Oracle Database Technology Night
～ 集え！オラクルの力（チカラ）～

高可用性と高拡張性を両立する
Oracle RAC
～ 改めて基礎からシンプルに理解する ～

日本オラクル株式会社
クラウド・テクノロジー事業統括
Cloud Platform ソリューション本部
Database ソリューション部
佐々木 亨

- 以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

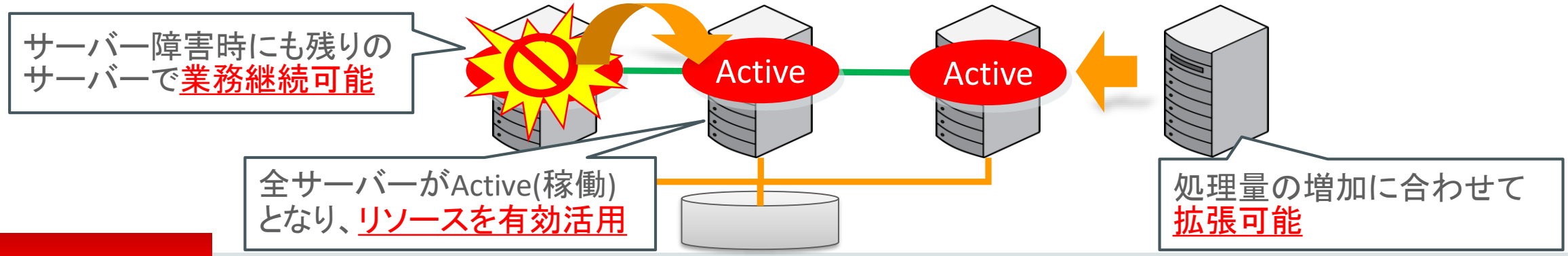
お伝えしたい内容

- RACを利用することで高可用性と拡張性が得られる
- RACもシングルインスタンスも基本的な考え方は同じ

Oracle Real Application Clusters (RAC) とは

- 複数ノードで構成する共有ディスク/共有キャッシュ型のデータベース
  - 全ノードが全データに直接アクセス可能
  - Cache Fusion Technologyで、キャッシュ・データの一貫性を維持
- 主な特徴

可用性	• サーバー障害時にも残りのサーバーで業務を継続可能
拡張性	• 負荷の増減に応じた処理性能の最適化が可能
投資コスト	• リソースの有効活用により 最適な投資コストを実現



