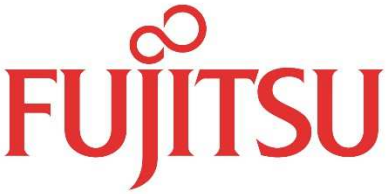
SPARC M10 による Oracle Multitenant の活用

～ Building Block　と　Pluggable Database による Database の統合と拡張 ～



目次

1. はじめに

企業を取り巻くビジネス環境は常に変化しています。ビジネスを支える ICT においても例外ではありません。今日のビジネスの変化に対応するために、現在の ICT 基盤への投資や IT 運用コストを削減し、ビジネスに直結したサービスを迅速に提供可能なクラウド型の ICT 基盤の構築に投資を実施することが急務となっています。

とりわけビジネスの中核となるデータベースにおいては、統合によって投資や IT 運用コストの削減を達成することや、新たなビジネスプロジェクトやサービスの開始時にデータベースを迅速に提供できる柔軟な拡張性を備えていることは、非常に重要です。

データベースの統合では、「仮想サーバ環境での統合」、「インスタンス統合」、「スキーマ統合」が以前からの方法ですが、Oracle Database 12*c* の登場により、「Oracle Multitenant による Pluggable database での統合」とういう方式が加わりました。

それぞれの方式で優劣はありますが、「Oracle Multitenant による Pluggable database での統合」は、集約率や拡張性で「仮想サーバ環境での統合」、「インスタンス統合」より優れ、管理性では 3 方式のなかで最も優れています。

Oracle Database 12*c* によりデータベースの統合が以前と比較し大幅に容易になるだけでなく、新規のサービスの追加に応じて、非常に迅速かつ柔軟に新たなデータベースを配備することが可能となりました。

一方、データベースが稼働する物理サーバにおいては、CPU やメモリ等の性能は向上し、より大規模な統合は可能にはなりましたが、ビジネスの変動に追随するためには、これまで、下記どちらかの方法を取るしかかありませんでした。

1. 将来予想される使用リソースに事前投資（初期コスト大、将来予測困難）
2. ビジネス拡大時に新規サーバ導入などの追加投資（システム構築費用大）

しかしながら、このような対応では、変化の速い現在のビジネス環境に追随するサーバ環境の提供や、投資や IT 運用コストの削減は、非常に困難です。

この問題に対処するため富士通は、将来のビジネスの拡大や状況の変化に柔軟に対応できるサーバとして、SPARC M10 を開発しました。

SPARC M10 は、コア単位で CPU リソースを追加可能な「CPU コア アクティベーション」機能や、筐体単位で動的なサーバの構成変更が可能な Building Block 方式の筐体接続による「Dynamic Reconfiguration」（物理パーティションの動的再構成）機能を採用しています。

以上のように Oracle Database 12*c* と SPARC M10 は、単体でもお客様の投資、IT 運用コストを削減し、柔軟な拡張性を提供いたしますが、Oracle Database 12*c* と SPARC M10 を組み合わせることで、最も効率的で柔軟なデータベースサーバ環境を提供することができます。

本資料では、Oracle Database 12*c* の Oracle Multitenant と SPARC M10-4S の CPU コアアクティベーションと Dynamic Reconfiguration を用いて、サーバやサービスの停止を伴わずに Pluggable database とハードウェアリソースが追加できることを確認しています。

また、データベースサーバを拡張していく過程で、レスポンスタイムは一定でありながら、リニアにトランザクション処理量が増加していくことを検証しています。

2. Oracle Database 12*c* Multitenant

2.1 製品概要

Oracle Database 12*c* の Oracle Multitenant[1]オプションは、革新的なデータベース内仮想化ソリューションで、Oracle 12*c* のデータベースを統合して投資と IT 運用コストを削減します。**図 1** に示すとおり、この革新的な技術は、Container database（CDB）のメモリとバックグラウンド・プロセスを有効利用することにより、高い効率性と、個々の Pluggable database（PDB）の高密度な統合を実現します。CDB に収容されている PDB テナントを組み合わせたものが Multitenant Container Database です。

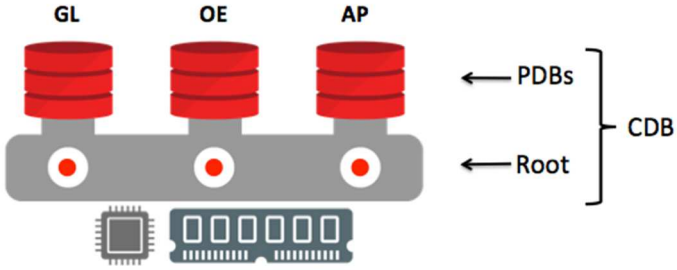


図 1 Oracle Multitenant Container Database

それぞれの PDB は移動可能な自己完結型データベースであり、独自のスキーマ、スキーマ・オブジェクト、非スキーマ・オブジェクトが個々の PDB の SYSTEM 表領域、SYSAUX 表領域、USER 表領域に格納されています。PDB ごとに、CDB とは別の独自のデータ・ファイルを持つことで、データベースを簡単にアンプラグまたはプラグインでき、クローニングによって PDB を迅速にプロビジョニングすることができます。ルート・コンテナ・データベース（CDB$ROOT）には、データ・ディクショナリ、SGA メモリおよびバックグラウンド・プロセスが保持されます。

Oracle Multitenant は、Real Application Clusters (RAC)、Oracle Data Guard、Oracle GoldenGate など、他の Oracle Database オプションを完全に補完します。このドキュメントに示すとおり、SPARC M10 の Dynamic Reconfiguration および CPU コア アクティベーションと RAC および Oracle Multitenant の CDB 初期化パラメータでの動的な CPU リソース変更の組み合わせで、PDB の迅速なオンライン・データベース・プロビジョニングが容易になり、追加の PDB は、簡単な SQL 文でプロビジョニングできます。

SPARC M10-4S の Dynamic Reconfiguration は、優れてコスト効率の良いデプロイメント・モデルであり、Oracle Multitenant の高密度な統合の利点をさらに強化します。

効率的なデータベース統合によって、このアーキテクチャには次のような実用的なメリットがあります。

- **一元管理によるデータベース管理の簡素化**
  Oracle Multitenant アーキテクチャには、データベース全体を管理出来る機能があり、全ての PDB を *common user* で管理できます。簡素化された管理タスクとは、たとえば、CDB のパッチ適用と全ての PDB のパッチ適用を 1 回のパッチ操作で実行したり、CDB のバックアップと全ての PDB のバックアップを 1 回の RMAN バックアップで実行したりするというものです。必要に応じて、管理を分離し、個々の PDB を

