Oracle GRID Center

Fujitsu SPARC Enterprise

Migrating from Oracle9*i* Database to Oracle Database 11*g*

Performance Comparison against Older-version System using Oracle Real Application Testing

Creation Date:	December 1, 2008
Last Update:	July 1, 2009
Version:	1.1

ORACLE

FUJITSU

THE POSSIBILITIES ARE INFINITE

Contents

1. Introduction

Since concluding an OEM agreement in 1989, Oracle Corporation Japan (hereinafter called "Oracle Japan") and Fujitsu Limited (hereinafter called "Fujitsu") have engaged in various joint efforts to provide solutions that deliver safety and security to clients, including system building, joint verification, and post-installation support.

In November 2006, Oracle Japan established the Oracle GRID Center (http://www.oracle.co.jp/solutions/grid_center/index.html) featuring cutting-edge technologies designed to help create next-generation grid-based business solutions and optimize corporate system infrastructures. Lending its full support to establish the Oracle GRID Center, Fujitsu is currently active in joint technical verification efforts with Oracle Japan at the Oracle GRID Center, based on its server and storage products.

To maximize the benefits of migrating to the newest Fujitsu's SPARC Enterprise UNIX servers and the latest version of Oracle Database and to design countermeasures against potential problems that may arise during migration, Oracle Japan and Fujitsu have recently undertaken various verification testing at the Oracle GRID Center. These tests sought to confirm the validity of Oracle Real Application Testing using Fujitsu's SPARC Enterprise M3000 and Oracle Database 11*g*. The results of the verification tests are reported below.

2. Introducing the Products and Product Features

The products used in our verification testing and their features are described below.

2.1. Fujitsu SPARC Enterprise M3000

IT systems now play a critical role for businesses. As the data volumes handled by corporations continue to grow, many companies have begun installing and using various purpose-specific servers, including web servers, application servers, and database servers. Meanwhile, corporations have been compelled to address various IT systems issues, including the cost increases incurred to improve processing performance and install additional servers. As part of these efforts, companies are seeking to integrate and optimize servers to reduce costs or improve efficiency.

Fujitsu's SPARC Enterprise series offers two groups of product. The SPARC Enterprise M9000, M8000, M5000, and M4000 offer superb mainframe-class reliability and are ideal for mission-critical applications, while the SPARC Enterprise T5440, T5240, T5220, T5140, T5120, T2000, and T1000 boast high throughput performance for use as web front-end and application servers.

Mid-size companies, which typically install entry-class UNIX servers as backbone servers, have expressed the need for products capable of handling a wide range of mission-critical corporate tasks, including databases and batch processing tasks.

- **Inheriting the mission-critical features of the midrange and high-end server family**

In response to these client needs, an entry-level model SPARC Enterprise M3000 offering the high performance and reliability required for mission-critical systems, this product can be used to handle various applications in addition to front-end operations.

The SPARC Enterprise M3000, a lineup of mission-critical servers, leverages the SPARC64 VII processor used in the SPARC Enterprise M4000 or later-end models to offer a multi-threading environment with up to four cores and eight threads and maximum capacity of 64 GB of memory. In its standard configuration, this compact, 2U-size unit offers one SAS port and four PCI Express slots while delivering top-tier performance at an entry-level price point to handle business ranging from database server to application server functions.

- **Delivering the high reliability required for mission-critical tasks**

The SPARC Enterprise M3000 inherits the acclaimed reliability of the higher-end models, M4000 through M9000. High reliability is built into the system at all levels from LSI to unit and system levels.

For example, at the LSI level, each CPU incorporates some 3,400 checkers for failure detection while offering strengthened hardware memory patrol functions to ensure data integrity.

The area of the processor most prone to malfunctions is cache memory. The SPARC Enterprise M3000 offers extensive data protection features. All secondary cache memory is protected by ECC. When a 1-bit error occurs frequently, dynamic degradation is implemented step-by-step in cache way units and CPU core units. As a result, the system can continue operating even with a single CPU. This minimizes further performance degradation.

In the unit level, the SPARC Enterprise M3000 offers outstanding fault tolerance performance by incorporating redundancy of disks, fans, power supply units, etc. and by supporting hot-swap and dynamic degradation. It also features status monitoring and malfunction notification functions. In the system level, the SPARC Enterprise M3000 offers cluster system support and storage/network redundancy to ensure uninterrupted operation.

- **Ecology server offering low power consumption, quiet operation, and space-saving footprint**

The SPARC Enterprise M3000 is a “Super Green” product that meets Fujitsu's own environmentally-aware product standards. The compact 2U (Unit) dimensions of the SPARC Enterprise M3000 reduces installation footprint and weight by 50% from the PRIMEPOWER 450 (4U). Power consumption is 470 W (at 100 V) maximum, 57% less than the PRIMEPOWER 450. With improved performance, the SPARC Enterprise M3000 can reduce annual CO2 emissions by up to 65%. At typical server sites with ambient temperatures of 25°C, operating noise is extremely low – 47 dB. Compared to quad-core servers manufactured by other companies, the SPARC Enterprise M3000 is the quietest and consumes the least power, which is the most ecology server.

To minimize power consumption, the SPARC Enterprise M3000 uses an air duct system for cooling. The components that generate the most heat such as CPUs and memory receive maximum cooling. There are two cooling zones inside the casing. One zone is cooled by the main unit fan, while the other zone is cooled by the power supply unit fan, a configuration that improves cooling efficiency. Cooling fans can operate at one of nine discrete steps. The SPARC Enterprise M3000 is 3 dB quieter than the PRIMEPOWER 250/450. With various innovative features and technologies, the SPARC Enterprise M3000 realizes energy conservation and quiet operation.

The SPARC Enterprise M3000 in its standard configuration incorporates a virtualization technology called Solaris Containers. Despite being an entry-level model, it is capable of consolidating resources through server integration to improve system efficiency.

Boasting high performance, reliability, and ecology, the SPARC Enterprise M3000 helps clients optimize system performance and expand business opportunities.

[Features of SPARC Enterprise M3000]

- Highest processor core performance in the SPARC/Solaris entry class
- Entry-class model inherits the high-reliability technologies of midrange models.
- “Green Policy Innovation” product offers energy- and space-saving features.

2.2. ETERNUS Storage System

Fujitsu offers a plethora of SAN-compatible disk arrays to meet a wide range of needs, including the ETERNUS8000 enterprise disk array with the world's largest capacity of 2.72 PB (petabytes), an ETERNUS4000 midrange disk array offering high cost-performance, and an ETERNUS2000 entry-level disk array with space-saving, energy-saving, and quiet operating characteristics, and an ETERNUS DX60/DX80 new entry-level disk array improving a storage capacity and Green IT features as the successor to an ETERNUS2000..

Assuring the highest levels of reliability and scalability at all levels from components to systems, the ETERNUS series realizes storage integration based on SAN. It also supports high-speed copy and data encryption functions for enhanced security and reliable data protection. The ETERNUS series also meets global environmental product standards and provides energy-efficient operations tailored to system requirements.

The ETERNUS disk arrays store important corporate data both reliably and efficiently and provide optimal storage solutions to significantly improve investment efficiency.

- **ETERNUS2000 Model 200**

The ETERNUS2000 Model 200 is an entry-level disk array that measures approximately 9 mm (2U) in thickness and mounts in a standard rack. Thanks to the reduced number of parts, this model curtails power consumption by about 40% compared to previous device of the ETERNUS4000 Model80 and offers the quiet operating sound of 42 dB (under the temperature of 29°C and lower).

This model incorporates the same high-reliability design as Fujitsu's midrange models. Main components feature redundant design, and the Redundant Copy function enables preventive replacement of a disk while ensuring data redundancy. The Advanced Copy Function*[1] that performs high-speed online data backup can be used to duplicate data immediately at a desired time.

Despite being positioned as an entry-level model, the ETERNUS2000 Model 200 offers a large capacity of up to 72 TB (terabytes) to achieve cost-effective data backup and to store ever-growing stores of corporate data.

[Features of ETERNUS2000 Model 200]

- Entry-level disk array with space-saving, energy-saving, and quiet operating characteristics
- Incorporates the same high-reliability design as midrange models.
- Supports Advanced Copy Function for high-speed data backups.
- Excellent scalability providing a maximum storage capacity of 72 TB

Besides, the ETERNUS DX80 expands a maximum storage capacity of 120TB. It also has achieved a further 8% reduction in power consumption and up to 6dB(under the temperature between 25℃ and 30℃) less noise output by improved fan rotation control compared to an ETERNUS2000.

2.3. Oracle Database 11*g*

As the leading relational database management system for business, Oracle Database has always incorporated leading-edge technologies to address information management issues in companies, government agencies, and other customers around the world. Fundamental features such as platform portability, read consistency that ensures data integrity, complete row-level locking without constraint, Real Application Clusters to enable both scalability and high availability, and support for a rich array of data types including XML data are integral to the architecture of Oracle Database, and are the basis of overwhelming technical advantages.

Oracle Database 11g is the latest version of Oracle Database and is a "Real Customer Release" that puts customer value first. This release evolves the base of leading-edge technologies while providing many sought-after features to address customer needs aggressively and resolve issues common to the modern IT environment. Some key features of Oracle Database 11g will be described in more detail below.

- **Minimize system failures while reducing Total Cost of Ownership (TCO)**

System failures can be extremely expensive, affecting both the bottom line cost and top line revenue of a typical enterprise. In some cases, addressing system failures can account for as much as 80% of IT costs. Oracle Database 11g makes available a solution

[1] This requires the optional ETERNUS SF Advanced Copy Manager software.

that can drastically reduce the potential for system failures, thereby decreasing TCO. Using Oracle Real Application Testing, customers can implement database changes with minimum risk by performing more accurate and comprehensive testing with fewer person-hours of effort.
Oracle Enterprise Manager Database Control is the GUI-based database management tool that provides an integrated view of overall system status and performance. It can minimize system failures, reduce TCO, and enable cost-effective system management, because it builds on its database management abilities by adding features that enable fault detection, diagnosis, and recovery. In addition, Database Control includes automated performance tuning and diagnosis features that bring us ever closer to the ideal of a self-managed database.

- **Enable higher availability with lower cost by sharing IT infrastructure**

In many companies, the IT infrastructure has grown organically into a dispersed and diverse set of systems and applications. This adds complexity that increases TCO because such heterogeneous environments often require separate management for each set of infrastructure, and because it is difficult to take advantage of economies of scale when trying to ensure high availability and meet service-level agreements.

Oracle Real Application Clusters (RAC) is Oracle's database cluster technology that enables high availability, high performance, scalability, and the efficient use of IT resources through workload management features in a grid computing environment
.
Automatic Storage Management (ASM) enables the virtualization and efficient use of storage resources, and can reduce workload in storage management.

Oracle Data Guard is a feature of Oracle Database that enables use of a standby database for disaster recovery. Starting with Oracle Database 11g, the Oracle Active Data Guard feature allows customers to use the standby database for queries, providing a combination of high availability and more efficient use of resources while off-loading query traffic from the primary production database.

- **Manage massive amounts of data with lower cost and higher performance / promote the utilization of information**

The proliferation of data throughout the enterprise, combined with data archive requirements of various regulatory regimes, has caused data management headaches in many companies. The increase in storage needed to meet data management and archive needs results in higher costs, and the need to access data for a broad spectrum of uses is more difficult when that data is spread both geographically and organizationally.

Oracle Advanced Compression is a breakthrough feature that can compress any type of data while actually increasing database query performance.

Oracle Partitioning is an option that many customers find essential because it enables them to increase database performance for all types of operations.

Oracle SecureFiles is a completely new implementation of Large Objects (LOBs) which can handle documents, pictures, maps and various unstructured data with very high performance. This feature is freely available in Oracle Database 11g, and allows customers to manage huge amounts of data with lower cost for both data warehousing and OLTP.

This combination of features promotes the utilization and integration of information throughout the enterprise.

- **Strengthen compliance and risk management with database security**

Information assets are often crucial to the success of a business. That is why today's demands of compliance and risk management make system security more important than ever. Oracle Database 11g includes features to help lock down a database environment and meet compliance and security needs while also fulfilling the needs of applications and database clients.

Oracle Database Vault can prevent over concentration of database privileges, ensuring that those who manage your systems do not receive inappropriate access to sensitive information. This lets customers align flexible access control with business needs.

For highly-confidential data, Transparent Data Encryption can automatically encrypt and store data without the need for manual intervention. Database Auditing can record audit trails flexibly and efficiently, ensuring that all data access is logged for compliance or security purposes.

Oracle Database has obtained not only ISO/IEC 15408 (Common Criteria), FIPS 140-2 certification and various international standards but ISO／IEC 15408 (EAL4) for the first time as a RDBMS product.

Systems of all sizes can enjoy the benefits of the many new features and capabilities in Oracle Database 11g. Its high performance, fault diagnostics, rich application development environments and automated management features contribute to build IT system infrastructure for small to mid-range systems. For larger and mission critical systems, the management features, high availability, security, and data warehousing make Oracle Database 11g the best in a long line of leading database releases. Oracle Database 11g can your competitive standing by bringing all the benefits of grid computing to your IT system infrastructure.

2.4. Real Application Testing Option

Today, enterprises have to make sizeable investments in hardware and software to roll out infrastructure changes. For example, a data center may have an initiative to move databases to a low cost computing platform, such as Oracle Enterprise Linux. Traditionally, this would require the enterprise to invest in duplicate hardware for the entire application stack (including web server, application server, and database) to test their production applications. Organizations therefore find it very expensive to evaluate and implement changes to their data center infrastructure.

Further, in spite of the extensive testing performed, unexpected problems are frequently encountered when a change is finally made in the production system. This is because test workloads are typically simulated and are not accurate or complete representations of true production workloads. Data center managers are therefore reluctant to adopt new technologies and adapt their businesses to rapidly changing competitive pressures.

The Oracle Real Application Testing option of Oracle Database 11g Enterprise Edition delivers two new functions, Database Replay and SQL Performance, which provide the most appropriate solution for these problems.

NOTE: For more information on the Oracle Real Application Testing, refer to the Database Manageability and Real Application Testing page on the Oracle Technology Network (URL:http://www.oracle.com/technology/products/manageability/database/index.html).

2.4.1. Database Replay

Database Replay provides DBAs and system administrators the ability to faithfully, accurately and realistically rerun actual production workloads, including online user and batch workloads, in test environments.

By capturing the full database workload from production systems, including all concurrency, dependencies and timing, Database Replay enables you to realistically test system changes by essentially recreating production workloads on the test system – something that a set of scripts can never duplicate. With Database Replay, DBAs and system administrators can test

- Database upgrades, patches, parameter, schema changes, etc.
- Configuration changes such as conversion from a single instance to RAC, ASM, etc.
- Storage, network, interconnect changes
- Operating system, hardware migrations, patches, upgrades, parameter changes

The benefits of Database Replay will be described in detail below.

Lower test infrastructure cost
DBAs now have a test infrastructure at their disposal to test their changes without the overhead of having to duplicate an entire application infrastructure. Database Replay does not require the set up overhead of having to recreate a middle-tier or a web server tier. Thus, DBAs and system administrators can rapidly test and upgrade data center infrastructure components with the utmost confidence, knowing that the changes have truly been tested and validated using production scenarios.

Faster deployment
Another major advantage of Database Replay is that it does not require the DBA to spend months getting a functional knowledge of the application and developing test scripts. With a few point and clicks, DBAs have a full production workload available at their fingertips to test and rollout any change. This reduces testing cycles from many months to days or weeks and brings significant cost savings to businesses as a result.

2.4.2. SQL Performance Analyzer

Changes that affect SQL execution plans (the access path chosen by the optimizer to execute SQL statements) can severely impact application performance and availability. As a result, DBAs spend enormous amounts of time identifying and fixing SQL statements that have regressed due to system changes. SQL Performance Analyzer (SPA) can predict and prevent SQL execution performance problems caused by environment changes.

SQL Performance Analyzer provides a granular view of the impact that environment changes will have on SQL execution plans and statistics. It does this by running the SQL statements one at a time before and after changes are made to the system. SQL Performance Analyzer generates a report outlining the net benefit on the workload due to the system change as well as the set of regressed SQL statements. For regressed SQL statements, SPA provides execution plan details along with tuning recommendations.

SQL Performance Analyzer is well integrated with existing SQL Tuning Set (STS) and SQL Tuning Advisor, and SQL Plan Management functionality. SQL Performance Analyzer completely automates and simplifies the manual and time-consuming process of assessing the impact of change on extremely large SQL workloads (the hundreds of thousands of SQL statements). DBAs can use SQL Tuning Advisor to fix the regressed SQL statements in test environments and generate new plans. These plans are then seeded in SQL Plan Management baselines and exported back into production. Thus, using SQL Performance Analyzer, businesses can validate with a high degree of confidence that a system change to a production environment in fact results in net positive improvement at a significantly lower cost.

Examples of common system changes for which you can use the SQL Performance Analyzer include:

- Database upgrade, patches, initialization parameter changes
- Configuration changes to the operating system, hardware, or database
- Schema changes such as adding new indexes, partitioning or materialized views
- Gathering optimizer statistics
- SQL tuning actions: for example, creating SQL profiles

Choosing the right solution – Database Replay and SQL Performance Analyzer – helps DBAs absorb and manage change efficiently. Database Replay is designed for concurrency and throughput testing. Meanwhile, SQL Performance Analyzer is designed for unit testing SQL and fixing the SQL regressions found in testing. Integrating SPA with SQL Plan Management, SQL Profiles, and/or stored outlines gives DBAs a robust set of tools to address potential performance issues. SQL Performance Analyzer and Database Replay are thus complementary solutions that are part of the same Real Application Testing option. We strongly recommend that customers use both these solutions to assess impact of change. Oracle Real Application Testing makes it easy for database administrators to manage and execute changes that are critical to the business and do it all at lower risk.

3. Migration Need

Continued use of older-version databases on an OS operating on aged hardware entails the following risks:

- Inadequate processing performance due to hardware limitations and expanded range of corporate tasks
- Rise in operating costs due to repeated addition of equipment and increased power consumption for server rooms
- Increased maintenance risks due to a lack of required technical support
- Difficulty in obtaining necessary technical information

Users often think only of the risk caused by upgrading seemingly stable operating existing systems, but a seemingly stable system operates in an ever-changing and unstable business environment that may introduce security risks, compliance requirements, and technological changes that can impose risks from outside the IT environment itself. It is important to recognize these ongoing operational risks when assessing the need for system upgrades.

Moreover, hardware maintenance contracts and OS support periods can expire from time to time over the system life cycle. Systems not upgraded for many years may complicate system migration or result in compatibility problems that add to the difficulty and risk of system migration. For example, the Oracle version in current use may be incompatible with a newly installed OS version.

To eliminate or resolve such problems, we recommend migrating to a system based on the newest SPARC/Solaris servers, latest Solaris OS, and the latest version of Oracle.

3.1. Benefits of Migration

Replacing existing Solaris servers with the newest SPARC/Solaris servers solves various problems associated with expiring hardware and OS maintenance contracts, while multi-core/multi-thread CPUs and the latest input/output interfaces such as PCI Express and SAS improve system performance and stability.

Additionally, the SPARC Enterprise M3000 offers both improved system and enhanced environmental performance over previous models. Migrating to the SPARC Enterprise M3000 dramatically improves performance and cuts electrical costs significantly.

In the verification tests described below, we replaced the PRIMEPOWER 250 with the SPARC Enterprise M3000 and upgraded Oracle9*i* Database to Oracle Database 11*g*. This change improved throughput for OLTP application execution by a factor of 3.5 and cut response time to roughly 1/6.

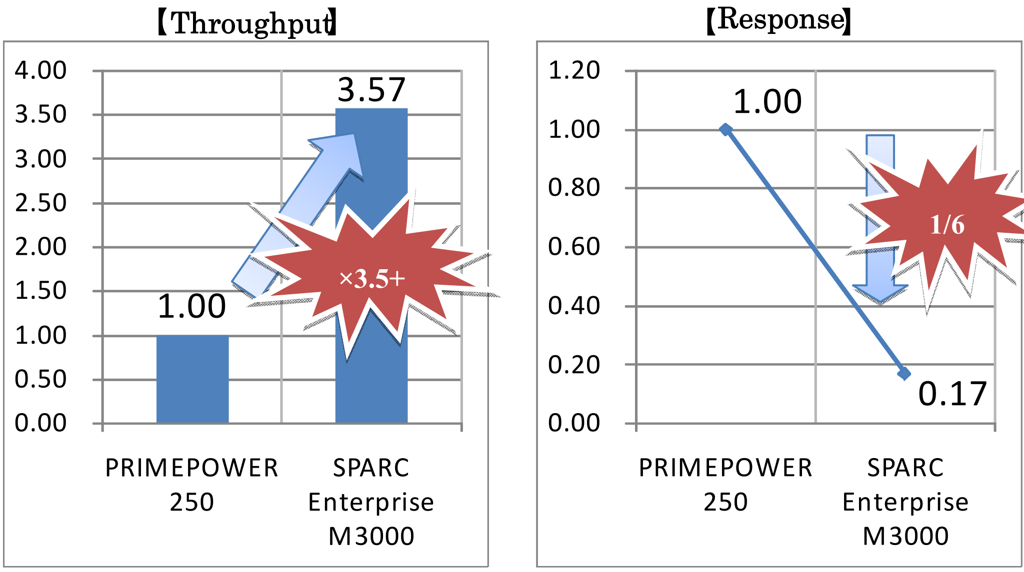


Figure 3-1 Performance comparison of PRIMEPOWER 250 and SPARC Enterprise M3000

Corporate IT systems can also benefit by upgrading from Oracle9*i* Database (or earlier releases) to Oracle Database 11*g*. First, upgrading the database system to the latest version gives users access to various cutting-edge technologies. For example, when the database performance has degraded and you must diagnose the cause of bottlenecking, Oracle9*i* Database requires a skilled and experienced database engineer to collect and analyze information to extract problems and to design remedies. Oracle Database 11*g* provides self-diagnostic capabilities and enhanced tuning advice functions.

In the area of security, Oracle9*i* Database requires rewriting application code to implement data encryption. In contrast, Oracle Database 11*g* offers a revamped data encryption function that allows users to establish a secure, robust database environment without modifying existing applications. In addition, Oracle Database 11*g* permits the use of various advanced technologies described earlier in this paper.

Continued use of older database versions can increase various risks. We strongly recommend that you examine and assess the risks associated with continued use of older versions, including the increasing difficulty of obtaining technical support, high maintenance costs, and potential difficulties obtaining necessary technical information.

Note that if a system has not been upgraded for quite some time since its initial deployment, procedures for migrating databases can be very complex, and compatibility problems may arise. This makes upgrading even harder.

3.2. Issues to Address When Planning Migration

Some issues are common to each component of a legacy system, and must be considered carefully before undertaking migration of hardware, OS, middleware, user data, and user applications.

To use existing assets (peripheral devices and other elements that will not be upgraded) in the new system, you must confirm that they work properly in combination with the hardware, OS, and middleware to be used with the new system.

Although Solaris 10 is guaranteed to be binary-compatible with previous versions, you must check for potential compatibility problems related to upgraded versions of middleware or to changes in items (such as obsolete commands and implementation methods) other than the binaries themselves made to take advantage of the enhanced feature set in Solaris OS.

Other issues must be considered as well when migrating from Oracle9*i* Database to Oracle Database 11*g*. In certain cases, compatibility between the two different Oracle Database versions can affect current applications. Such issues are primarily attributable to the addition of reserved words and changes in initialization parameters and the data dictionary.

Changes involving the query optimizer can cause the most significant problems. Versions up to Oracle9*i* Database supported the Rule Based Optimizer (RBO), but Oracle Database 10g Release 1 and later versions support only the Cost Based Optimizer (CBO). The RBO is a query optimizer which produces optimal SQL execution plans based on the rankings of available access paths. It is a legacy optimizer created before Oracle6 was developed. While the concept is easy to understand for developers, it has major drawbacks, including inability to closely follow data fluctuations or to account for data volumes and bias. Use of the RBO hinders not just the use of query processing engines released after Oracle 7.3, but improvements in important functions in the Oracle Database query processing engine. Moving to CBO makes it possible to improve application performance and reliability.