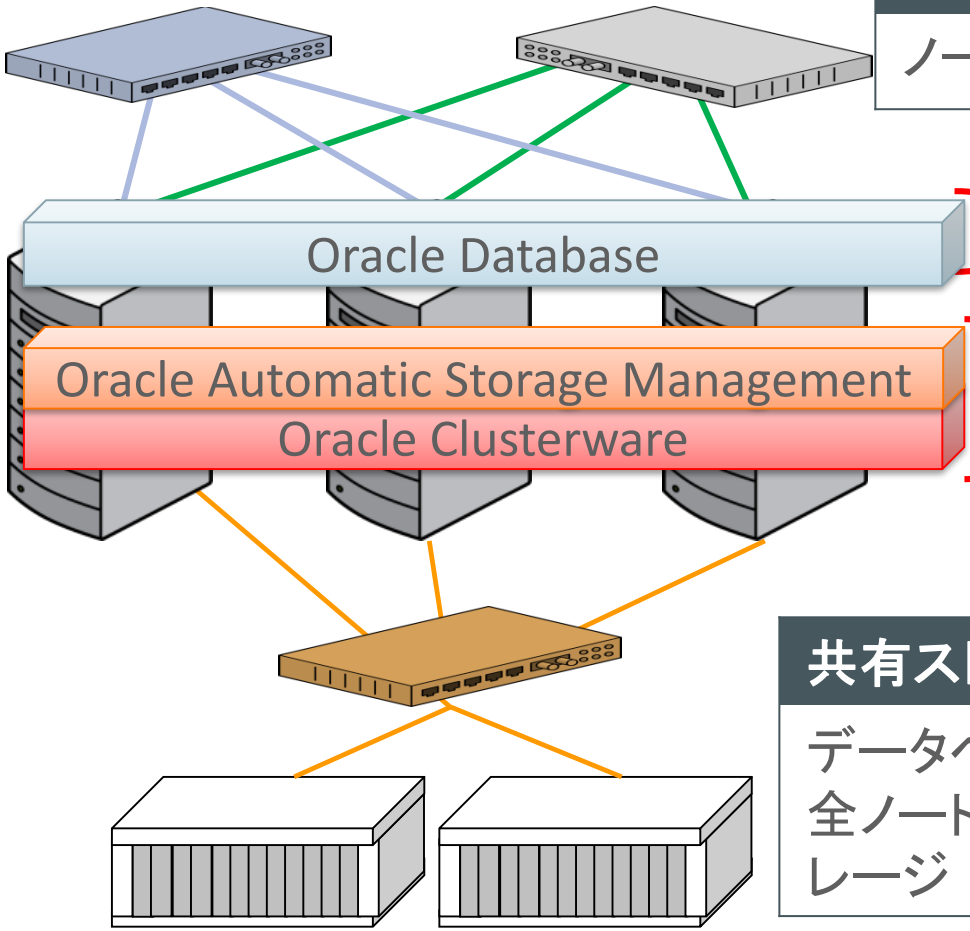
RAC に必要な H/W & S/W 構成

インターコネクト・ネットワーク

ノード間通信を行うネットワーク

パブリック・ネットワーク

クライアントとの通信を行うネットワーク



Database

Grid Infrastructure

ノード

RACを構成するデータベース・サーバー

共有ストレージ

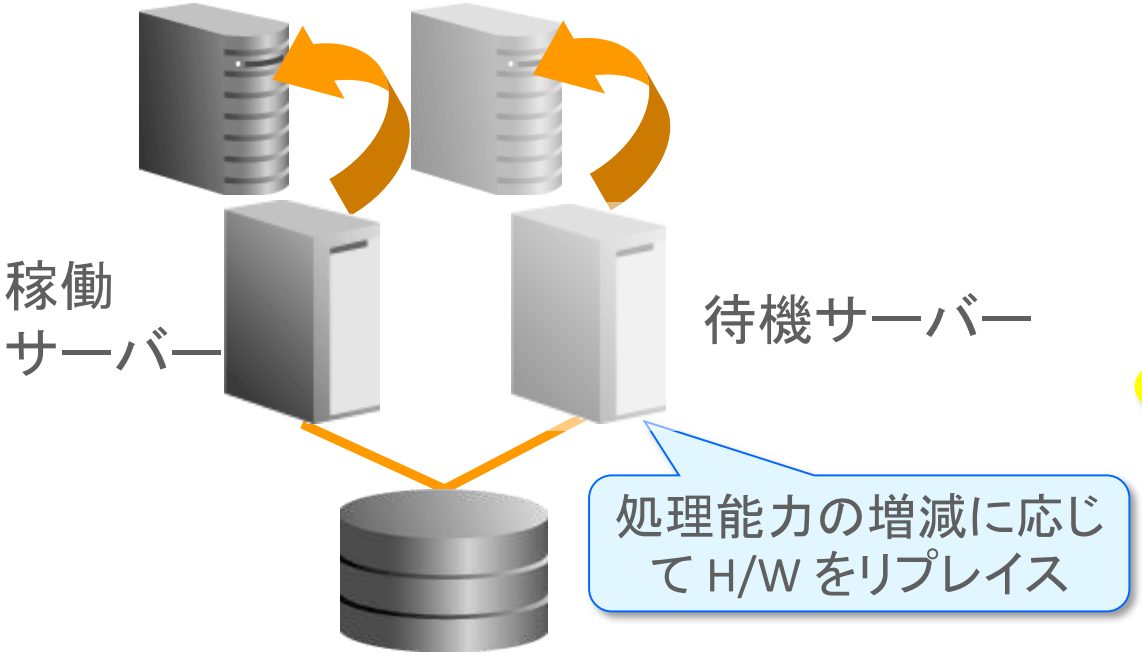
データベース・ファイルを配置する全ノードからアクセス可能なストレージ

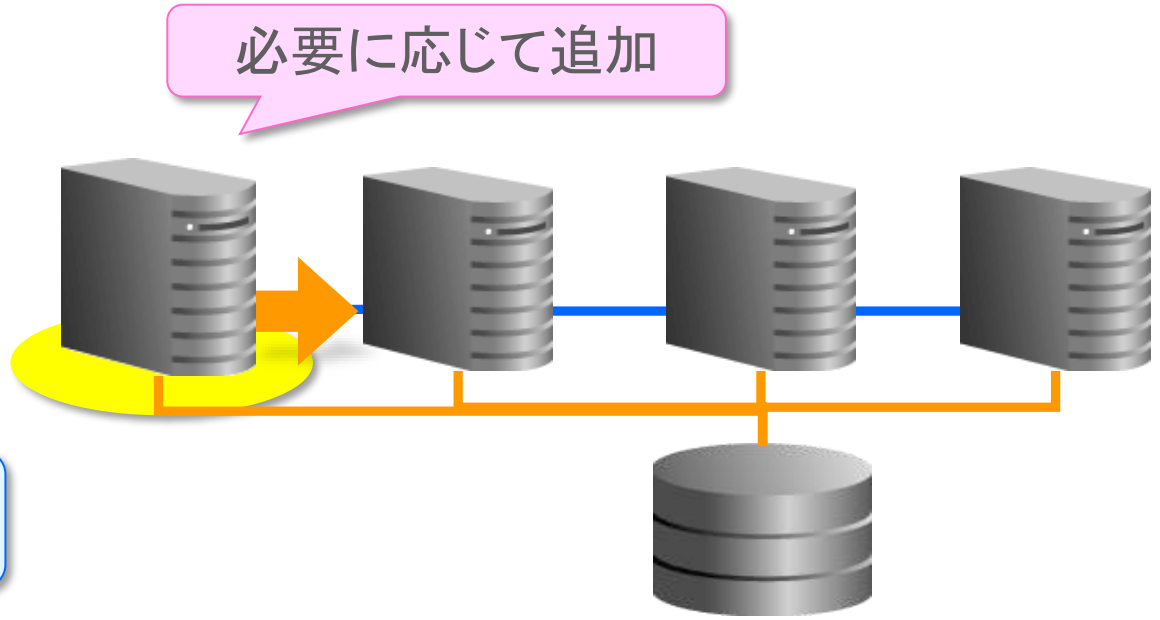
拡張性

RAC による高拡張性の実現

サーバー追加によるスケーラビリティの向上

- 必要に応じてサーバを追加し、処理能力の拡張が可能



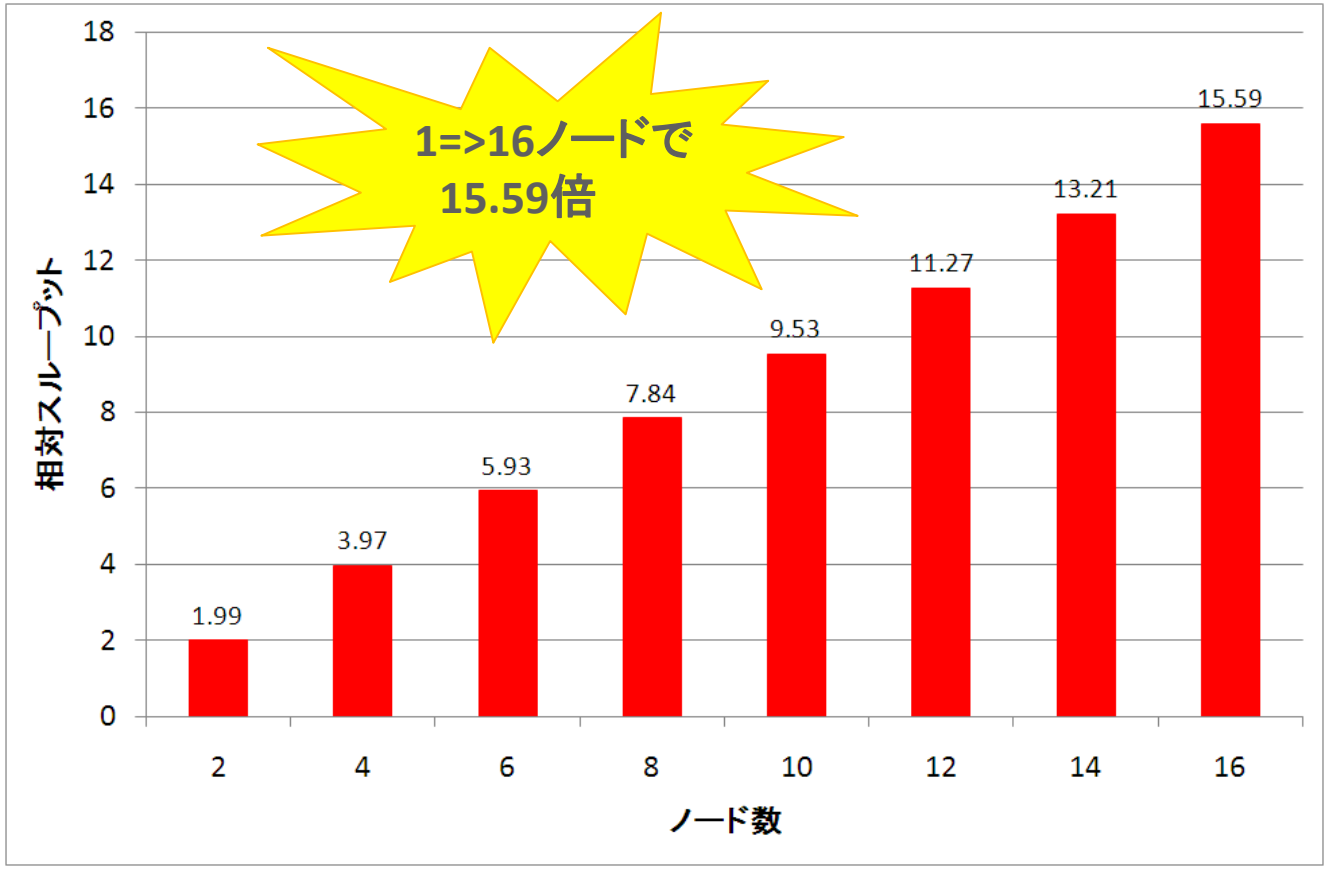


HA構成　入れかえ

RAC　買いたし

サーバー追加で処理能力は本当に向上するのか？

Oracle GRID Center での検証結果の例（検索処理中心、更新含むトランザクション10%）

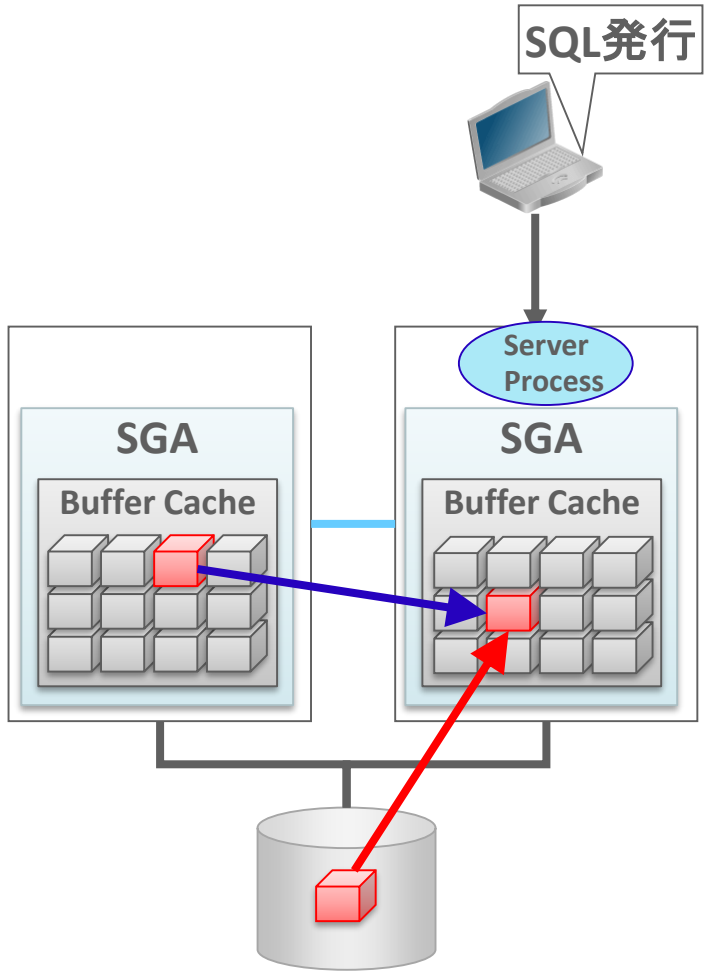


シングルインスタンスにおける
チューニングと同じ考え方に基づき
チューニングを実施することで
リニアにスケールした

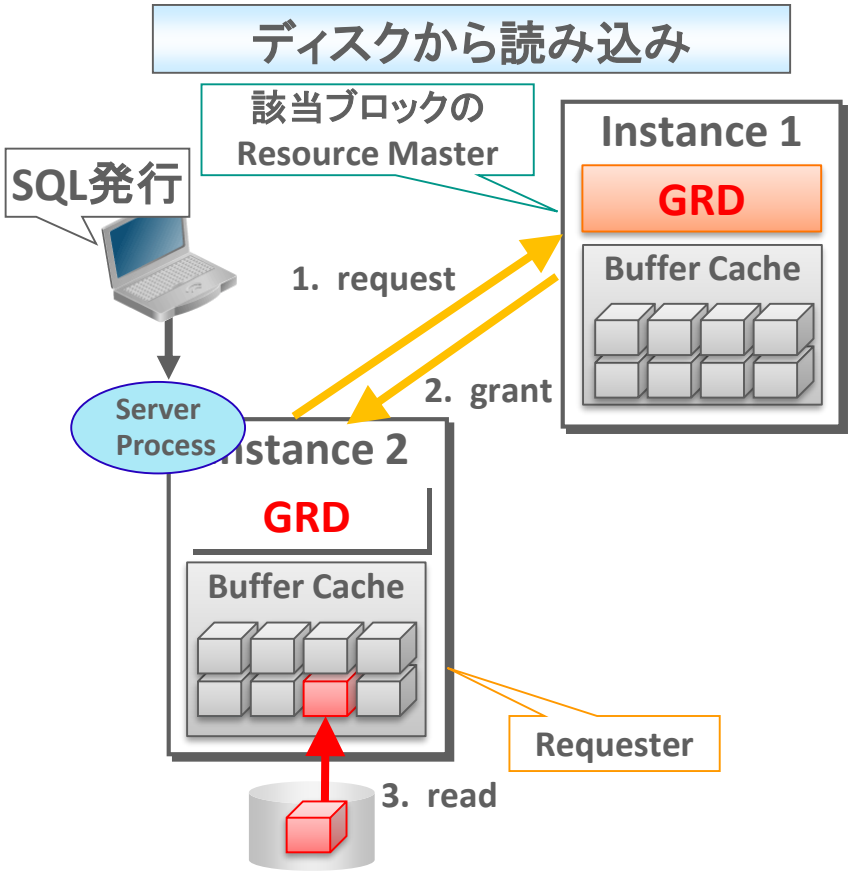
RACのデータアクセス

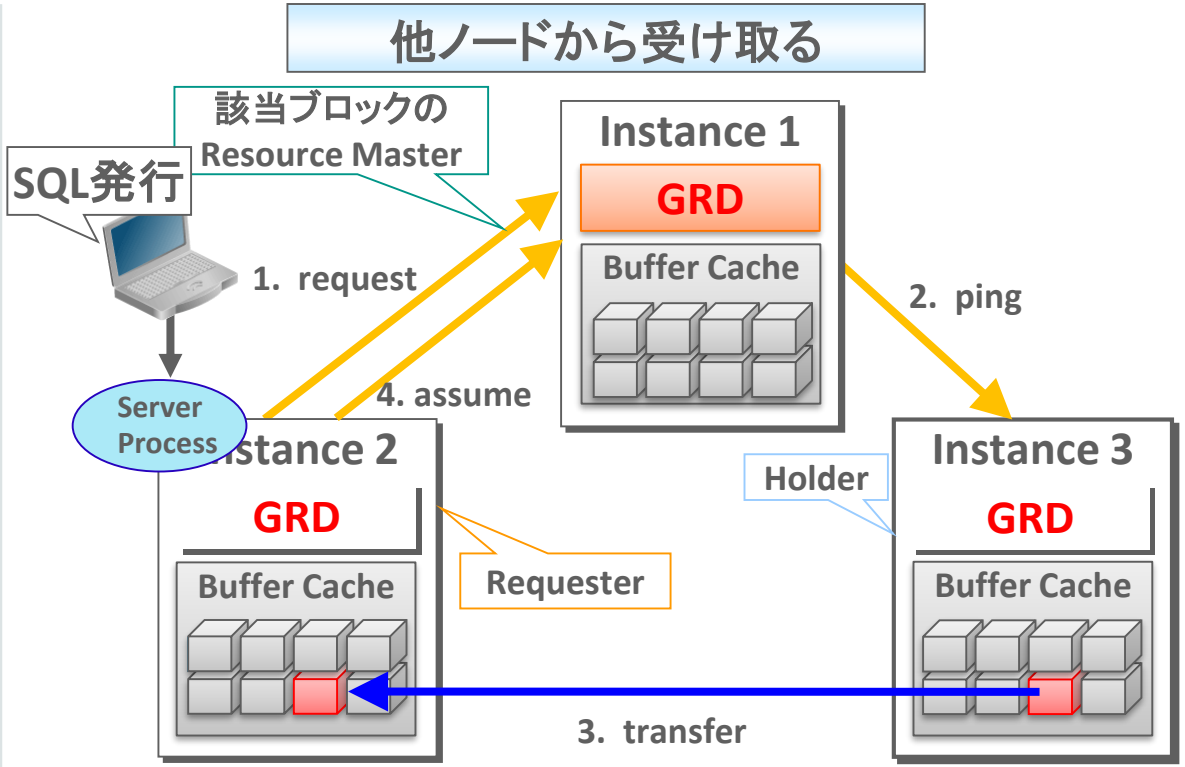
シングルインスタンスと同じように低速なディスクアクセスを減らすという考え方で動作

- キャッシュ・ヒットした場合
  - キャッシュのデータを使用
- キャッシュ・ミスした場合
  - シングルインスタンスの場合
    - ディスクから読み込む
  - RACの場合
    - （存在すれば）他ノードのキャッシュから受け取る
    - ディスクから読み込む



Cache Fusion の動作





- DBブロックAを必要とするインスタンス(Requester)
- Aのリソースマスターであるインスタンス (Resource Master)
- Aの最新イメージを保持しているインスタンス (Holder)

RACにおける考慮ポイント

アプリケーション・データベース設計

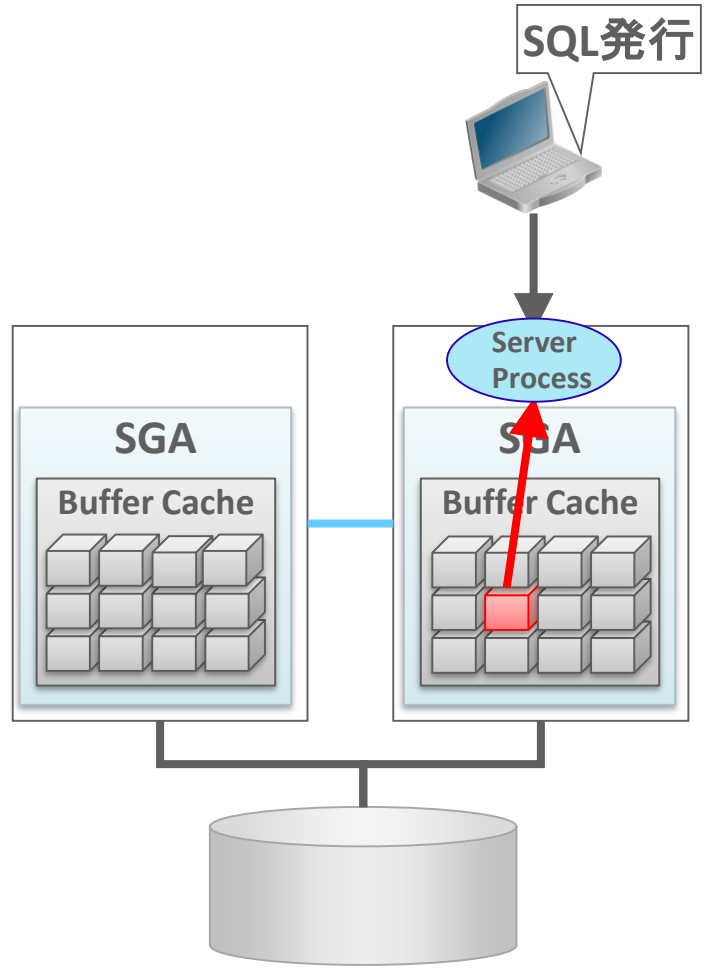