[1] Oracle Multitenant の詳細は、http://oracle.com/goto/Multitenant を参照してください。

local user で管理することもできます。

- **アンプラグ/プラグインによる迅速なデータベース移動**
  Oracle Multitenant のデータベース内仮想化によって、PDB 表領域とデータ・ファイルを個別に持つことができます。簡単な SQL コマンドを使用して、PDB を現在の CDB から別の CDB に再配置できます。
- **SQL コマンドラインまたは PDB セルフサービスアプリケーションでの高速シン・プロビジョニング**
  Oracle Multitenant は、完全なプロビジョニングと、PDB のシン・プロビジョニングをサポートします。ASM Cloud File System (ACFS) のようなコピーオンライトをサポートするファイル・システム・ストレージでは、簡単な SQL 文で PDB をスナップショット・クローニングできます。さらに、PDB クローニング機能には、特定スキーマのサブセット・クローニング、メタデータ・クローニングおよびソース・データベースからのリモート・クローニングがあります。
- **アプリケーションの変更なしに既存のデータベースを容易に PDB として導入**
  アプリケーションの観点からすると、Oracle Multitenant は、従来のデータベース（非 CDB）と同じです。アプリケーションの変更は必要ありません。

SPARC M10 で Oracle Multitenant を使用すると、効率的な統合と必要時にサーバ内のアプリケーション高密化を可能にするキャパシティ・オン・デマンド・プロビジョニングによって、投資を削減できます。また、Oracle Multitenant は、標準化、単純な SQL 文もしくは Oracle Multitenant Self-Service Provisioning アプリケーションによる迅速なデータベース・プロビジョニング、およびパッチ適用、アップグレード、バックアップを一元管理する CDB/PDB フレームワークによって運用コストを削減します。

2.2 SPARC M10 でのマルチテナントの使用例

Oracle Multitenant は、基本的には統合の場面で使用されます。IT 部門はサーバとソフトウェアを統合することで、エネルギー・コスト、メンテナンス・コスト、ライセンス・コストを削減します。そして、統合により標準化が重要になります。IT コストのさらなる削減は、効率的かつ低リスクな自動化されたプロビジョニングとメンテナンスによって実現されます。Oracle Multitenant は、これらの要件を両方とも満たします。これらは、サービス・ベースのアーキテクチャに移行する場合およびクラウド・デプロイメント・モデルにおいて、最初に必要なステップです。Oracle Multitenant は、Oracle の **Database as a Service** (DBaaS) の基盤となるように作られています。また、アプリケーションではなくデータベースにマルチテナント性を実装するので、Oracle Multitenant は **Software as a Service** (SaaS) に最適です。オーケストレーションによるクラウド・データベースのライフサイクル管理は、プライベート・クラウドとパブリック・クラウドの相互運用と、オンプレミスのプライベート・クラウドの両方に Oracle Multitenant を使用することで、より簡単になります。

DBaaS

Oracle Multitenant を使用して顧客は DBaaS を定義し、サービス・レベルごとに PDB のプロビジョニングを選択できます。たとえば、**図 2** に示すとおり、顧客は階層化された可用性のサービス・レベルに基づいて DBaaS カタログを定義できます。PDB は各層でプロビジョニングでき、PDB のアンプラグ/プラグインによる移動性を利用して、アプリケーションの可用性要件の変化に応じてサービス・レベル間を容易に移動できます。

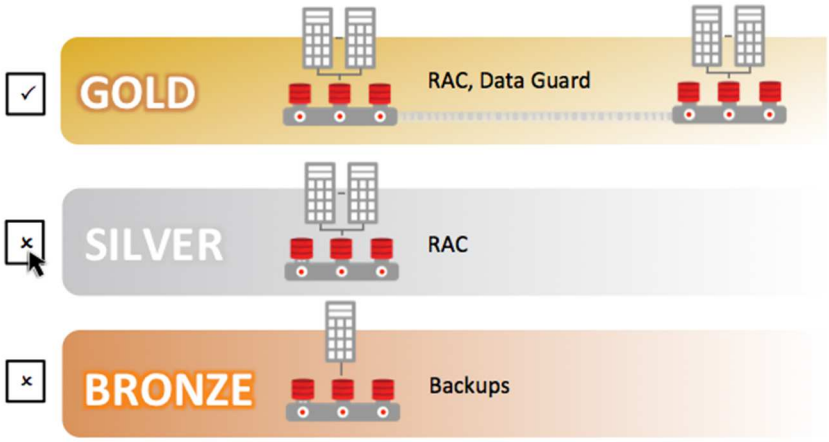


図 2 Database as a Service のための Oracle Multitenant

Oracle Multitenant と PDB によって向上する効率性とは次のようなものです。CDB が RAC または Data Guard のいずれかで構成される場合、ある PDB を単一インスタンスの CDB から RAC または Data Guard スタンバイ用に構成された CDB に接続すると、自動的に複数インスタンスの RAC PDB になり、Data Guard の場合は構成変更しなくても PDB の REDO が自動的にスタンバイにレプリケートされます。アンプラグ/プラグインのサービス・レベルを移行する API は簡単な SQL コマンドであるため、このような操作は任意のサービス・オーケストレーション・フレームワークとシームレスに統合されます。

SaaS

データの分離、データベース・セキュリティ、一般的なレポート作成ツールが必要な場合、アプリケーション・レベルでマルチテナント性を実装することが困難なときがあります。Oracle Multitenant に SaaS アプリケーションをデプロイすると、このような要件に対応できます。各 PDB は、他のテナントから物理的に分離されます。より高い分離性が必要とされる場合は、Database Vault や Label Security などの Oracle セキュリティ製品を Oracle Multitenant に構成できます。

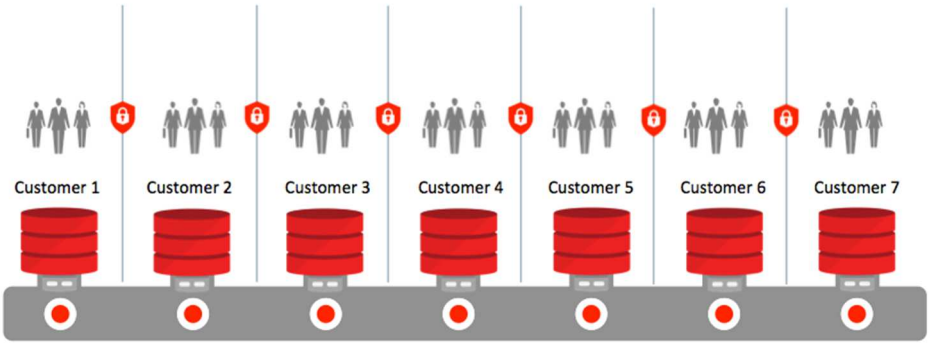


図 3 Software as a Service のための Oracle Multitenant

SaaS アプリケーションを使用する場合の多くは、アプリケーションで共通のデータ・モデルが共有されます。Oracle Multitenant では、SELECT 文に CONTAINERS()句を使用して実行することで、すべての PDB または一部の PDB を横断してレポート生成するための高速な問合せ操作を行うことができます。独立した個々の PDB に問合せを実行する機能と、全部または一部の PDB に対してより大きな規模で問合せを実行する機能の組合せが、Oracle