As shown above, to migrate from Oracle9*i* Database to Oracle Database 11*g*, clients must switch to the Cost Based Optimizer (CBO). The CBO estimates costs based on accurate statistical information prepared in advance and accounts for data value fluctuations and bias, then selects the optimal access paths for executing most SQL statements. However, in certain special cases, you may prefer to maintain the original RBO access paths. Switching to the CBO without performing adequate testing can generate serious performance degradation or problems after services are launched.

Performing extensive testing in advance of such a change and ensuring that the switchover from the RBO to CBO will not degrade database performance are critical in avoiding performance issues.

Table 3-1 RBO and CBO

	Rule Based Optimizer (RBO)	**Cost Based Optimizer (CBO)**
Overview	Create execution plans from a ranking of usable access path (for OLTP)	Estimates the cost from statistics and create execution plans with the lowest cost (for both OLTP and DSS)
advantages	• Easy for some developers to understand RBO • SQL execution plans change little	• Able to follow the change of data • Receives benefits of new database features • Create execution plan from the bias and the size of data
disadvantages	• Unable to follow the change of data • Cannot consider the bias and size of data • Not aware of new database features • Unsupported starting with Oracle Database 10g	• Need to gather statistics • May change database performance by gathering statistics again

3.3. Application Tests and Performance Tests before Migration

Application tests and performance tests should be planned and implemented to confirm in advance that no major compatibility or performance problems will arise in the new system. This reduces the risk of migration failure and other risks following migration.

For compatibility checks, plan and implement the following application tests:

- **Minimum tests**

  All or parts of applications are transferred from the current system to the new system (or test environment). Tests are performed without enabling new functions. In minimum tests, all possible potential problems cannot be discovered , but at least problems related to application launching and invocation can be checked.

- **Functional tests**

  Functional tests include testing of all databases, network, and application components. These are conducted after new and existing functions of the system are upgraded. Functional tests confirm that individual system components and new functionality work properly as they did before the upgrade.

- **Integration tests**

  Integration tests check interactions among system components. Specifically, these tests assess the following points:

  - Confirms that applications (Pro*C/C++ , JDBC, ODBC) and other software causes no problems in the new software.
  - Performs analysis to determine if changes in data type and data dictionary data affect front-end applications.
  - Checks connectivity when applications are connected through SQL*Net, Net8, or Oracle Net services.

- **Performance tests**

  To check performance, plan and carry out the following performance tests. Before performing the tests, you must obtain performance information*[2] on the current system based on the performance requirements of the new system. This will help determine whether the performance requirements of the new system are adequate and help identify factors leading to performance degradation, based on comparisons.

- **Confirming unit performance**

  To confirm unit performance, a test is conducted in the minimum task unit (e.g., SQL statement unit), and mainly the response for unit is analyzed. Once unit performance meets performance requirements, concurrent performance is confirmed. If the unit performance fails to satisfy performance requirements or if performance degrades, bottleneck analysis and tuning must be performed.

[2] Includes response and throughput for task (SQL), OS statistics, STATSPACK, and SQL trace.

- **Confirming concurrent performance**

  To confirm concurrent performance, multiple factors comprising actual business processes are used to generate loads similar to those in the actual system and the performance of the overall system is analyzed. If the system fails to meet the required unit performance, confirming concurrent performance will not help improve performance. Additionally, the following problems not generated during confirmation of unit performance can also emerge during confirmation of concurrent performance.

  - High CPU usage, CPU wait
  - High disk usage, disk wait
  - Exclusive control (lock) for files and databases
  - Process wait due to inter-process communication

As described above, migration to a new system requires planning and confirmation of compatibility of the current system (applicability, extent of results of any incompatibilities, countermeasures), application tests, and performance tests. This increases required man-hours, entails higher costs, and results in system failures and other problems in various work processes.

In our verification testing, we attempted to verify whether use of application tests, Database Replay to confirm concurrent performance, and SQL Performance Analyzer to confirm unit performance would reduce risks and facilitate the process of migration.

4. Verification System

The verification system used in our tests consisted of a single database client unit, two database servers, and one storage unit. SQL statements and others for each verification test were executed on database servers.

4.1. System Configuration

Figure 4-1 shows the configuration of the verification system. The client and database servers were connected by 1000Base-T. The database servers and storage were connected through two fiber channels, with an FC switch inserted between each database server and the storage unit.

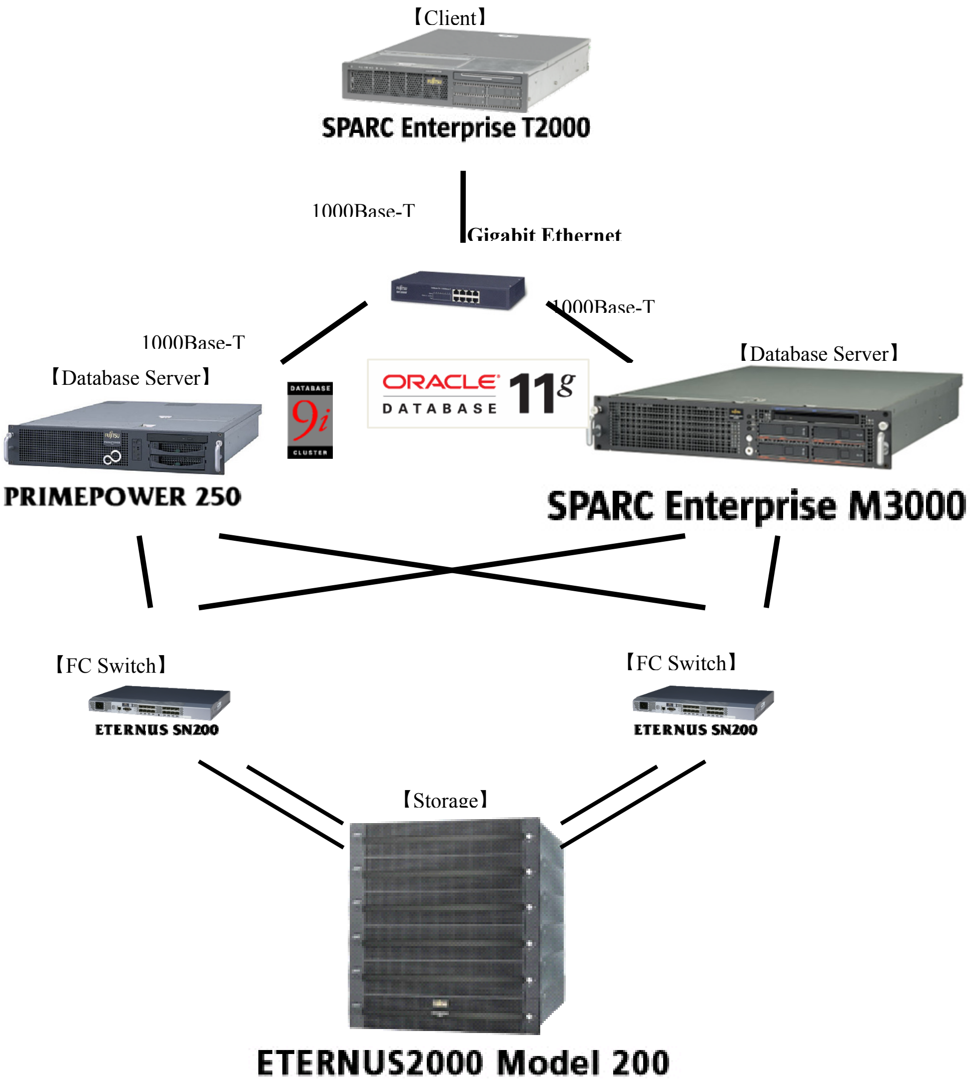


Figure 4-1 System configuration

Described below are the specifications for each device and the software used.

4.1.1. Database server (migration source)

Hardware

Model	Fujitsu PRIMEPOWER 250
CPU	SPARC64 Ⅴ 1.1GHz/1MB Cache×2
Memory	10GB
Internal HDD	73GB×1

Software

OS	Solaris 9 OS
Database	Oracle9i Database （9.2.0.8）

4.1.2. Database server (migration destination)

Hardware

Model	Fujitsu SPARC Enterprise M3000
CPU	SPARC64 Ⅶ 2.52GHz/5MB Cache 1CPU/4core/8thread
Memory	12GB
Internal HDD	146GB SAS Disk×2

Software

OS	Solaris 10 OS
Database	Oracle Database 11g （11.1.0.7） ※partly using 11.1.0.6

4.1.3. Storage

Model	Fujitsu ETERNUS 2000 Model 200
Disk Drive	146GB (15,000rpm) × 32

4.1.4. Client

Hardware

Model	Fujitsu SPARC Enterprise T2000
CPU	Ultra SPARC T1 1.2GHz/3MB Cache 1CPU/8core/32thread
Memory	8GB

Internal HDD	73GB （10,000rpm） SAS Disk×2

Software

OS	Solaris 10 OS
Database Client	Oracle Database 11g （11.1.0.7） ※partly using 11.1.0.6

4.2. Storage Configuration

Described below is the configuration of the storage used in the verification tests.

Figure 4-2 indicates the storage configuration. Storage consisted of 32 disks composing RAID 1+0 (4+4) x 4.

One RAID group was divided into two LUN-V units for use by the PRIMEPOWER 250 and SPARC Enterprise M3000. A file system was created for each LUN-V and mounted at the following mount points : /oradata1, /oradata2, /oradata3, and /oradata4.

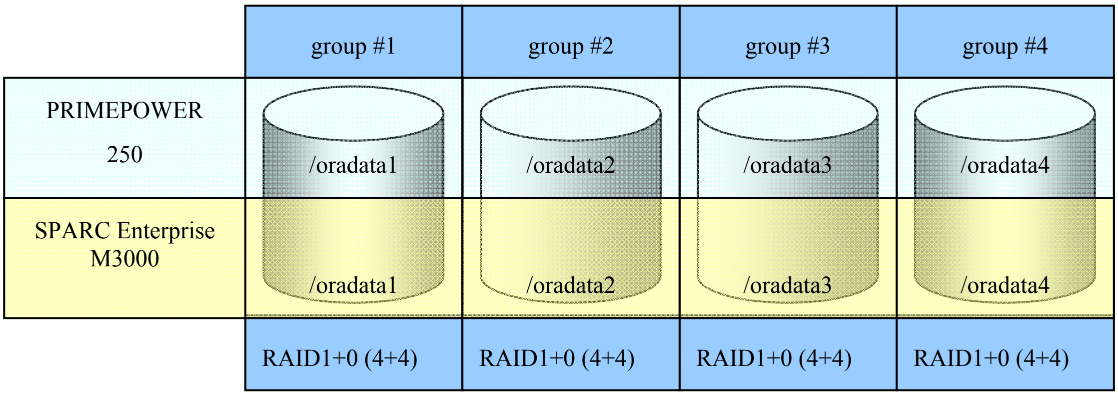


Figure 4-2 Storage configuration

These mount points were used for the purposes indicated in Table 4-1. The PRIMEPOWER 250 and SPARC Enterprise M3000 feature identical configurations.

Mount Points	Purpose
/oradata1	Tablespace for OLTP tables
/oradata2	Tablespace for OLTP tables
/oradata3	・the directory object for Database Replay ・for export/import dump files
/oradata4	SYSTEM, SYSAUX, UNDO tablespace, USER tablespace, Temporary tablespace, REDO logs, Control files

Table 4-1 Disk configuration

The RAID group for data files in the tablespace updated by online transactions in the verification tests was separated from the RAID group for other data files (SYSTEM tablespace, REDO logs, control files, etc.).

Similarly, directory objects and export/import dump files were also placed in another RAID group.

5. Migration Scenario

The general flow of Oracle Database migration, key points, and areas of verification focus are described below.

5.1. General Migration Flow

The Oracle Database migration involves work on both the test system and the production system. On the test system this includes preparing for the upgrade, producing the test database, and testing the upgrade process,. On the production system the migration work includes operational checks, backups, upgrading the production database, and production database adjustments. Each of these steps also involves a number of subtasks: understanding new functions, determining the proper upgrade path, and selecting the upgrade method, as shown for Step 1. (See Figure 5-1.)

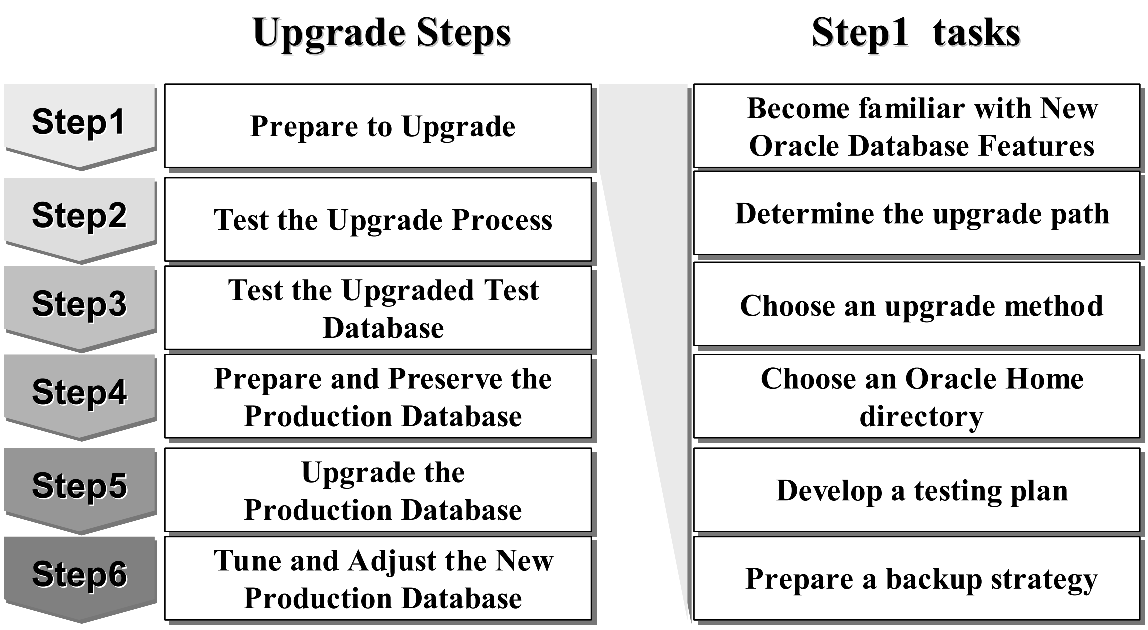


Figure 5-1 Upgrade steps

About upgrade paths

The versions that can be directly upgraded to Oracle Database 11*g* are 9.2.0.4 or later, 10.1.0.2 or later, and 10.2.0.1 or later. Prior versions must be upgraded to the above versions first, then upgraded to Oracle Database 11*g*. (See Figure 5-2.)

(Example)

- 8.0.6 → 9.2.0.8 → 11.1
- 8.1.7.4 → 10.2.0.x → 11.1
- 9.0.1.4 → 10.2.0.x → 11.1

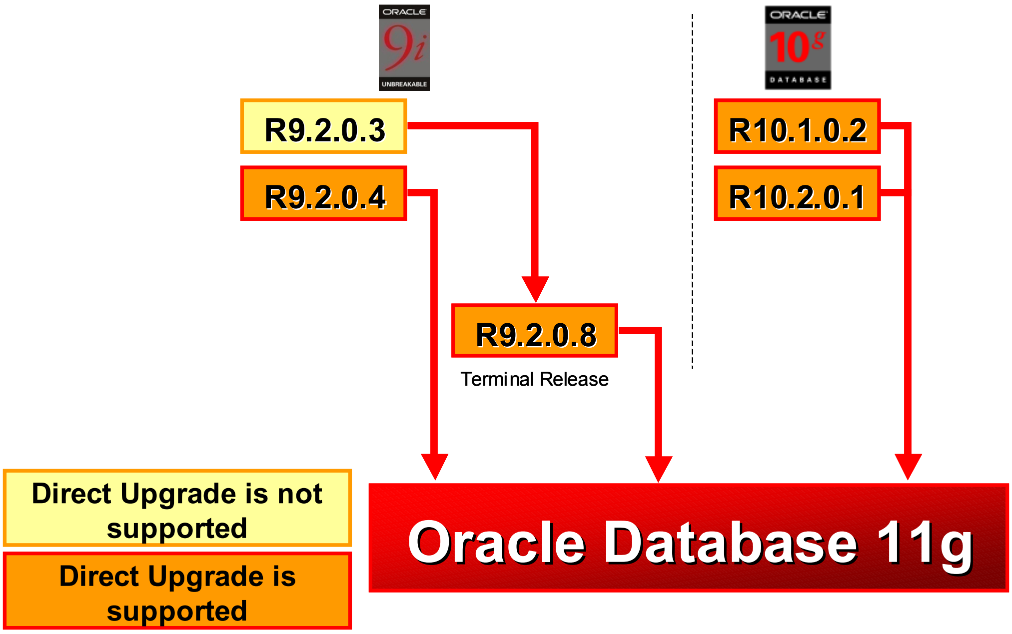


Figure 5-2 Upgrade path

Upgrade method

There are a number of approaches available for a database upgrade. Select the optimal method based on hardware/OS migration issues and requirements such as downtime, system migration, and fallback strategies.

- Upgrade the whole database
  - Database Upgrade Assistant (DBUA)
  - Upgrade script
- Migrate the data
  - export/import
  - migrate tablespaces (Transportable Tablespace [TTS])
  - unload/upload
  - data copy via DB Link

For detailed information on upgrade methods, refer to *Oracle Database Upgrade Guide 11g Release 1 (11.1) E05758-01* and MetaLink Note: 419550.1.

Table 5-1 Advantages and disadvantages of various upgrade methods

Method	Advantage	Disadvantage
DBUA	• Simple and concise using GUI • Fast: Independent of database size • Low space requirements for upgrade	• OS change impossible • Available version restricted
Manual (Upgrade script)	• Total control • Fast: Independent of database size • Low space requirements for upgrade	• OS change impossible • Available version restricted • More manual work required • Error-prone typing
Export/Import	• Defragment and reorganize data • Usable for upgrade from versions as early as Oracle 5 • Hardware and OS change possible	• Space requirements for dump files • Performance dependent on database size • Slow
TTS	• Can be very fast • Can provide minimal downtime • Hardware and OS change possible (Starting with Oracle Database 10g)	• Higher effort for preparation • Space requirements for datafiles • Available only since Oracle 8i
Unload/Load	• Defragment and reorganize data • Hardware and OS change possible	• Space requirements for unloaded files • Performance dependent on database size • Much manual work required • Slow
Data Copying	• Defragment and reorganize data • Hardware and OS change possible	• Performance dependent on database size • Everything such as structure and data must be produced manually • Connectable version restriced

5.2. Areas of Verification Focus

In our verification testing, we focused on preparing a test database for upgrading Oracle9*i* Release 2 (9.2.0.8) to Oracle Database 11*g* Release 1 (11.1.0.7) as well as performance and other tests performed with the test database.

What follows is an introduction to procedures for creating a test database using an online backup to minimize effects on the production database. For tests related to upgrading, Real Application Testing is used for efficient execution of workload and performance testing using the production database.

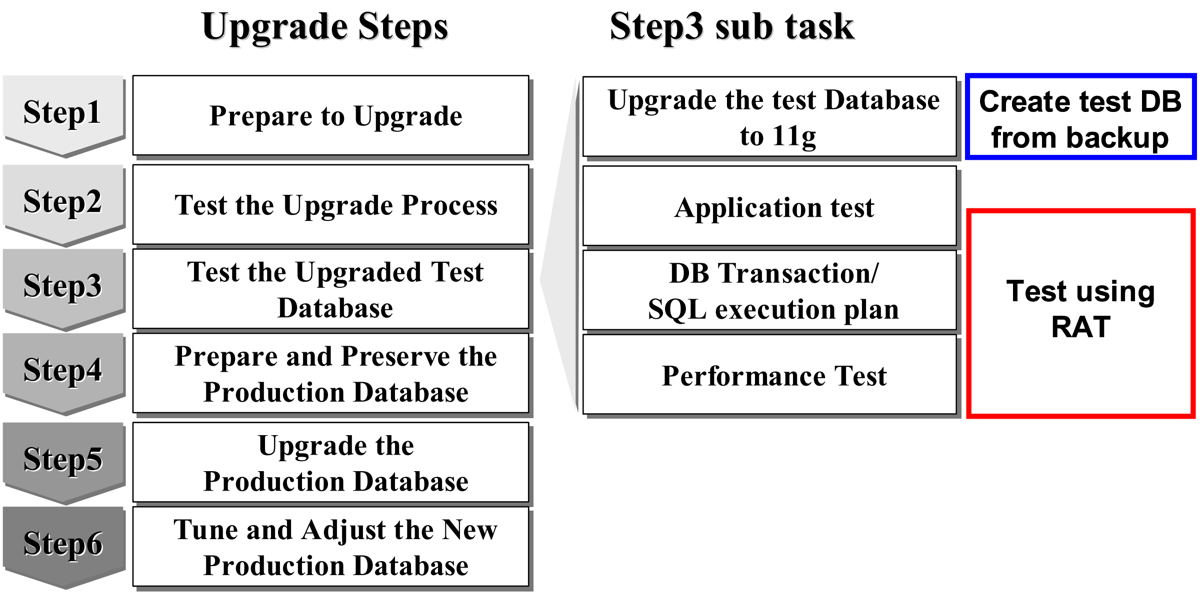


Figure 5-3 Areas of focus

6. Verification Details and Results

Comparisons of the performance of the PRIMEPOWER 250 and SPARC Enterprise M3000 and results of verifying Database Replay and SQL Performance Analyzer are described below.

6.1. Performance Comparison of PRIMEPOWER 250 and SPARC Enterprise M3000

Compared below are the online transaction processing and batch processing performance of systems configured with the PRIMEPOWER 250 + Oracle9*i* Database and with the SPARC Enterprise M3000 + Oracle Database 11*g*.

6.1.1. Online transaction processing

Verification details

An OLTP application simulating five processing operations (order processing, payment balance processing, order process confirmation, delivery batch processing, best-seller inventory check) of the inventory control system from the client (SPARC Enterprise T2000) was executed in various multiplicities and the order processing performance (CPU usage, throughput/response, disk busy rate) of the PRIMEPOWER 250 + Oracle9*i* Database and SPARC Enterprise M3000 + Oracle Database 11*g* compared.

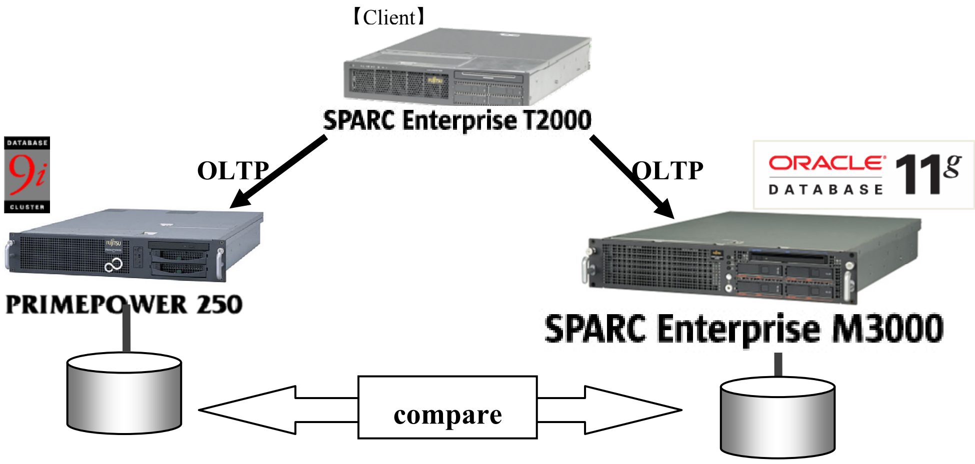


Figure 6-1 Overview of online transaction processing verification

The measurement conditions for the verification test are given below.

