

Oracle PeopleSoft on Oracle
Exadata Database Machine

*Oracle Maximum Availability Architecture White Paper
February 2011*

Maximum Availability Architecture

Oracle Best Practices For High Availability

| | |
|---|----|
| Executive Overview | 2 |
| Introduction | 2 |
| Summary of Best Practices | 3 |
| Exadata Database Machine..... | 3 |
| PeopleSoft..... | 4 |
| Oracle PeopleSoft MAA on Exadata Database Machine | 5 |
| Getting PeopleSoft to Exadata Database Machine | 6 |
| Installing a New Database | 6 |
| Migrating an Existing Database | 7 |
| Case Study: PeopleSoft HR Payroll on Exadata Database Machine . | 7 |
| Controlled Test Environment | 8 |
| Workload Profile | 9 |
| Scaling HR Payroll on Exadata..... | 10 |
| Exadata Smart Flash Cache..... | 16 |
| PeopleSoft Process Scheduler Placement..... | 17 |
| Conclusion | 21 |
| References | 22 |

Executive Overview

Oracle PeopleSoft deployed on Oracle Exadata Database Machine greatly benefits from the integrated Oracle Maximum Availability Architecture (MAA) and performance features. This paper provides the reader guidance for deploying Oracle PeopleSoft on Exadata Database Machine, to include MAA best practices, and provides a case study focused on Oracle PeopleSoft HR Payroll batch process performance. For generating workload, the PeopleSoft North America HR Payroll benchmark kit was used for the case study.

Oracle MAA [1] is Oracle's best practices blueprint for implementing Oracle high-availability technologies. The best practices for deploying Oracle PeopleSoft on Exadata Database Machine for maximum availability and disaster recovery have been documented and validated by the Oracle MAA team.

Introduction

Oracle PeopleSoft runs on Oracle Exadata Database Machine just as it runs on any other platform. However, unlike other platforms, the Exadata Database Machine offers a suite of technologies that has revolutionized the enterprise. The following suite of technologies addresses a number of challenges facing IT today:

- Oracle Real Application Clusters (Oracle RAC), and intelligent Exadata storage grid shorten deployment time (days) to achieve a fully functional and validated clustering solution that is integrated with an InfiniBand network.
- Exadata Smart Flash Cache understands the type of I/O requests being made from the database servers and optimizes the flash cache utilization. For instance, an RMAN backup will not flood the flash cache, thus reducing the impact on OLTP applications.
- Exadata Smart Flash Cache breaks up random I/O bottlenecks by caching frequently used data blocks that can increase read IO capacity (IOPS)¹ by as much as 20 times for OLTP applications.

¹ Input/Output Operations Per Second (IOPS)

- InfiniBand 40 Gb/sec high bandwidth network fabric is used for both Exadata storage cells and Oracle RAC to lower I/O and Oracle RAC interconnect latency, respectively.
- Higher memory, CPU and IO capacity for consolidation for both OLTP and analytic workloads onto a single Exadata Database Machine providing savings on floor space which translates into reduced cost.
- Exadata MAA provides validated best practices that are available to be deployed automatically and are described in the *Deploying Oracle Maximum Availability Architecture with Exadata Database Machine* MAA best practices.

This paper assumes you are familiar with Oracle PeopleSoft administration tasks, such as configuring process schedulers, and at least a functional knowledge of Oracle PeopleSoft HR Payroll. A basic knowledge of Oracle Real Application Clusters (Oracle RAC), Oracle Automatic Storage Manager (Oracle ASM), and a working knowledge of SQL is required to carry out some of the recommendations described in this paper. This paper is focused on running PeopleSoft HR Payroll on Exadata and discusses many aspects of Exadata. Therefore, an understanding of the major Exadata components such as compute nodes, storage cells, InfiniBand network and Smart Flash cache will be quite beneficial.

Summary of Best Practices

This section provides a summary of the best practices for implementing PeopleSoft HR onto the Oracle Exadata Database Machine.

Exadata Database Machine

The following recommendations provide links to My Oracle Support IDs and to other MAA white papers for more information:

- Follow My Oracle Support ID [1070954.1](#) for performing Exadata health checks.
- Review My Oracle Support ID [888828.1](#) for recommended software on Exadata Database Machine and Exadata Storage Server 11g Release 2 supported versions.
- Review and follow My Oracle Support ID [1110675.1](#) for Exadata Database Machine Monitoring and Automatic Service Request (ASR).
- Review Exadata testing and patching best practices My Oracle Support ID [1262380.1](#)
- Review and follow the recommendations in the “Deploying Oracle Maximum Availability Architecture with Exadata Database Machine” MAA white paper. This white paper includes other MAA topics that are not in this white paper.

PeopleSoft

The following recommendations include links to more details provided later in this white paper.