Multitenant の独自の特徴です。

クラウド・スケール・コンピューティング

Oracle Multitenant と SPARC M10 の Building Block 方式の筐体接続および CPU コア　アクティベーションは、プライベート・クラウドまたはパブリック・クラウドをデプロイするための強力な組合せです。DBaaS と SaaS のキャパシティを必要に応じて拡大/縮小、インスタンスを 1000 個とまではいかなくても 100 個まで増やせるようにスケールアウトしたデータベースをデプロイすることができます。本資料では、オンライン・キャパシティ要求を満たす 2 つのソリューションを示します。大規模な統合の実現と、アプリケーションのパフォーマンスを犠牲にすることなくトランザクション・ボリュームの総数を増やす例です。

3. SPARC M10

3.1 製品概要

SPARC M10 は、最先端テクノロジーによる世界 No.1 の拡張性と柔軟性、メインフレームおよび SPARC Enterprise M シリーズから継承する高い信頼性と高水準の技術力を備えた、データベースサーバ等の基幹システムに最適な高性能・高信頼 UNIX サーバです。

1U の省スペースに高い性能、信頼性、仮想化機能を備え、あらゆる用途に最適なエントリーサーバである SPARC M10-1、データセンターの統合に最適なミッドレンジサーバの SPARC M10-4、そして、「Building Block」方式の筐体間接続により、高いスケーラビリティを発揮するハイエンドサーバである SPARC M10-4S の 3 モデルをラインナップしています。

全てのモデルが、「CPU コア アクティベーション」に対応し、CPU リソースは、1 コア単位で段階的な拡張が可能です。なかでも、Building Block 方式の筐体間接続を採用した SPARC M10-4S は、ビジネスの成長に合わせて「Dynamic Reconfiguration（物理パーティションの動的再構成）」により筐体も追加が可能です。

3 タイプのモデル（図 4）や柔軟なリソース追加を実現する機能により、お客様のビジネスの成長に合わせた投資を可能にします。

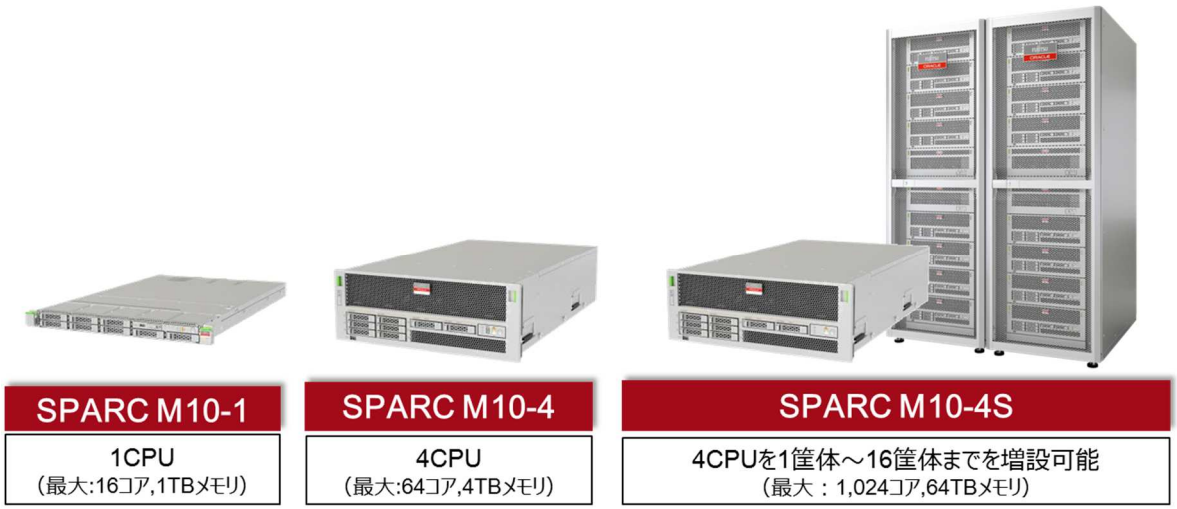


図 4 SPARC M10 のラインナップ

3.2 CPU コア アクティベーション

SPARC M10 では、業務量の増加に応じて、システムを停止せずに CPU 能力を段階的に増強可能な CPU コア アクティベーションに対応しています。
急な負荷増加にも対応する柔軟性と、業務計画に合わせてきめ細かく対応できる拡張性を実現します。

CPU コア アクティベーションにより 1 コア単位で段階的なリソース追加ができるため、サーバの使用開始当初、すぐに利用しないコアは未アクティベートのままにしておき、必要な時に増強することで、初期投資を抑え、お客様の TCO を最適化します。

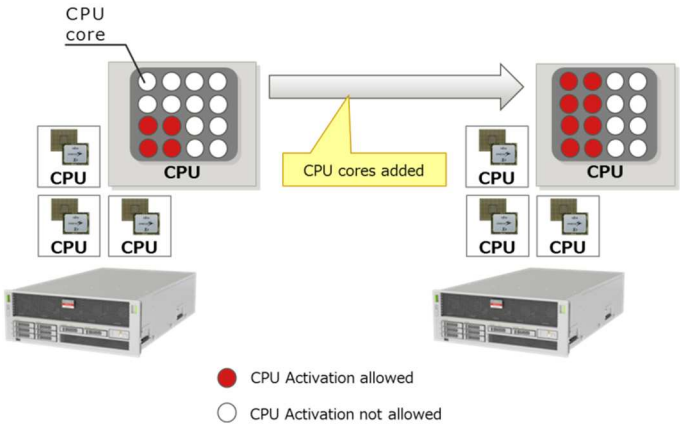


図 5　CPU コア アクティベーションによる CPU コアの追加

3.3 Building Block アーキテクチャ

SPARC M10-4S は、Building Block 方式の筐体間接続を採用しています。最大 16 台の筐体（Building Block）を高速インターコネクトで接続することにより、高い拡張性を実現する SPARC M10-4S は、1 つまたは複数の筐体で物理パーティションを作成できます。各物理パーティションは、物理的に独立した信頼性の高いシステム環境として利用することができます。