- ノード追加によって増えたCPUにデータを供給するI/O性能の向上は、RAC ノード追加と併せて検討すべき

- アプリケーションやデータベース設計はチューニングにおいて考慮すべき点
- チューニング・ポイントの例
  1. キャッシュ・ヒット率を改善
  2. ブロックへのアクセスを分散させる
  3. 不要なノード間通信を発生させない
- 上記はシングルインスタンスでも同様に重要なチューニング・ポイント

→ シングル・インスタンスのチューニング・ポイントはRACでも同じように通用する

1. キャッシュ・ヒット率を改善

- SQLが実行されるローカル・ノード上のメモリに必要なデータがあれば、そのデータブロックを利用
  - メモリ(高速)へのアクセスで完結
  - Disk I/O(低速)は起こらない
- 「ローカル・ノード上の」はひとまず置いておいて、まずは、データベース全体でのキャッシュ・ヒット率改善を目指す
  - OLTP システムにおける Full Scan を避ける
  - Buffer Cache を適切なサイズにする
  - Oracle Database のデータ圧縮機能を使う
- 上記のチューニングは、シングル・インスタンスと同じ



便宜上2Node構成の図を使います

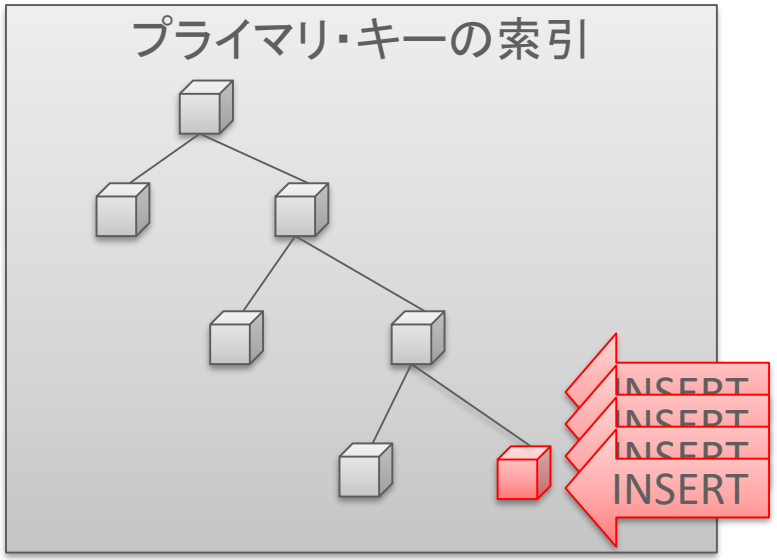
2. ブロックへのアクセスを分散させる

同一索引ブロックへのアクセスが集中する悪い例

Right Growing Index

- シーケンスから取得した単調増加する一意の値をINSERTする処理 (例:「注文番号」を INSERT)
- プライマリ・キーの索引の特定のリーフブロックに更新が集中する状況
- 同時実行数が増えればRACに限らず発生する、良くない状況

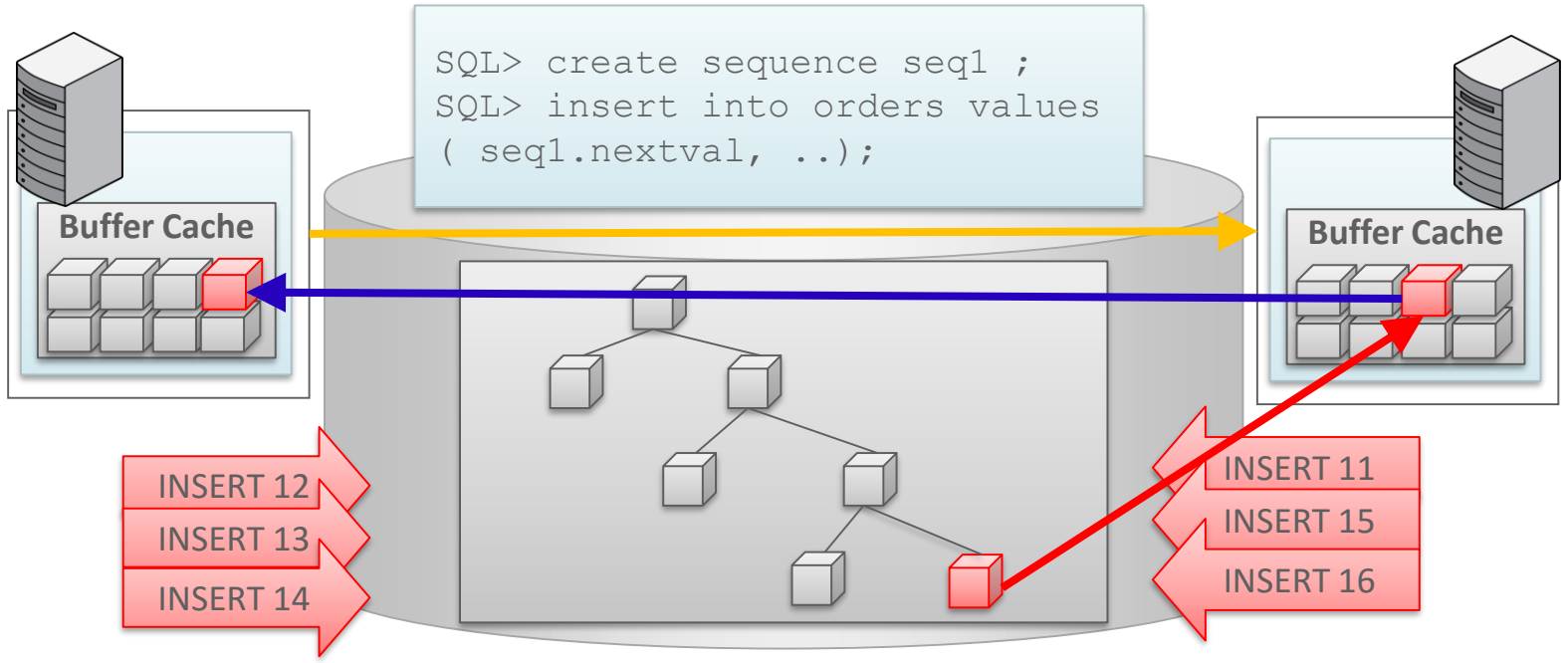
```
SQL> create table orders (
         orderid           number,
         customerid        number,
         orderdate         date,
                  :
         constraint orders_pk
         primary key (orderid)
     )
SQL> insert into orders values
     ( seq1.nextval,xxx,xxxxx,);
SQL> commit;
```