- Configure Linux huge pages on each database machine compute node. See My Oracle Support ID [744769.1](#) for further details. Configuring huge pages prevents larger database SGA from being subject to swapping.
- Create a new RAC enabled database or migrate an existing PeopleSoft database. Follow the instructions in the [“Installing or Migrating a PeopleSoft Database to Exadata Database Machine”](#) section in this white paper.
- Implement your PeopleSoft environment using the best practices in the [“Deploying a PeopleSoft Maximum Availability Architecture”](#) MAA white paper.
- Use table and index partitioning appropriate for the specific PeopleSoft application, as described in the [“Table and Index Partitioning”](#) section of this white paper.
- Define multiple run controls within PeopleSoft Process Scheduler, as described in the [“Workload Distribution with Multiple Payroll Run Controls”](#) section of this white paper.
- Configure client TNS load balancing to take advantage of Oracle RAC, as described in the ["Use Load Balancing across Oracle RAC Instances"](#) section of this white paper
- Place the Process Scheduler onto a separate server with sufficient CPU resources and with low network latency connection to the database machine, as described in the ["PeopleSoft Process Scheduler Placement"](#) section of this white paper.

Oracle PeopleSoft MAA on Exadata Database Machine

Figure 1 shows high availability, scale up and scale out, and the disaster-recovery sites.

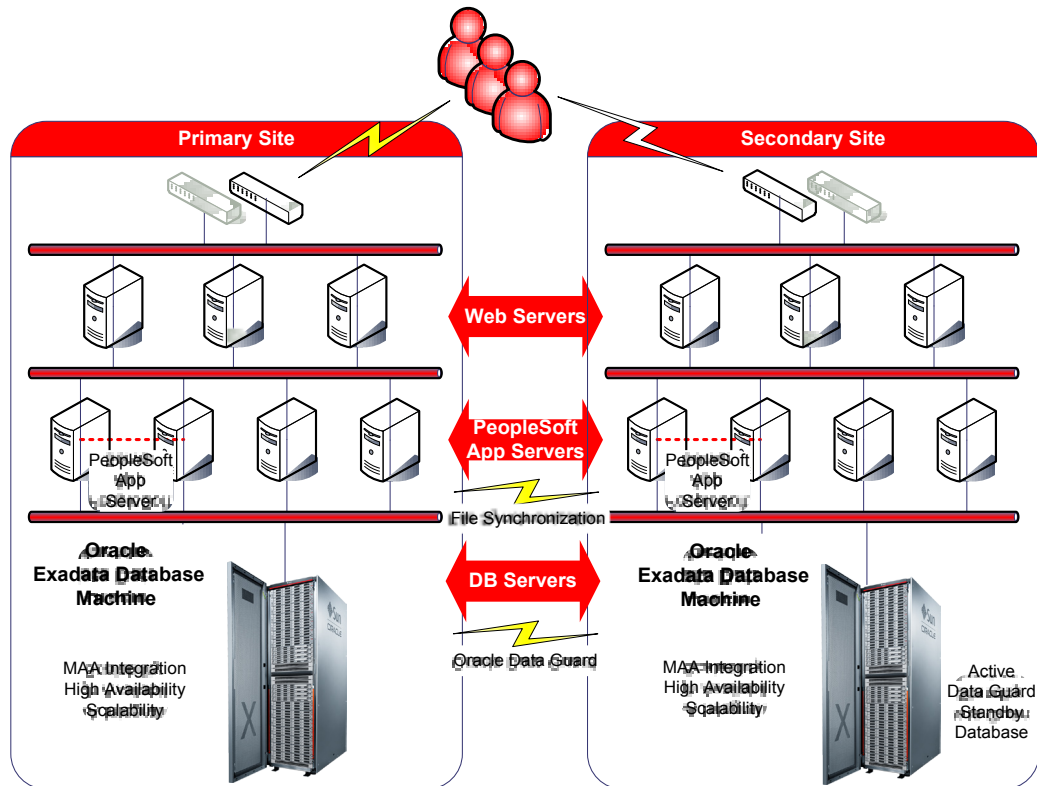


Figure 1: Oracle PeopleSoft High Availability Configuration and Disaster-Recovery Sites

All of the MAA best practices described in the “[Deploying a PeopleSoft Maximum Availability Architecture](#)” white paper can be applied to PeopleSoft applications running on Exadata Database Machine. Some of these MAA best practices are already integrated into the database machine, specifically the use of the Oracle Grid Infrastructure, Oracle RAC, and Oracle ASM. Oracle Data Guard can be implemented to provide data protection at a secondary disaster-recovery site. You can also implement Data Guard locally to provide fast failover in the event of database or entire Exadata Database Machine failure. Oracle Active Data Guard has the auto block repair feature that automatically repairs block corruptions without incurring application downtime and it is transparent to applications.

Also, beginning with release 8.51, Oracle PeopleSoft PeopleTools support includes:

- Oracle Active Data Guard to offload reports to the Active Data Guard database instance.