- Database size: 85 GB
- Application concurrency: *n*, *n* x 2, *n* x 4, *n* x 10 multiplicities
- Application execution time: 35 min (measurement time of 20 min, initiated 15 min after application start)

The table for OLTP application, process overview, and processing details are as follows:

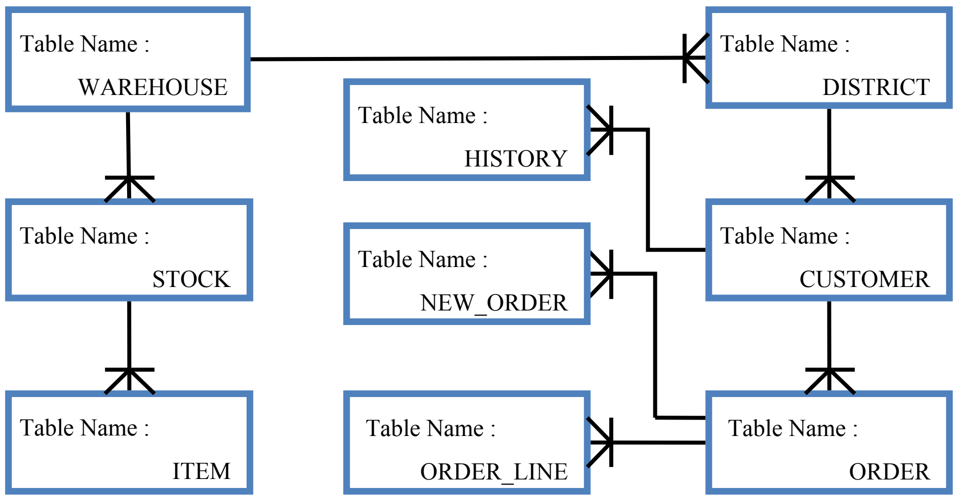


Figure 6-2 the table for OLTP application

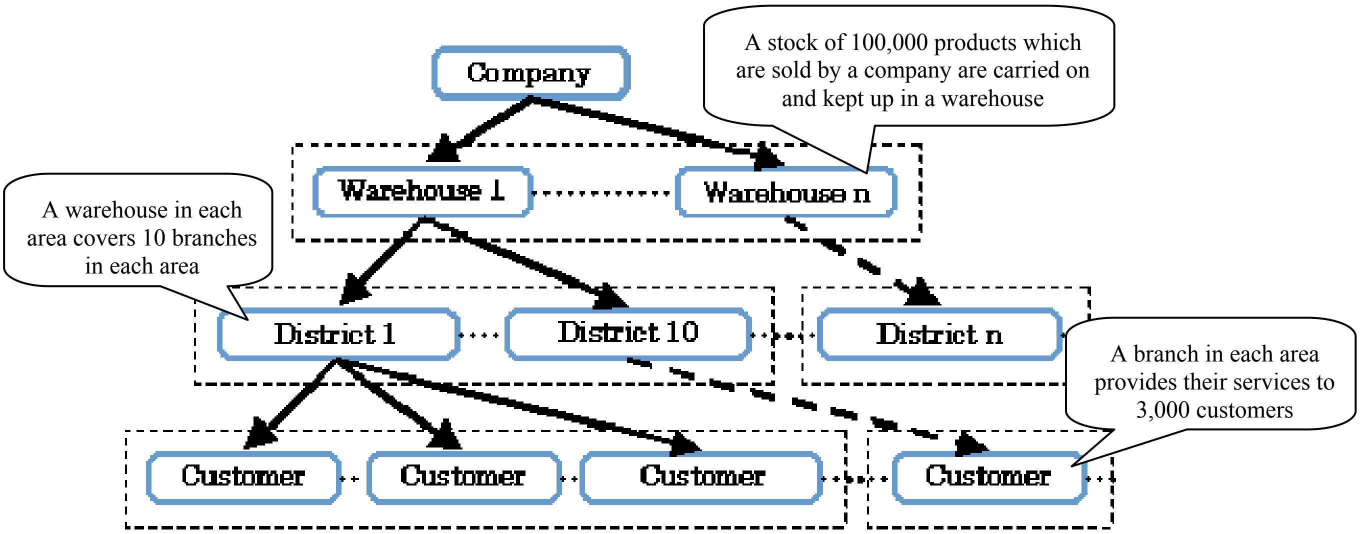


Figure 6-3 OLTP application process overview

Table 6-1 OLTP application processing details

transaction	load	frequency	response	read/write
New-Order	middle	high	short	read/write
Payment	low	high	short	read/write
Order-Status	middle	low	short	read only
Delivery	middle	low	long	read/write
Stock-Level	high	low	long	read only

Verification results

- Throughput

In a comparison of throughputs in online transaction processing, the system configured with the SPARC Enterprise M3000 + Oracle Database 11*g* achieved higher throughput than the system configured with the PRIMEPOWER 250 + Oracle9*i* Database. The difference was approximately a factor of 3.5 at a factor of *n* x 10.

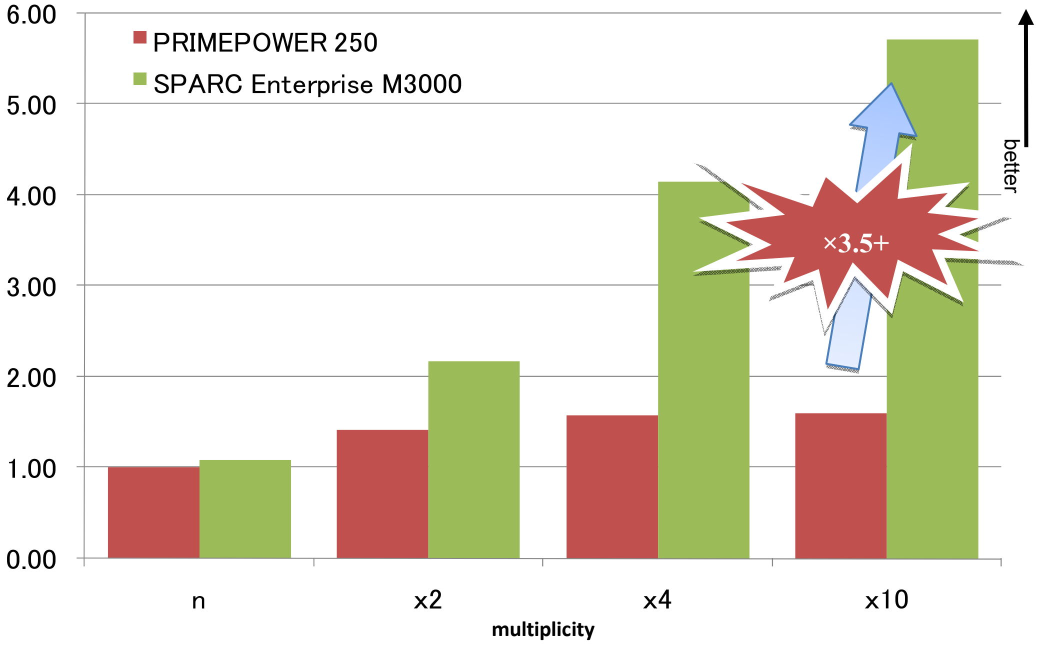


Figure 6-4 Throughput in online transaction processing

Table 6-2 Throughput volume in online transaction processing

multiplicity	**PRIMEPOWER 250**	**SPARC Enterprise M3000**
n	1.00	1.08
× 2	1.42	2.17
× 4	1.58	4.15
×10	1.60	5.71

* A value of “1.00” is assigned to the performance of the PRIMEPOWER 250 at a multiplicity of *n*.

- Response

  In a comparison of response times for online transaction processing, the system configured with the SPARC Enterprise M3000 + Oracle Database 11*g* has been proven to run much faster than the system configured with the PRIMEPOWER 250 + Oracle9*i* Database at all concurrencies. The server elapsed time was about 1/6 at a multiplicity of *n* x 10.

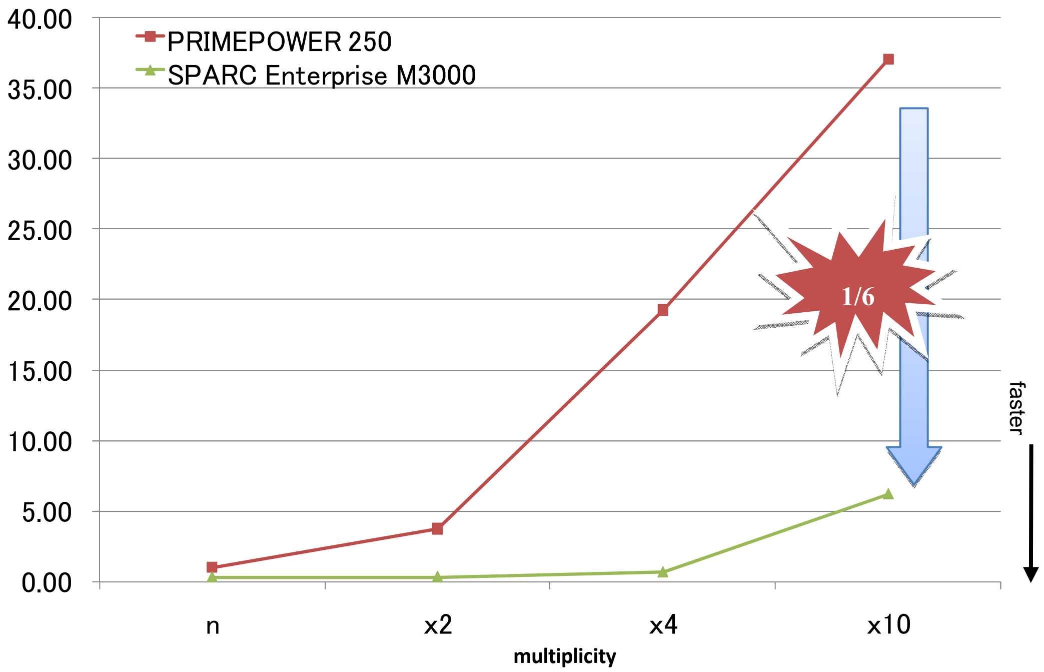


Figure 6-5 Response in online transaction processing

Table 6-3 Response times in online transaction processing

multiplicity	PRIMEPOWER 250	SPARC Enterprise M3000
n	1.00	0.31
× 2	3.73	0.33
× 4	19.27	0.66
×10	37.08	6.22

* A value of “1.00” is assigned to the performance of the PRIMEPOWER 250 at a multiplicity of *n*.

- CPU usage/disk busy rates

  In a comparison of average CPU usage at each multiplicity (Figure 6-6, Table 6-4), the system configured with the SPARC Enterprise M3000 + Oracle Database 11*g* processed online transactions at lower CPU usage rates than the system configured with the PRIMEPOWER 250 + Oracle9*i* Database at all multiplicities.

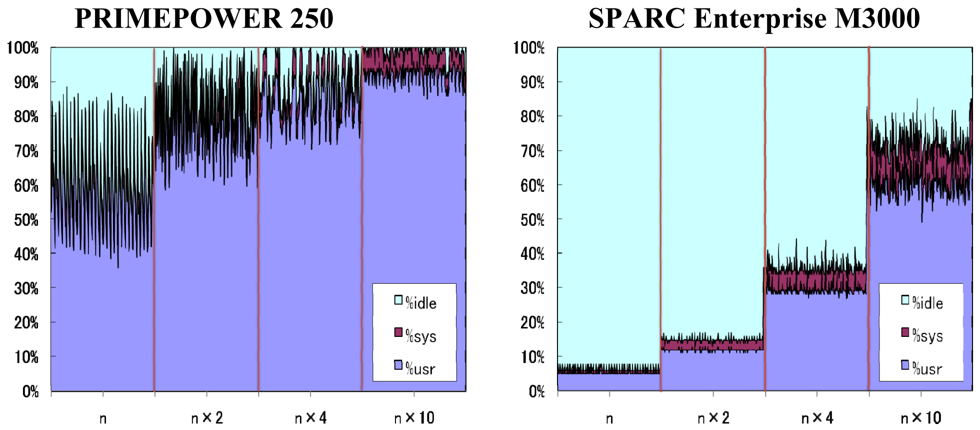


Figure 6-6　Changes in CPU usage by multiplicity

Table 6-4　Average CPU usage in online transaction processing

multiplicity	**PRIMEPOWER 250**	**SPARC Enterprise M3000**
n	59.9%	6.5%
× 2	83.4%	15.1%
× 4	89.7%	35.7%
×10	98.4%	71.7%

In a comparison of disk busy rates for the redo area at a multiplicity of *n* x 4 (Figure 6-7), the disk busy rate for the system configured with the SPARC Enterprise M3000 + Oracle Database 11*g* was elevated due to high transaction throughput. In the system configured with the PRIMEPOWER 250 + Oracle9*i* Database, the CPU created a bottleneck and reduced throughput and disk busy rates.

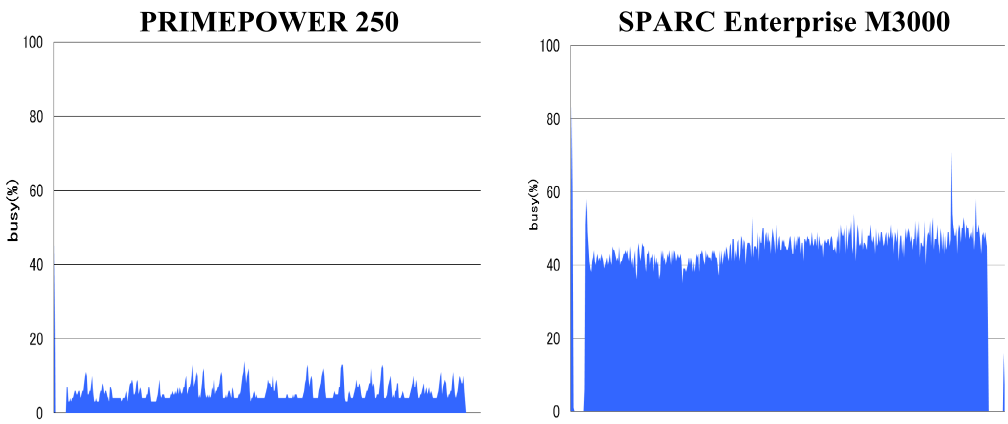


Figure 6-7 Changes in disk busy rates in redo log area

6.1.2. Batch processing

Verification details

A batch application was executed on the system configured with SPARC Enterprise M3000 + Oracle Database 11*g* and the system configured with PRIMEPOWER 250 + Oracle9*i* Database and the server elapsed times compared.

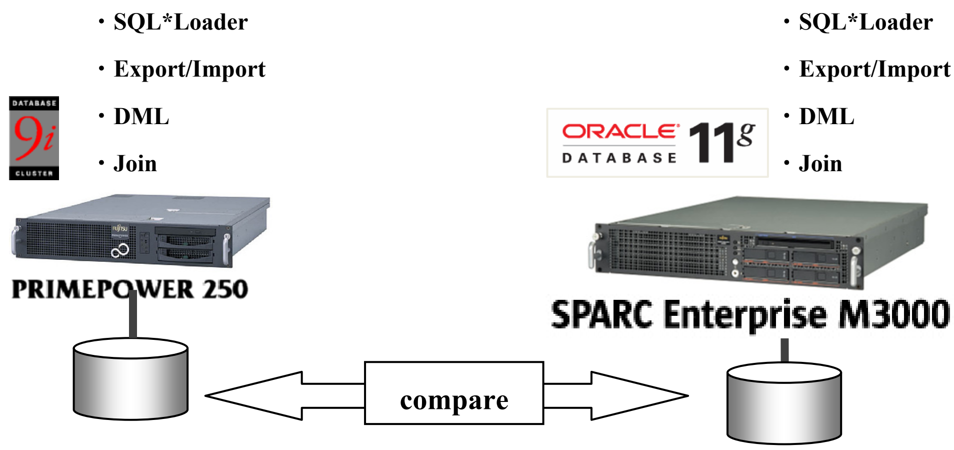


Figure 6-8　Overview of batch processing verification

The measurement items in batch processing verification were as follows:

- Export / Import

SQL*Loader	Load data of all tables except TEST_TABLE from CSV files
Export※/Import	Export data of all tables except TEST_TABLE by export utility, after that load data by import utility

- Data Manipulation Language（DML）

Insert	1,000,000 INSERT data into TEST_TABLE
Update	1,000,000 UPDATE data on TEST_TABLE which satisfy TEST_COMMENT1 column and TEST_NUM column as conditions
Delete	1,000,000 DELETE data which satisfy TEST_NUM column as conditions from TEST_TABLE
Fetch	1,000,000 FETCH data from TEST_TABLE

- Join processing

Nested Loop Join（full scan）	JOIN MASTER_TABLE and TRAN_TABLE as the driving table

※Original export is desupported except for downgrade purpose.

Shown below is the structure of the table subject to batch processing.

MASTER_TABLE (Rows 10,000, CSV 1.1MB)

No	Name	Type	Length	
1	M_ITEM1	CHAR	7	PK
2	M_ITEM2	CHAR	10	
3	M_ITEM3	NUMBER	10,2	
4	M_ITEM4	CHAR	7	
5	M_ITEM5	CHAR	10	
6	M_ITEM6	CHAR	7	
		amount	100	

TEST_TABLE (Rows 1,000,000)

No	Name	Type	Length	
1	TEST_NUM	NUMBER	8	PK
2	TEST_ID	NUMBER	2	
3	TEST_DATE	CHAR	8	
4	TEST_COMMENT1	VARCHAR2	50	
5	TEST_COMMENT2	VARCHAR2	50	Update
		amount	108	

TRAN_TABLE (Rows 1,000,000, CSV 383MB)

No	Name	Type	Length	
1	T_ITEM1	CHAR	7	PK
2	T_ITEM2	CHAR	10	PK
3	T_ITEM3	CHAR	20	
4	T_ITEM4	VARCHAR2	200	
5	T_ITEM5	NUMBER	10,2	
6	T_ITEM6	CHAR	20	
7	T_ITEM7	VARCHAR2	230	
8	T_ITEM8	DATE	7	
		amount	500	

LOG_TABLE (Rows 1,000,000, CSV 543 MB)

No	Name	Type	Length	
1	L_ITEM1	TIMESTAMP	11	
2	L_ITEM2	CHAR	10	
3	L_ITEM3	CHAR	20	
4	L_ITEM4	CHAR	52	
5	L_ITEM5	VARCHAR2	200	
6	L_ITEM6	VARCHAR2	200	
7	L_ITEM7	VARCHAR2	200	
8	L_ITEM8	DATE	7	
		amount	700	

Verification results

The comparison of SQL*Loader and export/import times shows that the SQL*Loader times and export/import times for the system configured with the SPARC Enterprise M3000 + Oracle Database 11*g* are about 1/2 and about 2/3, respectively, compared to the system configured with PRIMEPOWER 250 + Oracle9*i* Database.

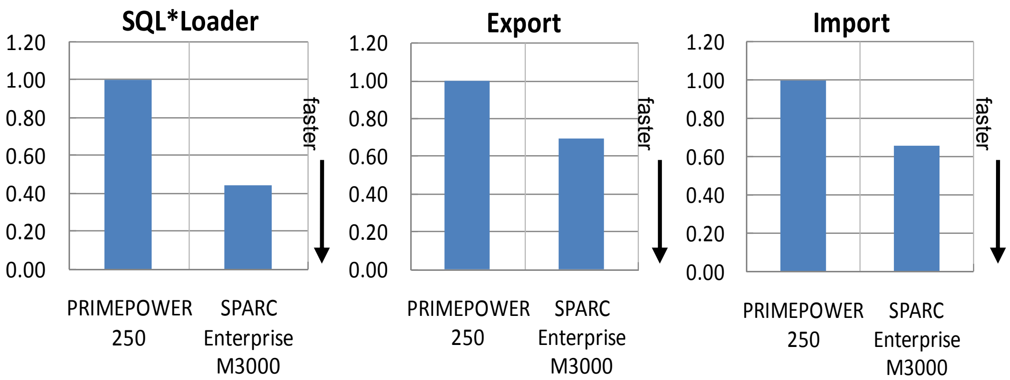


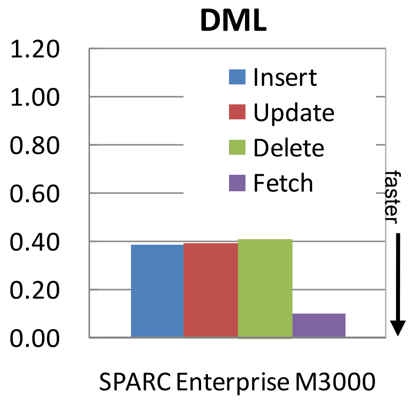
Figure 6-9 SQL* Loader and export/import times

Table 6-5 SQL* Loader and export/import processing times

	SPARC Enterprise M3000
SQL*Loader	0.44
export	0.69
import	0.66

*A value of "1.00" is assigned to the performance of the PRIMEPOWER 250.

Next, the data manipulation (DML) and join processing times were compared. The system configured with the SPARC Enterprise M3000 + Oracle Database 11*g* completed the total data manipulation in about 1/3 and the join processing in about 1/5 the time compared to the system configured with the PRIMEPOWER 250 + Oracle9*i* Database.



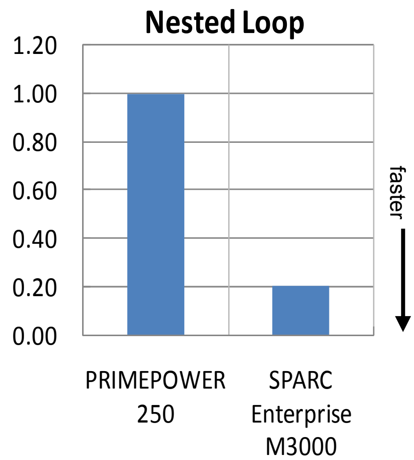


Figure 6-10 Data manipulation (DML) and join processing times

Table 6-6 Data manipulation (DML) and join processing times

	SPARC Enterprise M3000
insert	0.39
update	0.39
delete	0.41
fetch	0.10
average	0.38
join	0.21

*A value of "1.00" is assigned to the performance of the PRIMEPOWER 250.

The above results indicate that the batch processing performance of the system configured with the SPARC Enterprise M3000 + Oracle Database 11*g* is approximately twice that of the system configured with the PRIMEPOWER 250 + Oracle9*i* Database.

6.2. Database Replay

6.2.1. Details of functions

Database Replay enables the replay of workloads obtained with a production system on a test system.

Database Replay provides the following testing workflow which consists of a workload capture phase on the production system and a replay and analysis phase on the test system.

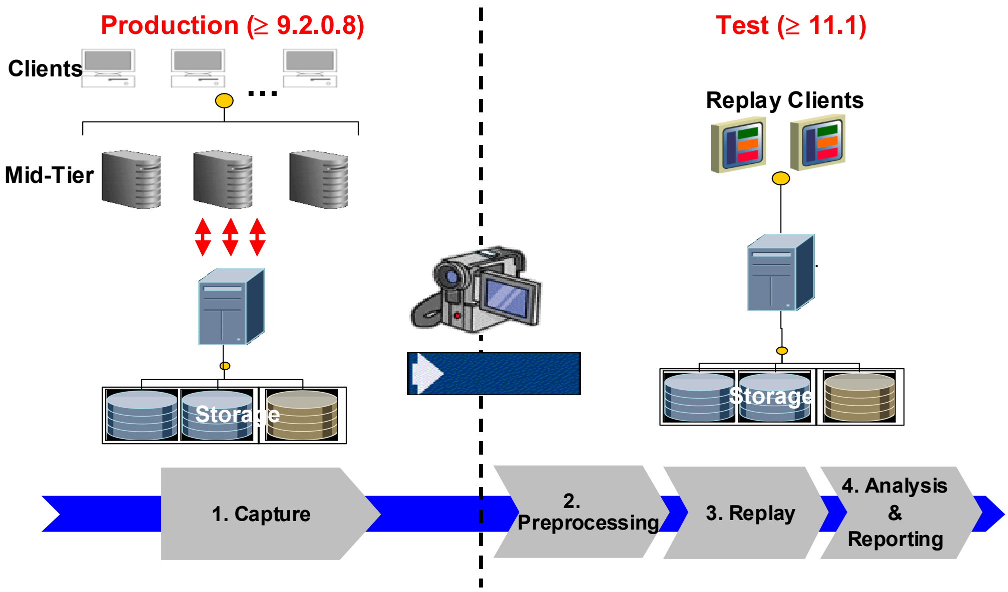


Figure 6-11 Main processing procedures of Database Replay

1. **Workload capture:** When workload capture is enabled, all external client requests directed to the Oracle Database are tracked and stored in binary files, called capture files, on the file system. Oracle recommends taking a backup of the entire database prior to the workload capture. The user specifies the location of the capture files and the workload capture start and end time. During this process, all information pertaining to requests from external database clients is written to the capture files.
2. **Workload preprocessing:** Once the workload has been captured, the information in the capture files has to be processed. This preprocessing transforms the captured data into replay files and creates all necessary metadata needed for replaying the workload. Workload preprocessing must be performed once for every captured workload for replay on a specific RDBMS version. If the replay database version changes preprocessing needs to performed again. Preprocessing is an intensive operation and should not be done on the production system. .
3. **Workload replay:** After the captured workload has been processed, it is now ready for replay. A client program, called Workload Replay Client (wrc executable), then interprets the replay files and submits calls to the database with the exact same timing

and concurrency as in the capture system. Depending on the captured workload, you may need one or more replay clients to properly replay the workload. A calibration utility is provided to help determine the number of replay clients needed for a workload. It should be noted that since the entire workload is replayed including DML and SQL queries, it is important that the data in the replay system is in the same logical state to that in the production system just before capture was started to enable reliable analysis for reporting purposes.

