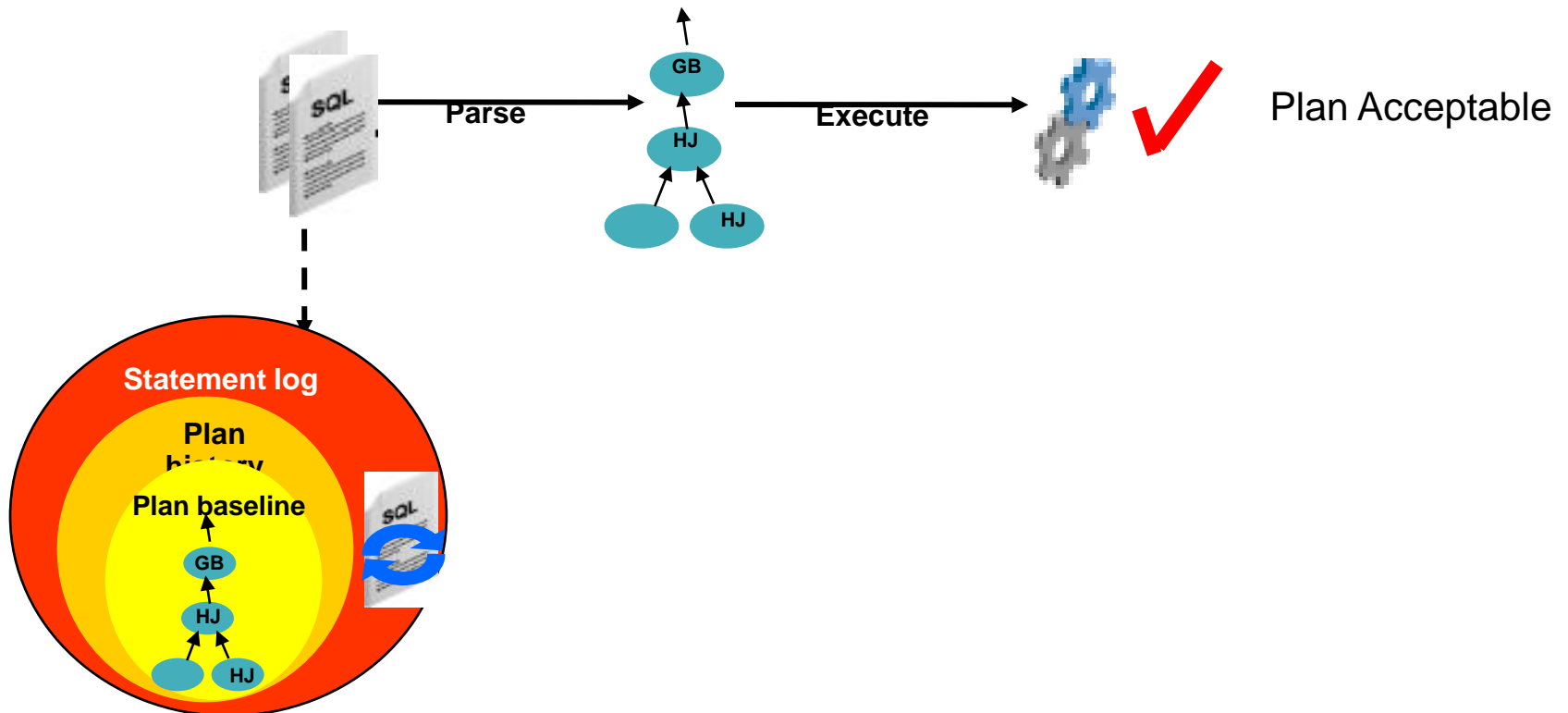
- Caches results of queries, query blocks, or pl/sql function calls
 - Cache is shared across statements and sessions on server
 - Significant speed up for read-only / read-mostly data