If the Oracle Active Data Guard database is running on Exadata Database Machine, then those queries and reports can realize all of the benefits with cell offload, cell storage indexes, Smart Scans, and other query optimizations while still functioning as a disaster-recovery database.

- Fast Application Notification (FAN) (starting with release 8.5.0.9) and Transparent Application Failover (TAF) to expedite client failover in the event that the primary database service is lost. For further details about Oracle PeopleSoft MAA, see [“Deploying a PeopleSoft Maximum Availability Architecture” MAA white paper](#).

Getting PeopleSoft to Exadata Database Machine

Getting a PeopleSoft database to Exadata Database Machine can be achieved either by performing a fresh database installation on Exadata Database Machine or by migrating an existing PeopleSoft database to Exadata Database Machine.

You must use the following minimum software versions:

- Oracle Database 11g release 2 (11.2.0.1) or later
- PeopleTools version 8.49 or later

PeopleTools version 8.49 is the minimum version that is supported on Oracle Database 11g Release 2 (11.2). The PeopleSoft application versions are dependent on the version of PeopleTools that is implemented. It is recommended to implement PeopleTools 8.51 as this provides support for MAA features such as FAN and client failover and it also supports Active Data Guard. Consult the PeopleSoft application-specific installation PeopleBooks to determine the minimum supported version of PeopleTools.

Installing a New Database

Performing an Oracle PeopleSoft installation on Exadata Database Machine uses the standard tools provided in the PeopleTools software suite, such as DataMover and Application Designer. The tools and methods for building Oracle database and installing the PeopleSoft schema are the same as for all other Oracle platforms.

On Exadata, it is recommended to first create an empty RAC enabled database using DBCA, then use the PeopleTools tool set to complete the installation of the PeopleTools schemas, and application schemas. The PeopleTools setup scripts use file system path names in the tablespace

creation SQL statements. When instructed to edit these scripts, you will need to modify them to use ASM instead.

Note: Do not install PeopleTools, Application Server, and Process Scheduler directly onto Exadata Database Machine. See Section [“PeopleSoft Process Scheduler Placement”](#) later in this paper for more details.

See the PeopleTools PeopleBook [\[5\]](#) documentation for more information about setup and installation.

Migrating an Existing Database

To migrate an existing PeopleSoft database, several methods are available depending on the source platform of the database being migrated. These methods are discussed in the MAA presentation: [Best Practices for Migrating to Exadata Database Machine](#). It is important to test the migration strategy to ensure a successful migration before implementation on a production system.

If an existing PeopleSoft application is using a version of PeopleTools prior to release 8.49, then the database is also on an older release version. You have these options:

- Upgrade PeopleTools to release 8.49 or higher on the source system and migrate to Exadata using one of the migration strategies described in the [Best Practices for Migrating to Exadata Database Machine](#) presentation. For example, you could use the transportable tablespace method of migration.
- Upgrade PeopleTools to release 8.49 or higher on the source system. Then, use Oracle Data Pump to migrate the data to the new database on Exadata Database Machine.

Case Study: PeopleSoft HR Payroll on Exadata Database Machine

This section describes a PeopleSoft HR payroll batch processing case study that focuses on optimizing the PaySheet, PayCalc, and PayConfirm phases of payroll on Exadata Database Machine. There are other key payroll process phases that are not covered in this paper. This case study is an on-going MAA project for deriving best practices that can be applied to most real-world PeopleSoft environments.

The HR payroll is an iterative process due to various employee benefit changes, business governance, legal requirements, and the need to recalculate pay for each pay cycle throughout the year. Some countries have very complex legal, tax, and pay calculation requirements. Therefore, it is often necessary to rerun the pay calculation phase several times before performing a final run.

Being an iterative process, payroll can take quite a long time to complete during each pay cycle. This paper discusses specific approaches for achieving scalability and reducing overall elapsed times when running payroll on Exadata Database Machine:

- [Controlled Test Environment](#)
- [Payroll Workload Profile](#)
- [Scaling Payroll on Exadata](#)
- [Exadata Smart Flash Cache](#)
- [Process Scheduler Placement](#)

Controlled Test Environment

The Exadata Database Machine test environment includes the following:

Server - Quarter Rack Oracle Exadata X2-2 Database Machine

- 2 compute nodes (SunFire X4170)
 - 2 sockets, 4 cores with Symmetric Multi-Threading (SMT) enabled yielding 16 logical CPUs per compute node
 - 72 GB of physical RAM per compute node
- 3 storage cells (X4275)
 - 2 sockets, 4 cores with SMT enabled yielding 16 logical CPUs per storage cell
 - 24 GB physical RAM per storage cell
 - 12 SATA disks per storage cell
 - 384 GB of Smart Flash Cache per storage cell

Software

- Oracle Database 11g Release 2 (11.2.0.2)
- Oracle PeopleSoft PeopleTools 8.49
Requires 32-bit Oracle client (11.2.0.2)
- Oracle PeopleSoft HCM 9.0

Test Environment Configuration

- Database parameter BUFFER_CACHE_SIZE was set to 32 GB on both Oracle RAC instances

- Database parameter SGA_TARGET was set to 37 GB on both Oracle RAC instances
- Huge pages set for 42 GB on both database machine compute nodes

Workload Profile

The workload used in this case study is based on the PeopleSoft North America HR Payroll benchmarking kit. However, this case study is not a benchmark. Instead, the case study is used as an MAA exercise to derive best practices for optimal performance and scalability.