SPARC M10-4S の物理パーティションは、1 つの物理パーティションで業務負荷が高くなった場合や万が一のトラブル時にも、他の物理パーティションに影響を及ぼしません。ファームウェア層での仮想化機能「Oracle VM Server for SPARC」や、ソフトウェアレベルでの仮想化機能「Oracle Solaris ゾーン」よりも、高い障害隔離性を実現します。

3.4 Dynamic Reconfiguration

Building Block 方式で筐体間接続された SPARC M10-4S は、標準機能で「Dynamic Reconfiguration」（物理パーティションの動的再構成）を採用しており、追加費用なしに使用できます。Dynamic Reconfiguration は、CPU やメモリ、I/O などのハードウェアリソースを、システムを動作させたまま物理パーティションへ追加・削除することができる機能です。システム監視機構（XSCF）を通じて、物理パーティションの構成の変更や、ハードウェアの組込み/切り離しなど、様々な構成変更を行うことができます。（図.6）

物理パーティションの動的再構成の利用により、業務拡張や新規業務の追加などの要求に応じたタイムリーなリソースの追加（活性増設）や、ハードウェアの活性保守が可能になります。

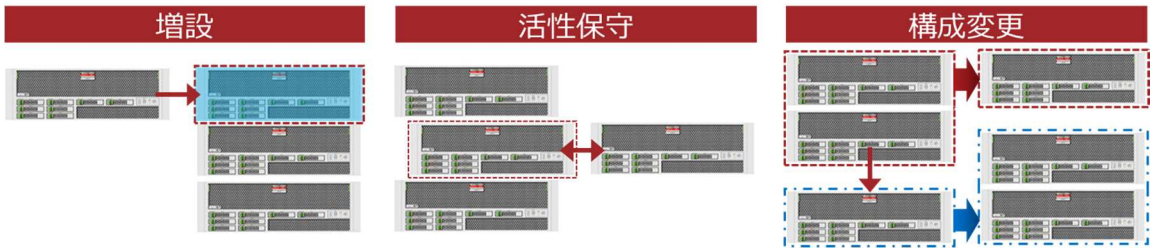


図 6　Dynamic Reconfiguration による様々な構成変更

4. 検証

前章までに述べたように、Oracle Database 12*c* は Oracle Multitenant による Pluggable Database、SPARC M10 は、CPU コア アクティベーションと Dynamic Reconfiguration によるハードウェアリソースの追加削除を実装しています。

本章では、Oracle Database 12*c* の Oracle Multitenant と SPARC M10 の CPU コア アクティベーションおよび Dynamic Reconfiguration を使用したデータベースサーバの拡大手法の詳細を記載します。

4.1 検証環境

本検証では、データベースサーバとして、SPARC M10-4S（Solaris11.1） 、Oracle Databas 12*c*、ストレージとして ETERNUS DX410 S2、負荷発生装置として PRIMERGY RX200 S7、Oracle Load Testing Accelerator for Oracle Database を使用しています。

4.1.1 使用ハードウェアおよびソフトウェア

Database Server

SPARC M10-4S(8units)
- Processor : 3.7GHz SPARC 64X+, 4 sockets(16cores per soket)
- Memory :1TB
- Operating System: Oracle Solaris 11.1.17.5.0

Storage System

ETERNUS DX410 S2(4units)
- HDD : 300GB (15,000rpm)
- I/O :8Gbps FC

Oracle Databese

Oracle 12.1.0.1.4

Network

CISCO Catalyst4900M : Cluster interconnect Network switch
Fujitsu SR-X324T1 : Local Area Network switch

4.1.2　システム構成

データベースサーバは、Oracle Real Applications Clusters を用いて 2 ノードの構成となっています。（図 7）

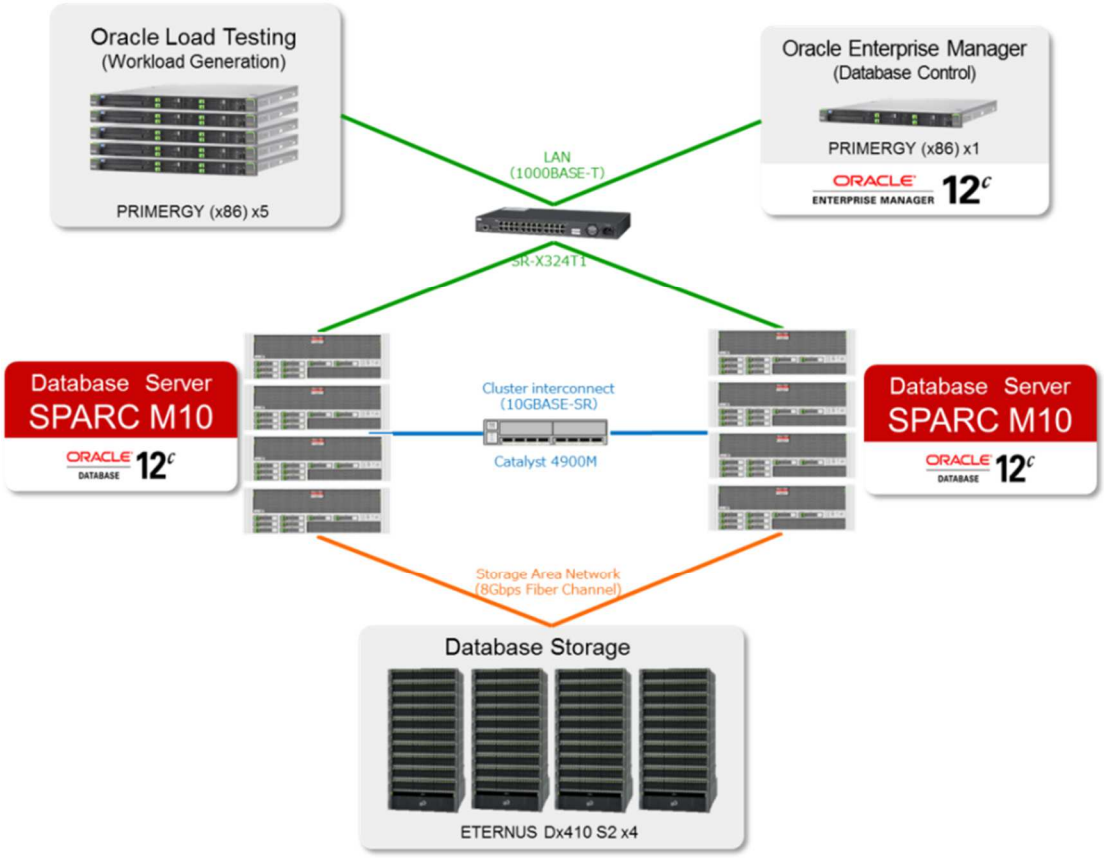


図 7　システム構成図

4.1.3 ストレージ設定

Oracle Automatic Storage Management と Oracle 12*c* Plugable Database は、下記の様に構成されています。