Q2: Use it transparently

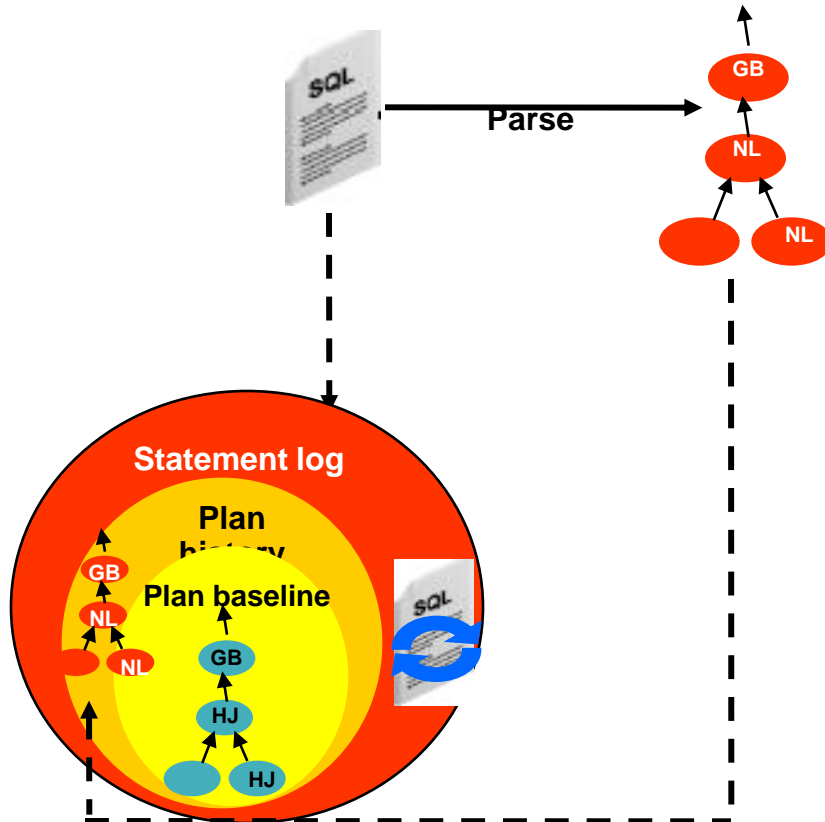
SQL Plan Management

- Repeatable SQL populates statement log, plan history and creates a baseline



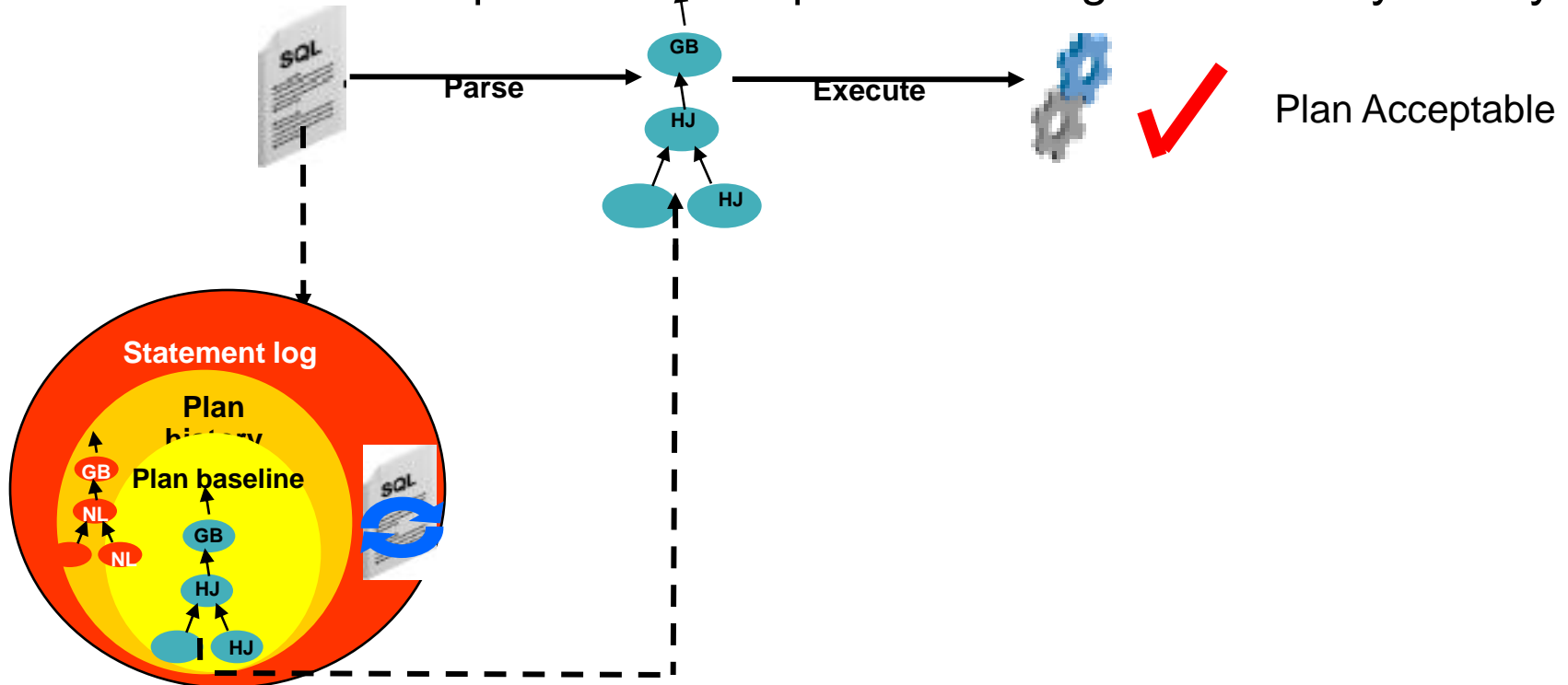
With SQL Plan Management

- Something changes in the environment
- SQL statement is parsed again and a **new plan is generated**
- New plan is not the same as the baseline – **new plan is not executed** but marked for verification



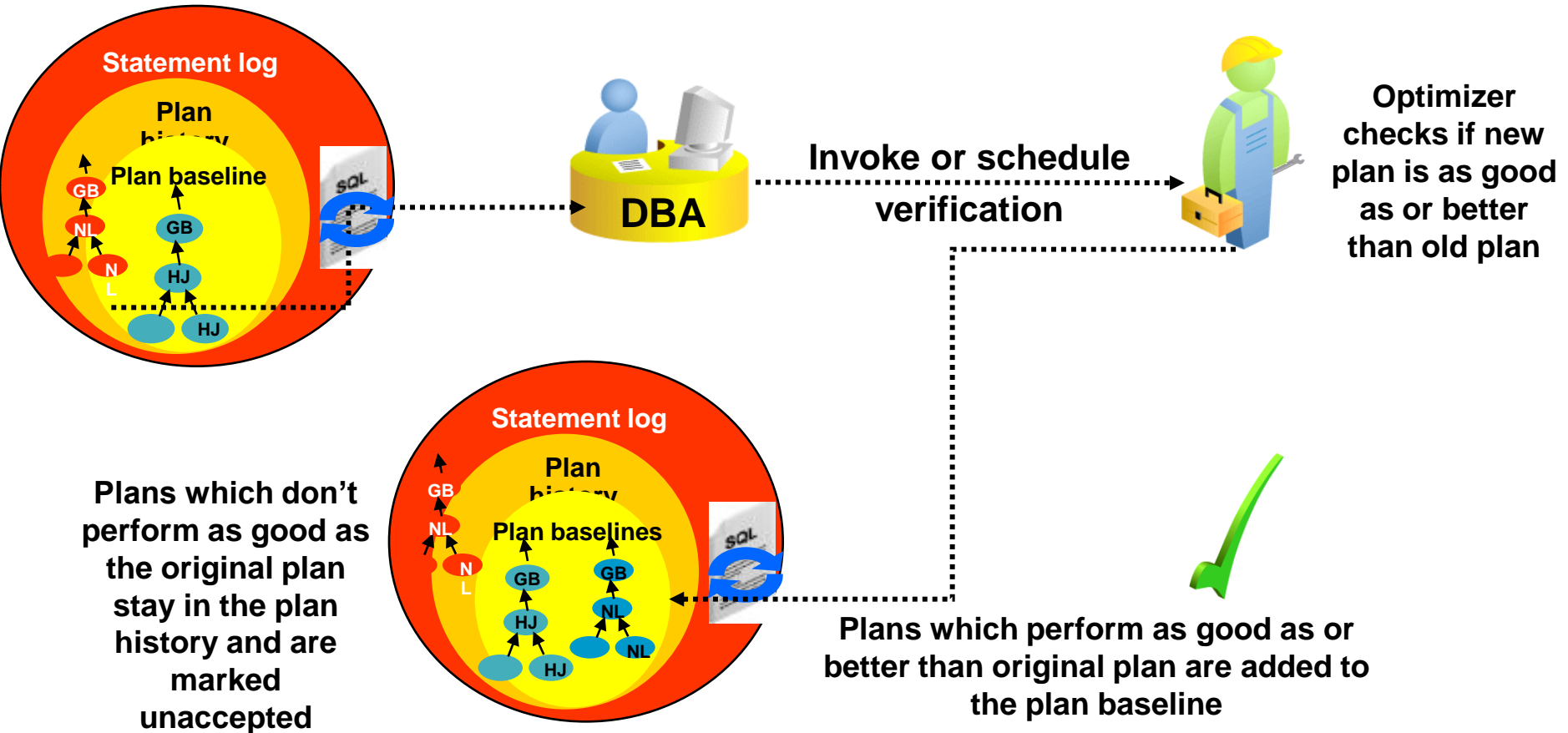
With SQL Plan Management

- Something changes in the environment
- SQL statement is parsed again and a **new plan is generated**
- New plan is not the same as the baseline – **new plan is not executed** but marked for verification
- Execute the known plan baseline “performance guaranteed by history”

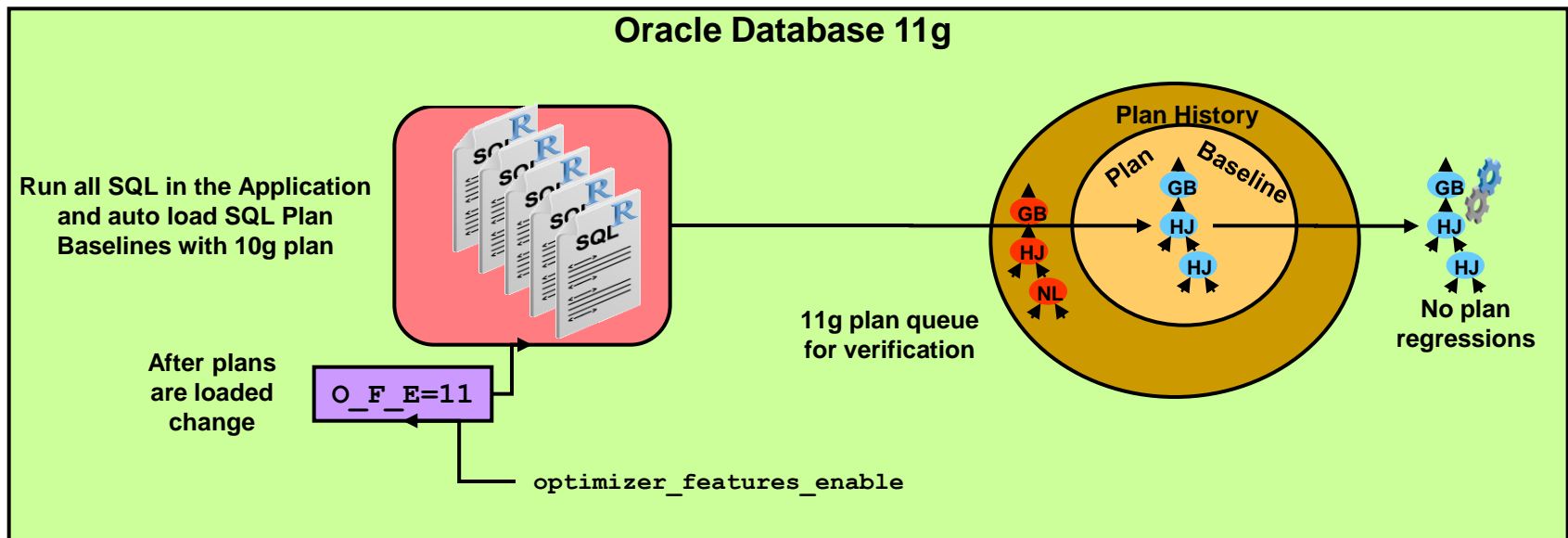


Verifying the new plan

- Non-baseline plans will not be used until verified
- DBA can verify plan at any time