Only the batch workload portion of the payroll was run, with no online users. The following batch processes were run:

- PaySheet: a COBOL process that generates payroll data worksheets for employees
- PayCalc: a COBOL process that looks at Paysheets, and calculates checks for those employees
- PayConfirm: a COBOL process that takes the information generated by Payroll Calculation and updates the employees' balances with the calculated amounts

These batch processes were executed as if the payroll was run for the first time in the last month of the pay year cycle (the month of December). The first time an actual payroll is run typically has the longest elapsed time excluding any human processing. In this study, the PayCalc process was never run iteratively for any given payroll run. The PeopleSoft North America Payroll kit has 500,000 employees across 128 pay groups all within a single company, and 11 months of pay history.

The HR payroll in this test environment was configured with the setting SINGLE CHECK = "NO". This allows the PayCalc batch process to run multiple run-controls in parallel.

The payroll workload for all phases shares the following common characteristics:

- Sweep style processing in which the process *sweeps* through all employees, with very little data being reread
- Single row processing (fetch, insert and update) works on one row at a time
- High rates of small I/Os due to index scans
- Index bound operations - no full table scans
- High network round trips between the payroll COBOL processes and the database server
- COBOL processes execute business logic with moderate CPU usage

The following sections describe best practice recommendations that can be implemented on Exadata Database Machine. In addition, Exadata provides further performance benefits for Oracle PeopleSoft.

Scaling HR Payroll on Exadata

This case study is focused on PeopleSoft North America HR Payroll running on Exadata. It does not consider other PeopleSoft application suites such as Financials, CRM and EPM. The reader should find after following the recommendations in the example provided below that:

- PeopleSoft HR Payroll scales on RAC
- PeopleSoft HR Payroll does not require RAC instance affinity
- Data partitioning and workload redistribution provides excellent scalability on Exadata

The key to scaling on Exadata Database Machine is to take advantage of Oracle RAC, which is pre-configured on the database machine. It is important to minimize any inter-instance table or index block contention between Oracle RAC instances. Using large monolithic tables and indexes can raise the possibility of higher cluster-wide contention where many processes running on multiple Oracle RAC nodes are contending for the same index leaf blocks and data blocks for tables.

For this reason, most ERP batch processes do not execute across multiple Oracle RAC instances. Many applications which do run on Oracle RAC databases, assign batch workloads to a static Oracle RAC instance to ensure minimum cluster contention. This requires some specific customizations.

One of the objectives for this study was to scale the HR payroll without forcing workload affinity to a specific Oracle RAC instances. The batch processes should be able to run across multiple Oracle RAC instances with consistent performance. To achieve this objective, three key strategies were implemented:

- [Partitioned tables and indexes](#)
- [Redistributed workload with multiple payroll run controls](#)
- [Configured load balancing across Oracle RAC instances](#)

These topics are discussed as best practices in the following sections.

Table and Index Partitioning

Partitioned tables and indexes coupled with directing specific workload processes to work on specific partitions (via partition key) provide an excellent scaling strategy for the payroll batch processes. Payroll is processed by pay groups and each payroll batch process is assigned one or more pay groups. Because our payroll tests have only one company and 128 pay groups, it was logical to range partition by pay group using the PAYGROUP column as the partition key. Each pay group was placed into its own partition and thus, each partitioned table had 128 partitions. Note that not all payroll tables were partitioned in our case study, only those provided by the installation scripts.

Tables and indexes can be partitioned while the application is running but we will be rebuilding indexes as locally partitioned indexes. Therefore, it is recommended that the application be taken offline while the tables and indexes are partitioned.

At a high level, the steps for building partitions are:

1. Create a shadow table with one partition that has the same shape (same columns) as that of the non-partitioned live table
2. Exchange the live table with the new partitioned table
3. Split the data into the final partitions
4. Create the locally partitioned indexes
5. Rebuild any global indexes for the table, if any

The example given below is specific to our MAA case study. However, the partitioning scheme requires knowledge of the range boundaries of the PAYGROUP column. When determining the best partition strategy, it may be required to partition by COMPANY and then by PAYGROUP if your enterprise consists of more than one company. The data will not be uniformly distributed in each partition. How to partition the payroll tables and the best scheme to use is out of scope of this article and will require an analysis by those with functional expertise of PeopleSoft HR Payroll.