4. **Analysis and Reporting:** Extensive reports are provided to enable detailed analysis of the capture and replay. Any errors encountered during replay are reported. Any divergence in rows returned by DML or queries is shown. Basic performance comparisons between capture and replay are provided. For advanced analysis, AWR reports are made available to allow detailed comparison of performance statistics between capture and replays.

Replay parameters

By default, workloads are replayed so that the captured transaction order/timing is retained. However, the transaction order/timing can be changed by adjusting the replay parameter settings.

Depending on the testing objective, the parameters described below can be changed to accommodate the particular requirements.

- Synchronization parameter [TRUE (default) / FALSE]

  This parameter specifies whether to enable synchronization during workload replay. When set to TRUE, the captured workload commit order is retained during replay, and all replay actions are executed only after all dependent commit actions are completed. In other words, there is a much bigger guarantee that each replayed call will do the same work it and be a useful workload for application testing(extracting SQL-level incompatibilities). When set to FALSE, there could be potential workload error and data divergence but it might still be useful in case of some applications for load testing. The default value is TRUE.

- Think_time_auto_correct parameter [TRUE (default) / FALSE]

  This parameter is used to adjust think times between calls (based on the think_time_scale parameter) when the completion of a user call in replay takes longer than capture. The think time between consecutive calls is decreased so that replay can catch up with the captured timings.

- Connect_time_scale parameter [0 to 100 (default)]

  This parameter changes the scale of the elapsed time from the start of workload capture to the connection of a session based on a specified value, interpreting it as a percentage value. If the parameter is set to 100 then all the users connect to the db at roughly the same rate. If you set this to 50 then the users will try to connect at twice the captured rate.

- Think_time_scale parameter [0 to 100 (default)]

  This parameter changes the scale of elapsed time between two consecutive user calls from the same session and interprets it as a percentage value. When this parameter is set to 0 (zero) there is no think time between replayed requests. This means that requests are sent to the RDBMS at the highest possible rate.

Replay result evaluation method

The method for evaluating the results of replay execution are discussed here. The general evaluation flow involves checking whether database changes or replay preparations were accurately completed and confirming whether the workload to be replayed (captured) was appropriate.

The target success rate for user calls is 90%. Although this value should be as close to 100% as possible, it depends on the captured workload (inclusion of unsupported calls, background workload, problems due to changes in the database, etc.). Thus, the causes of call failures must be identified by analyzing the report. Additionally, you must determine whether the failed calls are permissible given the workload.

Refer to the following flowchart for more information on the evaluation method.

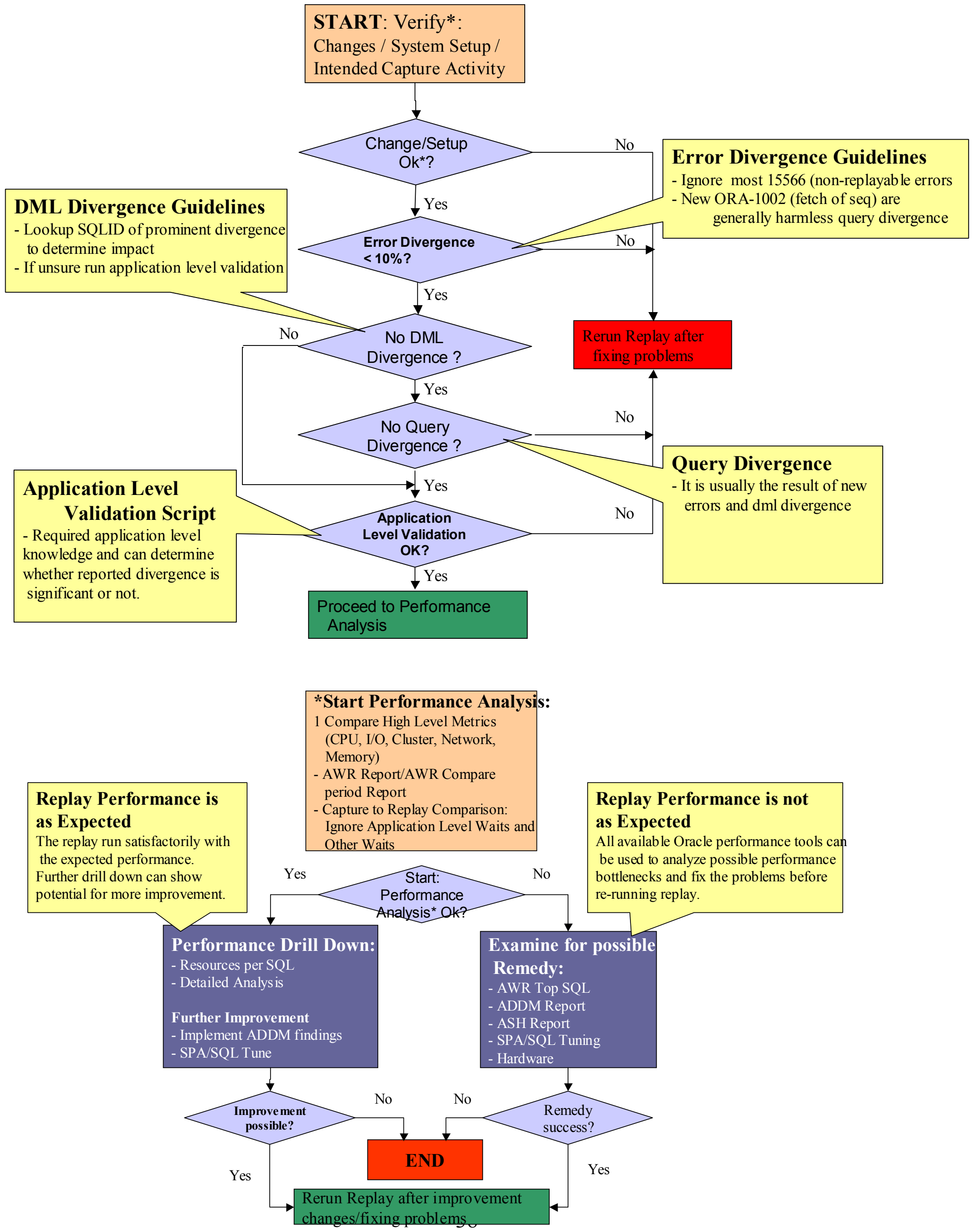


Figure 6-12 Flowchart of replay result evaluation method

6.2.2. Prerequisites for Database Replay

Account for the following when executing Database Replay.

- **Data match**

  The application/data conditions in the test environment must be logically identical to those in the production environment.

- **Patch application**

  To capture workloads using Oracle9*i* Database and to replay them using Oracle Database 11*g*, the Oracle9*i* Database used must be 9.2.0.8 + One-Off Patch #96973309.

- **Estimating the number of replay clients**

  The captured workload is issued to the test database during replay by a multi-thread program called Workload Replay Client (hereinafter called “wrc”).

  To determine the optimal number of replay clients for the workload in question and the number of CPUs, the wrc must be executed in calibrate mode.

  Shown below is an example launching Workload Replay Client in calibration mode.

```
$ wrc mode=calibrate replaydir= <WORKLOAD_DIRECTORY_NAME>
Workload Replay Client: Release 11.1.0.7.0 - Production on Wed Nov 5 15:22:30 2008


Copyright (c) 1982, 2007, Oracle.    All rights reserved.


Report for Workload in: /export/home/oracle/replay/REP40_RMAN_9208_2


Recommendation:
Consider using at least 4 clients divided among 1 CPU(s)
You will need at least 150 MB of memory per client process.
If your machine(s) cannot match that number, consider using more clients.
Workload Characteristics:
- max concurrency: 40 sessions
- total number of sessions: 40
Assumptions:
- 1 client process per 50 concurrent sessions
- 4 client process per CPU
- 256 KB of memory cache per concurrent session
- think time scale = 100
- connect time scale = 100
```

- synchronization = TRUE

* WORKLOAD_DIRECTORY NAME: Directory where preprocessed workload files are located

The results of evaluations in calibrate mode recommend four replay clients. It may be possible to use more or fewer clients since replay will not check for the recommended number of clients. For our tests replay was conducted based on the Oracle recommended values above.

6.2.3. Preparing the Database Replay execution environment

Procedure summary

Described below are the procedures for preparing the test environment and for executing Database Replay in the verification tests. (See Figure 6-13.)

1. Create a backup of the production database
2. Capture a workload from the production database
3. Create a test database (9*i* R2) from the production backup
4. Upgrade the test database from Oracle Database 9*i* R2 to 11*g*
5. Replay the previously captured workload on the test database (11*g*)

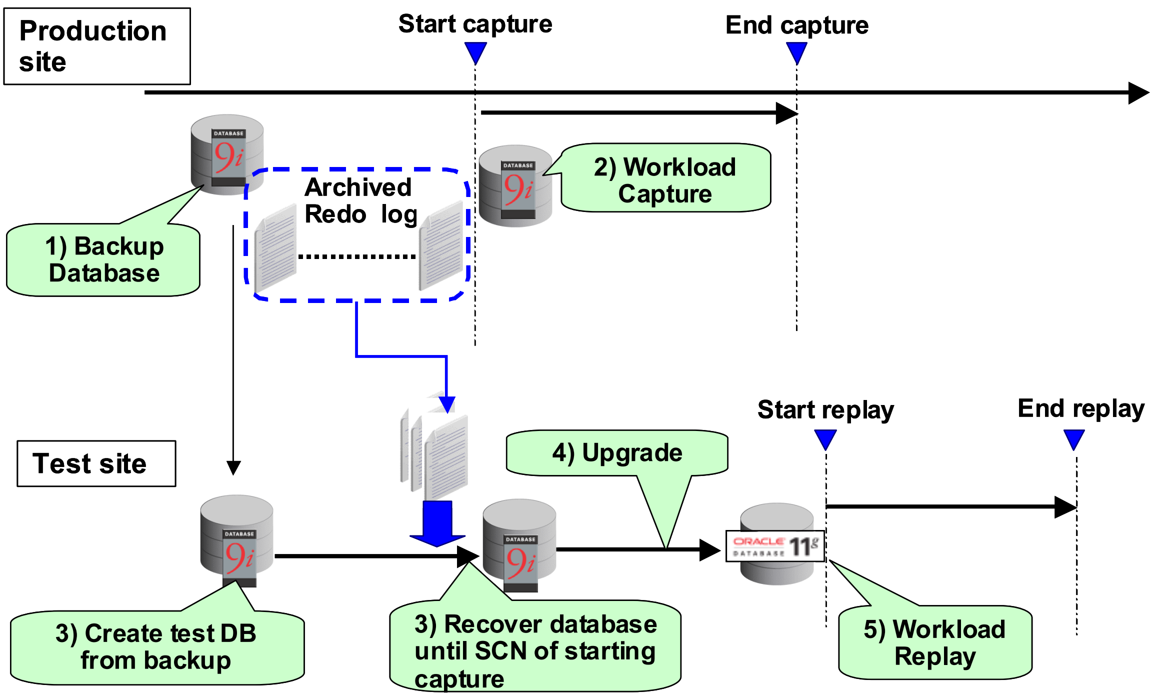


Figure 6-13 Procedure summary

0. Prerequisites

Confirm the following prerequisites before starting the procedure.

- Prepare test environment featuring the same platform (OS) and version of the Oracle environment (Oracle9*i* Database R2)
- Ensure that the database is in ARCHIVELOG Mode

1. Create a backup of the production database.

NOTE: In the following discussion of preparations for duplicating the production database, the production database is referred to as “old” and the test database as “new.”

1-1. Create the initialization parameter file, init<sid>.ora, for the test database

(1) Edit the initialization parameter file.

Old: db_name = “old”

New: db_name = “new”

Change all directories specified in the initialization parameter file to the directories for the test database (control_files, background_dump_dest, user_dump_dest, log_archieve_dest_n, etc.).

(2) After changing parameters, store the initialization parameter file in $ORACLE_HOME/dbs of the destination database.

Change all directories specified in the initialization parameter file to directories for the test database (control_files, background_dump_dest, user_dump_dest, log_archieve_dest_n, etc.).

1-2. Prepare scripts for control file production

Execute the following commands for the production database.

```
SQL> alter database backup controlfile to trace;
```

A trace file will be created in the directory of the production database specified by the user_dump_dest parameter.

1-3. Edit the scripts

Edit the scripts produced in the step 1-2 above.

(1) Correct the following sections:

Old

```
CREATE CONTROLFILE REUSE DATABASE "OLD" NORESETLOGS ARCHIVELOG
    MAXLOGFILES 16
```

```
      MAXLOGMEMBERS 3
      MAXDATAFILES 100
      MAXINSTANCES 8
      MAXLOGHISTORY 292
LOGFILE
   GROUP 1 '/oradata4/redo01.log'   SIZE 1024M,
   GROUP 2 '/oradata4/redo02.log'   SIZE 1024M,
   GROUP 3 '/oradata4/redo03.log'   SIZE 1024M
-- STANDBY LOGFILE
DATAFILE
   '/oradata4/system01.dbf',
   '/oradata4/sysaux01.dbf',
   '/oradata4/undotbs01.dbf',
   '/oradata4/users01.dbf'
CHARACTER SET JA16EUC;
```

New

```
CREATE CONTROLFILE SET DATABASE "NEW" RESETLOGS ARCHIVELOG
      MAXLOGFILES 16
      MAXLOGMEMBERS 3
      MAXDATAFILES 100
      MAXINSTANCES 8
      MAXLOGHISTORY 292
LOGFILE
   GROUP 1 '/oradata4/redo01.log'   SIZE 1024M,
   GROUP 2 '/oradata4/redo02.log'   SIZE 1024M,
   GROUP 3 '/oradata4/redo03.log'   SIZE 1024M
-- STANDBY LOGFILE
DATAFILE
   '/oradata4/system01.dbf',
   '/oradata4/sysaux01.dbf',
   '/oradata4/undotbs01.dbf',
   '/oradata4/users01.dbf'
CHARACTER SET JA16EUC;
```

- Change the directories for the LOGFILE and DATAFILE files to those for the test database.
- Delete all except for CREATE CONTROLFILE statements (trace file header section, RECOVER statements, ALTER statements).

(2) After making the changes, store the scripts in the test environment (any desired location)

1-4. Acquire a hot backup of all data files in the production database

(1) Confirm the data files comprising the production database.

```
SQL> select file_name,tablespace_name from dba_data_files;
```

(2) Set all tablespaces confirmed in the above step (1) to backup mode.

```
SQL> alter tablespace <TABLESPACE_NAME> begin backup;
```

(3) Confirm that the tablespaces subject to backup are in backup mode.

```
SQL> select tablespace_name,file_id,v$backup.status
2>      from dba_data_files,v$backup
3>      where dba_data_files.file_id = v$backup.file#;
```

(4) Obtain the backup using the OS copy command.

```
$ cp <FILE_NAME> <BACKUP_FILE_NAME>
```

(5) Cancel backup mode set in step (2) above.

```
SQL> alter tablespace <TABLESPACE_NAME> end backup;
```

Move the online backup to the directory for the test database.

NOTE: You do not need to copy the control file or online redo log file.

TIPS: You can also use other mechanisms such as datapump, logical /physical standby, etc for restoring to the same logical point in time as capture.

2. Capture the workload from the production database

Described below is the method for obtaining the workload from the production database.

2-1. Create directory objects

Create directory objects using the production database.

```
SQL> create directory <DIRECTORY_OBJECT_NAME>
2>      as '<DIRECTORY_OBJECT_PATH>';
```

2-2. Set filters for workload capture

Example:

```
SQL> begin
2>     DBMS_WORKLOAD_CAPTURE.ADD_FILTER (
3>     fname          => 'TPCCONLY',
4>     fattribute     => 'USER',
5>     fvalue         => 'TPCC'
6>     );
7>     end;
8>     /
```

2-3. Initiate workload capture

```
SQL> begin
2>     DBMS_WORKLOAD_CAPTURE.START_CAPTURE (
3>     name           => '<CAPTURE_NAME>',
4>     dir            => '<DIRECTORY_OBJECT_NAME>',
5>     duration       => NULL,
6>     default_action => 'EXCLUDE',
7>     auto_unrestrict => TRUE
8>     );
9>     end;
10>    /
```

2-4. Terminate workload capture

```
SQL> exec DBMS_WORKLOAD_CAPTURE.FINISH_CAPTURE ();
```

3. Create the test database

3-1. Archive current online redo logs for the production database

```
SQL> alter system archive log current;
```

3-2. Copy the archived redo logs

Copy the archived redo logs for the production database to the location where archived logs of the test database are stored (initialization parameter log_archive_dest_n specified in step 1-1).

3-3. Create the test database

Change the environment variable, ORACLE_SID, to the SID of the migration destination database using the migration destination machine.

(1) Launch SQL* Plus as a sys user or internal user and start the instance using NOMOUNT.

```
SQL> startup nomount;
```

(2) Execute the CREATE CONTROLFILE scripts produced in step 1-3.

(3) Obtaining the SCN at the startup of workload capture from the production database.

NOTE: The SCN at the startup of workload capture can be obtained by querying DBA_WORKLOAD_CAPTURES static data dictionary views, and checking wcr_xx.text or wcr_xx.html located in the directory object created in step 2-1.

(4) Using RMAN, perform an incomplete recovery until just before the startup of workload capture from the production database and create a test database.

```
$ rman target /
RMAN> recover database until scn    *the SCN number that was obtained above (3)
RMAN> alter database open resetlogs;
```

(5) Create a temporary tablespace.

Example

```
SQL> ALTER TABLESPACE TEMP ADD TEMPFILE
2>      '/oradata4/tpc1_9i/temp01.dbf'
3>      SIZE 1024M REUSE AUTOEXTEND OFF;
```

4. Upgrade the Test Database

Upgrade the test database from 9.2.0.8 to 11.1.0.7.

4-1. Installing Oracle Database 11*g* to the test environment

Install Oracle Database 11*g* R1 to the test environment.

NOTE: For more information on the installation method, refer to the installation guide for the applicable platform.

4-2. Upgrade the Test Database

Upgrade the test database from 9*i* to 11*g*.

- DBUA was used in our verification testing.

NOTE: A database can be upgraded to Oracle Database 11*g* R1 (11.1) only if the version of the time zone files is 4 or later. To replace older time zone files with version 4 files, apply “patch5632264 - TZ V4 file.” For detailed information on patch5632264 and its implementation, refer to “Note 413671.1: Applying 'version 4' Time Zone Files on an Oracle Database,” “Note 359145.1: Links to download utltzuv2.sql,” “Note 396397.1: Explanation,” and “Note 414590.1: FAQ and Issues.”

4-3. Apply Patch Set Release (PSR) 11.1.0.7 and upgrade the database

Apply Patch Set Release (PSR) 11.1.0.7 and upgrade the database. Read the README file for PSR for more information on applying the patch.

5. Executing Database Replay with the test database

5-1. Preprocessing workloads

5-2. Replaying workloads

5-3. Analysis and report

For more information on procedures for pre-processing workloads, replaying the workload, and analyzing and producing reports using Enterprise Manager, refer to the manual entitled *Oracle® Database Real Application Testing User's Guide 11g Release 1 (11.1) E12253-01*.

6.2.4. Load created by workload capture

Verification details

When workload capture starts, the existing server processes do more work which is required to capture the workload data, generating overhead. The effects of workload capture on business transactions can be examined by comparing the order processing performance (throughput, response, CPU usage, workload output device disk I/O) of the OLTP model while workload is captured and while no workload is captured. Capture files are directed to an IO system different than the one on which database files are stored.

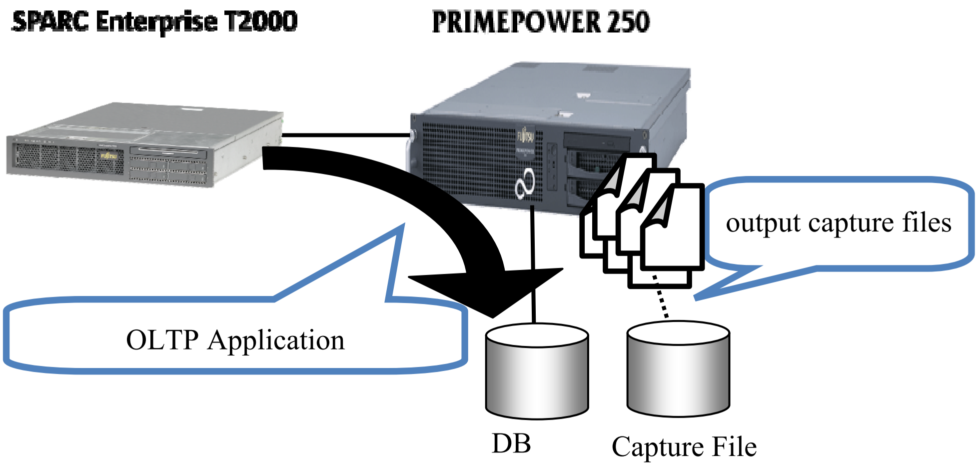


Figure 6-14 Overview of measurement method

Verification results

- **Throughput, response**

  Figure 6-15 shows application throughput and response performance during workload capture and no workload capture in progress.

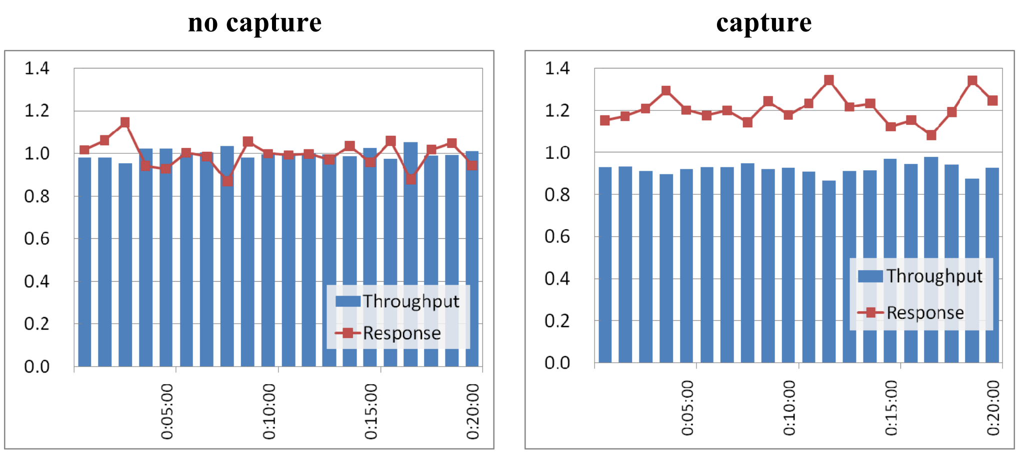


Figure 6-15 Throughput and response with and without workload capture

Table 6-7 Average throughput and average response

capture	throughput	response
No	1.00	1.00
Yes	0.93	1.21

*A value of "1.00" is assigned to performance without workload capture.