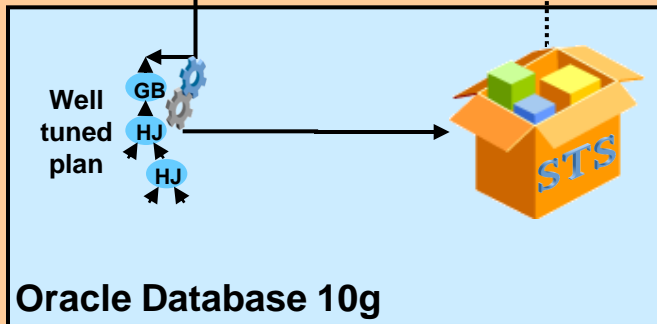
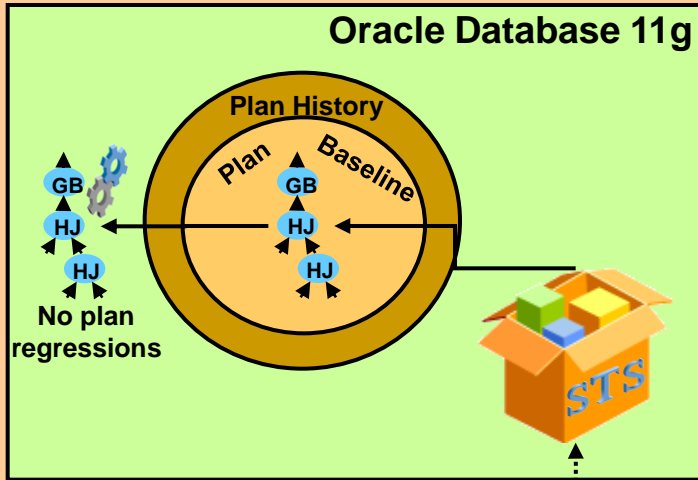
SQL Plan Baseline general upgrade strategy



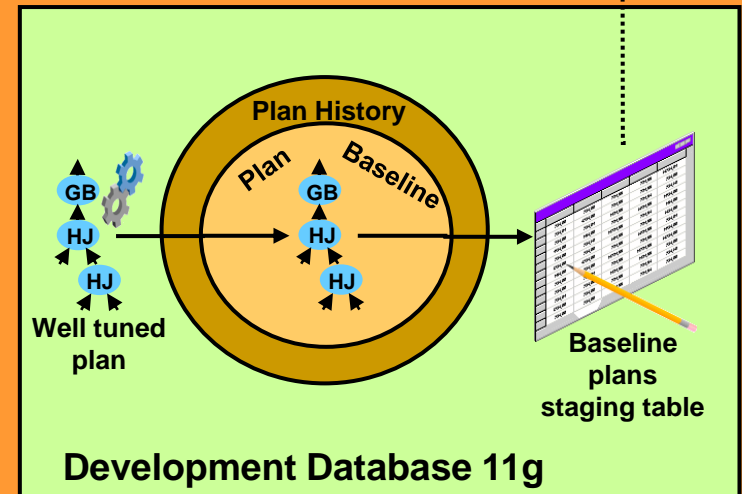
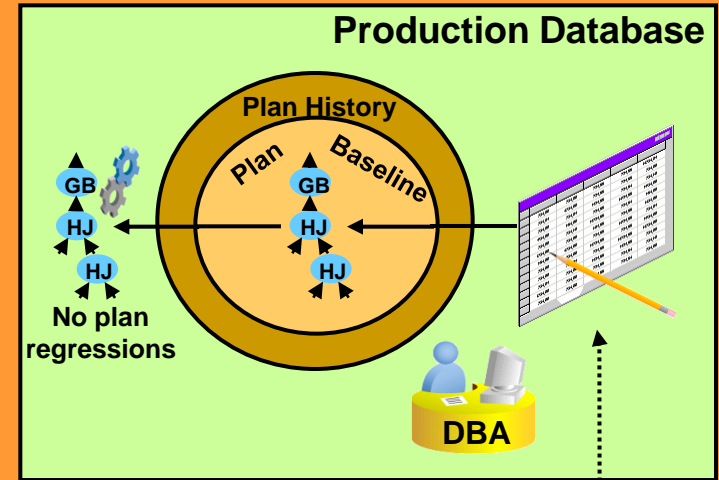
- Seeding the SQL Plan Baselines with 10g plans No plan change on upgrade
- After all SQL Plan Baselines are populated switch Optimizer_Features_Enable to 11g
 - new 11g plans will only be used after they have been verified

Possible SQL Plan Manageability Scenarios

Database Upgrade
using Tuning Pack



New Application Deployment,
no Tuning Pack required

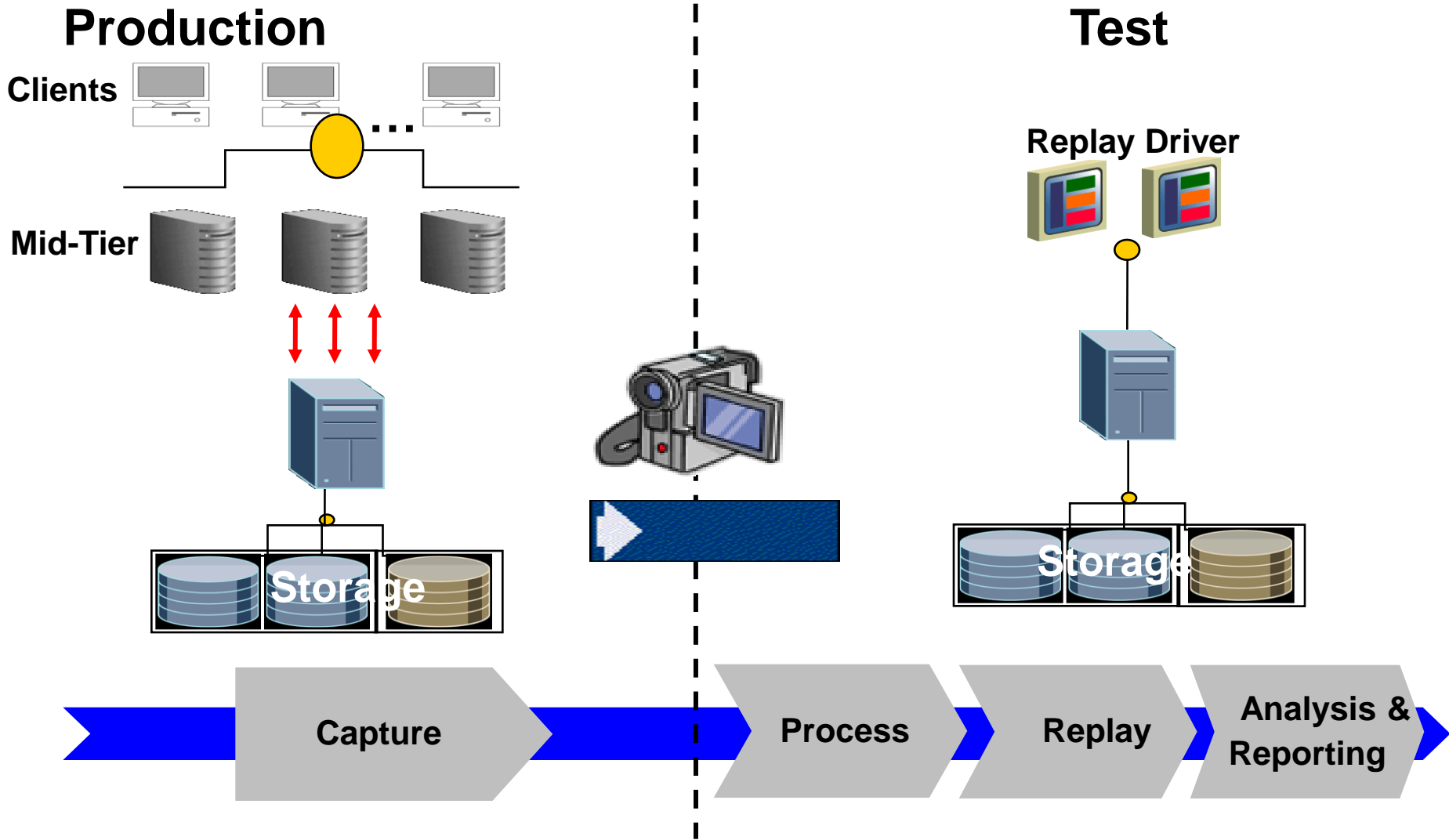


Database optimization

- Top sqls were identified and added to base lines

```
SET SERVEROUTPUT ON
DECLARE
  l_plans_loaded  PLS_INTEGER;
BEGIN
  l_plans_loaded := DBMS_SPM.load_plans_from_cursor_cache(
                    sql_id => '15rp2as8g68py');
  DBMS_OUTPUT.put_line('Plans Loaded: ' ||
l_plans_loaded);
END;
/
```