2. ブロックへのアクセスを分散させる

同一索引ブロックへのアクセスが集中する悪い例

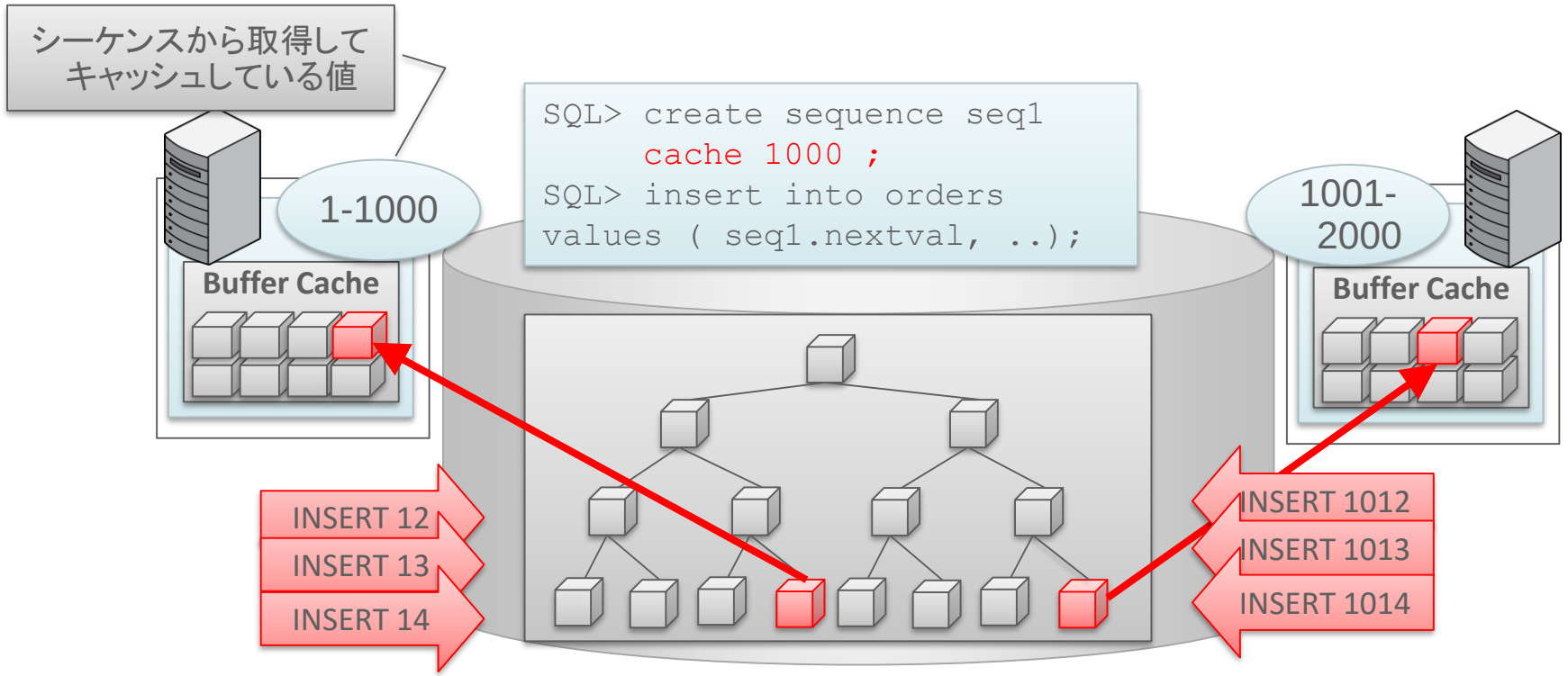
- 特定のリーフ･ブロックのノード間の転送が頻発する
- 他ノードのブロックの持つ転送を待つ部分が出てくる



2. ブロックへのアクセスを分散させる

シーケンスのキャッシュを増やす

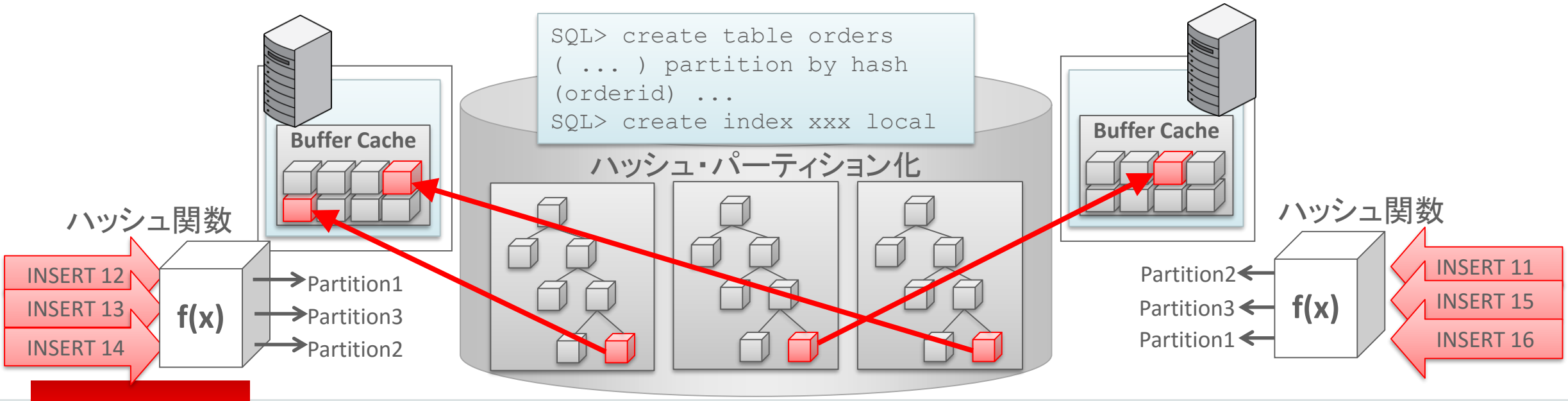
- シーケンスのキャッシュを増やすことで、各ノードからのINSERT処理が別のリーフ・ブロックに分散される



2. ブロックへのアクセスを分散させる

パーティショニング

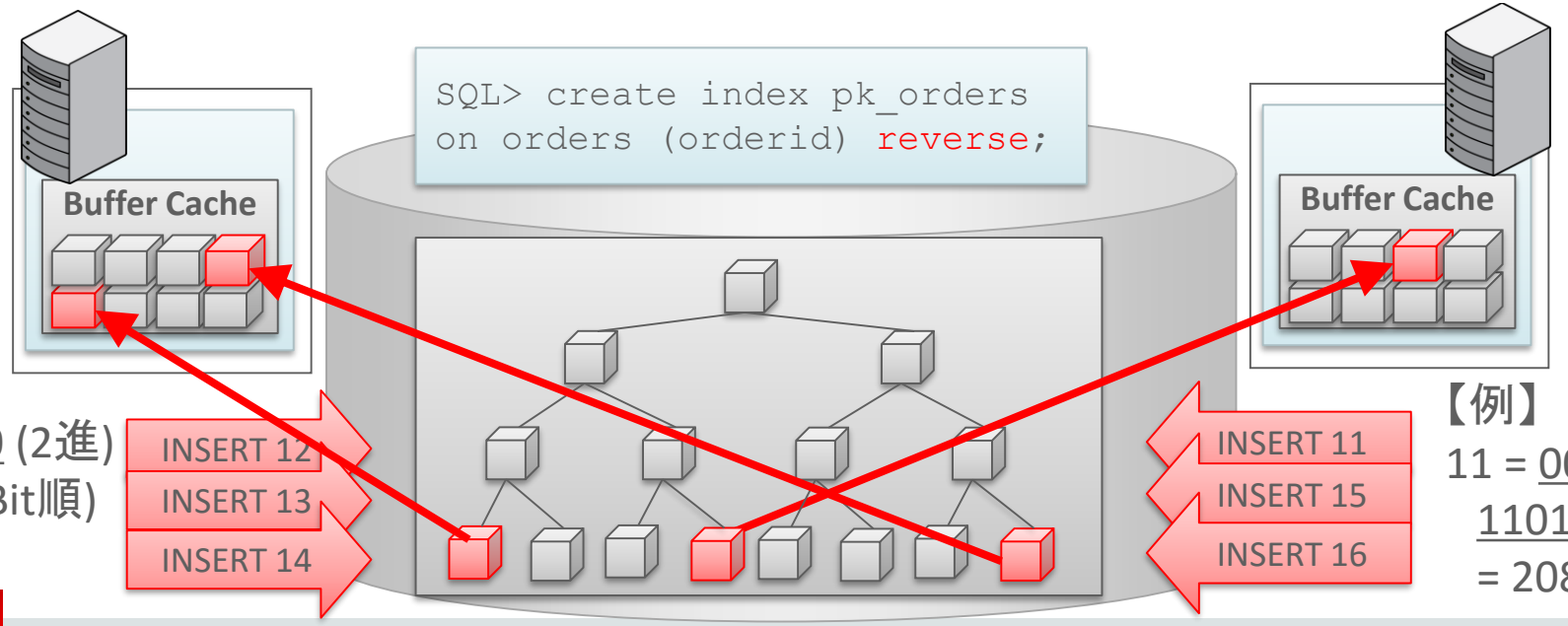
- プライマリ・キーの索引をパーティション化
- アクセス対象のリーフ・ブロックを全体で分散可能
  - シングルインスタンスにおいても有効なチューニング