When compared to the performance achieved while no workload is captured, throughput merely decreased by approximately 7% due to workload capture, but response times increased about 20%. We suspect workload capture had such large effects because the response was originally less than 100 milliseconds.

- **CPU usage**

  Figure 6-16 shows the change in CPU usage over time.

  The average CPU usage without workload capture was 83.4%, while the average CPU usage with workload capture was 85.7%.

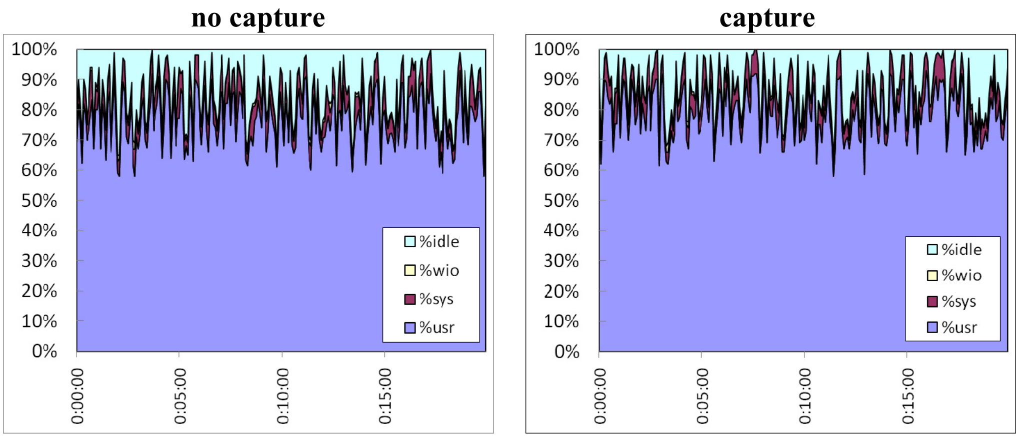


Figure 6-16 CPU usage with and without workload capture

Table 6-8 Average CPU usage during measurement

capture	CPU usage（sys+user）
No	83.4%
yes	85.7%

Compared to the performance achieved while no workload was captured, the CPU usage increased only 2.7% due to workload capture, indicating virtually no effects on CPU usage.

- **Disk I/O for workload output device**

  Figure 6-17 shows a comparison of the disk I/O for the workload output device and the disk busy rate.

  When work load was captured, the average write amounts to the workload output device was about 230 KB/s, and the disk busy rate was nearly 0%. This indicates minimal effects.

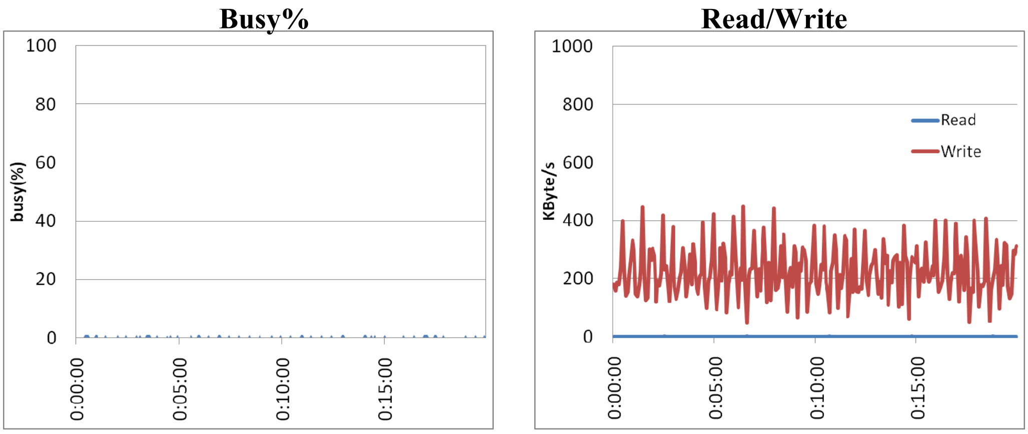


Figure 6-17 Workload output device disk I/O

These results indicate that workload capture generated relatively modest overhead in CPU usage and disk I/O for the workload output device. Nevertheless, note that actual trends may vary, depending on the nature of the transactions to be captured.

6.2.5. Effects of replay parameters

Verification details

The think_time_scale replay parameter enables changes in the scale of elapsed time between user calls. By setting a scale different from the one used during workload capture, the intervals for SQLs being issued can be changed to control system loads.

Replay was performed with the think_time_scale parameter set to 100 and set to 50 (see Fig. 6-18). When the parameter was set to 100, SQL statements were issued at the same intervals as when the workload was captured. However, when the parameter was set to 50, the SQL statement issuance intervals decreased to half compared to when the workload was captured. Reducing the value set in the think_time_scale parameter executes SQL statements in shorter intervals, increasing system loads.

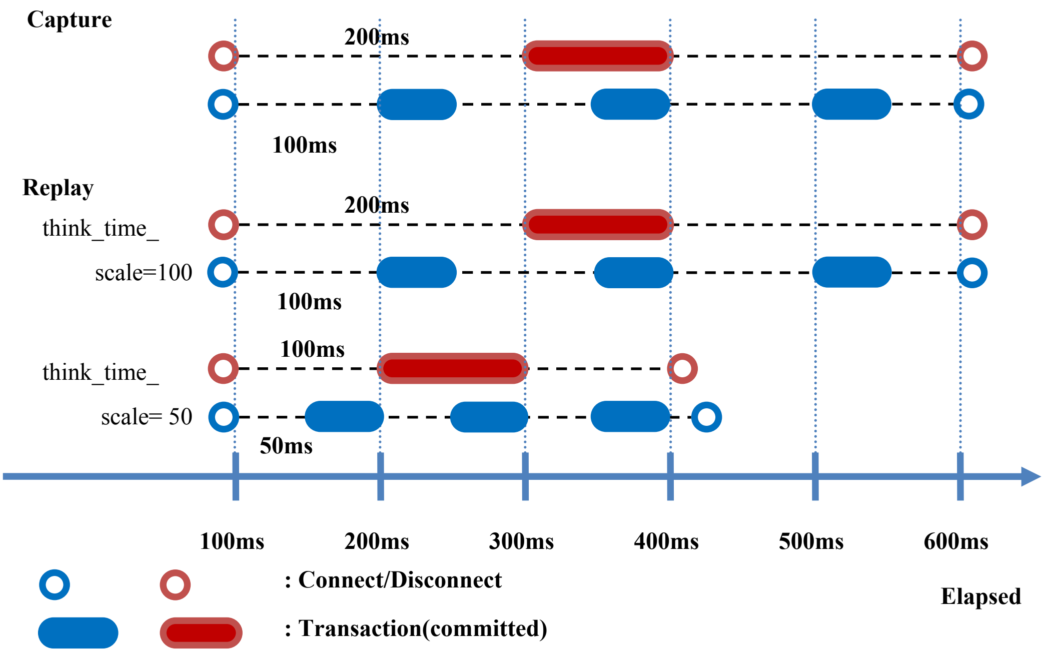


Figure 6-18 Effects of think_time_scale parameter

Verification results

- **CPU usage**

  Figure 6-19 shows the change in CPU usage over time when the think_time_scale parameter is set to 100 and 50.

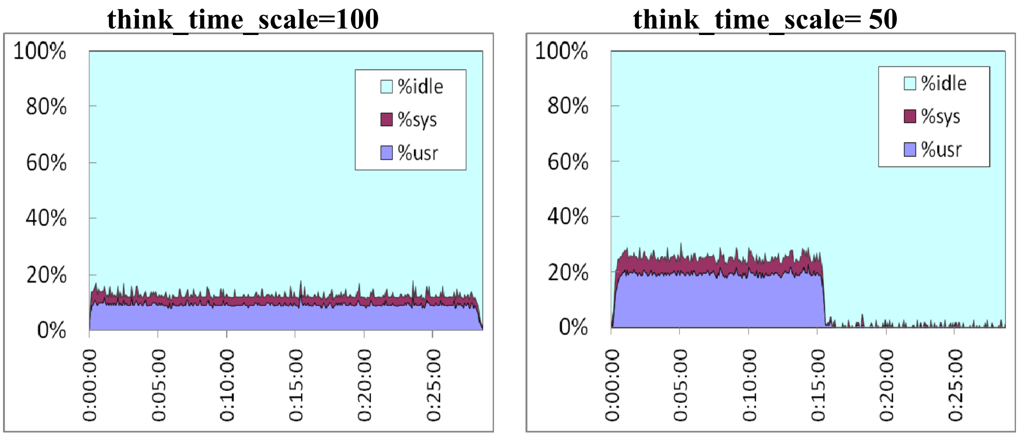


Figure 6-19 Comparison of CPU usage based on parameter value

When the think_time_scale parameter is set to 100, the average CPU load is about 12% and the replay time for the entire process is approximately 28 minutes. However, when the think_time_scale parameter is set to 50, the average CPU usage shows approximately 23%, while the replay time for the entire process is approximately 16 minutes.

This confirms that setting the think_time_scale parameter to 50 halves the SQL statement issuance intervals for this particular application being tested, resulting in shorter replay times and increasing CPU usage. Note that in some cases, the call issuance rate may be application dependent.

- **Network traffic and disk busy rate**

  Figure 6-20 illustrates network traffic, while Figure 6-21 shows disk busy rates.

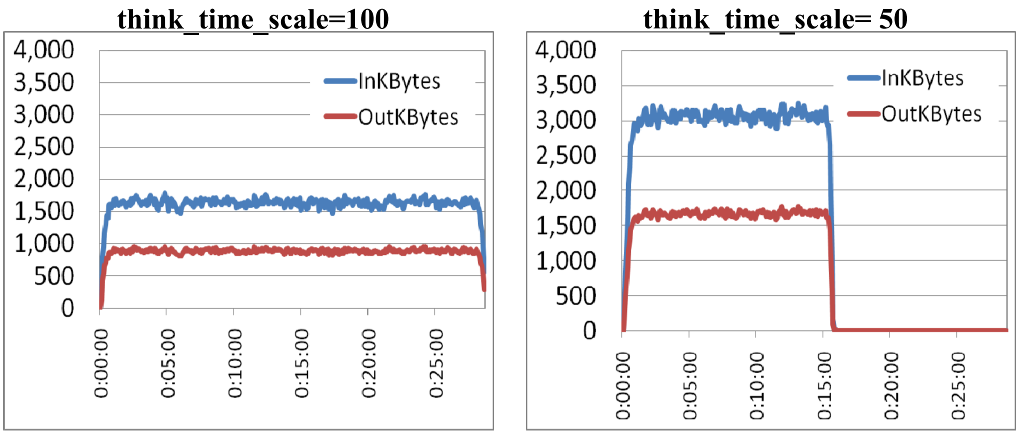


Figure 6-20 Comparison of network traffic based on parameter value

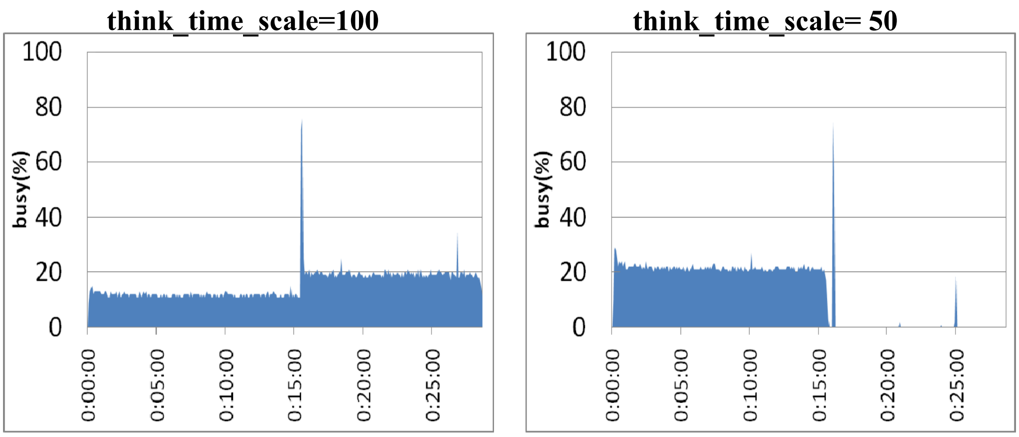


Figure 6-21 Comparison of disk busy rates based on parameter value

Figure 6-20 and Figure 6-21 indicate that decreasing the SQL statement issuance intervals by half also results in a corresponding increase in CPU usage.

These indicate that system loads in general can be controlled by adjusting the think_time_scale parameter to change the SQL statement issuance intervals in the case of transactions that include user think time (e.g., the OLTP model).

6.2.6. Workload replay accuracy

Verification details

When workload is replayed in a system identical to the system used to capture that workload, no data or error divergence should be observed. A small number of divergence instances may be OK. The amount of tolerable divergence depends on the environment. It is recommended that the user of database replay check key that reflect key application data after the workload replay is complete. For example, a table that contains the number of completed orders is a candidate for determining the accuracy of replay.

Described below is a verification of the effects of the synchronization parameter for capture and replay using the same hardware.

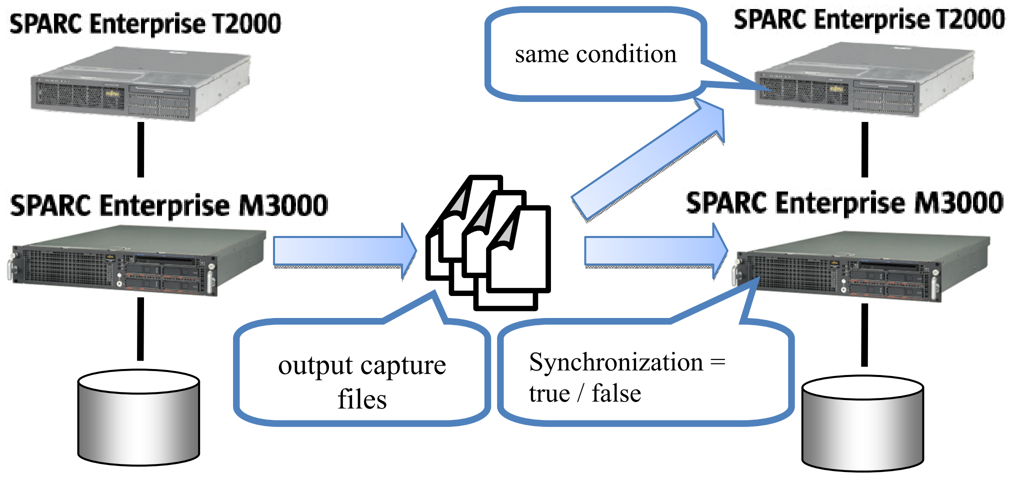


Figure 6-22 Overview of verification test

Verification results

- **Errors and data divergence**

  The synchronization parameter controls whether to retain the commit ordering for captured workload (see Figure 6-23). If this parameter is true, transactions to be replayed are executed after the commits of dependent transactions.

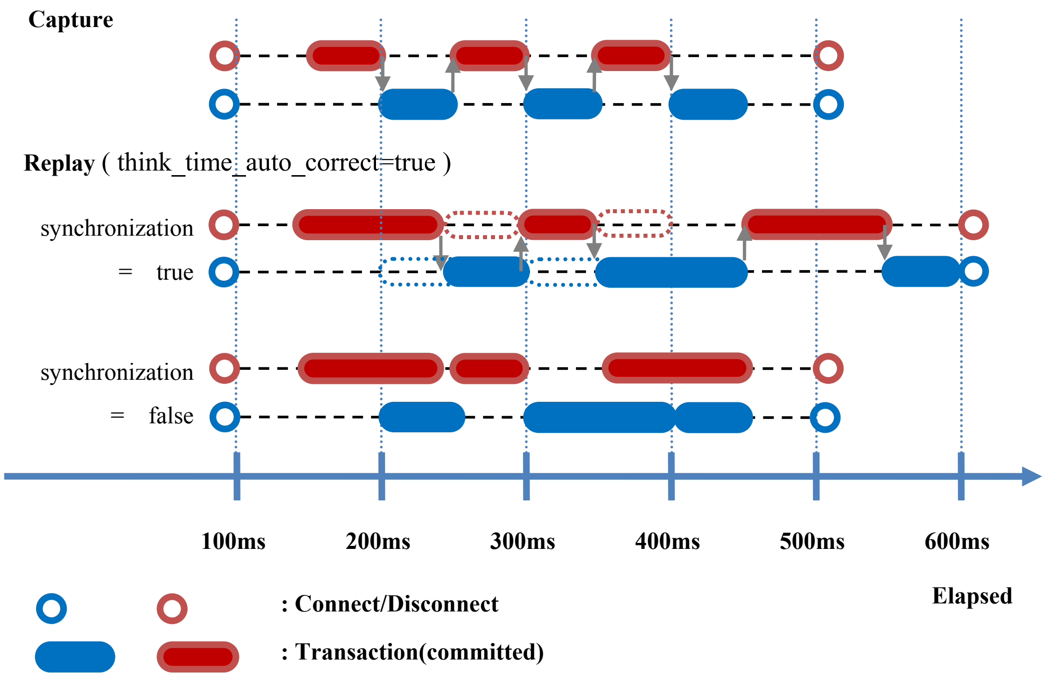


Figure 6-23 Effects of synchronization parameter

Shown below is a comparison of results when the commit ordering is and is not retained.

Figure 6-24 shows the measurements obtained when the commit ordering is and is not retained.

synchronization=true

Errors Mutated During Replay	0	0.00
DMLs with Different Number of Rows Modified	0	0.00
SELECTs with Different Number of Rows Fetched	0	0.00

the total number of replayed call: 1,654,410

Synchronization=false

Errors Mutated During Replay	0	0.00
DMLs with Different Number of Rows Modified	1	0.00
SELECTs with Different Number of Rows Fetched	27	0.00

the total number of replayed call: 1,660,787

Figure 6-24 Report output result (section where data differs)

When the commit ordering is retained, there is no data divergence on replay as well as on capture.

When the commit order is not retained, there are 28 data divergence on replay.

These results indicate that not retaining the commit order show that data differences can arise in DML and SELECT statements since the sequence of transactions is ignored during execution.

- **Effects of replay accuracy improvement patch**

  Figure 6-25 shows the results for replaying transactions in the Oracle Database 11g OLTP model with the commit order retained (synchronization = TRUE).

  The elapsed time on replay takes longer than on capture, but to rectify the replay performance problem, Oracle provided an interim patch (One-Off Patch #7240198) for Oracle Database 11*g* (11.1.0.7).

  Figure 6-25, Figure 6-26, and Figure 6-27 compare measurements before and after the application of the interim patch.

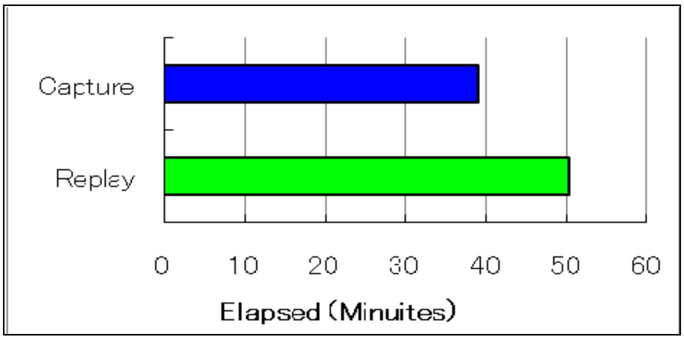


Figure 6-25 Capture and replay times before patch application

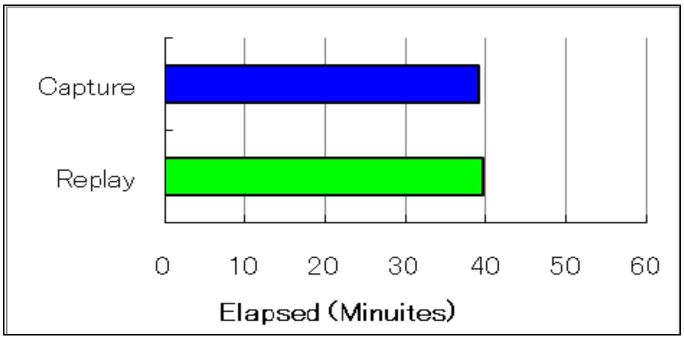


Figure 6-26 Capture and replay times after patch application

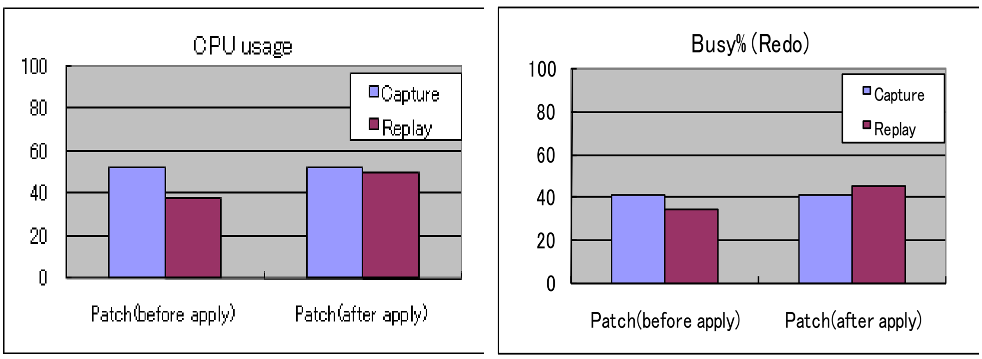


Figure 6-27 CPU usage and disk busy rates before and after patch application

The comparison of replay times (see Figure 6-25 and Figure 6-26) indicates that replay times and capture times were roughly equal following patch application and that CPU usage and disk busy rates during replay remained roughly equal to those during capture.

These results demonstrate the need to apply the interim patch, because this patch improves the accuracy of the elapsed time on replay and renders capture time and replay time roughly equal. This is particularly important if you expect to use Database Replay to compare performance between the old and new system configurations.

6.2.7. Summary of Database Replay

Summarized below are the results of the database replay verification tests.

- The overhead generated during capture is minimal. Its effects on production environment performance is negligible.
- In the case of OLTP model transactions, the think_time_scale parameter enables changes in the scale of elapsed time between user calls, thereby allowing replays with different loads.
- Setting the synchronization parameter to TRUE allows the confirmation of differences in errors and differences in data between capture and replay. Using Database Replay to evaluate the destination system, this parameter enables confirmation of SQL-level application operations and incompatibilities.
- When the synchronization parameter is set to FALSE, data divergence may arise because this does not retain the commit order. This parameter should be set to FALSE only when measuring application loads.
- Check with Oracle Support if there are any recommended patches on top of Oracle Database Release 11.1.0.7, this will minimize effort and rediscovery of any already known product defects. At the time of writing this paper, applying the Oracle recommended interim patch #7240198 improved replay performance when the synchronization parameter is set to TRUE, rendering replay times approximately equal to capture times. The interim patch is being incorporated by Oracle development into a future release of the product.

6.3. SQL Performance Analyzer

In verification tests performed using SQL Performance Analyzer (hereinafter called "SPA"), we obtained SQL workloads using Oracle9*i* Database, then examined the procedure for implementing SPA by Oracle Database 11*g*, remediating regressed SQL statements using SQL Tuning Advisor (hereinafter called "STA"), and SQL Plan Baselines functionalities.

6.3.1. Details of functions

Executing SQL statements before and after a change in the system environment allows SPA to analyze and report on the detailed results of changes in the system environment on the SQL statement execution plan and performance based on a number of measurement criteria.