Database Replay Workflow

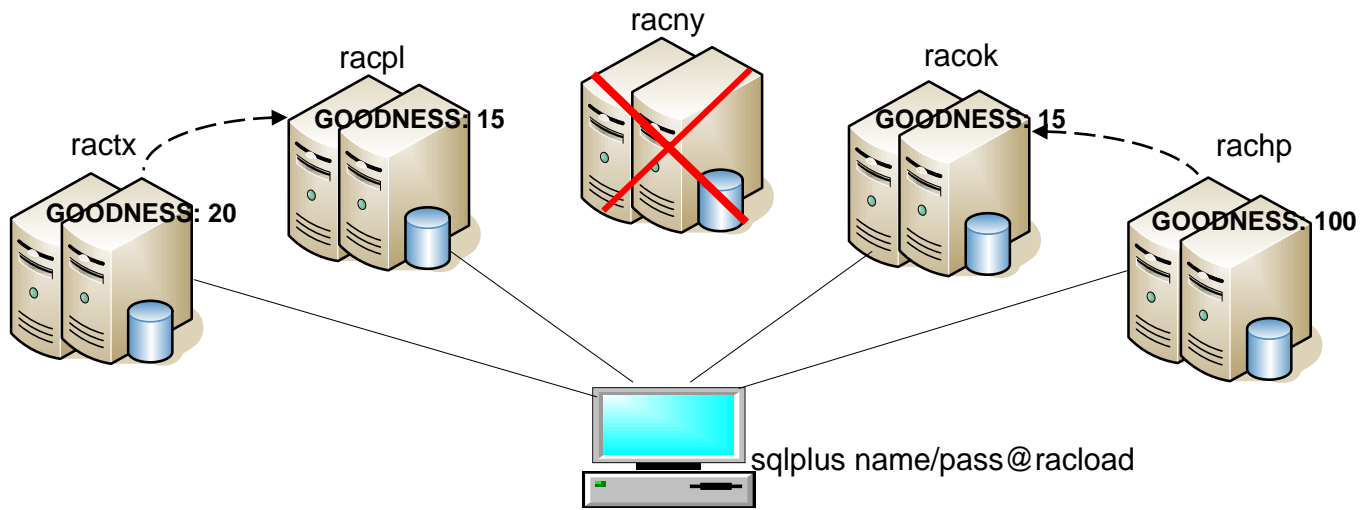


Load Balancing



Load Balancing

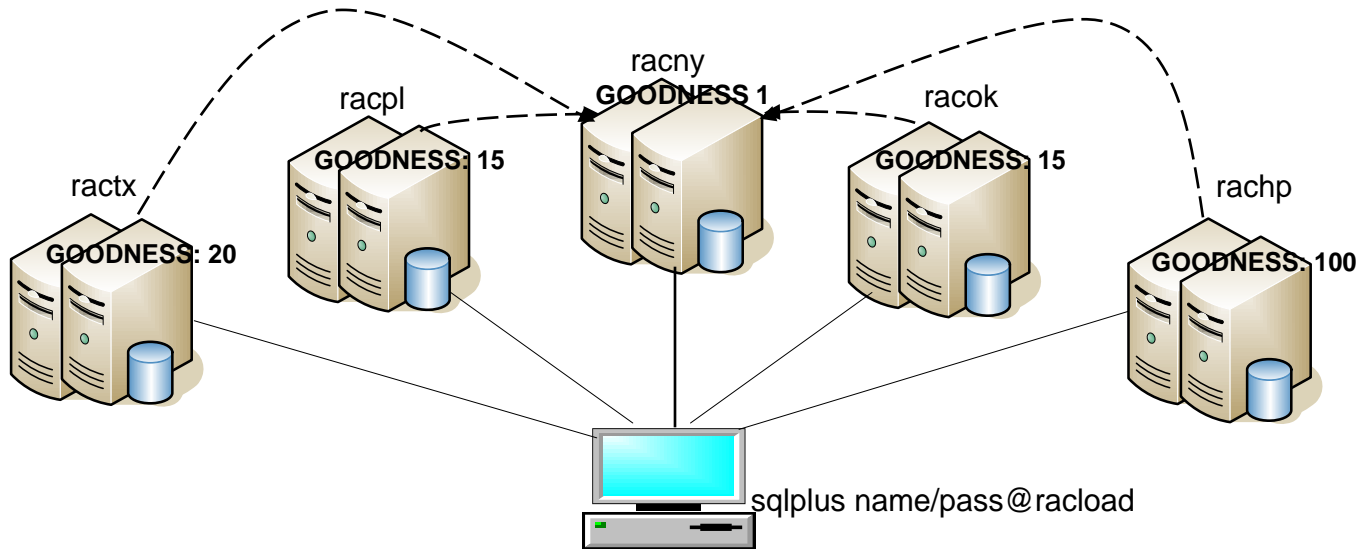
- Server side Load Balancing across clusters



Makes use of Load Balancing Advisory feature

Load Balancing

- Server side Load Balancing across clusters

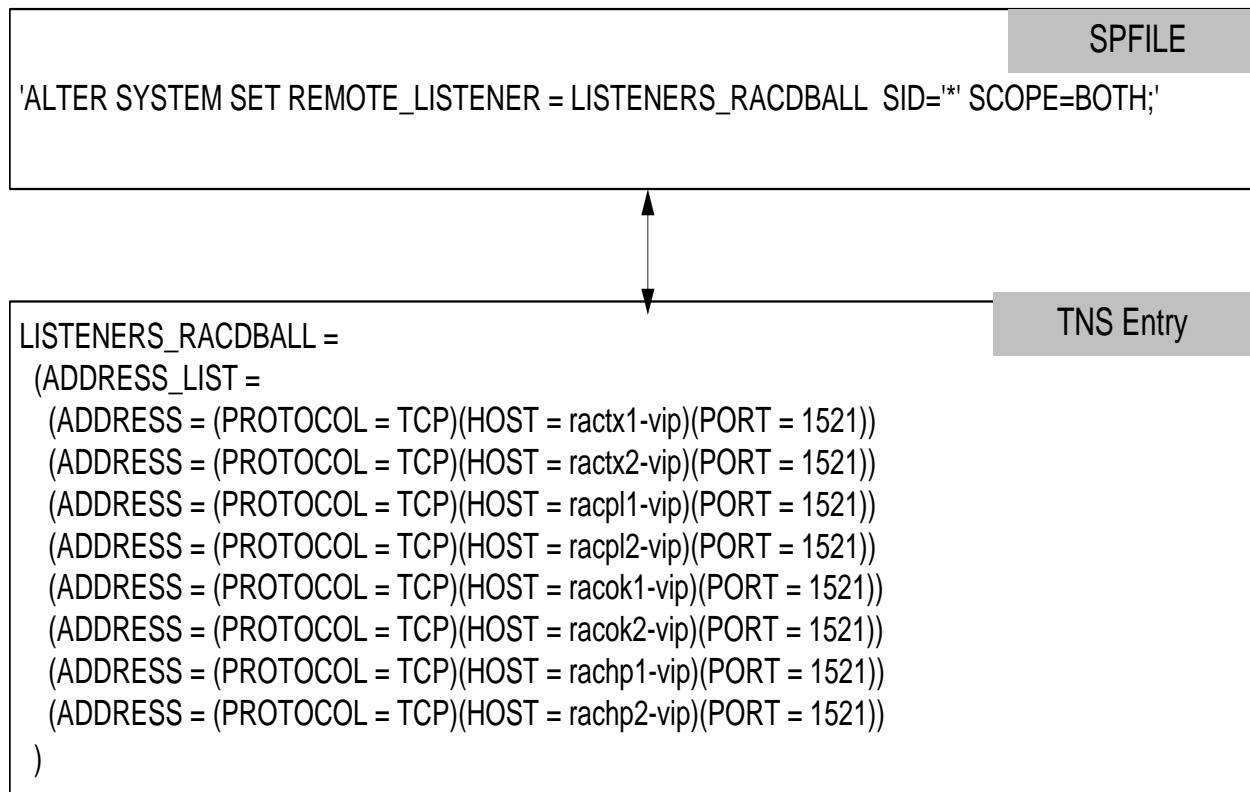


Verify service in listener: lsnrctl status
List services in listener: lsnrctl services