ストレージ容量	：	62TB (total) = 1.3TB/LUN　x 48
ASM Disk Group　構成：		S.A.M.E (Stripe And Mirror Everything) 全ての DISK を 1 つの DISK Group に配置（図 8）
格納データ	：	Data Base　ファイル、REDO Log

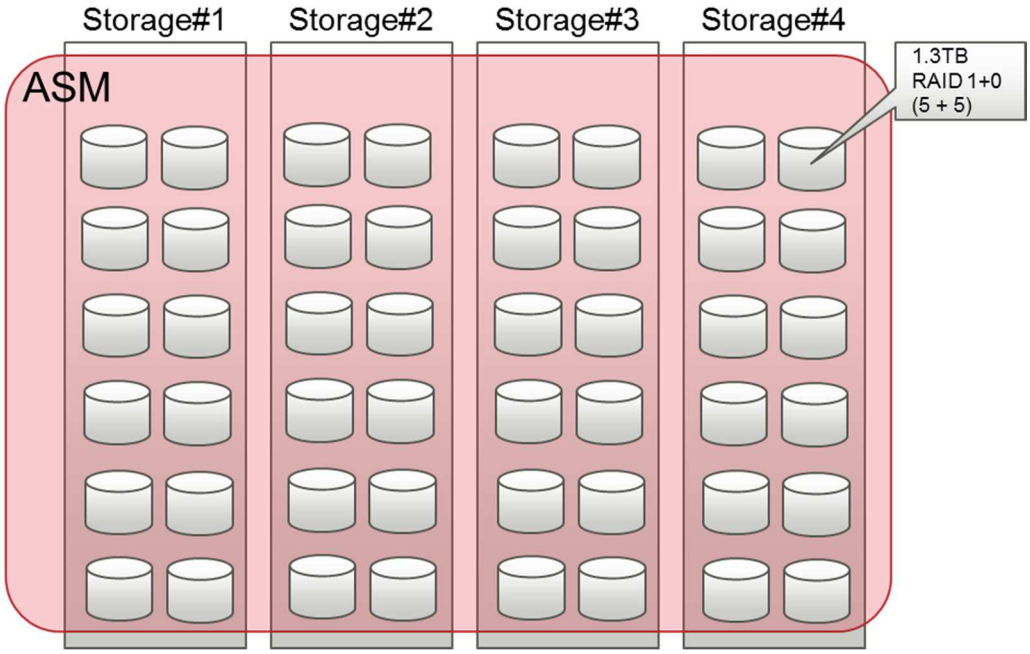


図 8　RAID および ASM ディスクグループ構成図

Plugable Database は、252 個全てを ASM 内に作成し、PDB あたり 10GB の容量を確保しています。(図 9)　また、CPU は、PDB あたり 4 CPU コアを割り当て、各 PDB には 10User が接続します。

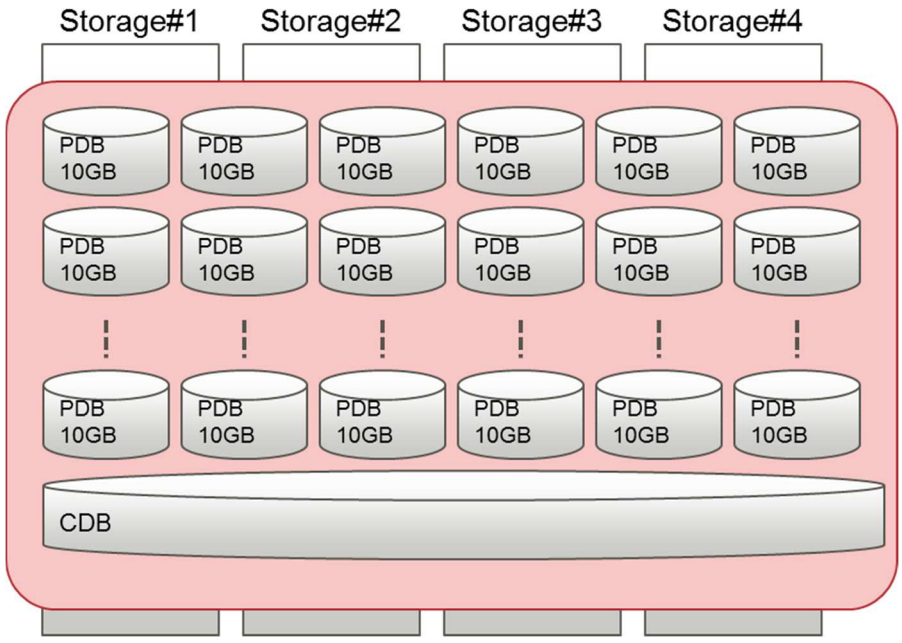


図 9　PDB 構成図

4.1.4 テストツール

テストツールは、Oracle Load Testing Accelerator (OLT) for Oracle Database を使用しています。ワークロードは、OLTP model (Online Shopping – JpetStore -)を使用しています。
今回、Web サーバとアプリケーションサーバは使用していません。
また、負荷を増加させる方法として、トリガーが起動し INSERT が実行されるたびに、LOOP 処理と SELECT 処理が実行されます。

4.2 検証内容

本節では、検証において設定した SPARC M10 のリソース追加手順と Oracle Database 12*c* の Pluggable Database 追加手順を示します。

4.2.1 CPU コア アクティベーションによるリソース追加

データベースの拡大に応じて、SPARC M10-4S の CPU コア アクティベーション機能を活用し、ハードウェアリソース(CPU コアの追加)を実施します。

稼働中の SPARC M10-4S の CPU コアを追加する設定を以下に示します。
SPARC M10-4S のシステム監視機構（eXtended System Control Facility：以下 XSCF)から有効になっている CPU コアの数を確認します。

```
XSCF> showcodusage
Resource In Use Installed CoD Permitted Status
-------- ------ --------- ------------ ------
PROC         16        64            16 OK: 0 cores available

PPAR-ID/Resource In Use Installed Assigned
---------------- ------ --------- --------------
0 – PROC             16        64       16 cores
Unused – PROC         0         0        0 cores
```