To use the SPA, the following five generic steps must be followed:

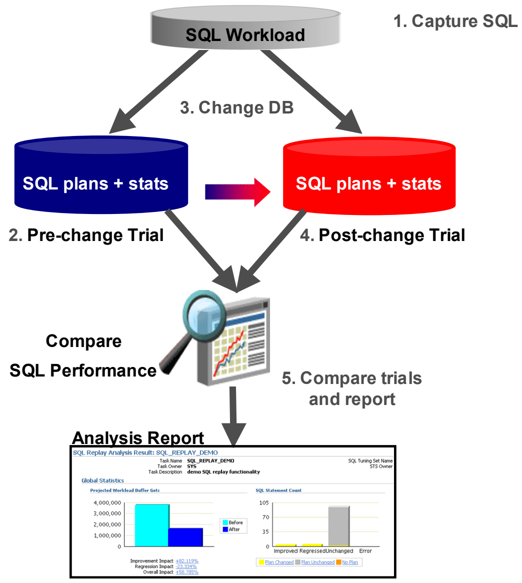


Figure 6-28 Main processing procedures for SQL Performance Analyzer

1. Capture the SQL workload that you want to analyze with SPA. The Oracle database offers ways to capture SQL workload from several sources, such as cursor cache and Automatic Workload Repository, SQL Trace files into a SQL tuning set (STS). This would typically be done on a production system and the STS would then be transported to the test system where SPA analysis will take place.
2. Measure the performance of the workload before a change by executing SPA trial on the SQL tuning set.
3. Make the change, such as database upgrade or optimizer statistics refresh.
4. Measure performance of the workload after the change by executing SPA trial on the SQL tuning set again.
5. Compare performance of the two executions of the SQL tuning set to identify the SQL statements that have regressed, improved, or were unchanged.

If the SPA detects SQL performance degradation, the user can correct the degradation-causing SQL statement using SQL Tuning Advisor (STA)[3] or the SQL plan baseline (a plan stability feature first introduced in Oracle Database 11*g*).

The SPA enables multi-faceted comparison and analysis of SQL performance before and after a change is made in database using a number of metrics.

Shown below are the available metrics.

Table 6-9 SQL Performance Analyzer comparison metrics

Metric	Dimension	Repeatable	Comprehensive
PARSE_TIME	CPU Used in Parsing Time	YES	NO
ELAPSED_TIME	Time	NO	YES
CPU_TIME	CPU Used in Executing	YES	YES
BUFFER_GETS	Logical I/O	YES	YES
DISK_READS	Physical I/O Reads	NO	NO
DIRECT_WRITES	Direct Path Writes	NO	YES
OPTIMIZER_COST	Cost Estimate	YES	NO

Since certain comparison metrics depend heavily on the environment (including buffer cache conditions), repeated testing will not necessarily produce the same results. Those metrics are indicated by “Repeatable: No.” The comparison metrics which fully describe one dimension of a SQL statement’s performance are indicated by “Comprehensive: Yes.”

“BUFFER_GETS” metric while repeatable and comprehensive, does not accurate translate physical I/O. “CPU_TIME” metric that indicates “Repeatable: Yes” and obtains statistic for SQL performance. However, this metric does not account for I/O, and the CPU times required to execute SQL statements in an Oracle9*i* environment can be affected by SQL trace capture. We strongly recommend performing analyses using both “CPU_TIME” and “BUFFER_GETS”.

[3] SQL Tuning Advisor requires Tuning Pack license to be used. Diagnostic Pack license is the prerequisite for Tuning Pack license.

6.3.2. Prerequisites for SQL Performance Analyzer

Account for the following when executing SPA.

- **Data match**

  The application/data conditions in the test environment are most useful when they are logically identical to those in the production environment.

- **Patch application**

  Acquiring workloads using Oracle9*i* Database and executing SPA using Oracle Database 11*g* requires Oracle Database 11*g* version 11.1.0.6 + One-Off Patch #6865809, or 11.1.0.7 or later. No patch is required for Oracle9i database.

  For detailed information on patch application, refer to "MetaLink Note: 560977.1."

6.3.3. Preparing the SPA executing environment

Described below are the preparation procedures, ranging from capture of SQL workloads from the production environment (Oracle9*i* Database) to the creation of STS using the test environment (Oracle Database 11*g*).

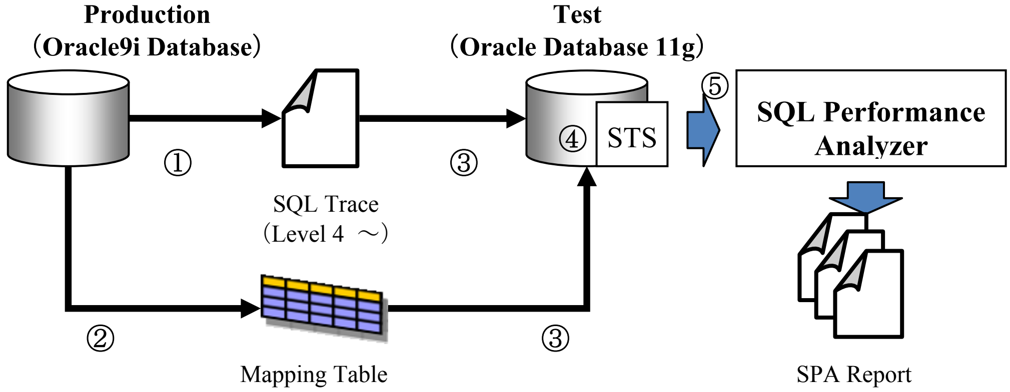


Figure 6-29 Overview of SQL Performance Analyzer

① To capture SQL workloads from the production environment (Oracle9*i* Database), enable the SQL trace and obtain an SQL trace file.

The SQL trace can be enabled by the following SQL statement.

```
SQL> alter session set events '10046 trace name context forever, level 4';
```

The SQL trace can be disabled by the following SQL statement.

```
SQL> alter session set events '10046 trace name context off';
```

② In the SQL trace file, objects and users are identified by object ID. Converting the SQL trace to STS in the test environment (Oracle Database 11*g*) requires mapping information for object IDs and object names. This means a mapping table must be produced in the production environment (Oracle9*i* Database). Use the following SQL statement to produce a mapping table.

```
SQL> connect system/password
SQL> create table <MappingTable> as
2>       select object_id id, owner, substr(object_name, 1, 30) name
3>       from dba_objects where object_type not in
4>          ('CONSUMER GROUP','EVALUATION CONTEXT',
5>           'JAVA DATA','JAVA RESOURCE','LIBRARY',
6>           'LOB','OPERATOR','PACKAGE','PACKAGE BODY',
7>           'PROCEDURE','QUEUE','RESOURCE PLAN',
8>           'TRIGGER','TYPE','TYPE BODY')
9>       union all
10>      select user_id id, username owner, null name from dba_users;
```

Produce a mapping table in the production environment (Oracle9*i* Database), then export the table.

③ Copy the SQL trace file to a desired directory in the test environment (Oracle Database 11*g*), and create a directory object for that directory.

```
SQL> create or replace directory <DirectoryObject> as
2>    '< SQL Trace file Path>';
```

Import the mapping table produced in step ② above into the test environment (Oracle Database 11*g*). *In our verification test, the table was imported into the system schema.

④ Create STS from the SQL trace file obtained in the production environment (Oracle9*i* Database).

```
SQL> connect / as sysdba
SQL> declare
2>       mycur dbms_sqltune.sqlset_cursor;
3>    begin
4>       dbms_sqltune.create_sqlset ('<STS>');
5>       open mycur for
6>          select value (P)
7>          from table (dbms_sqltune.select_sql_trace (
8>             directory => '< DirectoryObject >',
9>             file_name => '%ora%',
10>            mapping_table_name => '< MappingTable >',
```

```
            mapping_table_owner => 'SYSTEM',
            select_mode => dbms_sqltune.SINGLE_EXECUTION)) P;
    dbms_sqltune.load_sqlset ('<STS>', mycur);
    close mycur;
  end;
  /
```

⑤ Using the STS created in the test environment (Oracle Database 11*g*), execute SPA on the test environment to analyze workloads in the production environment (Oracle9*i* Database) and test environment.

6.3.4. Analysis using SPA

Verification details

To perform SPA verification, we prepared four patterns. Table 6-10 provides an outline of each of these patterns.

Table 6-10 Overview of verification patterns

Pattern	Overview
1	Select the result which joins LIST_1 table and MASTER_1 table, and select data
2	Select the result of subquery which joins LIST_1_2 table, MASTER_1_2_1 table, MASTER_1_2_2 table, and MASTER_1_2_3 table
3	Select row numbers of the result which joins LIST_4 table, MASTER_4_1 tabele, MASTER_4_2 table and MASTER_4_3 table
4	Select all data which satisfy conditions of WHERE clause ※Statistics on SALES table are stale (They were gathered when it had 124 rows, but it actually has 1837810 rows)

For these four patterns, using SPA, we analyzed changes in execution plan resulting from the transition from Oracle9*i* Database's Rule Based Optimizer to Oracle Database 11*g*'s Cost Based Optimizer. We also compared performance.

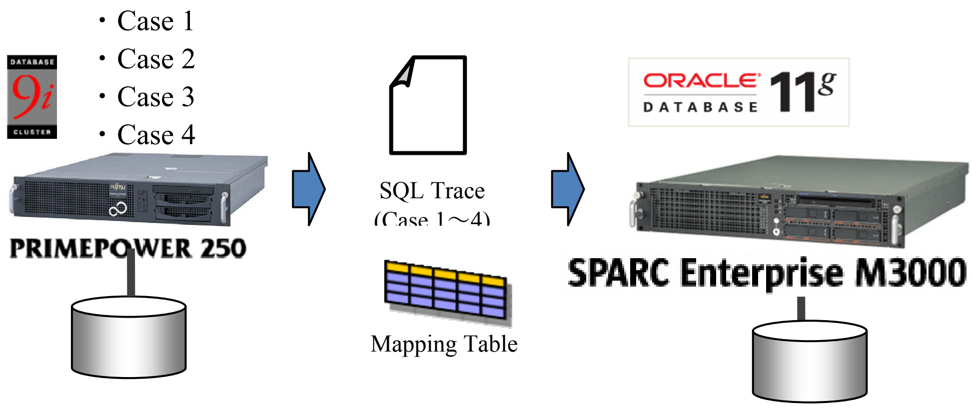
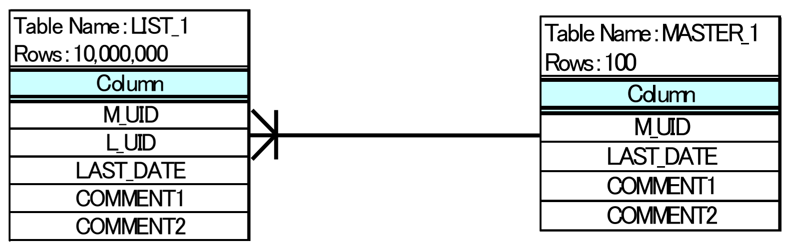


Figure 6-30 Overview of batch processing verification test

Figure 6-31 through Figure 6-34 show the tables and SQL statements prepared for the four patterns.

Case 1

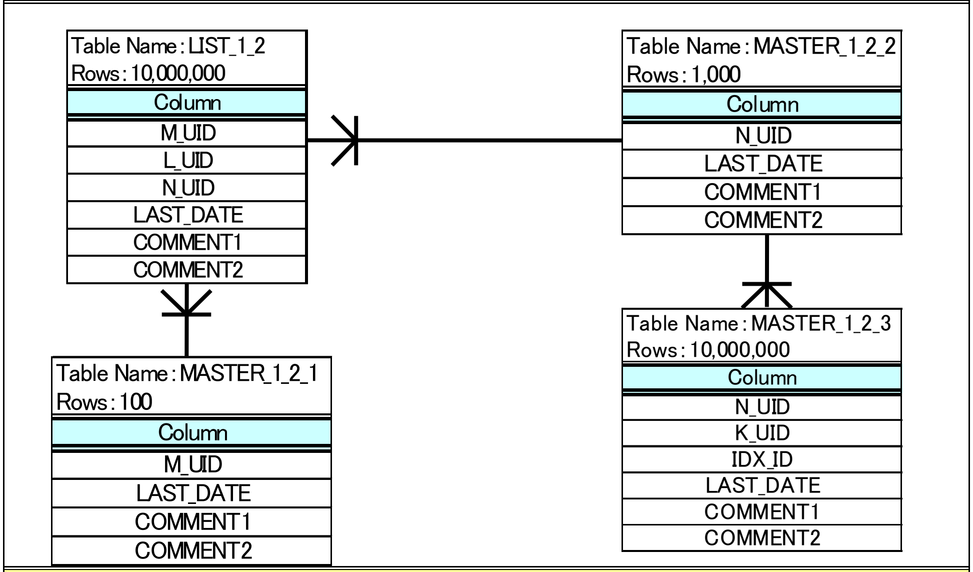


SQL

```
SELECT /* PROBLEM_SQL 1_1 */ T1.COMMENT1, T2.COMMENT1,
COUNT(*)
FROM LIST_1 T1, MASTER_1 T2
WHERE T1.M_UID = T2.M_UIDAND T1.LAST_DATE > SYSDATE –31
GROUP BY T1.COMMENT1, T2.COMMENT1;
```

Figure 6-31 Case 1

Case 2

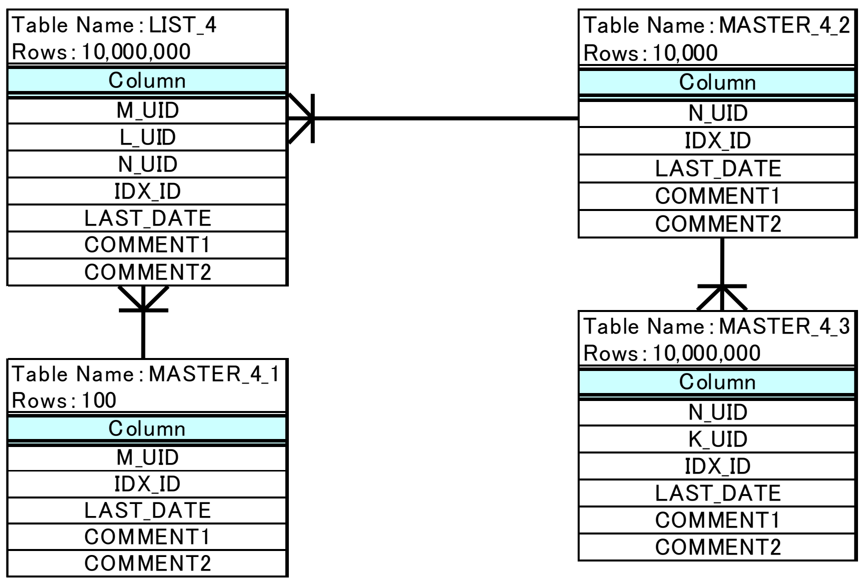


SQL

```
SELECT /* PROBLEM_SQL 1_2 */ * FROM (
  SELECT T2.L_UID, T2.M_UID, T2.N_UID, T4.K_UID,
      T4.IDX_ID, T1.LAST_DATE, T2.COMMENT1, T3.COMMENT2
  FROM MASTER_1_2_1 T1, LIST_1_2 T2, MASTER_1_2_2 T3,
      MASTER_1_2_3 T4
  WHERE T1.M_UID = T2.M_UID AND T2.N_UID = T3.N_UID
      AND T3.N_UID = T4.N_UID AND T4.IDX_ID = 1
  ORDER BY T2.M_UID, T2.N_UID, T2.L_UID)
WHERE ROWNUM < 11;
```

Figure 6-32 Case 2

Case 3



SQL

```
WITH
  SAMPLE4 AS (
    SELECT /* PROBLEM_SQL 4 */ T4.L_UID, T4.M_UID, T4.N_UID,
      T1.K_UID, T1.IDX_ID, T4.LAST_DATE, T3.COMMENT1, T2.COMMENT2
    FROM MASTER_4_3 T1, MASTER_4_2 T2, MASTER_4_1 T3, LIST_4 T4
    WHERE T3.M_UID = T4.M_UID AND T2.N_UID = T4.N_UID
    AND T1.N_UID = T2.N_UID AND T4.IDX_ID BETWEEN 101 AND 200
    ORDER BY T4.M_UID, T4.N_UID, T4.L_UID
) SELECT COUNT(*) FROM SAMPLE4;
```

Figure 6-33 Case 3

Case 4
Table Name : SALES Rows : 1,837,810 Column: PROD_ID, CUST_ID, TIME_ID, CHANNEL_ID, PROMO_ID, QUANTITY_SOLD, AMOUNT_SOLD
SQL
SELECT * FROM SCOTT.SALES WHERE TIME_ID < '1998-01-10';

Figure 6-34 Case 4

Verification results (Pattern 1)

The comparison report (Figure 6-35) for Pattern 1 shows improvements in buffer read performance and degradation in elapsed time and CPU time. The comparison of execution plans indicates a specific change: NESTED LOOP was selected in Oracle9*i* Database, while HASH JOIN was selected in Oracle Database 11*g*.

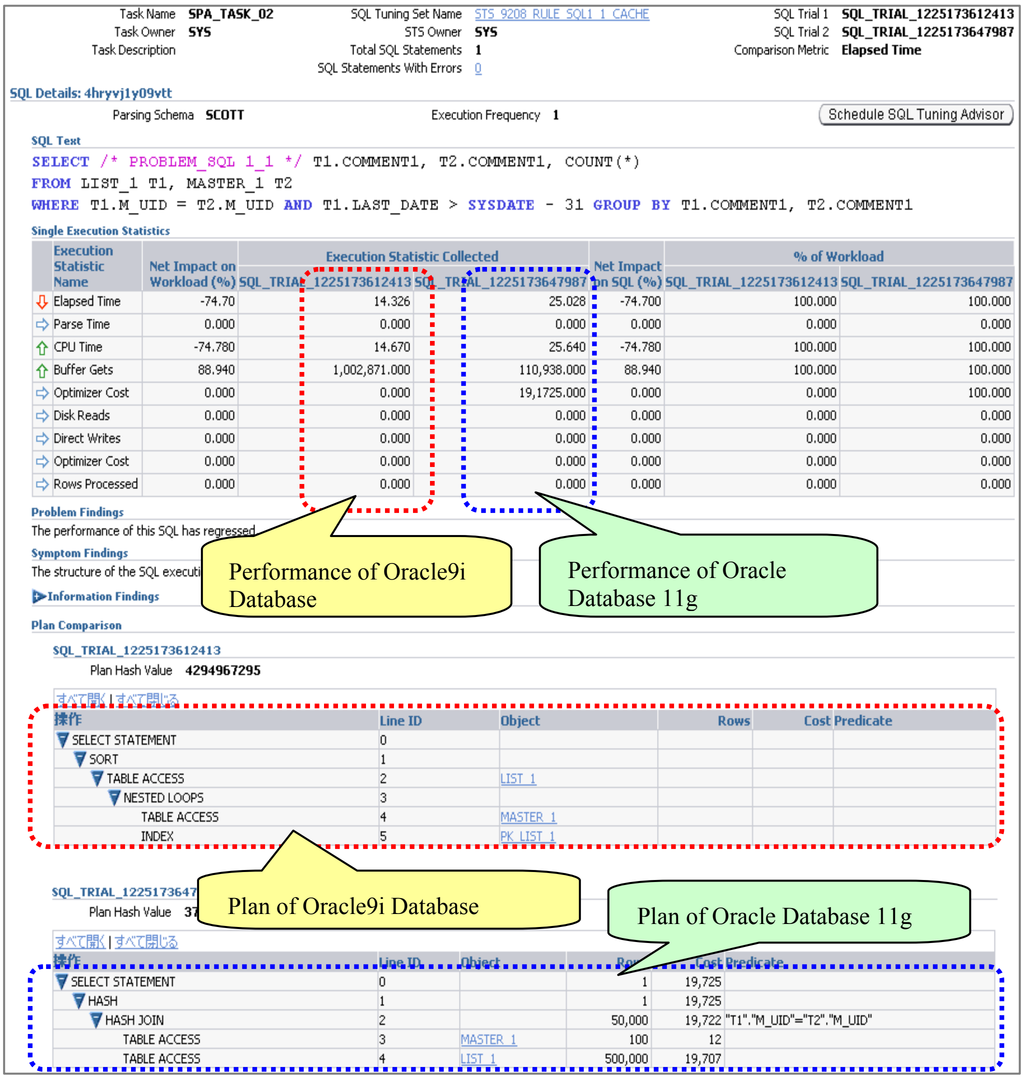


Figure 6-35 Comparison report for Pattern 1

Figure 6-11 Elapsed time for Pattern 1

Oracle9i Database	Oracle Database 11g
0:00:14.32	0:00:25.02

Verification results (Pattern 2)

The comparison report (Figure 6-36) for Pattern 2 indicates improvements in elapsed time, CPU time, and buffer read performance. The comparison of execution plans indicates a specific change: as in the case of Pattern 1, NESTED LOOP was selected in Oracle9*i* Database, while HASH JOIN was selected in Oracle Database 11*g*.

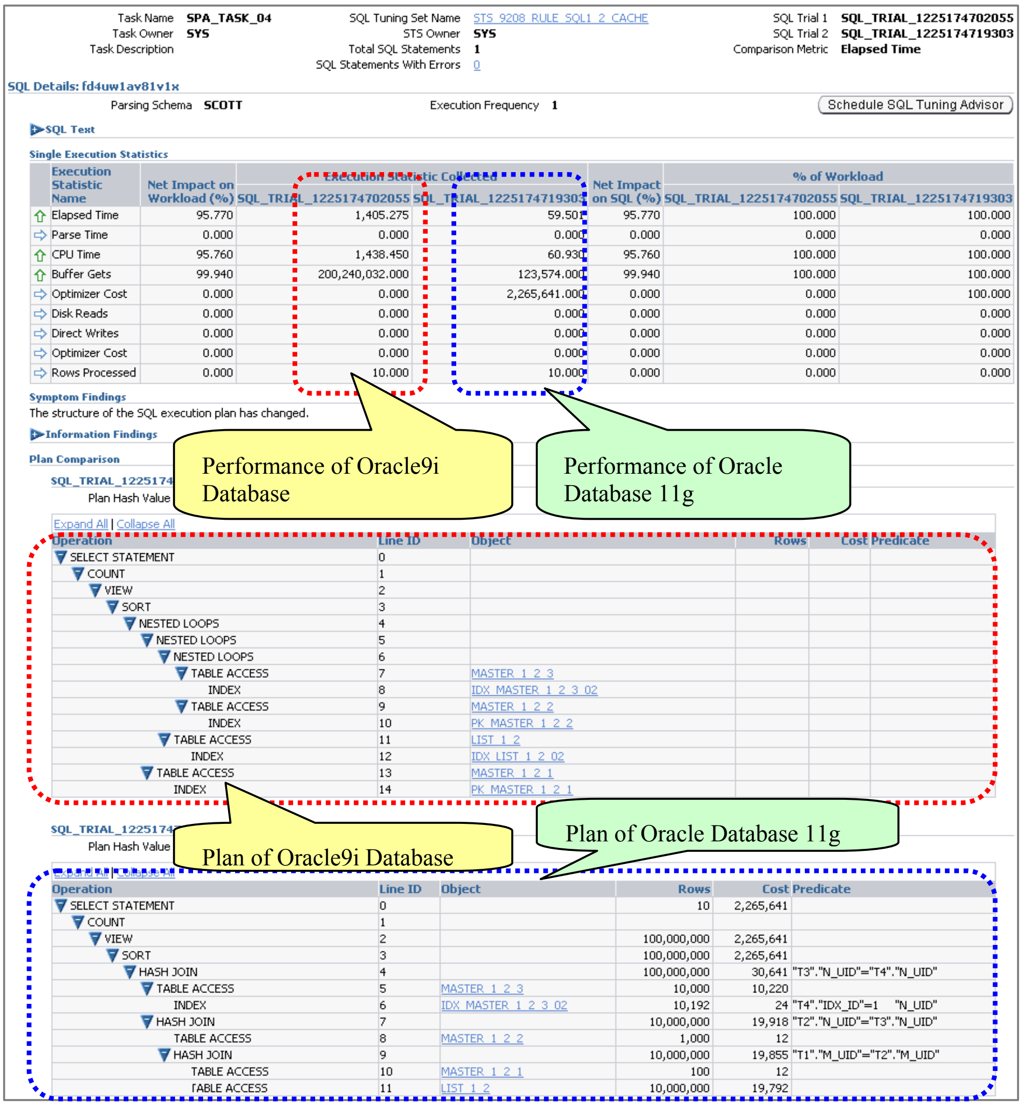


Figure 6-36 Comparison report for Pattern 2

Table 6-12 Elapsed time for Pattern 2

Oracle9i Database	Oracle Database 11g
0:23:25.28	0:00:59.50

Verification results (Pattern 3)

The comparison report (Figure 6-37) for Pattern 3 indicates improvements in elapsed time, CPU time, buffer read performance, and disk read performance. The comparison of execution plans indicates a specific change: NESTED LOOP was selected in Oracle9*i* Database, while HASH JOIN and MERGE JOIN were selected in Oracle Database 11*g*.