Makes use of Load Balancing Advisory feature

Load Balancing

- Server side Load Balancing across clustered DBs



Load Balancing

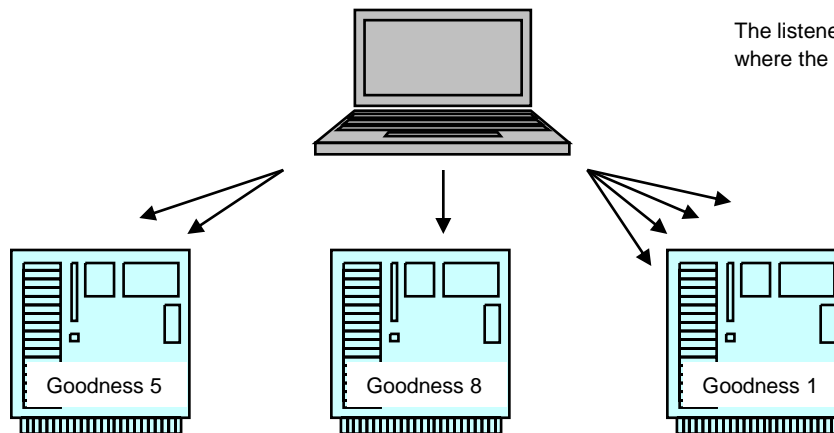
On Connection – Server side Load balancing with Load Balancing Advisor Routed to least busy on a service by service basis

Service Goals

- None
- Service Time - Best response time – elapsed time for work done in the service plus available bandwidth. (per call metric) Example: Internet shopping
- Throughput - better throughput – rate that work is completed plus available bandwidth. (per second metric) Example: Batch

Connection Load Balancing

- **Short** – Connection load balancing uses Load Balancing Advisory, when Load Balancing Advisory is enabled with Service Goals.
- Long – Balances the number of connections per instance using session count per service.



The listener uses the load balancing advisory when load balancing connections where the service has `clb_goal_short` and `goal=service_time` or `throughput`

Goodness: ranking for the quality of service that the service is experiencing at an instance level, including whether access has been restricted from an instance. Lower number is better

Load Balancing

- Server side Load Balancing across clusters.

Create RAC service (from unix srvctl command)

```
srvctl config service -d ractx -a
srvctl add service -d ractx -s racload -r ractx1, ractx2 -P basic
srvctl start service -d ractx -s racload
srvctl config service -d ractx -a
```

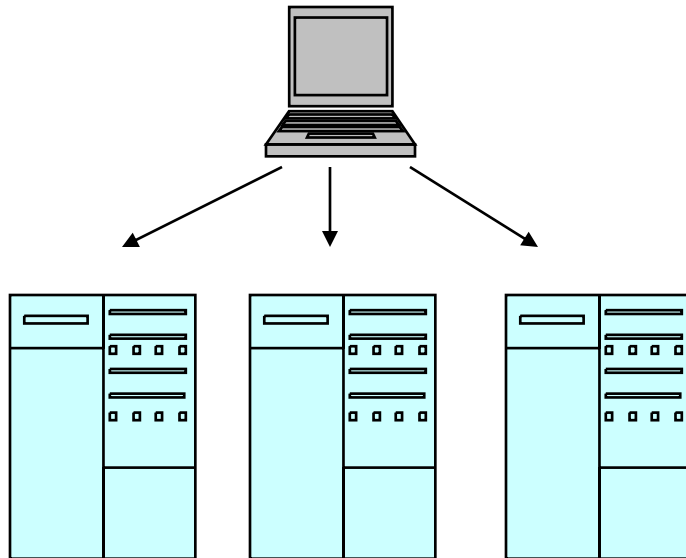
From sqlplus, modify service for goals:

```
BEGIN
DBMS_SERVICE.MODIFY_SERVICE(
    service_name => 'racload'
    ,failover_method => DBMS_SERVICE.FAILOVER_METHOD_BASIC
    ,failover_type => DBMS_SERVICE.FAILOVER_TYPE_SESSION,
    ,failover_retries => 10,
    ,failover_delay => 1,
    ,goal => DBMS_SERVICE.GOAL_SERVICE_TIME
    ,clb_goal => DBMS_SERVICE.CLB_GOAL_SHORT
);
END;
/
```

Load Balancing

Client Side Connection Load balancing utilizing TNSNAMES

```
CLIENT1 =  
(DESCRIPTION =  
  (LOAD_BALANCE = ON)    # ON=Random Connections OFF=Sequential connections  
  (FAILOVER = ON)       # Activates failover when TCP timeout is reached  
  (ADDRESS = (PROTOCOL = TCP)(HOST = racny1-vip)(PORT = 1521))  
  (ADDRESS = (PROTOCOL = TCP)(HOST = racny2-vip)(PORT = 1521))  
  (ADDRESS = (PROTOCOL = TCP)(HOST = ractx2-vip)(PORT = 1521))  
  (ADDRESS = (PROTOCOL = TCP)(HOST = ractx2-vip)(PORT = 1521))  
  (CONNECT_DATA =  
    (SERVER = DEDICATED)  
    (SERVICE_NAME = racall) ) )
```



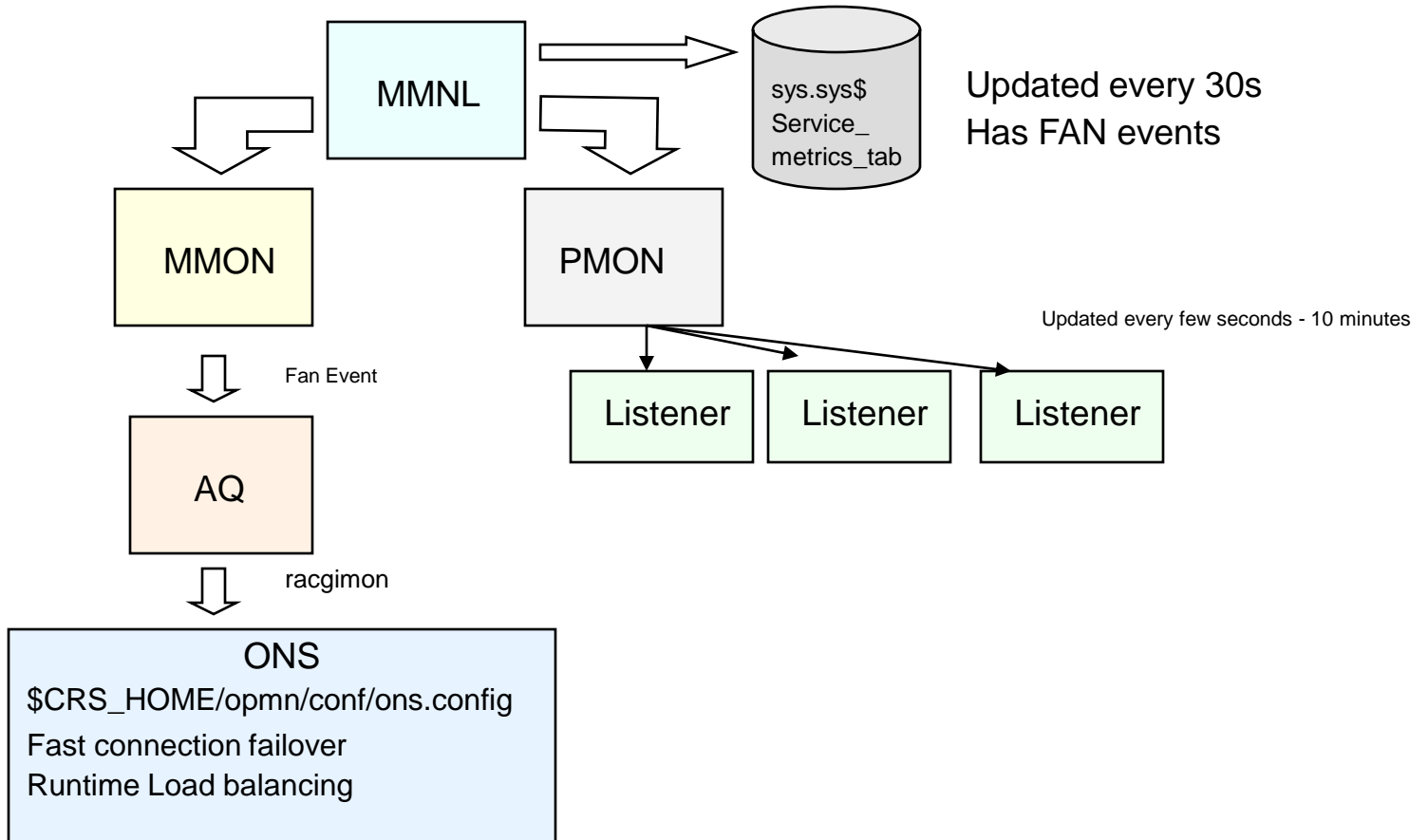
Random connections
One listener is not overwhelmed

Set **SQLNET.OUTBOUND_CONNECT_TIMEOUT** to specify the time, in seconds, for a client to establish a TCP connection to the database server/service. When a cluster is completely down, this timeout will be used since all vips will be down.