Suppose for example, we want to partition the PS_JOB table. Assuming that the table is already created and populated, the table now needs to be partitioned. To partition this table into 128 partitions—one partition per pay group—perform the following steps:

Step 1: Create a shadow version of the PS_JOB_PART table that has only one partition

This table must have the same columns as the PS_JOB table. Here is a snippet of the DDL used to create this table:

```
CREATE TABLE PS_JOB_PART (
  EMPLID VARCHAR2(11) NOT NULL,
  EMPL_RCD SMALLINT NOT NULL,
  EFFDT DATE NOT NULL,
  EFFSEQ SMALLINT NOT NULL,
  PER_ORG VARCHAR2(3) NOT NULL,
  ...
  AUTO_END_FLG VARCHAR2(1) NOT NULL,
  LASTUPDDTTM DATE,
  LASTUPDOPRID VARCHAR2(30) NOT NULL
)
```

```

PARTITION BY RANGE (PAYGROUP)
(
PARTITION JOB_P128 VALUES LESS THAN (MAXVALUE) TABLESPACE PSTABLE)
/

```

Step 2: Use EXCHANGE PARTITION to exchange with the original PS_JOB table

```

alter table PS_JOB_PART
exchange partition JOB_P128
with table PS_JOB
/

```

Step 2 converts the existing PS_JOB table segment into one partition that will be split into all of the remaining partitions in Step 3.

Step 3: Split the partitions on pay groups

The value of the partition key in the following partition split SQL statements are specific pay groups for example: 102, 103, 430, and so on. Each pay group should be placed into its own partition with no overlap. Thus, it is important to specify the range key at pay group boundaries. If any of the existing pay groups are merged or split, then the partitions can be merged or split at their new boundaries. If new pay groups are to be added, then if possible, this should be done by splitting the last partition (MAXVALUE). If new pay groups are added with values between the existing ones, it will be necessary to merge and re-split existing partitions into the new partitions. Note that you should have N+1 partitions where N is the number of pay groups.

```

ALTER TABLE PS_JOB_PART split partition JOB_P128 at (102) into (
partition JOB_P00 tablespace pstable, partition JOB_P128 )
parallel 8;

ALTER TABLE PS_JOB_PART split partition JOB_P128 at (103) into (
partition JOB_P01 tablespace pstable, partition JOB_P128 )
parallel 8;

...

ALTER TABLE PS_JOB_PART split partition JOB_P128 at (430) into (
partition JOB_P126 tablespace pstable, partition JOB_P128 )
parallel 8;

ALTER TABLE PS_JOB_PART split partition JOB_P128 at (431) into (
partition JOB_P127 tablespace pstable, partition JOB_P128 )
parallel 8;

```

Note the use of the PARALLEL clause in the SQL statement. This clause scales the partition split operation and completes in a much shorter time period. The parallel process makes use of the wide I/O bandwidth and the InfiniBand network on Exadata Database Machine for scalability.

Step 4: Build local partitioned indexes

Build indexes on the PS_JOB table with local partitions. For example:

```
CREATE INDEX PS0JOB ON PS_JOB (PER_ORG,
    EMPLID,
    EMPL_RCD,
    EFFDT DESC,
    EFFSEQ DESC)
LOCAL (
    PARTITION JOB_P00 TABLESPACE PSINDEX,
    PARTITION JOB_P01 TABLESPACE PSINDEX,
    PARTITION JOB_P02 TABLESPACE PSINDEX,
    ...
    PARTITION JOB_P127 TABLESPACE PSINDEX,
    PARTITION JOB_P128 TABLESPACE PSINDEX
)
PCTFREE 10 PARALLEL NOLOGGING
/
ALTER INDEX PS0JOB NOPARALLEL LOGGING
/
```

Step 5: Build global indexes

Note: Build a global index only if the required index would not be appropriate as a locally partitioned index.

For PS_JOB, the PS_JOB unique index is a global index. For example:

```
CREATE UNIQUE INDEX PS_JOB ON PS_JOB (EMPLID,
    EMPL_RCD,
    EFFDT DESC,
    EFFSEQ DESC)
TABLESPACE PSINDEX
PCTFREE 10
PARALLEL NOLOGGING
/
ALTER INDEX PS_JOB NOPARALLEL LOGGING
/
```

Executing the above steps to partition the data should take tens of minutes on Exadata Database Machine, instead of hours. In our testing, partitioning 9 payroll tables including all index builds took less than 40 minutes from start to finish on a 300 GB database. These 9 tables contained the bulk of the payroll data.

Workload Distribution with Multiple Payroll Run Controls

After partitioning the data, it is important to distribute the workload as evenly as possible to achieve good scalability. If the payroll processes were assigned pay groups in a simplified round-robin fashion, some of the streams would finish sooner while others would take much longer to complete.

The partitioning strategy provides the division of the pay group data to specific partitions. Because the payroll data is not uniformly distributed across all pay groups (partitions), we need to devise a scheme to distribute the workload evenly. Doing so allows for better scaling and higher throughput.