上記の例では、SPARC M10-4S　1 台において、物理パーティションは 1 つの状態で、16 個の CPU コアが割り当てられている状態です。残りの 48 個については CPU コアがアクティベートされていない状態です。

本検証では、データベースの拡大に合わせて、ハードウェアリソースを追加していきますので、CPU コアを有効化するために、CPU コアアクティベーションキーを追加します。

```
XSCF>addcodactivation "Product: SPARC M10-4S
SequenceNumber: 116
Cpu noExpiration 48
Text-Signature-SHA256-RSA2048:
SBxYBSmB32E1ctOidgWV09nGFnWKNtCJ5N3WSlowbRUYlVVySvjncfOrDNteFLzo
:
:
1TSgrjnee9FyEYITT+ddJQ=="
Above Key will be added, Continue?[y|n]:y
```

上記の例では、48 個の CPU コア分を有効化しました。

CPU コアが有効化されているか状態を確認します。

```
XSCF> showcodusage
Resource In Use Installed CoD Permitted Status
-------- ------ --------- ------------ ------
PROC         16        64            64  OK: 48 cores available

PPAR-ID/Resource In Use Installed Assigned
---------------- ------ --------- --------------
0 – PROC             16        64       16 cores
Unused – PROC         0         0       48 cores
```

CPU コアは有効化されましたが、物理パーティションに CPU コアが割り当てれれていませんので、引き続き CPU コアの物理パーティションへの割り当てを実施します。

```
XSCF> setcod -p 0 -s cpu 64
```

CPU コアが割り当てられているか確認します。

```
XSCF> showcodusage
Resource In Use Installed CoD Permitted Status
-------- ------ --------- ------------ ------
PROC         64        64            64 OK: 0 cores available

PPAR-ID/Resource In Use Installed Assigned
---------------- ------ --------- --------------
0 – PROC             64        64       64 cores
Unused – PROC         0         0       0 cores
```

次に、Oracle データベースが稼働する論理ドメインに CPU コアを追加します。
Oracle データベースが稼働する論理ドメイン上で、ldm list-devices コマンドを実行し、追加されたハードウェアリソースの状態を確認し、ldm add-core コマンドを実行して Oracle データベースが稼働する論理ドメインに CPU コアを追加すます。

```
# ldm list-devices
CORE
    ID          %FREE          CPUSET
    128         100            (256, 257)
    132         100            (264, 265)
    136         100            (272, 273)
    140         100            (280, 281)
....
MEMORY
    PA                                      SIZE                   BOUND
    0x700000000000                          32G
    0x720000000000                          32G
    0x740000000000                          32G
    0x760050000000                          31488M
....

# ldm add-core 48 OracleDB-domain
```

システムの運用を中断することなく CPU コアが物理パーティションに割り当てられることが確認できました。
CPU コアを段階的に追加していく場合は、同様の手順を繰り返します。

4.2.2 DR によるリソース追加

筐体中の CPU コアを全て有効化した後、さらにハードウェアリソースを追加する場合は、DR による筐体追加を実施することになります。（図 10）

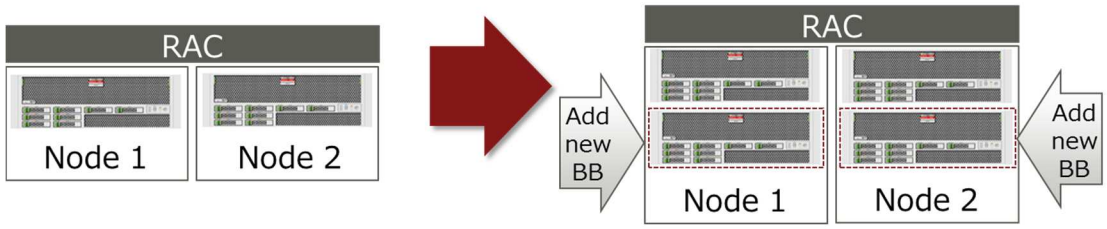


図 10　DR による SPARC M10-4S の追加と PDB の追加

DR による SPARC M10-4S 筐体追加設定を以下に示します。
SPARC M10-4S の XSCF から現在の物理パーティション構成情報（PPAR 構成情報）を確認します。

```
XSCF> showpcl -p 0
PPAR-ID   LSB    PSB     Status
00                       Running
          00     00-0
```

物理パーティション構成情報に追加する SPARC M10-4S を登録します。

```
XSCF> setpcl -p 0 -a 1=01-0
```

設定した物理パーティション構成情報を確認します。

```
XSCF> showpcl -p 0
PPAR-ID   LSB    PSB     Status
00                       Running
          00     00-0
          01     01-0
```

SPARC M10-4S の状態を確認します。
下記では、新規に登録した SPARC M10-4S が 01-0 として、システムボートプール状態になっています。

```
XSCF> showboards -p 0
PSB  PPAR-ID(LSB) Assignment  Pwr   Conn Conf Test    Fault
---- ------------ ---------- ---- ---- ---- ------- --------
00-0 00(00)       Assigned    y    y    y    Passed  Normal
01-0 SP           Available   n    n    n    Passed  Normal
```

新規に登録した SPARC M10-4S（01-0）を、稼働中の物理パーティションに追加します。

```
XSCF> addboard -c configure -p 0 01-0
PSB#01-0 will be configured into PPAR-ID 0. Continue?[y|n] :y
Start connecting PSB to PPAR. [3600sec]
0..... 30..... 60..... 90.. ...120.....150.....180.....210.....240.....
270.....300.....330.....360.....390.....420.....450.....480.....510.....
540.....570.....600.....630.....660.....690.....720.....750.....780.....
810.....840.....870.....900.....930.....960.....end
Connected PSB to PPAR.
Start configuring PSB to Logical Domains (LDoms) Manager. [1800sec]
0.....end
Configured PSB to Logical Domains (LDoms) Manager.
Operation has completed.
```

SPARC M10-4S の状態を確認します。

```
XSCF> showboards -p 0
PSB  PPAR-ID(LSB) Assignment  Pwr  Conn Conf Test    Fault
---- ------------ ---------- ---- ---- ---- ------- --------
00-0 00(00)       Assigned    y    y    y    Passed  Normal
01-0 00(01)       Assigned    y    y    y    Passed  Normal
```

ここで、新規に登録した SPARC M10-4S の CPU コアがアクティベーションされていない場合は、「5.1　CPU コア アクティベーションによるリソース追加」と同様の設定を実施し、新規登録した SPARC M10-4S 上の CPU コアが使用可能な状態となるようにして下さい。