Load Balancing

Load Balancing Advisor processes

MMNL - Memory Monitor Light
MMON - Manageability Monitor
ONS - Oracle Notification Service



Load Balancing

LBSCORE – Surge Protection

- Oracle 10.2 and higher: Listener uses a local value "lbscore"
- Lbscore : two dynamic values "goodness" and "delta"
- Continuously updated by PMON.

When an instance is started, PMON contacts the listener and provides starting values for goodness and delta...

Listener Lbscore = Goodness from PMON

Listener Delta = Delta from PMON

To account for changes in load between PMON updates, the listener will increment the lbscore after each new incoming connection...

Listener Lbscore = Listener Lbscore(previous) + Listener Delta

```
2009-03-02 11:54:40.019022 : nsglb:instance:RACTX11 new lbscore:2400
2009-03-02 11:54:41.073505 : nsglb:instance:RACTX12 new lbscore:2400
2009-03-02 11:54:42.131014 : nsglb:instance:RACNY11 new lbscore:100
2009-03-02 11:54:43.179900 : nsglb:instance:RACNY11 new lbscore:200
2009-03-02 11:54:44.230763 : nsglb:instance:RACNY11 new lbscore:300
2009-03-02 11:54:45.277785 : nsglb:instance:RACNY11 new lbscore:100
2009-03-02 11:54:46.325604 : nsglb:instance:RACNY11 new lbscore:200
2009-03-02 11:54:47.373624 : nsglb:instance:RACNY11 new lbscore:300
2009-03-02 11:54:48.422227 : nsglb:instance:RACTX11 new lbscore:100
2009-03-02 11:54:49.477567 : nsglb:instance:RACTX11 new lbscore:200
2009-03-02 11:54:50.536465 : nsglb:instance:RACTX11 new lbscore:300
2009-03-02 11:54:51.589902 : nsglb:instance:RACTX11 new lbscore:400
2009-03-02 11:54:52.642743 : nsglb:instance:RACNY11 new lbscore:400
2009-03-02 11:54:53.703777 : nsglb:instance:RACTX11 new lbscore:500
```

Load Balancing

Update listener information

- PMON updates the listener with instance load and dispatcher information
- PMON SERVICE_UPDATE's vary according to the workload of the instance
- The maximum interval between these service updates is 10 minutes.

Check Listener log. When busy, it updated every few seconds

```
05-MAR-2009 13:31:15 * service_update * RACNY11 * 0
05-MAR-2009 13:31:16 * service_update * RACTX22 * 0
05-MAR-2009 13:31:19 * service_update * RACTX21 * 0
05-MAR-2009 13:31:31 * service_update * RACTX22 * 0
05-MAR-2009 13:31:52 * service_update * RACNY12 * 0
```

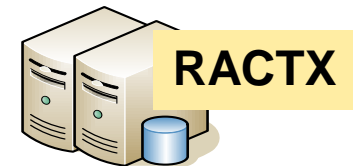
Listener Trace: 'lsnrctl set trc_level 16'
produces a trace log that shows the Load Balance Advisory updates

```
2009-03-05 16:24:17.185722 : nsglgrDoRegister:inst loads: ld1:102 mld1:81920 ld2:17
mld2:6000
2009-03-05 16:24:17.185746 : nsglgrDoRegister:service:RACNY_STAGING what:4 value:1
2009-03-05 16:24:17.185763 : nsglgrDoRegister:service:RACNY_STAGING what:2 value:0
2009-03-05 16:24:17.185799 : nsglgrDoRegister:service:RACNY_RTG what:4 value:100
2009-03-05 16:24:17.185817 : nsglgrDoRegister:service:RACNY_RTG what:2 value:100
2009-03-05 16:24:17.186112 : nsglgrDoRegister:service:RACNY1 what:4 value:1
2009-03-05 16:24:17.186129 : nsglgrDoRegister:service:RACNY1 what:2 value:2
2009-03-05 16:24:17.186157 : nsglgrDoRegister:exit
2009-03-05 16:24:17.186193 : nsdo:entry
```

What:4 = Delta
What:2 = Goodness
ld1 = node load - load average
ld2 = instance load - number user sessions
mld1 = max load data/ max cpu capacity
mld2 = max load data/sessions init.ora

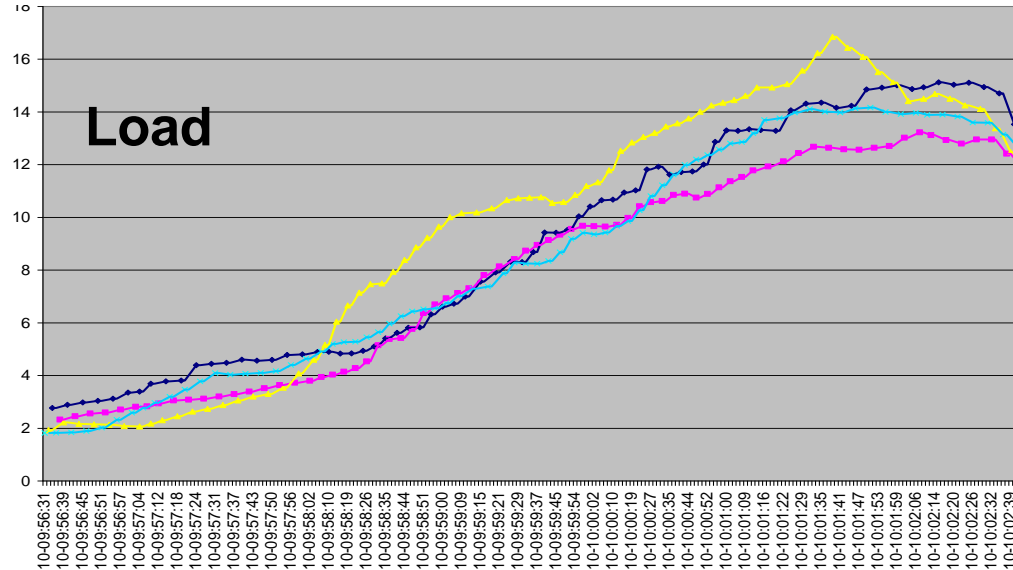
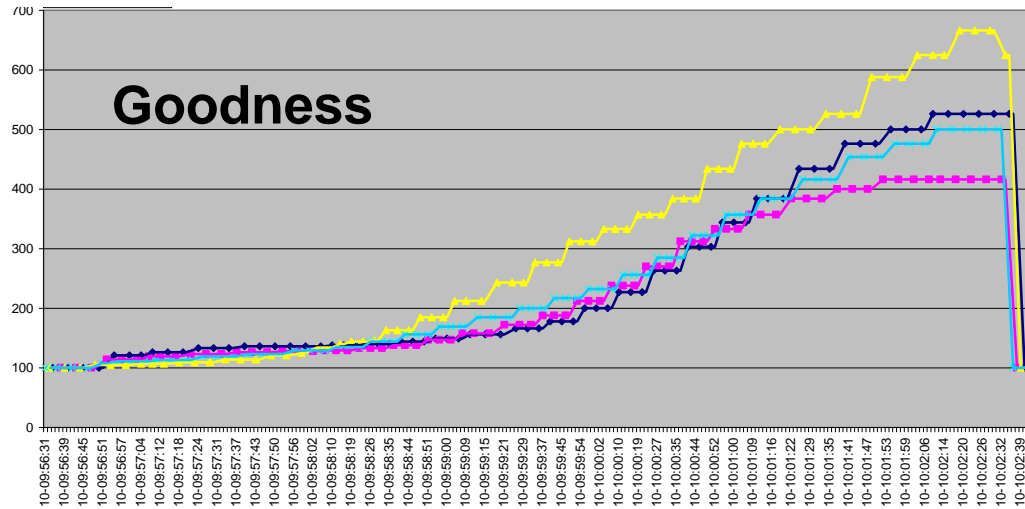
Server Side Load Balancing Test Results