In this case study, there are four types of pay groups in the following ranges:

- 101 – 132 has 10,948 jobs within this range
- 201 – 232 has 6,256 jobs within this range
- 301 – 332 has 3,128 jobs within this range
- 401 – 432 has 3,128 jobs within this range

The above distribution shows that pay group type in the 100 series across 32 pay groups has 10,948 jobs. This range has substantially more jobs than the 200, 300, or 400 pay group series, illustrating an uneven distribution. You can define run controls within PeopleSoft to distribute the work across multiple payroll work streams. To achieve a near equal workload distribution, 64 run controls were defined. These were then divided into two groups:

- Run control streams 1 to 32 were assigned all pay groups in the 100 series (101 to 132).
- Run control streams 33 to 64 were assigned the remaining 200, 300, and 400 series pay groups.

In the first group, run control 1 was assigned pay group 101 while in this second group, run control stream 33 was assigned 3 pay groups: 201, 301, and 401. Note that no single run control shares pay groups with any other run control. The defined run controls align with our partitioning scheme above. Two key benefits are achieved with this scheme:

- Each payroll process works exclusively on a set of partitions
- Each payroll process can run on any given Oracle RAC instance with no affinity required

These run controls must be defined for all phases of payroll: PaySheet, PayCalc, and PayConfirm.

PayCalc and Run Control

The PayCalc process can issue a single check that covers all of an employee's calculations for multiple jobs in the same organization. This affects payroll performance in the following ways:

- If PayCalc issues a single check then it should be run serially. Run controls would be run one at a time and one after the other for PayCalc.
- If PayCalc calculates individual check then it can be run in parallel. You can start all run controls at the same time for PayCalc. This is the preferred way for achieving high scalability and significantly reducing elapsed run time.

To enable PayCalc to run parallel processes with multiple run controls, you must configure the HR Payroll with “SingleCheck” set to “NO” when setting up Payroll.

For further details on the SINGLE_CHECK feature, please refer to the PeopleSoft Enterprise Payroll for North America PeopleBook at <http://www.oracle.com/pls/psft/homepage>

You will need to locate the specific PeopleBook: “Handling Employees with Multiple Jobs in the same Organization” for the PeopleSoft HRMS version you have implemented.

Use Load Balancing across Oracle RAC Instances

To scale across Oracle RAC instances, ensure that the TNS alias connect string specifies LOAD_BALANCE = YES. On Exadata Database Machine, the TNS alias connect string should specify the Single Client Access Name (SCAN)² for the HOST parameter.

To use SCAN, PeopleTools must use Oracle Database 11g release 2 (11.2) or higher database client software installation. When using SCAN, the Grid Infrastructure on Exadata distributes the connections across all Oracle RAC instances that support the service specified in the connect descriptor.

The following example shows a TNS connect string:

```
PSFT =  
  (DESCRIPTION =  
    (SDU=32767)
```

² For additional information about SCAN, consult the [Oracle Real Application Clusters 11g Release 2 Overview of SCAN whitepaper](#).

```

    (ADDRESS = (PROTOCOL = TCP) (HOST = sclcz-scan2.us.oracle.com) (PORT =
1521) )
    (LOAD_BALANCE = yes)
    (CONNECT_DATA =
        (SERVER = DEDICATED)
        (SERVICE_NAME = PSFT)
    )
)
)

```

Results of Partitioning, Workload Distribution, and Load Balancing

The following table shows the percentage distribution of the top wait events from our testing of a two-node Oracle RAC cluster on the Oracle Exadata Database Machine:

| DISTRIBUTION OF WAIT EVENTS | | | |
|-----------------------------|----------|---------|------------|
| WAIT EVENT % | PAYSHEET | PAYCALC | PAYCONFIRM |
| DB CPU % | 50.97 | 44.44 | 68.54 |
| I/O % | 36.88 | 47.48 | 24.02 |
| RAC Cluster % | 13.15 | 8.54 | 8.30 |

Because most of the wait time is spent on either DB CPU or I/O, workload can be scaled across Oracle RAC instances and perform more productive work. There are minimal cluster waits across all three phases of the payroll processes. This is achieved by not having any of the run controls overlap pay groups with another run control. Therefore, load balancing across Oracle RAC instances allows scaling without regard to which Oracle RAC instance each payroll process runs on.

Exadata Smart Flash Cache

Both the Exadata Database Machine X2-2 and X2-8 come with 384 GB of Smart Flash Cache on each storage cell. The total amount of flash cache depends on the number of storage cells within the Exadata Database Machine rack. The quarter rack used in this study has 1.01 TB of flash cache across three storage cells.

Exadata Smart Flash Cache monitors the I/O requests and intelligently optimizes the flash cache. For example, random I/O bottlenecks can be broken up by caching frequently used data blocks that can increase read IO capacity. It can prevent flash cache flooding by intelligently managing the type of I/O's. For instance, RMAN will not impact OLTP applications by flooding the flash cache while performing a backup.