また、筐体追加の場合メモリも追加可能となりますので、ldm add-memory コマンドを実行し、Oracle データベースが稼働する論理ドメインにメモリを追加します。

```
# ldm set-memory 64G OracleDB-domain 1
```

4.2.3　Oracle Database の操作

Solaris OS が認識している CPU 情報を OS 上から確認します。

```
# psrinfo -vp
The physical processor has 16 cores and 32 virtual processors (0-31)
The core has 2 virtual processors (0 1)
The core has 2 virtual processors (2 3)
The core has 2 virtual processors (4 5)
The core has 2 virtual processors (6 7)
…
SPARC64-X+ (chipid 0, clock 3700 MHz)
The physical processor has 16 cores and 32 virtual processors (32-63)
…
```

同様に、OS が認識しているメモリサイズを確認します。

```
# prtconf | grep Memory
Memory size:  8388608 Megabytes
```

上記のように、SPARC M10-4S の CPU コア アクティベーション機能や DR によるリソース追加によって、Solaris OS が認識する CPU/メモリを動的に拡張することが可能です。

続いて、Oracle 12*c* が認識する CPU/メモリ資源を確認します。
まず CPU リソースについては、Solaris OS が認識している CPU リソースを自動的に Oracle データベースが検知し、初期化パラメータ cpu_count（スレッド単位）を自動調整します。

```
$ sqlplus / as sysdba
…
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, Automatic Storage Management, OLAP, Advanced Analytics
and Real Application Testing options

SQL> show parameter cpu_count

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
cpu_count                            integer     512
```

上記の例では、SPARC M10-4S(SPARC64 X+ CPU チップ)を使用し 16CPU/256 コア(512 スレ

ッド)を Solaris OS に割り当てた場合に、Oracle 12*c* が CPU スレッド数を自動認識している状況を確認しています。

次に、Oracle 12*c* が利用するメモリサイズの設定と確認をします。Oracle 12*c* が利用するメモリサイズは自動変更されないため、初期化パラメータ memory_target で調整します。（データベースには、動的拡張で使用する可能性のある最大のメモリサイズをあらかじめ memory_max_target パラメータに設定しておきます。Oracle 12*c* は、memory_max_target の値まで動的にメモリサイズを拡張できます。）

```
SQL> show parameter memory_

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
memory_max_target                    big integer 6144G
memory_target                        big integer 3072G

SQL> alter system set memory_target = 6144G;

System altered.

SQL> show parameter memory_

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
memory_max_target                    big integer 6144G
memory_target                        big integer 6144G
```

上記の通り、SPARC M10-4S では、CPU/メモリのハードウェアリソースが動的に増強可能で、稼働中の Solaris OS/Oracle 12*c* も正しくリソースを認識することが可能です。

Containar Dabase(CDB)が利用できる CPU/メモリが動的拡張されたので、CDB 上で稼働する Pluggable Database(PDB)を追加起動します。

```
SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB001                         READ WRITE NO
         4 PDB002                         READ WRITE NO
…
```

```
SQL> alter pluggable database pdb003 open;

Pluggable database altered.

SQL> show pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB001                         READ WRITE NO
         4 PDB002                         READ WRITE NO
         5 PDB003                         READ WRITE NO
…
```

上記の例では、PDB003 を追加起動し、その状態(OPEN MODE)が”READ WRITE”に設定されていることが確認できます。

本検証では、上述したリソース追加操作を実施し、下記のように段階的にリソースを追加しながら、トランザクションの増加とレスポンスを測定しました。

・SPARC M10-4S　1 台の状態で CPU コアのリソース追加（16 コアから 64 コア）
・SPARC M10-4S　1 台づつの 2node RAC 構成から開始し、両ノードに SPARC M10-4S を 1 台づつ追加し、4 台の SPARC M10-4S で 1 ノードを構成する RAC 環境まで拡大（図 11）

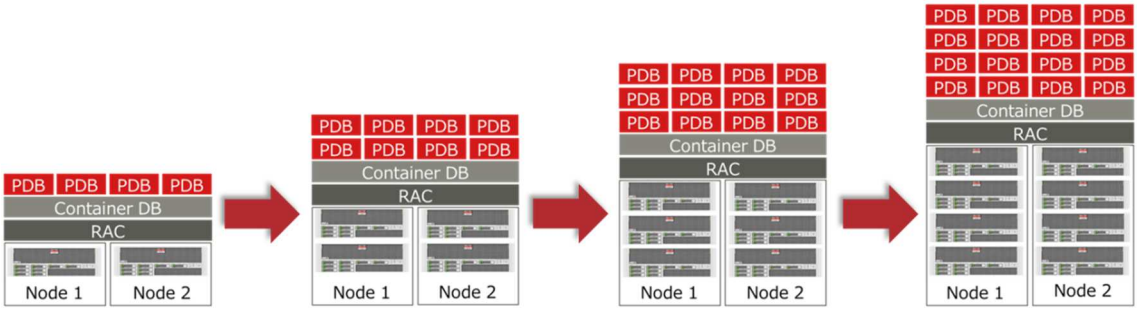


図 11　DR による SPARC M10-4S と PDB の段階的な追加

注-　本書に記載されているコマンドや設定項目、表示例は、あくまでも一例です。
実際のシステムにおいては、その構成等により、必要な設定や表示が異なります。
必ず、各種マニュアルをご参照下さい。

4.3 検証結果

CPU コア アクティベーションによるリソース追加（16 コアから 64 コア）と DR によるリソース追加（2BB から 8BB）時のトランザクション処理量（TPS）と平均レスポンスタイムを計測した。（表.1） リソースの追加と共に、平均レスポンスタイムは一定に保ちながら、トランザクション処理量が増加しています。