HOST	LOAD	COUNT
racny1	0.269	156
racny2	0.729	92
ractx1	2.379	7
ractx2	0.419	119



Load Balancing

Swingbench with a slow ramp up of 250 sessions



Load Balancing

gv\$active_services	Active services
gv\$session	Service_name by session
gv\$service_stats	Stats by service calls, execute, parse, etc
dba_services	goal, failover, etc
gv\$servicemetric	Goodness and flags Use group_id=10 for latest entry Updated every 5 seconds
gv\$servicemetric_history	Servicemetric history Contains 1 hour of history
sys.sys\$service_metrics_tab	LBA advisory events Updated every 30 seconds Has RLB fan event

TCO

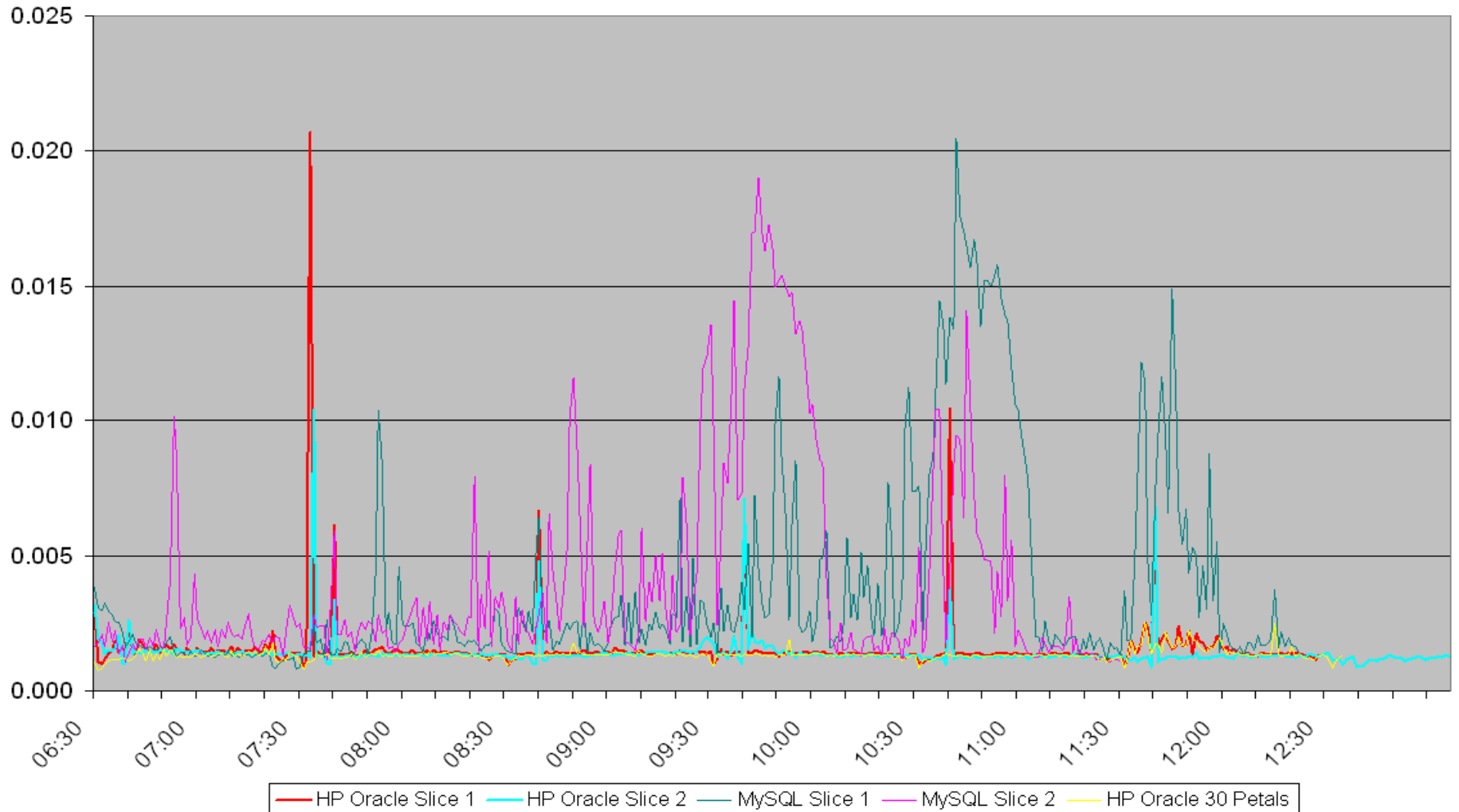


TCO

- By tracking wait events and Time Model statistics we could pin point exactly where the time is spent in the database and even estimate the percentage of performance benefit by tuning.
- The power of the management packs gives the DBAs the ability to optimize resource utilization more than any other database.

TCO

HP Lab - SELECT Query Average Response Times from 8 Petals/1 POD (pimhp015)
6 Hour (8 Slice) Test - 60 Petal Total DB Workload - 22GB Oracle SGA



TCO

- 1/10th of the MySQL servers
- 1/5th of the MySQL processors
- Improve application server to database server ratio 10 fold
- Cost Reduction
- > 20,000 executions a second per node - as high as 27,000+
- Approx 4000 connections per node
- 8 copies of the databases vs the 160 copies
- Minimal wait/cluster wait

- Simplified management
 - Index creations done online
 - Table change takes less than 5 minutes and requires just 1 resource (vs. 4hrs & 3 heads)
 - Adding a column requires the running of a simple alter statement that can be replicated
 - The data is the same on all nodes (vs. different images by node)
 - Test system refreshes require 1 FTE working 4 hours (vs. 5 FTEs working 48 hours)
 - Change management is much easier

TCO

- Standard tools provide detail analysis of performance in real time
- Database instrumentation in place and provided
- Services approach to connection management using even distribution of services though usage patterns and load balancing
- 1/20th or less recovery time using Flash Back

TCO

Oracle

- Oracle Real Application Server – scalability on the same data copy
- Fully utilize all available CPU cores
- Memory utilization greater than 40 GB per server
- Throughput: 181 million queries returning 1.1 trillion rows in one hour
- Large application server to db server ratio with large connection management capability

MySql

- Availability in numbers scalability by duplication only
- CPU Core utilization restricted
- Memory utilization has not been as high
- Throughput: Difficult to say without having the proper tools

TCO

Oracle

- Reduction of 2 FTE
- Reduction in severity outages
- Index creations done online.
- Average table change in Oracle takes less than 5 minutes and requires two resources.
- Adding a column requires the running of a simple alter statement.
- Data segmentation achieved through services i.e. different access paths to the same server.

MySql

- No staff reduction
- 5 out of 7 database severity related to current infrastructure
- Index creations result in table locks.
- Average table change in MySQL takes more than 4 hours and requires three resources.
- Adding a column requires the building of an additional table and then an outage related cutover.
- Data segmentation achieved through scenarios i.e. additional servers.