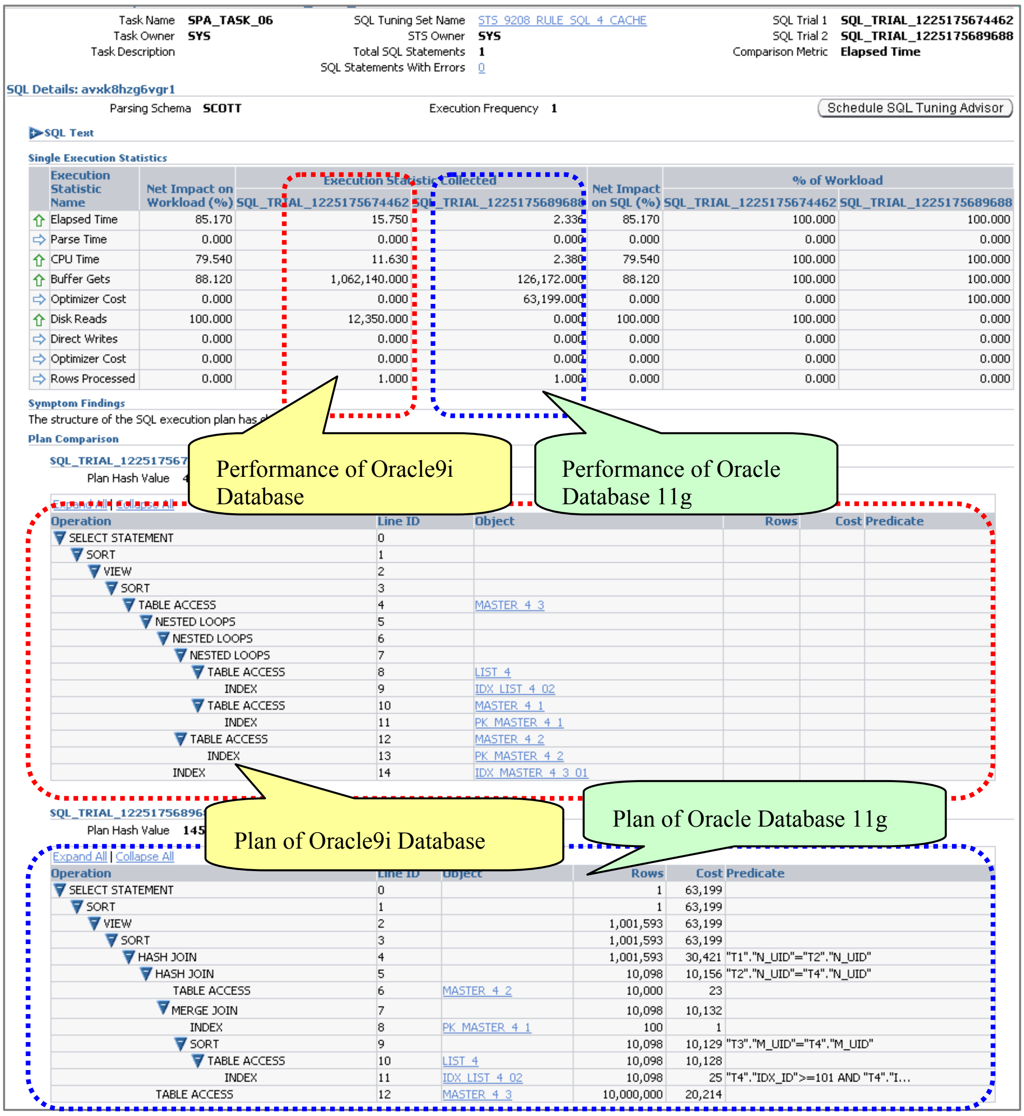


Figure 6-37 Comparison report for Pattern 3

Table 6-13 Elapsed time for Pattern 3

Oracle9i Database	Oracle Database 11g
0:00:15.75	0:00:02.34

Verification results (Pattern 4)

The comparison report (Figure 6-38) for Pattern 4 indicates degradation in elapsed time, CPU time, buffer read performance, and disk read performance. The comparison of execution plans indicates a specific change: INDEX SCAN was selected in Oracle9*i* Database, while FULL SCAN was selected in Oracle Database 11*g*.

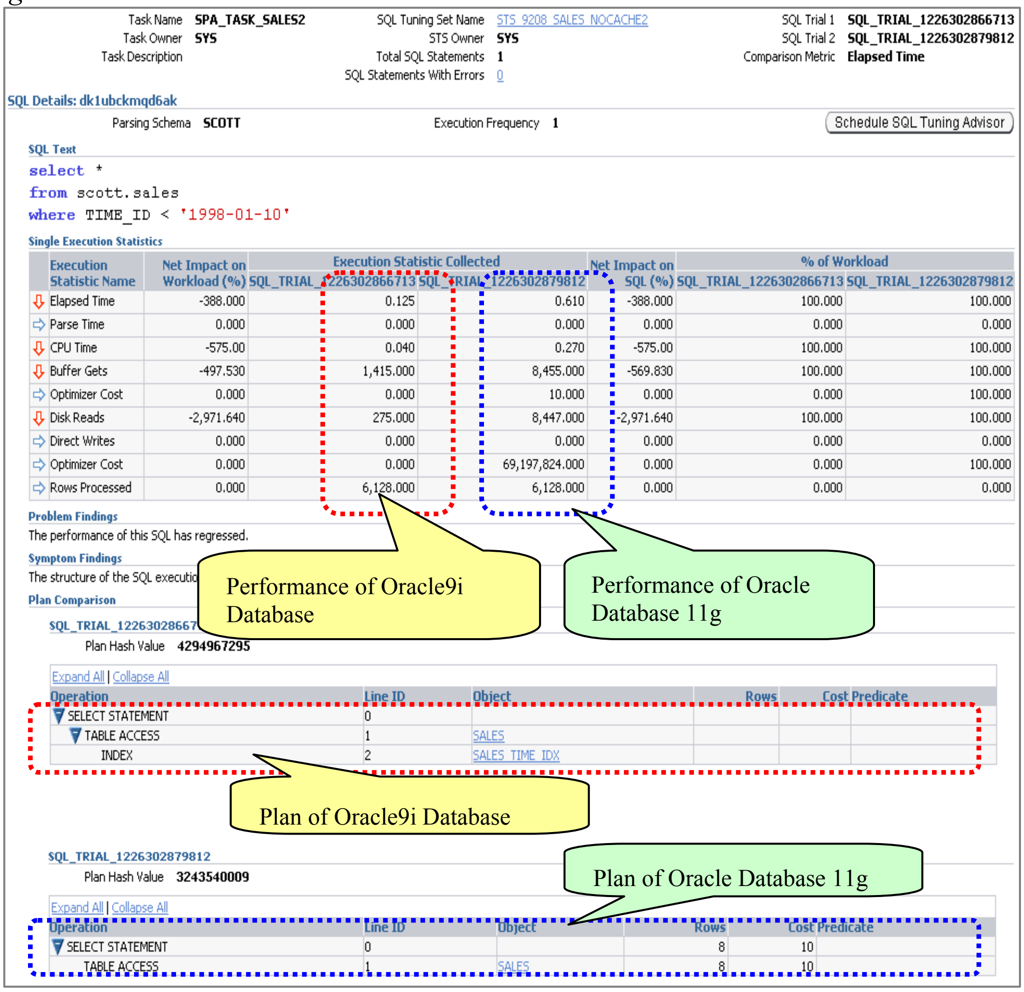


Figure 6-38 Comparison report for Pattern 4

Table 6-14 Elapsed time for Pattern 4

Oracle9i Database	Oracle Database 11g
0:00:00.13	0:00:00.61

As indicated above, performance improvements were realized in Pattern 2 and Pattern 3 due to the upgrade from Oracle9*i* Database to Oracle Database 11*g*. In particular, in Pattern 2, performance improved by some 23-fold after the migration due to the selection of the most suitable execution plan. In Pattern 1 and Pattern 4, the migration degraded performance despite higher hardware performance.

Introduced below are countermeasures for SQL statements with degraded performance and the results for verification tests.

6.3.5. SQL tuning

Execution of STA from SPA result report

If performance degradation is observed after SPA execution, the report screen displays the Run SQL Tuning Advisor button. This button can be used to initiate performance tuning from the STA.

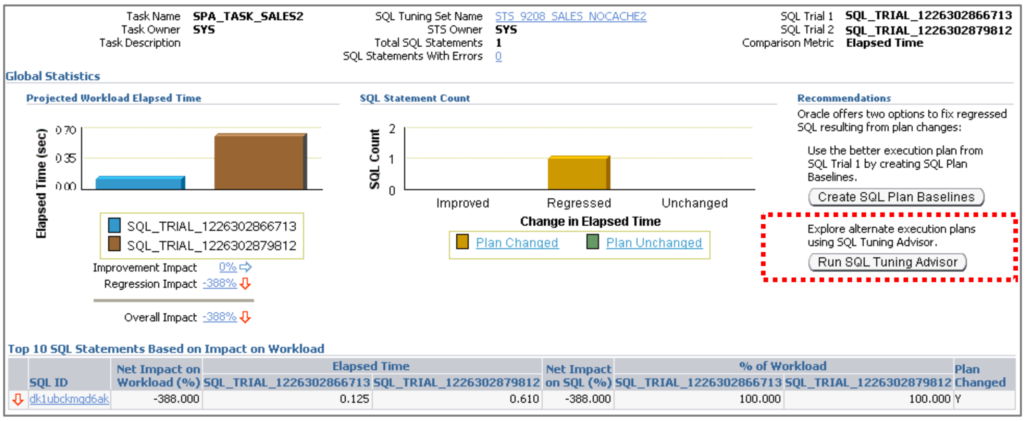


Figure 6-39 STA execution

Shown below are the results for executing STA tuning for Pattern 4 in which performance degradation was observed.

Verification results

The execution of STA clearly shows that the optimizer's statistical information is outdated for the SALES table and indexes, and that a more suitable execution plan is possible. The improved execution plan is implemented.

The items for which statistical data were to be obtained for the table were checked and implemented.

Figure 6-40 STA advice

Following implementation, statistical data is collected once again. At this point, a retrial can be carried out for applicable SPA tasks and the results compared.

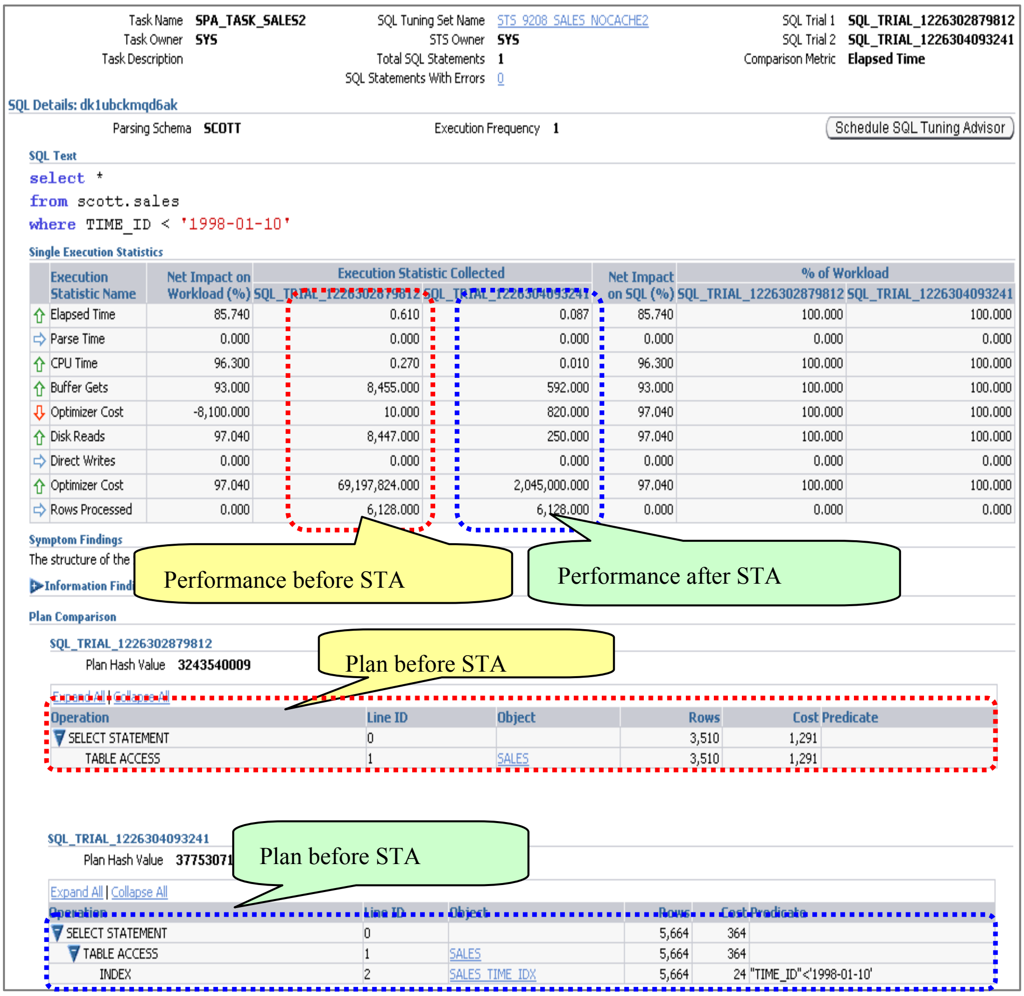


Figure 6-41 SPA comparison reports before and after STA implementation

The results of SQL comparison in Pattern 4 before and after STA implementation show that FULL SCAN was selected before STA implementation but INDEX SCAN after, a change that improved performance in areas such as elapsed time and CPU time.

6.3.6. Stabilizing the execution plan

Cases requiring execution plan stabilization

If your current system provides adequate performance and you wish to minimize the risk of performance degradation due to changes in execution plans during the upgrade to a new system, it is possible to capture stored outlines before the upgrade using hint clauses or plan stability. This will prevent changes in execution plans during the migration to a new system.

However, note that if execution plans are stabilized by hint clauses or plan stability, old execution plans will continue to be used even if more suitable execution plans or new indexes become available.

Oracle Database 11*g* delivers a feature called SQL Plan Management that gives the best of both worlds. It not only prevents performance degradation caused by unexpected changes in execution plans but also manages changes in execution plans. By setting outlines as SQL plan baselines, this maintains the execution plans for the current system and allows performance comparisons with the new system.

Shown below are verification results for the migration of execution plans using outlines and SQL plan management. These verification tests were performed for Pattern 1 in which performance degradation occurred due to changes in execution plans.

Verification procedure and results

Steps (1) through (3) are performed with an existing system (Oracle9*i* Database).

(1) We granted the system privilege “Create any outline” to the schema that creates outlines for plan stability.

```
SQL> connect / as sysdba
SQL> GRANT CREATE ANY OUTLINE TO SCOTT;
```

(2) We stored all SQL statements executed by specific sessions*[4] in outlines. Here, SQL statements for Pattern 1 were stored.

```
SQL> ALTER SESSION SET create_stored_outlines=OUTLN_920;
SQL> SELECT /* PROBLEM_SQL 1_1 */ T1.COMMENT1, T2.COMMENT1,
2>         COUNT(*)FROM LIST_1 T1, MASTER_1 T2
3>         WHERE T1.M_UID = T2.M_UID
4>         AND T1.LAST_DATE > SYSDATE – 31
5>         GROUP BY T1.COMMENT1, T2.COMMENT1;
SQL> ALTER SESSION SET create_stored_outlines=false;
```

The stored outlines can be checked using the following SQL statement.

```
SQL> SELECT category, version, sql_text FROM USER_OUTLINES;
```

(3) We exported the outlines (OL$ table, OL$HINTS table, OL$NODES table) using the export utility.

```
$ exp outln/outln file=outln_920.dmp tables='OL$' 'OL$HINTS'
'OL$NODES' log=exp.log
```

This completed the procedures to be performed with the existing system

[4] Outlines can store SQL statements for the entire system or specific SQL statements.

(Oracle9*i* Database). The steps from (4) were implemented with the new system (Oracle Database 11*g*).

(4) We imported the outlines (OL$ table, OL$HINTS table, OL$NODES table) into the new system (Oracle Database 11*g*) using the import utility.

```
$ imp outln/outln file=outln_920.dmp ignore=Y tables='OL$'
'OL$HINTS' 'OL$NODES' log=imp.log
```

(5) Once outlines produced by a previous release are imported, they must be updated using DBMS_OUTLN.UPDATE_SIGNATURES.

```
SQL> connect outln/outln
SQL> execure DBMS_OUTLN.UPDATE_SIGNATURES;
```

Shown below are the results for the confirmation of the updated USER_OUTLINES table.

```
SQL> connect scott/tiger
SQL> SELECT category, used, version, sql_text FROM USER_OUTLINES;


CATEGORY     USED     VERSION     SQL_TEXT
--------     ----     -------     --------------------------
OUTLN_920    UNUSED   9.2.0.8.0   SELECT /* PROBLEM_SQL 1_1,
```

(6) Using the imported outlines, we executed SQL statements.

Note that the following parameter settings must be consistent before and after migration to ensure appropriate functioning of the outlines.

- QUERY_REWRITE_ENABLED
- STAR_TRANSFORMATION_ENABLED
- OPTIMIZER_FEATURES_ENABLE

```
SQL> ALTER SESSION SET query_rewrite_enabled='TRUE';
SQL> ALTER SESSION SET star_transformation_enabled='FALSE';
SQL> ALTER SESSION SET optimizer_features_enable='9.2.0';
SQL> ALTER SESSION SET use_stored_outlines=OUTLN_920;
SQL> SELECT /* PROBLEM_SQL 1_1 */ T1.COMMENT1, T2.COMMENT1,
2>       COUNT(*)FROM LIST_1 T1, MASTER_1 T2
3>       WHERE T1.M_UID = T2.M_UID
4>       AND T1.LAST_DATE > SYSDATE –31
5>       GROUP BY T1.COMMENT1, T2.COMMENT1;
```

```
SQL> ALTER SESSION SET use_stored_outlines=false;
```

When SQL statements are executed using outlines, “UNUSED” changes to “USED” in the “USED” column of USER_OUTLINES. This confirms the use of outlines.

(7) Using Enterprise Manager (hereinafter called “EM”), we stored execution plans that use the outlines of the cursor cache in the SQL management base during SQL plan baseline loading.

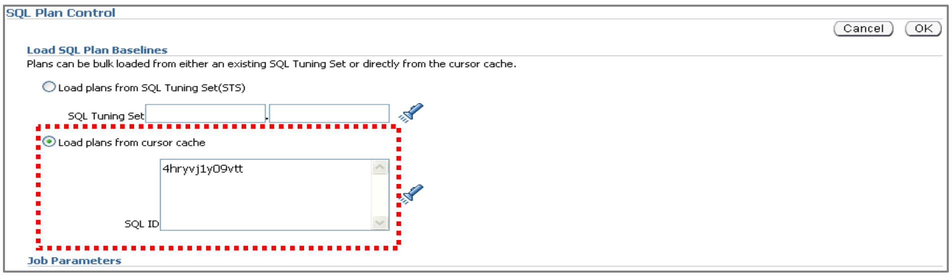


Figure 6-42 Loading SQL plan baselines

(8) The SQL plan baseline created is automatically approved, and the “Confirmed” status changes to “YES.” Hereafter, this execution plan is granted higher priority for selection than new execution plans generated by the optimizer in the subsequent execution of SQL statements.

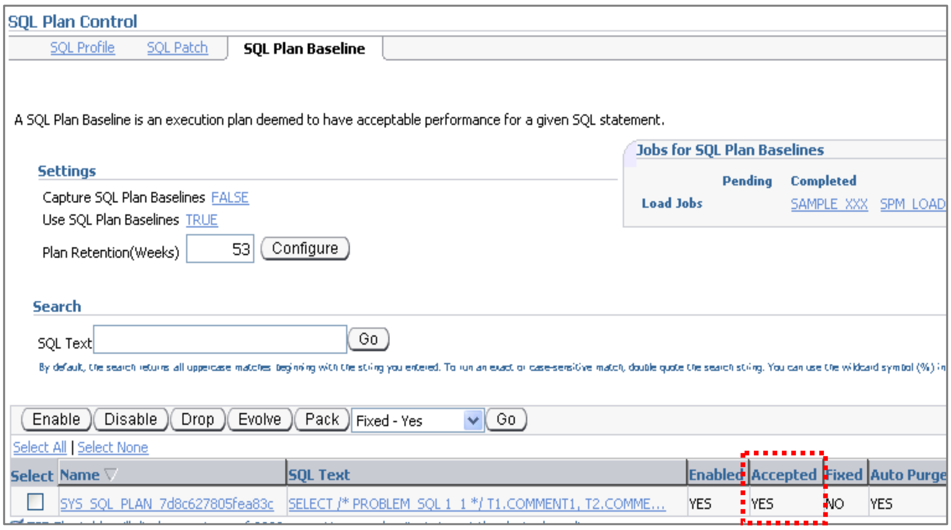


Figure 6-43 SQL plan management

(9) Stored execution plans can be checked by examining links to the SQL text described in (8) above.

A NESTED LOOP join was used in our verification tests, and the execution plans were the same as those in the existing system (Oracle9*i* Database).

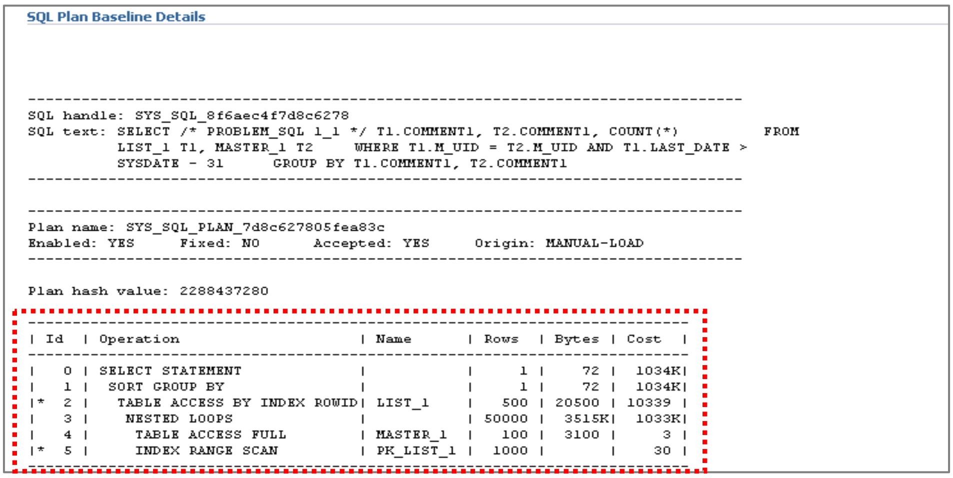


Figure 6-44 Details of SQL plan baseline

(10) When an SQL statement is executed, if the optimizer creates an execution plan that differs from the execution plan in the SQL plan base line, the newly created execution plan is stored in the SQL management base as an SQL plan baseline.

However, since the new SQL plan baseline stored in the SQL management base has not been approved, the “Confirmed” status is set to “NO.”