Typically, I/O latency for SAN and NAS storage arrays (assuming SCSI, fiber or GigE respectively) averages 6 to 8 ms. There are two factors that drive I/O performance: IOPS (I/O operations per second) and I/O bandwidth. Testing shows that PeopleSoft payroll benefits from the Exadata Smart Flash Cache in the storage cells. PeopleSoft takes advantage of the wide I/O bandwidth and higher IOPS capacity offered by the Smart Flash Cache on the Exadata Storage Cells.

No special tuning or application modifications are required to explicitly use the flash cache.

Flash Cache Results

The payroll tests with flash cache have shown a reduction of I/O latency from an average of 7 ms to 3 ms, with a 30% hit ratio. This improved the overall throughput and resulted in a reduced elapsed time of 15%. Because the HR payroll is a *sweep* style workload, few tables are re-read and most of the data is processed once.

Application performance will vary but you should see improvement in overall performance with Exadata Smart Flash Cache.

PeopleSoft Process Scheduler Placement

Our testing shows that 32 concurrent PayCalc and PayConfirm processes can consume between 25 to 40% of the CPU resources on an eight core server. This is because much of the business logic is in the COBOL application processes that run payroll.

The database machine is tuned for best performance and scaling of the database. Installing application components onto the compute nodes of the database machine will eventually lead to CPU resource contention and lower throughput when trying to achieve high scalability. The objective is to balance CPU resources between the database server instances and the PeopleSoft payroll COBOL processes, in order to scale both.

It is recommended to place all application components: application server and process scheduler onto separate middle tier servers outside of Exadata. A discussion on how to overcome network latency that can impact performance follows.

As mentioned previously, much of the business logic for the payroll process resides in the COBOL processes in the application tier. This design results in a high amount of network round trips, both to and from the database.

The following may indicate some performance issues:

- When the database idle wait-event time becomes significant (several thousands of seconds as reported in AWR). This is calculated by summing “Total Wait Time Seconds” for both SQL*Net message to client and SQL*Net messages from client wait events.

This may point to a possible performance bottleneck with the batch processes due to CPU resource contention, network bandwidth and latency constraints, or both.

- When "SQL*Net round trip" and "SQL*Net message from / to client" wait events are returned, with wait times per round trip registering in the order of several tenths of milliseconds (0.3 ms or higher)..

This case study tested the following configurations:

- Process scheduler placed on a separate middle tier connected via GigE with an average latency of 0.264 ms latency
- Process scheduler placed on a separate middle tier connected via GigE with an average latency of 0.121 ms latency

Results for Process Scheduler Running On a Separate Server

Placing the process scheduler onto a separate server requires sufficient CPU resources to support the COBOL process side of the workload. In addition, the network links for these batch processes play a significant role in throughput and performance. As mentioned earlier, these COBOL processes are very chatty. Therefore, higher network latency will prolong the processing elapsed time. MAA testing on a separate server was configured with two network topologies: GigE two hops away from the Exadata server with 0.264 ms network latency and GigE single-hop away from the Exadata server with 0.121 ms network latency.

The pie charts in Figure 3 and Figure 4 illustrate the impact that network latency has on the payroll processing. The pie charts compare the differences in the percent of time spent in the client between the two network configurations.

In Figure 3, the pie chart shows that for a two-hop network where the latency is 0.264 ms, of the total elapse time, we spend 43% in the client.

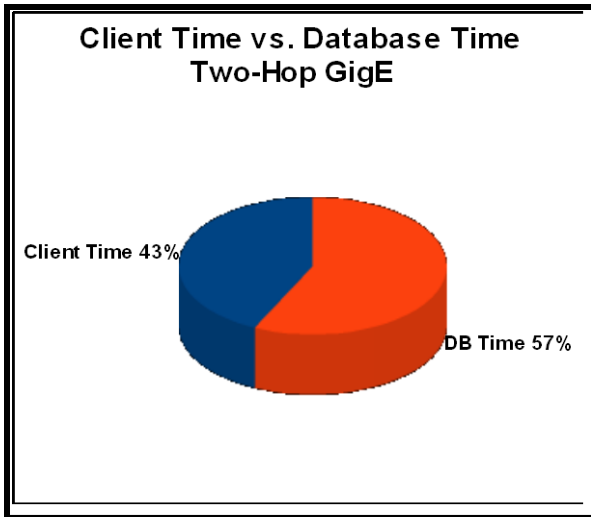


Figure 2: CPU Statistics for Exadata Database Machine Compute Nodes

In Figure 4, the pie chart shows that when configured with the single-hop network (0.121 ms latency), there is a significant reduction of over 50% of the client time. Only 27% of the overall time was spent in the client while the remaining 73% was spent in the database.