2. ブロックへのアクセスを分散させる

逆キー索引

- プライマリ・キーの索引を逆キー索引化(逆ビット順に格納)
  - 大小関係が変化し、連続したキーでも異なるブロックに挿入され易くなる
  - シングルインスタンスでも有効なチューニング
  - 索引を使った範囲検索ができなくなる点には注意



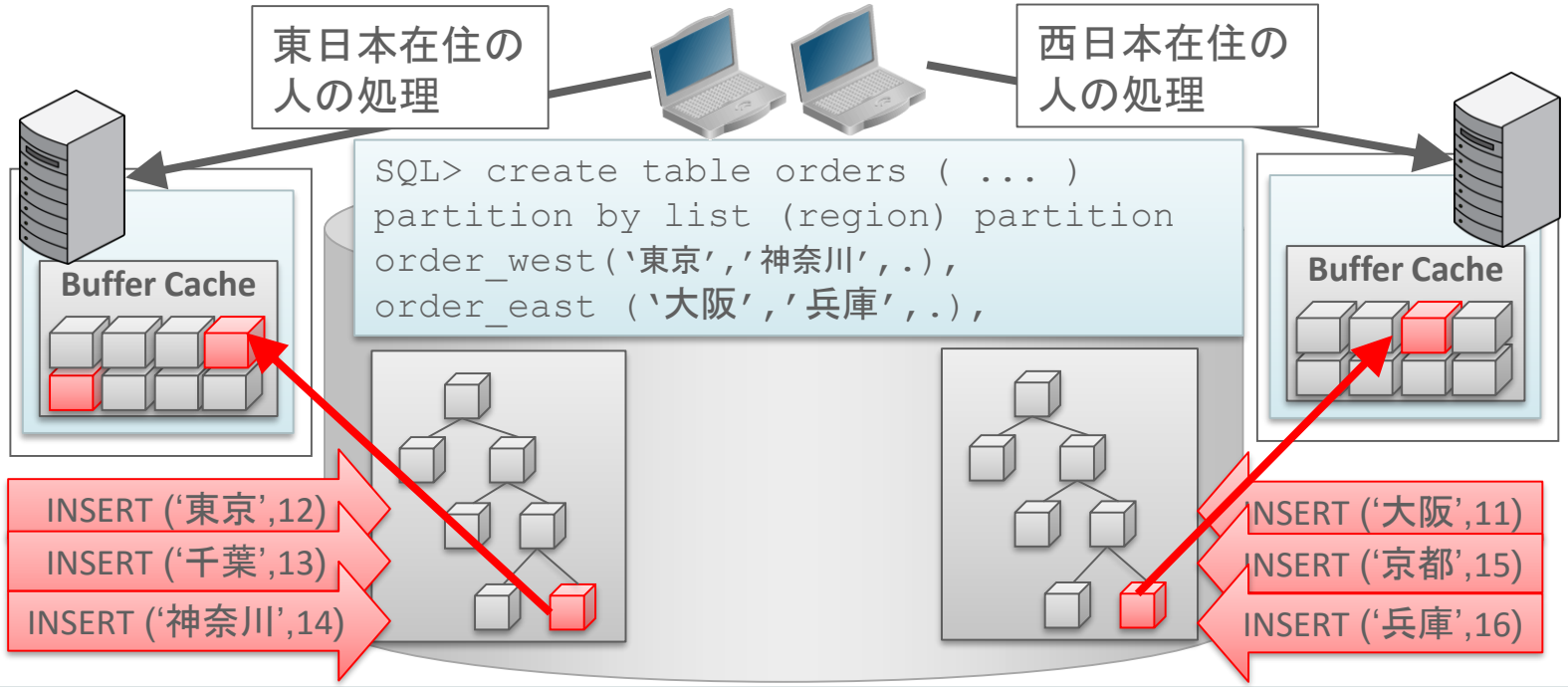
【例】
12 = 00001100 (2進)
00110000 (逆Bit順)
= 48

【例】
11 = 00001011 (2進)
11010000 (逆bit順)
= 208

2. ブロックへのアクセスを分散させる

アプリケーション・パーティショニング

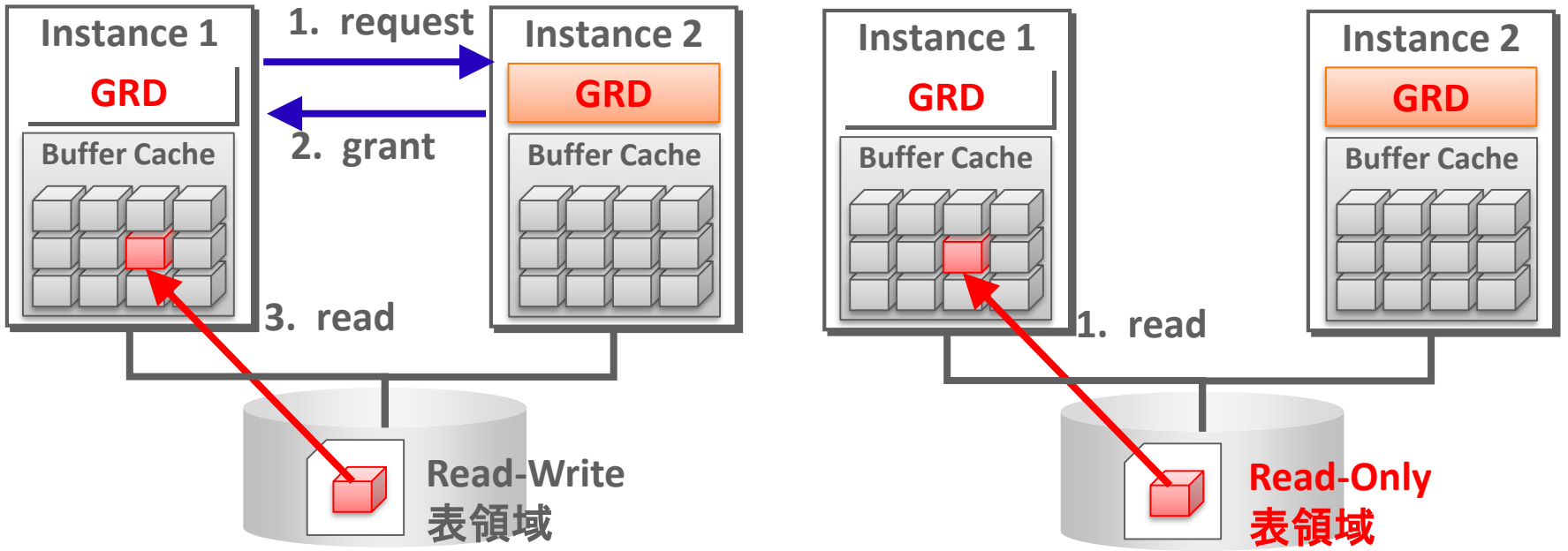
- 異なるノードから別のブロックにアクセスするようアプリケーションで制御(アプリケーション・ロジック的に可能な場合のみ)
- レンジ、リストパーティションなどと組合せると効果的



3. 余分な処理を起こさない

読み取り専用のデータはRead-Only表領域に配置

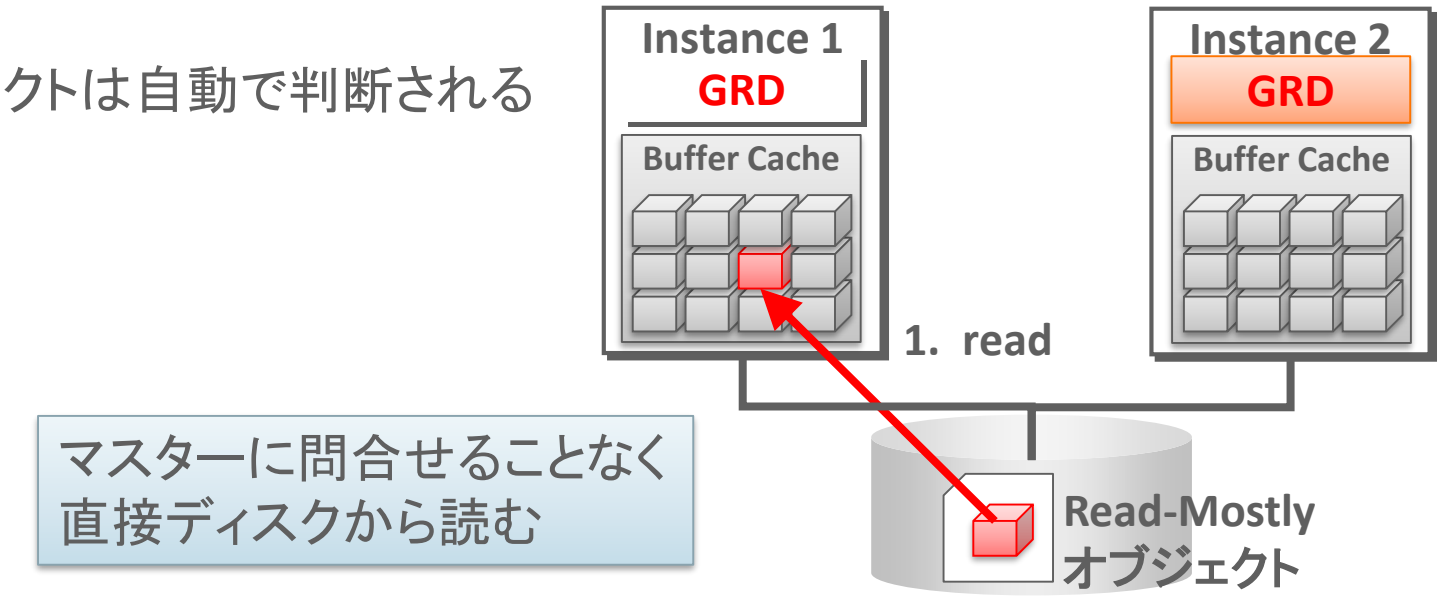
- 読み取り専用の表領域中のテーブルについては、Disk からの読み取りは Global Cache Operation を伴わない
  - 余分な通信によるオーバーヘッドを抑えることができる