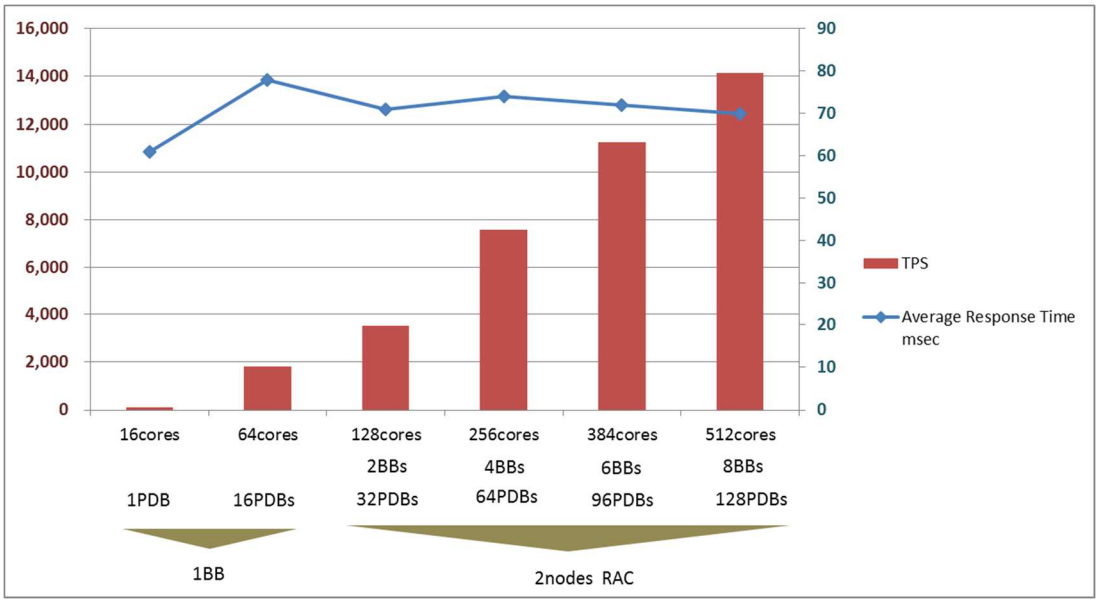


表 1 DR による SPARC M10-4S と PDB の段階的な追加

しかし、リソースの追加を中断し負荷を継続して増加させると、平均レスポンスタイムは悪化し、トランザクション処理量も増加しない結果となりました。（表 2）

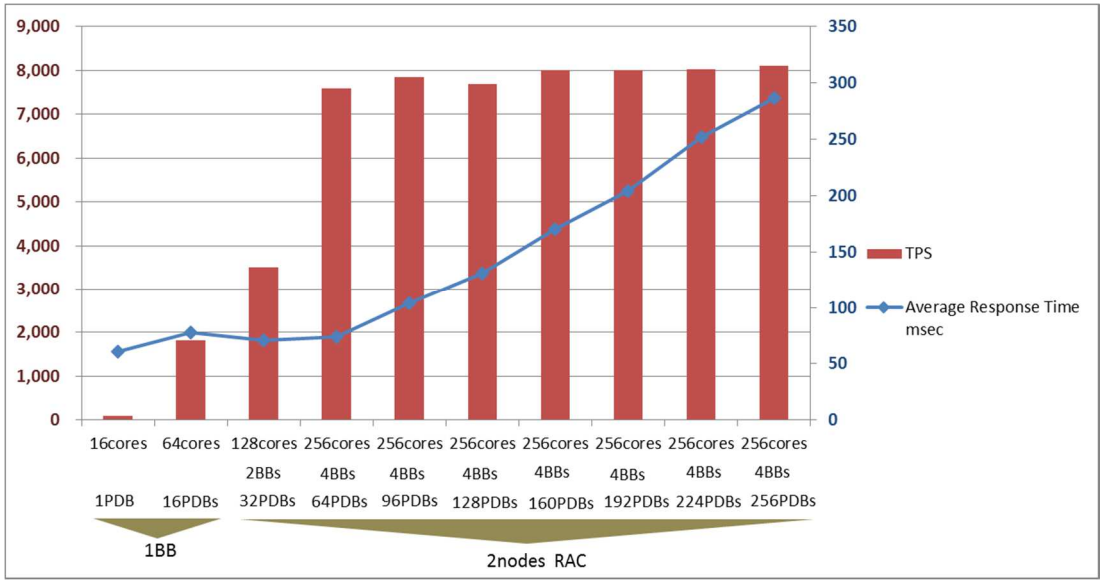


表 2 SPARC M10-4S を 4BB までのリソース追加で中断

本結果より、Oracle Database 12*c* の Oracle Multitenant と SPARC M10-4S の CPU コア アクティベーションおよび DR によるサーバリソース増強は、データベースの統合・拡張に、非常に有効であることを実証できました。

5. 総括

本検証から、Oracle Database 12*c* の Oracle Multitenant による PDB 追加、SPARC M10-4S の CPU コア アクティベーション機能、DR 機能によるリソース追加を利用することで、稼働中のサーバ（Oracle データベースのサービス）を停止することなく拡張することが確認出来ました。

Oracle Database 12*c* の Oracle Multitenant でのデータベース統合は、CPU の効率的な使用や、メモリ使用領域、ストレージ I/O の削減などにより、必要なハードウェアリソースの抑制、効率的な管理を実施するための機能により、投資の抑制や運用コストの削減が可能です。

また、SPARC M10-4S は、高性能 CPU、大容量メモリが搭載可能で、さらにビジネスの拡大や、他サーバを統合する段階に合わせて SPARC M10-4S を 16 台まで拡張できるため、初期導入費用を抑えることができ、ビジネス規模に合わせた適切な投資が可能です。

それぞれの製品単体でご利用されても、従来製品以上のメリットが見込まれますが、本資料で述べたように、Oracle Database 12*c* と SPARC M10-4S で構築されたデータベースサーバであれば、従来のデータベースの統合やビジネスの成長に合わせてパフォーマンスを維持するためのデータベースの拡張・拡大が非常に簡易に実現できます。これは、Oracle Database 12*c* と SPARC M10-4S の相乗効果により、ソフトウェアやハードウェアの費用に限らず、システムインテグレーション費用、運用・保守費用まで含めて、さらに適切な投資が行えることになります。

Oracle Database 12*c* と SPARC M10-4S を組み合わせてデータベースサーバとすることは、効率性、柔軟性、投資抑制、ICT コスト削減の面から、単なるデータベースサーバ統合の基盤というだけではなく、新たなビジネスプロジェクトやサービスの開始時にデータベースを迅速に提供できるデータベースクラウド基盤として、最適と言えるでしょう。

富士通株式会社
〒105-7123
東京都港区東新橋 1-5-2
汐留シティセンター

本書に記載されている内容は改善のため、予告なく変更することがあります。
富士通株式会社は、本書の内容に関して、いかなる保証もいたしません。
また、本書の内容に関連した、いかなる損害についてもその責任を負いません。
UNIX は、米国およびその他の国におけるオープン・グループの登録商標です。
SPARC64、SPARC64 ロゴ、およびすべての SPARC 商標は、米国 SPARC International, Inc. のライセンスを受けて使用している、同社の米国およびその他の国における商標または登録商標です。
その他各種製品名は、各社の製品名称、商標または登録商標です。
本書に記載されているシステム名、製品名等には、必ずしも商標表示（(R)、TM）を付記していません。

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries
Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US


blogs.oracle.com/oracle


facebook.com/oracle


twitter.com/oracle


oracle.com

Hardware and Software, Engineered to Work Together