TCO

Oracle

- Weekly and scheduled system refreshes accomplished through backup & recovery or transportable tablespaces. 1 FTE working 4 hours
- End-to-end automation w/fewer engineers
- Highly qualified Oracle engineering resources available

MySql

- Weekly and scheduled test system refreshes require a separate extract and load process for MySQL. 5 FTEs working 48 hours
- Difficult to introduce end-to-end automation
- Difficult to find qualified MySQL engineers

TCO

Oracle

- Standard tools provide detail analysis of system performance in real time
- Oracle is ahead in database instrumentation i.e. its timing model.
 - Automatic Workload Repository
 - Oracle Enterprise Manager
 - Automatic Database Diagnostic Monitor /w Repository
 - Active Session History
 - Automatic SQL Tuning
 - Automatic Memory Management
 - Session Level Tracing

MySql

- Dedicated specialized team in place to provide manual monitoring with home grown tools.
- MySQL limited on it's database instrumentation.
 - Enterprise Monitor
 - Explain Plans
 - Slow log

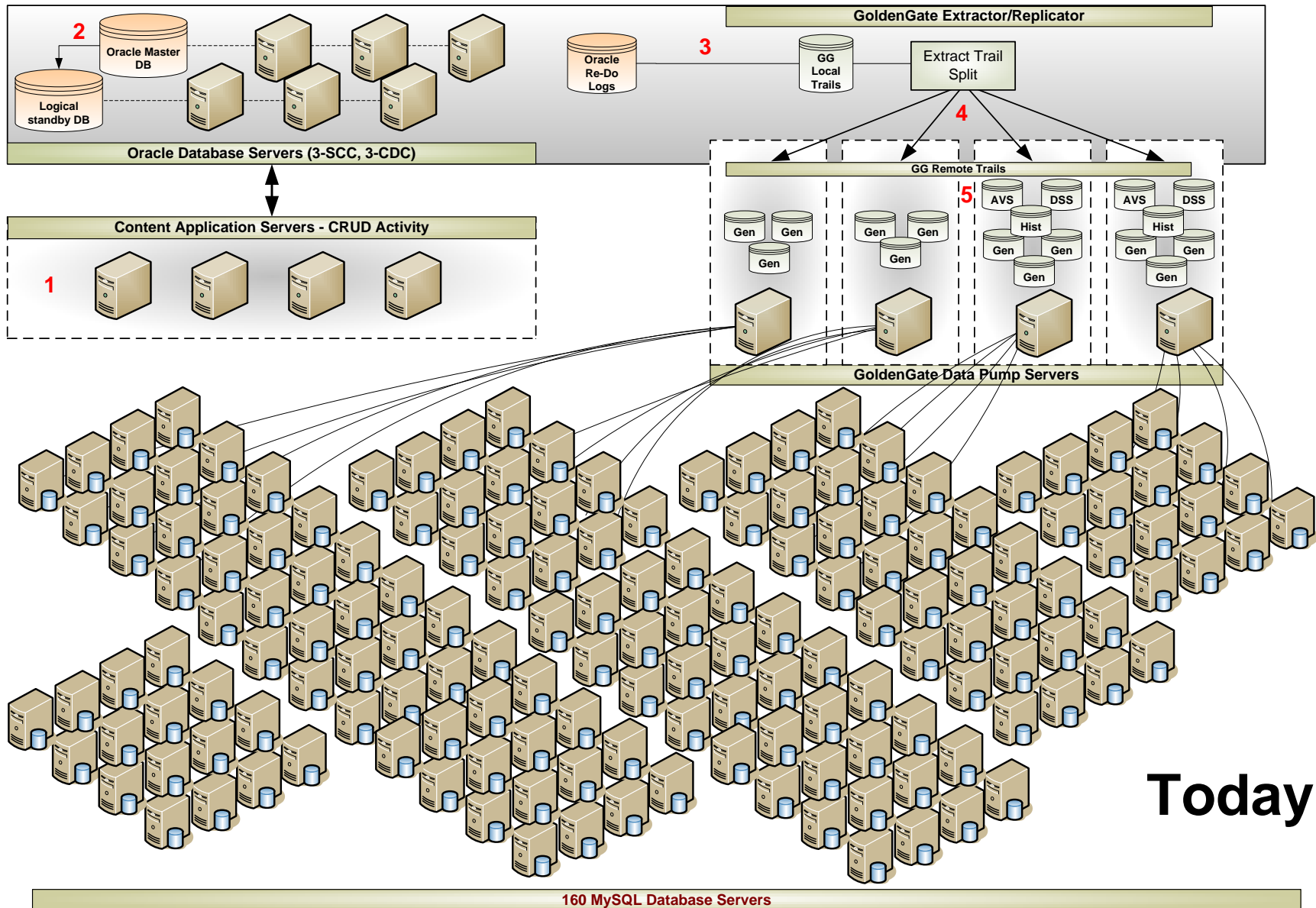
TCO

Oracle

- SQL Plan Management
- Partitioning
- Automatic SQL Tuning
- Automatic Shared Memory Management
- Compression
- Database Resident Connection Pooling

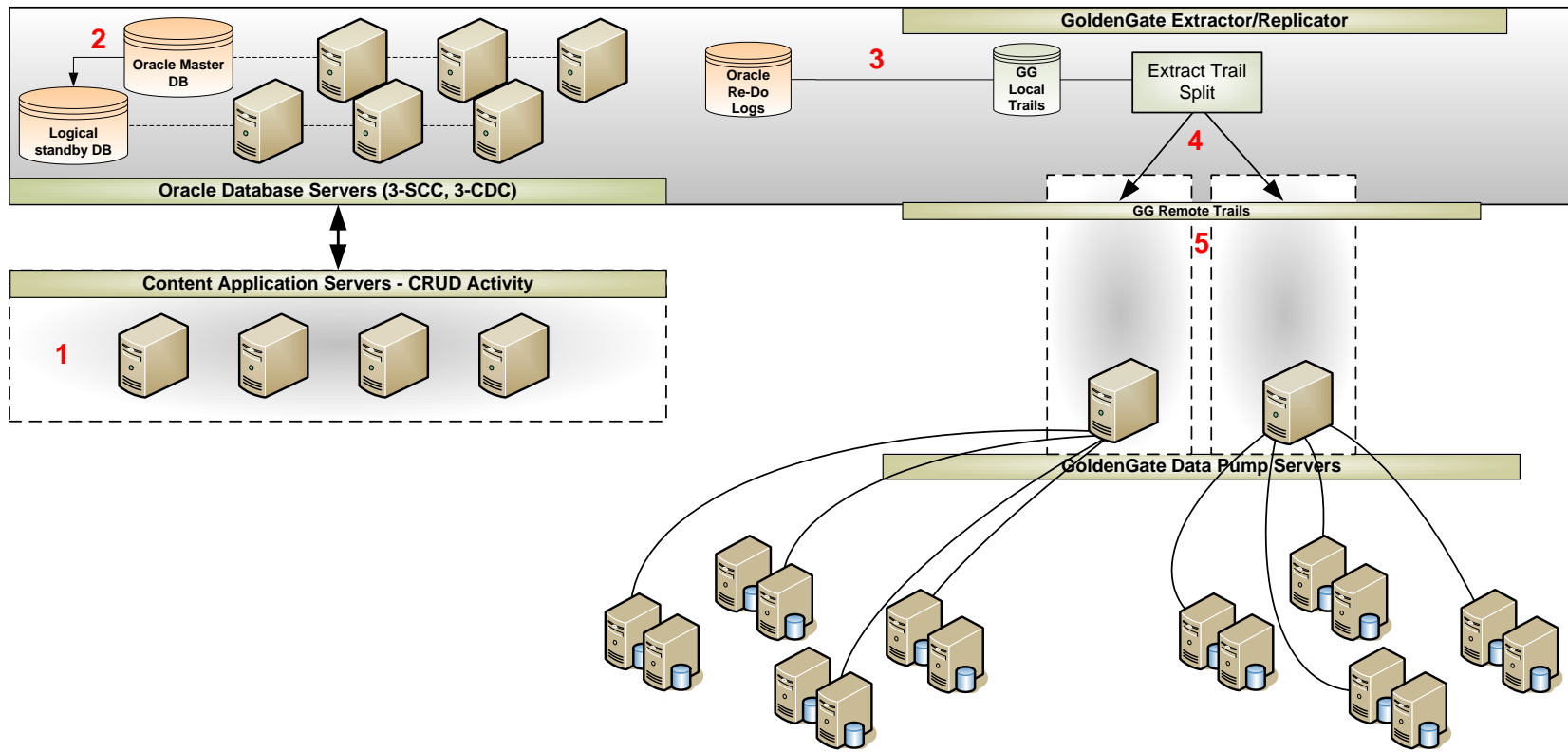
MySql

- MySQL 5.1 (not yet GA) Partitioning
- InnoDB (Oracle) Plug in
 - Fast Index Creation
 - Data Compression



Today

160 MySQL Database Servers



Tomorrow

8 Oracle 2 Node RAC Database Clusters - Blades