3. 余分な処理を起こさない

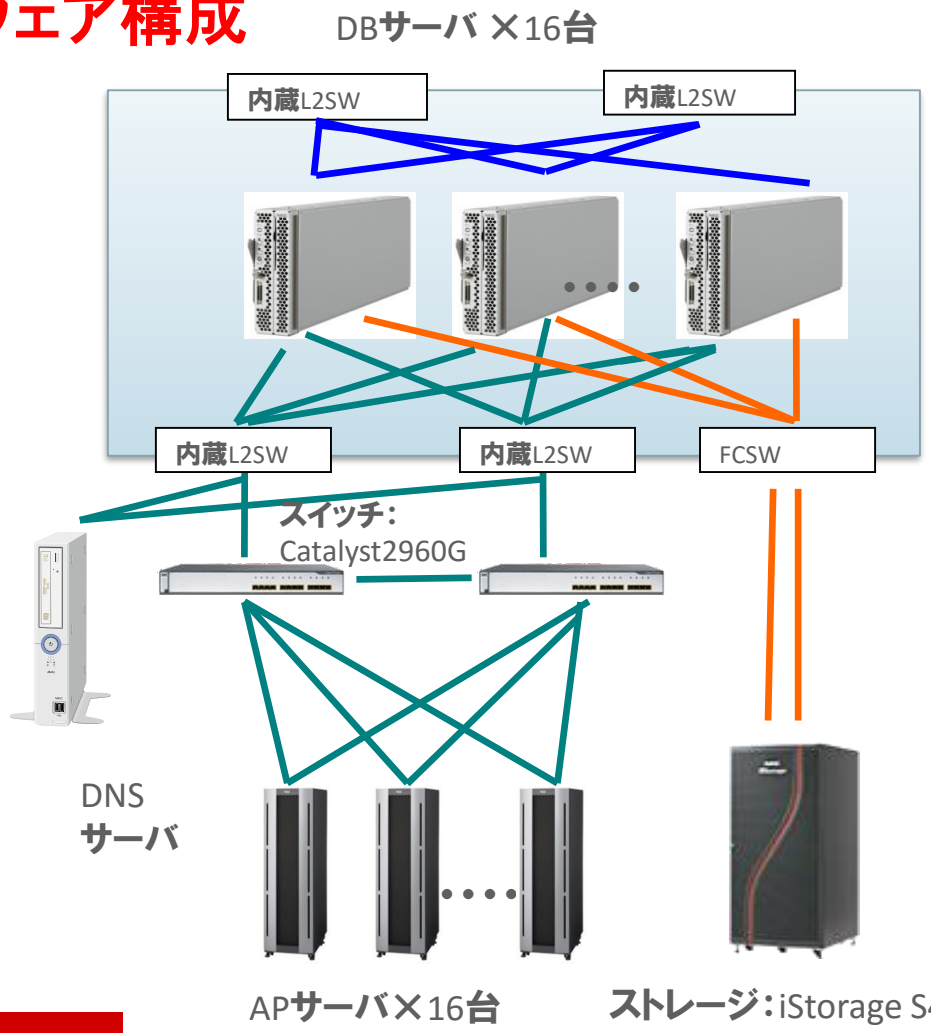
Read-Mostly Locking

- 99%はRead処理だけど、1%はWrite処理というオブジェクト(Read-Mostly) については、Read-Only にはできない
- Read-Mostly であるオブジェクトについては、マスターに問い合わせることなく直接ディスクから読み出す
  - Read-Only オブジェクトは自動で判断される



拡張性の検証

ハードウェア構成



DBサーバー（1台あたり）	**本体:** Express5800/B120a-d (N8400-089) **CPU:** インテル Xeon プロセッサー X5550 4Core * 2スレッド* 2CPU **メモリ:**48GB
クライアント（1台あたり）	**本体:** ECOCENTER(NE1000-001) **CPU:** 8Core **メモリ:**16GB
ストレージ	**本体:** iStorage S4900 **キャッシュメモリ:** 100GB

検証に用いたアプリケーション

Webショッピングサイトを模したアプリケーション

TX1（更新あり）

1. ユーザー･サインオン
   - SELECT ... FROM account, profile, signon, bannerdata ...
2. 商品検索
   - SELECT ... FROM category ...
   - SELECT ... FROM product ...
3. 商品選択
   - SELECT ... FROM item, product ...
4. 在庫数チェック
   - SELECT ... FROM inventory ...
5. 注文
   - (SELECT ordernum.nextval FROM dual)

更新処理

- INSERT INTO orders ...
- INSERT INTO orderstatus ...
- INSERT INTO lineitem ...
- UPDATE inventory ...
- COMMIT

TX2（検索のみ）

1. ユーザー･サインオン
   - SELECT ... FROM account, profile, signon, bannerdata ...
2. 商品検索
   - SELECT ... FROM category ...
   - SELECT ... FROM product ...
3. 商品選択
   - SELECT ... FROM item, product ...
4. 在庫数チェック
   - SELECT ... FROM inventory ...

TX1とTX2をそれぞれ「1トランザクション」とする

アプリケーションパーティショニングは実施しない

商品検索は平均100件程度がヒットする

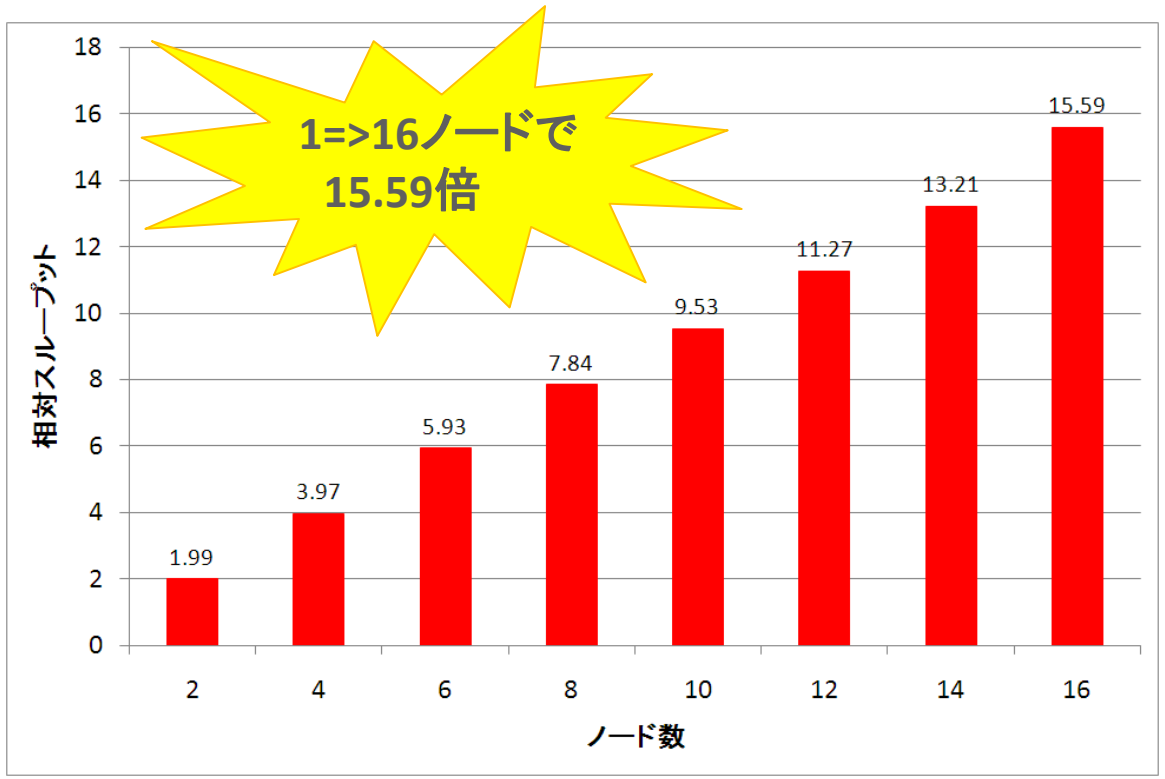
チューニング内容

使用したチューニング・ポイントの例

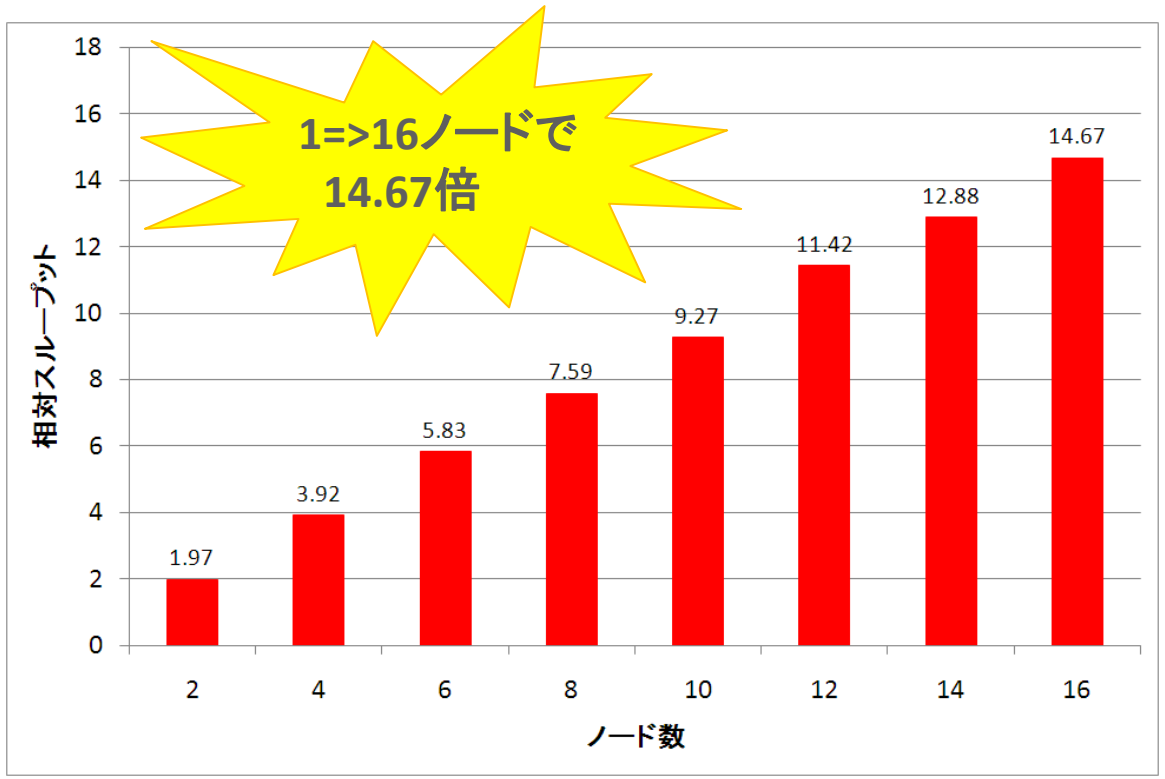
- キャッシュ・ヒット率を改善
- ブロックへのアクセスを分散させる
  - シーケンスのキャッシュ
  - パーティショニング
  - 逆キー索引
- 余分な処理を起こさない
  - 読み取り専用のテーブルをRead-Only
  - Read Mostly Lock