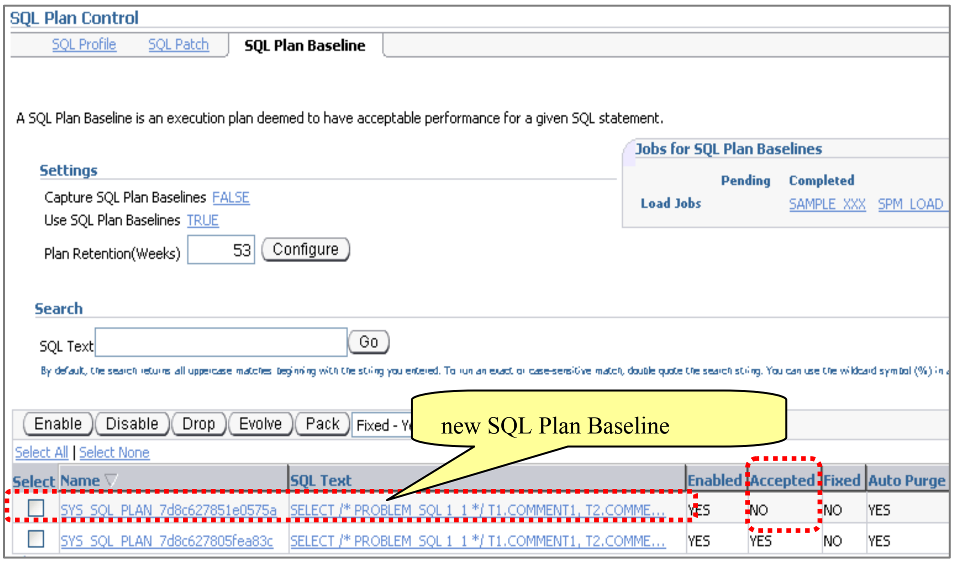


Figure 6-45 New SQL plan baseline

(11) A comparison of execution plans for this SQL plan baseline shows HASH JOIN

was selected in the newly stored SQL plan baseline.

```
SQL Plan Baseline Details

--------------------------------------------------------------------------------
SQL handle: SYS_SQL_8f6aec4f7d8c6278
SQL text: SELECT /* PROBLEM_SQL 1_1 */ T1.COMMENT1, T2.COMMENT1, COUNT(*)         FROM
          LIST_1 T1, MASTER_1 T2     WHERE T1.M_UID = T2.M_UID AND T1.LAST_DATE >
          SYSDATE - 31      GROUP BY T1.COMMENT1, T2.COMMENT1
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Plan name: SYS_SQL_PLAN_7d8c627805fea83c
Enabled: YES     Fixed: NO      Accepted: YES     Origin: MANUAL-LOAD
--------------------------------------------------------------------------------

Plan hash value: 2288437280

--------------------------------------------------------------------------
| Id  | Operation                    | Name       | Rows  | Bytes | Cost  |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |            |     1 |    72 |  1034K|
|   1 |  SORT GROUP BY               |            |     1 |    72 |  1034K|
|*  2 |   TABLE ACCESS BY INDEX ROWID| LIST_1     |   500 | 20500 | 10339 |
|   3 |    NESTED LOOPS              |            | 50000 |  3515K|  1033K|
|   4 |     TABLE ACCESS FULL        | MASTER_1   |   100 |  3100 |     3 |
|*  5 |     INDEX RANGE SCAN         | PK_LIST_1  |  1000 |       |    30 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - filter(INTERNAL_FUNCTION("T1"."LAST_DATE")>SYSDATE@!-31)
   5 - access("T1"."M_UID"="T2"."M_UID")

Note
-----
   - cpu costing is off (consider enabling it)

--------------------------------------------------------------------------------
Plan name: SYS_SQL_PLAN_7d8c627851e0575a
Enabled: YES     Fixed: NO      Accepted: NO      Origin: AUT
--------------------------------------------------------------------------------

Plan hash value: 3790705535

--------------------------------------------------------------------------------
| Id  | Operation              | Name       | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |            |     1 |    72 | 19725   (3)| 00:03:57 |
|   1 |  HASH GROUP BY         |            |     1 |    72 | 19725   (5)| 00:03:57 |
|*  2 |   HASH JOIN            |            | 50000 |  3515K| 19722   (3)| 00:03:57 |
|   3 |    TABLE ACCESS FULL| MASTER_1 |   100 |  3100 |    12   (0)| 00:00:01 |
|*  4 |    TABLE ACCESS FULL| LIST_1   |   500K|    19M| 19707   (3)| 00:03:57 |
--------------------------------------------------------------------------------
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("T1"."M_UID"="T2"."M_UID")
   4 - filter(INTERNAL_FUNCTION("T1"."LAST_DATE")>SYSDATE@!-31)
```

new Plan

Figure 6-46 Details of SQL plan baseline (new execution plan)

(12) Using the two SQL plan baselines described in (10), we examined the improvements achieved by changing the currently effective SQL plan baseline to the new SQL plan baseline.

This verification was initiated using the Expand button.

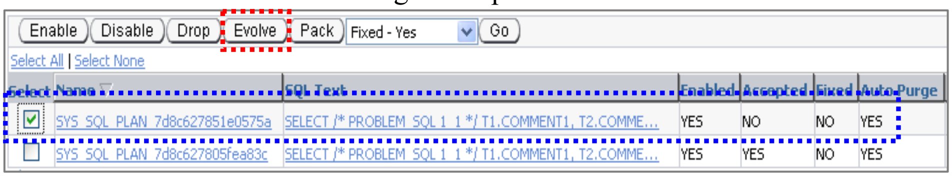
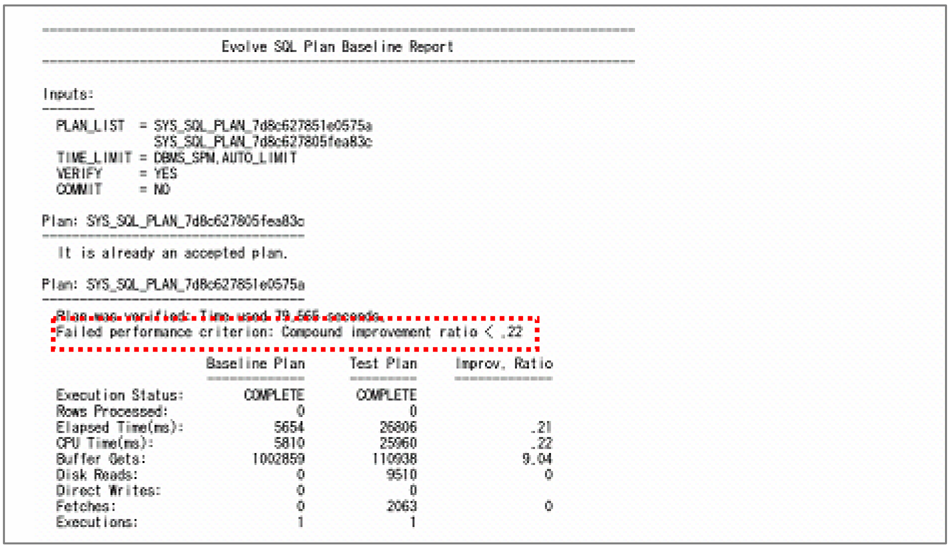


Figure 6-47 Expansion of SQL plan baseline

(13) According to the verification results, the improvement of the new SQL plan

baseline from the current effective SQL execution baseline was less than 0.22, and the indication, “Failed performance criterion,” was displayed. We determined this to be unacceptable (we recommend against making this change).



```
-------------------------------------------------------------------------------
                        Evolve SQL Plan Baseline Report
-------------------------------------------------------------------------------

Inputs:
-------
  PLAN_LIST  = SYS_SQL_PLAN_7d8c627851e0575a
               SYS_SQL_PLAN_7d8c627805fea83c
  TIME_LIMIT = DBMS_SPM.AUTO_LIMIT
  VERIFY     = YES
  COMMIT     = NO

Plan: SYS_SQL_PLAN_7d8c627805fea83c
-----------------------------------
  It is already an accepted plan.

Plan: SYS_SQL_PLAN_7d8c627851e0575a
-----------------------------------
  Plan was verified: Time used 79.566 seconds.
  Failed performance criterion: Compound improvement ratio < .22

                      Baseline Plan      Test Plan     Improv. Ratio
                      -------------      ---------     -------------
  Execution Status:        COMPLETE       COMPLETE
  Rows Processed:                 0              0
  Elapsed Time(ms):            5654          26806               .21
  CPU Time(ms):                5810          25960               .22
  Buffer Gets:              1002859         110938              9.04
  Disk Reads:                     0           9510                 0
  Direct Writes:                  0              0
  Fetches:                        0           2063                 0
  Executions:                     1              1
```

Other comparison methods

In addition to use of SQL plan management, we can also compare performance by creating an STS that stores pertinent SQL statements and by comparing retrials carried out with the SQL plan baseline enabled and disabled using the SPA.

We executed SPA under the following conditions: only the SQL plan baseline for the newly created execution plan was enabled for the first trial; only the SQL plan baseline created from the outline execution plan was enabled for the second trial. We then compared the results based on elapsed times. This showed better results with the SQL plan baseline created based on the outline execution plan.

Comparisons using the SPA provides visual, easy-to-understand results.

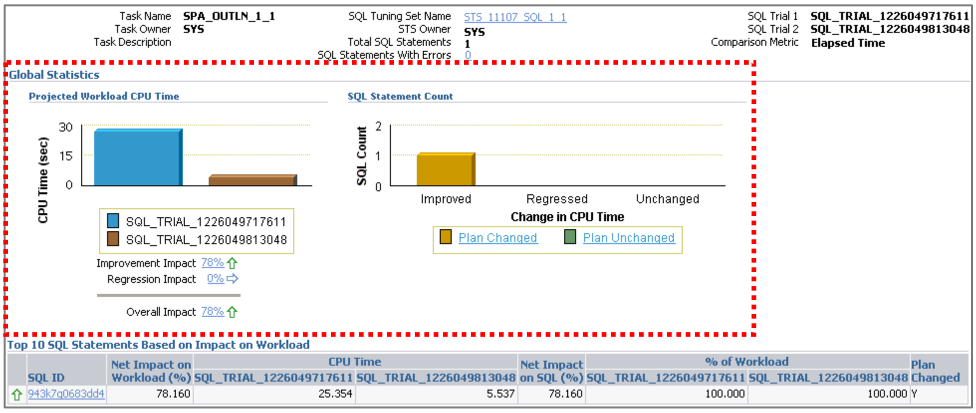


Figure 6-48 Comparison using SPA

6.3.7. SPA Summary

Upgrading databases can alter execution plans due to changes in optimizer operations, system I/O characteristics, or other factors. In particular, Oracle Database 10*g* or later versions support only Cost Based Optimizer. This means special care is required when changing from the Rule Based Optimizer supported by Oracle9*i* Database and earlier versions to Cost Based Optimizer.

Although the Cost Based Optimizer in most cases selects the optimal execution plan, it can also select suboptimal execution plans in light of stale statistics or various other reasons as shown by our verification tests. Maintaining execution plans before upgrading may be prudent in certain cases.

With Oracle Database 11*g*, SQL Performance Analyzer (SPA) enables comparisons of performance in terms of SQL response time before and after an upgrade.

After SPA execution, SQL statements that indicate performance degradation are easily tuned by combining SQL Tuning Advisor and SQL Plan Management. In addition, SPA can be used to make comparisons when examining whether to store outlines in SQL plan baselines and stabilize them. As such, it can be combined and linked to various functions.

7. Considerations and Conclusion

In our verification tests involving migration from an old system to a new system, we replaced a PRIMEPOWER 250 with a SPARC Enterprise M3000, among the latest SPARC/Solaris servers. We also upgraded Oracle9*i* Database to Oracle Data 11*g*.

The results of performance comparisons show that the online transaction processing performance of the system configured with the SPARC Enterprise M3000 and Oracle Database 11*g* is approximately 3.5-times better than the performance of the system configured with the PRIMEPOWER 250 and Oracle9*i* Database. Batch processing performance is about twice as good. This confirms the benefits of migrating to SPARC/Solaris servers and Oracle Database 11*g*.

However, updating an old system may entail various risks, including compatibility problems and performance degradation. Application tests and performance tests play an important role in reducing these risks.

We confirmed the effectiveness of Oracle Real Application Testing in solving the above-mentioned problems. Database Replay enables the confirmation of compatibility and entire system loads by acquiring the workload actually executed and replaying it in the migrated/upgraded environment. We believe Database Replay's capacity to change the scale of elapsed times between user calls is especially convenient, since it can test workload behavior under higher loads and identify potential problems in the future.

SPA allows easy production of SQL performance comparison reports from Oracle Enterprise Manager, which previously had to be produced manually. SPA can improve performance comparison efficiency. If performance degradation occurs, SQL Tuning Advisor and SQL plan management function enable tuning of SQL statements and stabilization of execution plans.

Using Oracle Real Application Testing in migrations from Oracle9*i* Database to Oracle Database 11*g* should improve the execution efficiency of application tests and performance tests, thereby reducing risks during and after migration.

We hope you will find the information provided in this document useful when migrating to a new system.

8. Appendix

8.1. Use of Real Application Testing after Migration

Users who have already upgraded to Oracle Database 11*g* can use Database Replay and SQL Performance Analyzers to perform other changes made in their database environments.

Some examples of changes in the database environment where Real Application Testing can be used are as follows:

- Use of new functions in Oracle Database 11g
- Database upgrades, patch applications, parameter/schema changes
- Configuration changes, such as conversion from single instance to RAC or ASM
- Changes in storage, networks, or interconnects
- Changes in operating system, hardware, patch applications, upgrades, or parameters
- Collection of optimizer statistical information, SQL profile production or other SQL tuning operations
- Changes in database initialization parameters

Database Replay and SQL Performance Analyzer can also be used to evaluate the effects on performance resulting from adding a new business application to an existing database environment. You can also manage the history of the results of such tests.

8.2. Effective Use of Hardware Resources

Replacing existing hardware with the latest models at the time a database upgrade is performed can deliver various benefits. Figure 8-1 shows the CPU usage for a workload replayed by Database Replay before and after migration from the PRIMEPOWER 250 to the SPARC Enterprise M3000. The new system reduced system loads and improved the availability of hardware resources.

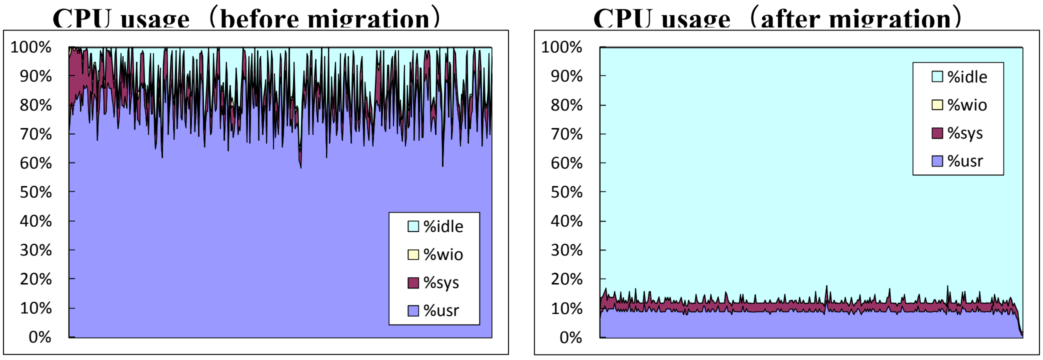


Figure 8-1 CPU usage before and after migration

Increased availability of hardware resources provides room for future growth in business transactions and data traffic. It can also be used to deploy new functions, such as security and compression functions.

Oracle Advanced Compression*[5] is a new option of Oracle Database 11*g* Enterprise Edition. In versions up to Oracle Database 10*g* Release 2, a data segment compression function was provided for data warehousing applications, but this function has been enhanced in Oracle Database 11*g* to include the capacity to compress ordinary DML statements.

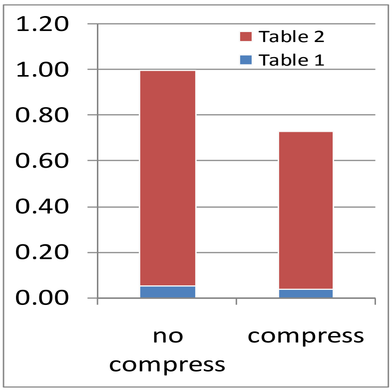


Figure 8-2 Effects of compression

Figure 8-2 shows the results of compressing two tables in the workload using Oracle Advanced Compression.

The results indicate that Oracle Advanced Compression compressed target tables by about 30%.

Since Oracle Advanced Compression also compresses ordinary DML statements, you mustcheck the loads that can affect the system.

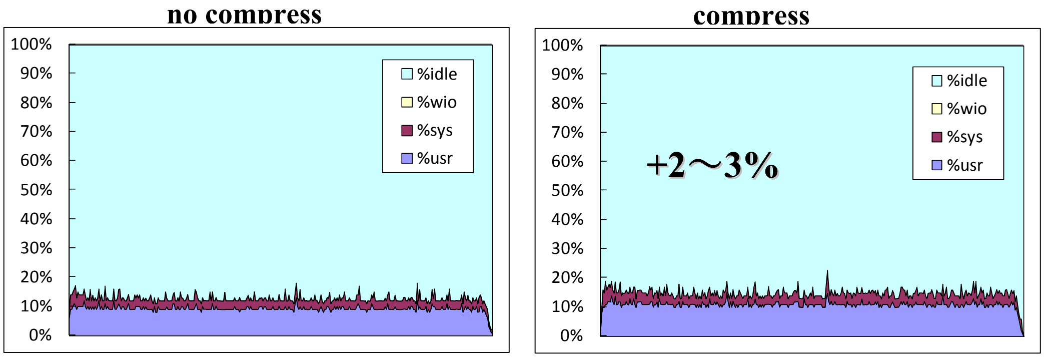


Figure 8-3 Change in CPU usage

The graph on the right shows the results for Database Replay replaying the workload with Oracle Advanced Compression disabled. The graph on the left shows the results for replay with Oracle Advanced Compression enabled. With the compression function enabled, the CPU usage rate increases, but by no more than 2 to 3%.

Next, Table 8-1 shows the CPU usage time for Insert/Update processing for the tables

[5] For more information, refer to the white paper entitled "Verification of Oracle Database 11g for Data Warehousing Using Fujitsu SPARC Enterprise" (URL: http://primeserver.fujitsu.com/sparcenterprise/documents/data/).

compressed. Enabling the compression function affects both Insert and Update processing, but greater effects are observed with the Update processing, as shown in Table 8-1.

Table 8-1 CPU usage time for Insert/Update processing

compression	insert	update
No	1.0	1.0
Yes	1.1	5.6

*A value of “1.0” is assigned to CPU usage time without compression.

As shown above, using Real Application Testing (Database Replay) makes it possible to confirm the effects of changes in the system or of adding a new function on the entire system or individual SQL processing.

8.3. Notice Regarding Upgrade to Oracle Database 10*g* Release 2

Thus far, this white paper has described migration from Oracle9*i* Database to Oracle Database 11*g*. However, certain clients may wish to upgrade their Oracle9*i* Database systems to Oracle Database 10*g* Release 2 (for example, perhaps a particular third-party business application currently in use does not support Oracle Database 11*g*). To assist such clients, described below is a method for testing the effects of such an upgrade on SQL performance.

SPA functions have been enhanced to enable testing of the effects on SQL performance resulting from an upgrade from Oracle9*i* Database to Oracle Database 10*g* R2.[6]

Specifically, this involves the following six steps.

[6] For more information, refer to the white paper entitled “Testing the SQL Performance Impact of an Oracle 9*i*/10*g*R1 to 10*g*R2 Upgrade with SQL Performance Analyzer” (URL: http://www.oracle.com/technology/products/manageability/database/pdf/owp_spa_9i_10g.pdf).

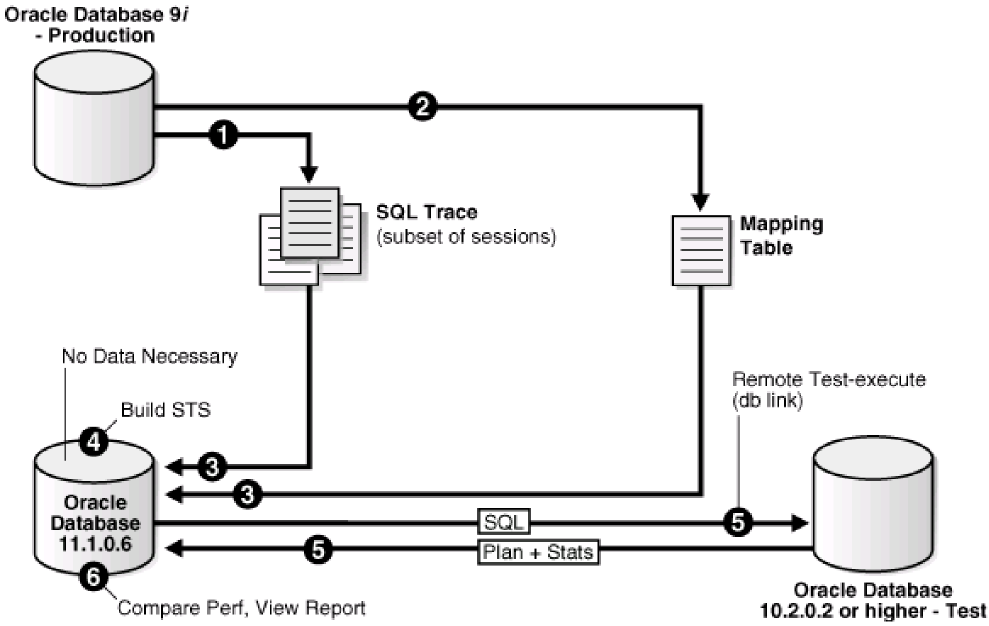


Figure 8-4 SQL performance test for upgrade from 9*i* to 10*g* R2

1. Enable the SQL Trace facility on the production system running Oracle Database 9i.

   To minimize the performance impact on the production system and still be able to fully capture a representative set of SQL statements, consider enabling SQL Trace for only a subset of the sessions, for as long as required, to capture all important SQL statements at least once.

2. Create a mapping table on the production system running Oracle Database 9i.

   This mapping table will be used to convert the user and object identifier numbers in the SQL trace files to their string equivalents.

3. Move the SQL trace files and the mapping table from the production system running Oracle Database 9i to the system running Oracle Database 11g.
4. On the system running Oracle Database 11g, construct a SQL tuning set using the SQL trace files.

   The SQL tuning set will contain the SQL statements captured in the SQL trace files, along with their relevant execution context and statistics.

5. On the system running Oracle Database 11g, use SQL Performance Analyzer to build a pre-upgrade SQL trial and a post-upgrade SQL trial:
   1. Convert the contents in the SQL tuning set into a pre-upgrade SQL trial that will be used as a baseline for comparison.
   2. Remotely test execute the SQL statements on the test system running Oracle Database 10g over a database link to build a post-upgrade SQL trial.
6. Compare SQL performance and fix regressed SQL:

SQL Performance Analyzer compares the performance of SQL statements read from the SQL tuning set during the pre-upgrade SQL trial to those captured from the remote test execution during the post-upgrade SQL trial. A report is produced to identify any changes in execution plans or performance of the SQL statements.

If the report reveals any regressed SQL statements, you can make further changes to fix the regressed SQL. You can then repeat the process of executing the SQL tuning set and comparing its performance to a previous execution to test any fixes or additional changes made. Repeat these steps until you are satisfied with the outcome of the analysis.

The following describes the detailed tuning cycle for SQL statements with performance degradation.

1. From the SPA comparison report, execute SQL Tuning Advisor for SQL statements that show performance degradation.
2. If SQL Tuning Advisor provides useful recommendations, examine the implementation of those recommendations.
3. Implement the recommendations, create a replay trial, and execute the SQL statements.
4. Compare performance before and after implementation.
5. If SQL Tuning Advisor does not provide useful recommendations, produce stored outlines for pertinent SQL statements in the Oracle9*i* Database production environment.
6. Transfer the created stored outlines to the Oracle Database 10*g* R2 test system and stabilize the execution plans.

Oracle Real Application Testing provides an environment that allows sufficient pre-testing of migration even if Oracle9*i* Database can be upgraded only to Oracle Database 10*g* R2 for some unavoidable reasons.

Oracle Corporation Japan
Oracle Aoyama Center
2-5-8, Kita Aoyama, Minato-ku, Tokyo
107-0061 Japan

FUJITSU LIMITED
Shiodome City Center
1-5-2, Higashi-Shimbashi, Minato-ku, Tokyo
105-7123 Japan