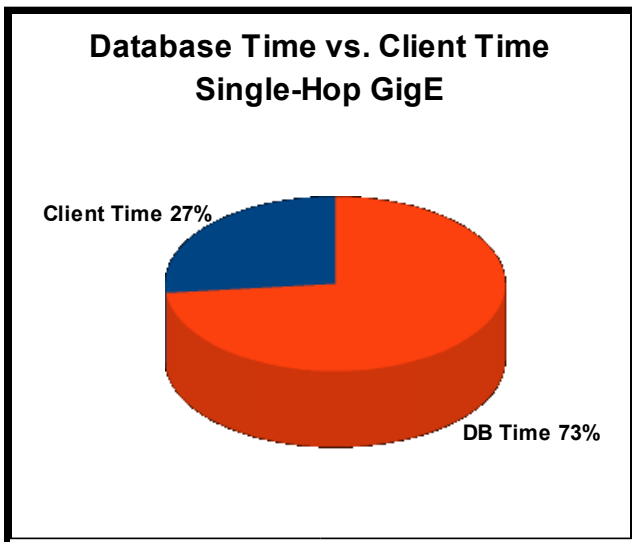


Figure 3: CPU Statistics for Exadata Database Machine Compute Nodes

Another way to illustrate this is shown in the bar graph in Figure 5. The bar graph compares the client side time (in seconds) for each network topology. The same workload was used in both cases. The same 32 concurrent payroll streams processes 500,000 employees. The “DB Time” is the same for both tests. The higher latency network causes higher network wait times. The bar graph shows that for the 0.264 ms network, the time spent on the client side was approximately 77,000 seconds (aggregated across all 32 streams). In the 0.121 ms latency network, the time dropped to approximately 37,000 seconds, which is a 50% reduction in network wait time.

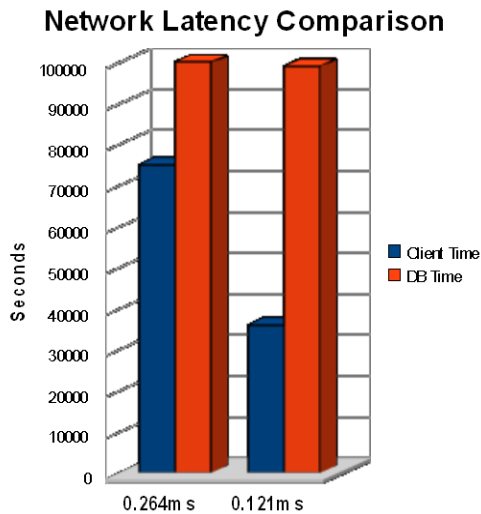


Figure 4: CPU Statistics for Exadata Database Machine Compute Nodes

Based on these tests, it is recommended to place the Process Scheduler onto a separate server with a low-latency network. This allows the payroll processing to both scale up and scale out.

PeopleSoft has the flexibility to configure more than one Process Scheduler where each may be deployed on its own server. The best practice is to configure each Process Scheduler with its own queue for this purpose. For high availability, configure a “master” Process Scheduler such that when any one of the Process Schedulers fail, the master scheduler can assume the workload.

Conclusion

Customers considering running their ERP OLTP applications on Oracle Exadata Database Machine will find that their applications will benefit in higher availability and performance. Changes to the application are minimal, as illustrated in our case study. Such changes will be specific to the application being deployed.

It is recommended to keep the application-tier components separate from Exadata Database Machine for achieving higher scalability. The application-tier server should have sufficient CPU capacity for the COBOL process and use a high bandwidth, low latency network to connect to the Database Machine, such as GigE, 10GigE, or InfiniBand. This configuration offers high scalability and easier performance management for "chatty" batch application processes

Customers such as CBA are finding that their critical PeopleSoft Financials and other applications are seeing excellent benefits running on Exadata. Because customers no longer need to install and configure the hardware and software, the overall time required to move to Exadata Database Machine is reduced. Teams are freed up to test and validate post-installation and/or migration tasks. Once running in production, online users see stable performance and batch processes enjoy significantly reduced elapsed run times.

References

1. Oracle Maximum Availability Architecture Web site
<http://www.otn.oracle.com/goto/maa>
2. *Oracle Database High Availability Overview (Part #B14210)*
<http://otn.oracle.com/pls/db111/db111.toc?partno=b28281>
3. *Oracle Database High Availability Best Practices (Part B25159)*
<http://otn.oracle.com/pls/db111/db111.toc?partno=b28282>
4. *Oracle Exadata Bundle Patch Support Note:*
[My Oracle Support ID 888828.1 - Database Machine and Exadata Storage Server 11g Release 2 \(11.2\) Supported Versions](#)
5. *Oracle PeopleSoft PeopleBooks*
<http://www.oracle.com/pls/psft/homepage>



Oracle PeopleSoft on Oracle Exadata Database
Machine

February 2011

Author: Darryl Presley

Contributing Authors: Lawrence To, Lyn Pratt,
Richard Exley, Ray Dutcher, Marc Varennes,
Michelle Lam, Viv Schupmann

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com



Oracle is committed to developing practices and products that help protect the environment

Copyright © 2011, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.