検証結果

Tx1:Tx2=1:9



Tx1:Tx2=5:5



可用性

RACの高可用性

サーバー障害時の継続稼働に必要な機能

1. 障害検知

- 各サーバー内のプロセス死活監視
- クラスタ内メンバーの死活監視

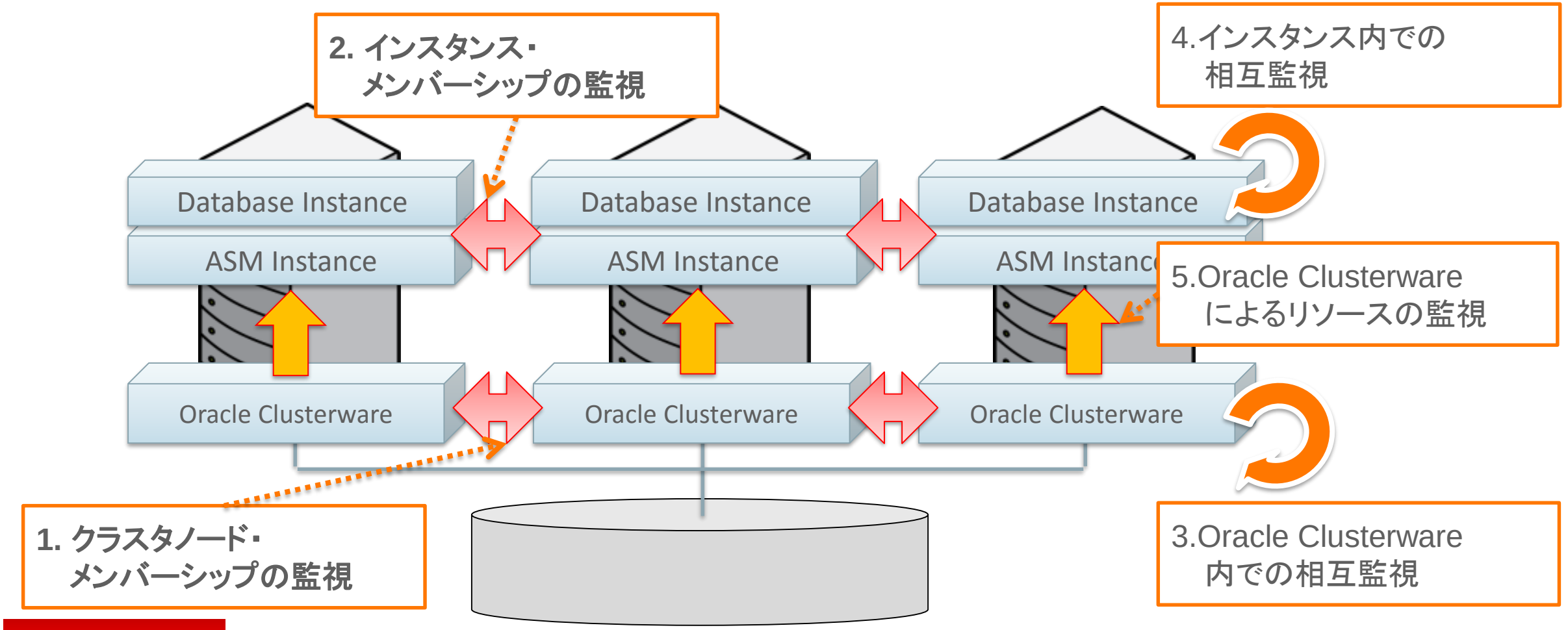
2. 障害サーバー切り離し

- クラスタ・ノード・メンバーシップ管理
- RACリソース再構成
- インスタンス・リカバリ

3. 生存サーバーへの接続

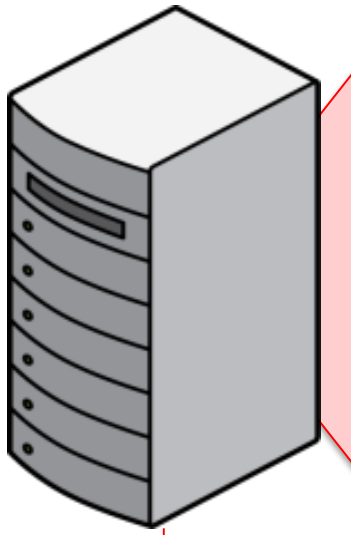
- サービス名を用いたRACデータベースへの接続
- 接続時のフェイルオーバーと仮想IPアドレス
- FANイベントとFCF

RAC が障害検知のために実施している監視



RACを構成するプロセス群

Oracle Clusterware と Oracle Database



Oracle Database スタック

RAC関連プロセス
(LMON, LMS, LMD, LCK, LMHB)

Oracle Clusterware スタック

CRS エージェント

Oracle Clusterware プロセス
(CSS/ CRSなど)

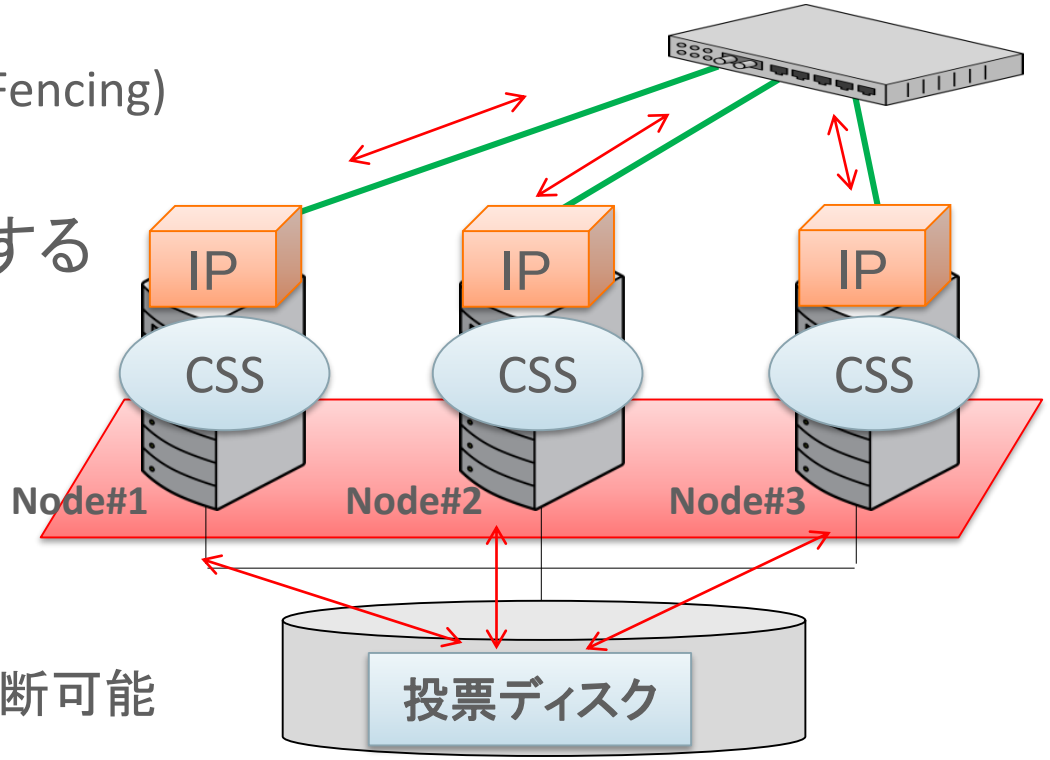
OHAS エージェント

Oracle High Availability
Services (OHAS)

クラスタ・ノードのメンバーシップ管理

Cluster Synchronization Services (CSS) によるクラスタノードの異常検知

- 各ノードの CSS がクラスタ・ノード構成の一貫性に責任を持つ
  - 相互に生存監視を実施し、異常ノードを切り離す
  - 異常ノードが勝手に協調せずにIOするのを防ぐ(IO Fencing)
- CSSは2種類のハートビートをして互いを監視する
  - ネットワーク・ハートビート（1秒毎）
    - タイムアウト misscount=30
  - ディスク・ハートビート
  - ネットワークとディスク双方に対してハートビートが一定時間途絶えた場合、相手はノード停止したと判断可能

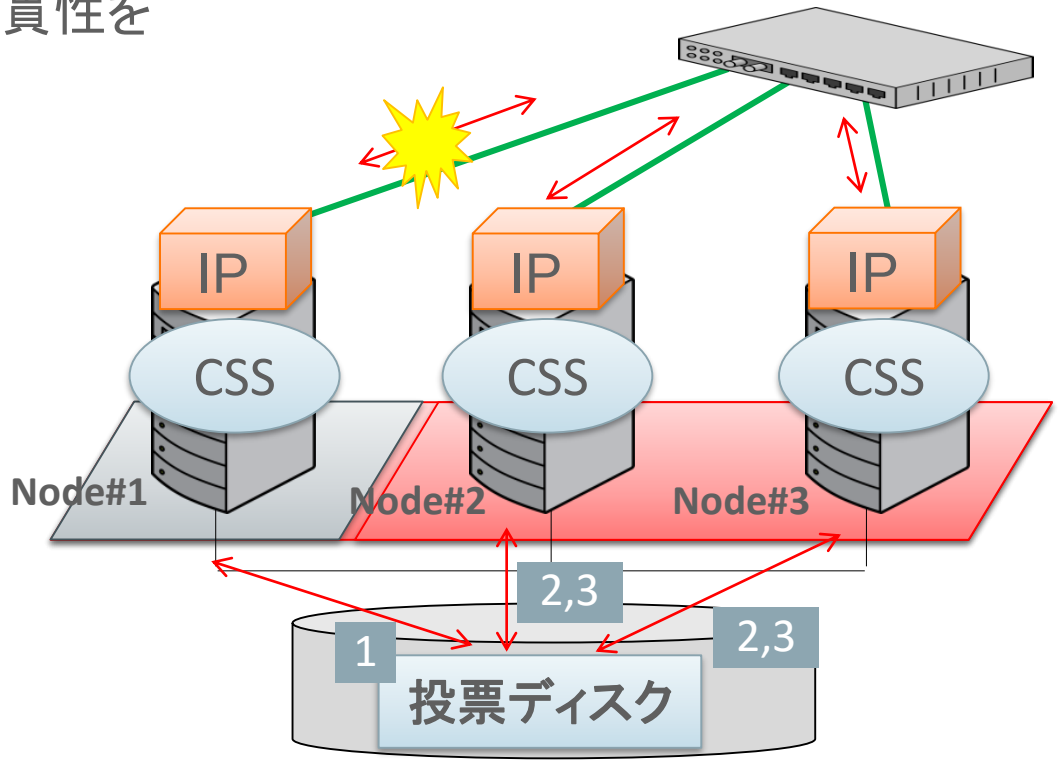


クラスタ・ノードのメンバーシップ管理

排除ノードの決定

- ノード間の通信にのみ問題がある場合
  - それぞれのノードが協調せず動作して、データの一貫性を損なわないように対処する
    → スプリット・ブレイン解決

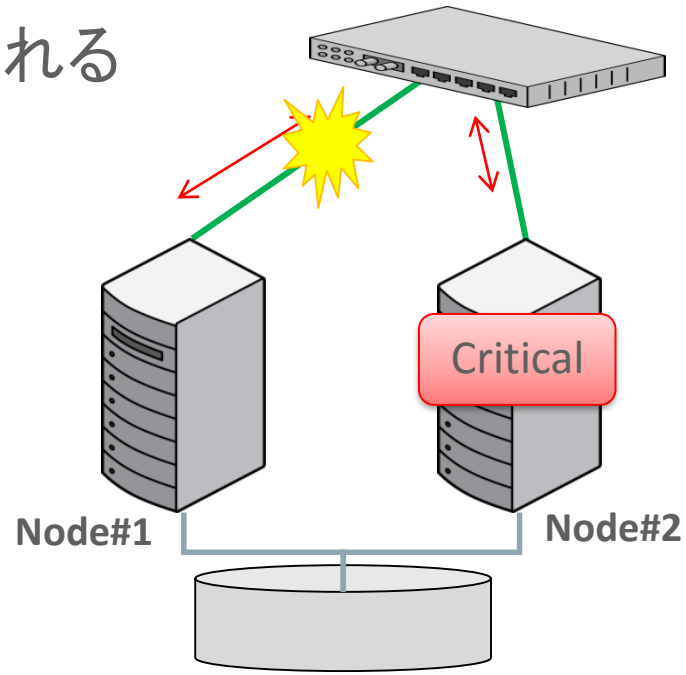
- 投票ディスクを介して、生き残るサブ・クラスタを調停する
  - 自分から見えるノードを投票する
  - 所属するノード数が最も多いサブ・クラスタを残す
  - 所属するノード数が同一の場合は、ノード番号が最小のノードを含むサブ・クラスタを残す



クラスタ・ノードのメンバーシップ管理

排除ノードの決定 Server Weight-Based Node Eviction（12.2～）

- クラスタ内のノードに対する負荷を考慮して重みづけする
- 具体的には、クラスタウェアの管理リソース（データベース、サービス）にフラグ(css_crticalというリソース属性)を付ける
- 同数のサブクラスタに分離された場合は次の基準で判断される
  - 重みづけの大きいノードを残す
  - Public 障害の発生していないノードを残す
- 右の例だと
  - 12.1まで：Node#1 が残る(若番を残すポリシー)
  - 12.2から：Node#2 が残る(重みづけにより)



クラスタ・ノードのメンバーシップ管理

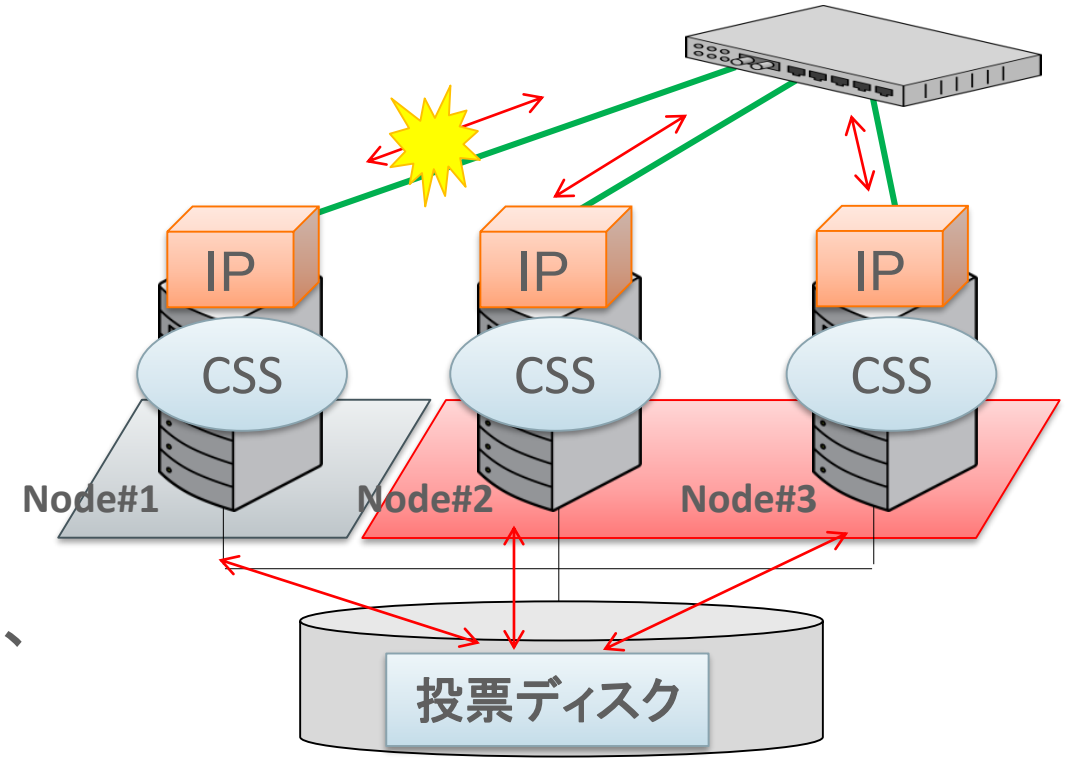
排除ノードの停止

排除ノードに対して停止命令を送る系統は3種類

A) ネットワーク・ハートビート経由

B) ディスク・ハートビート経由
（投票ディスク中のKILLブロック）

- 命令を受けるとACKを返して自分で停止
  - OS再起動
  - Grid Infrastructure の再起動 (11.2.0.2~)
- 停止命令を出したマスターにACKが返り次第、クラスタメンバーの再構成を実施

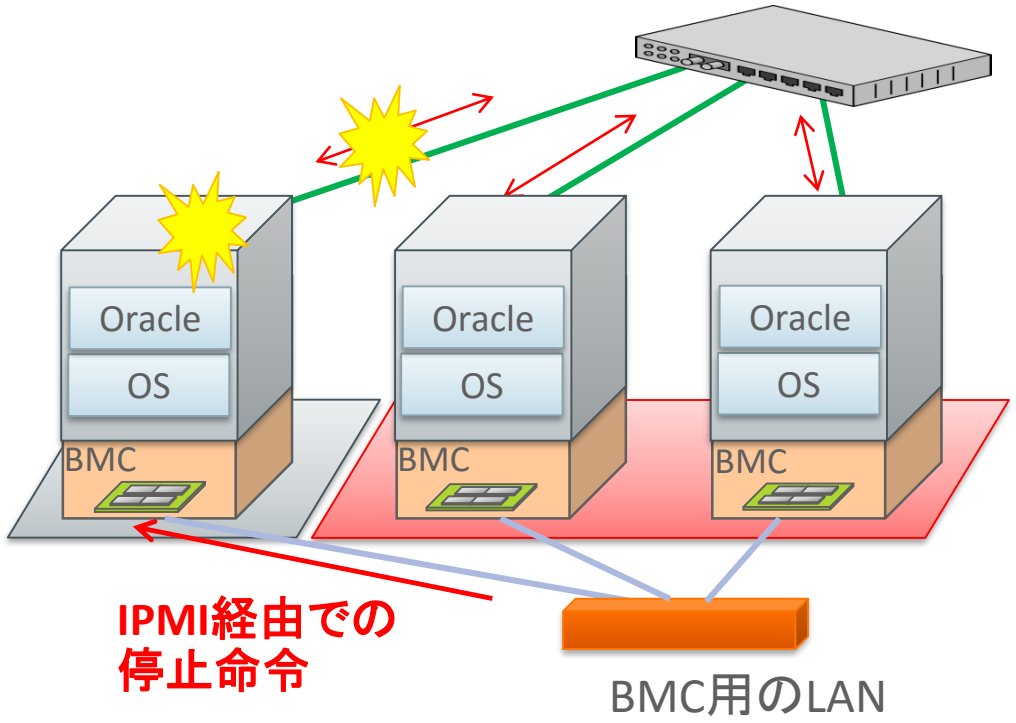


クラスタ・ノードのメンバーシップ管理

排除ノードの停止

C) IPMI（対応H/Wであれば使用可11.2 ~）

- BMC（Baseboard Management Controller）を経由して、サーバーを停止、再起動
- BMC は OS とは独立してサーバーを管理
- 障害ノードのOSが異常ケースでも、OS を介さないため確実にサーバーを停止可能

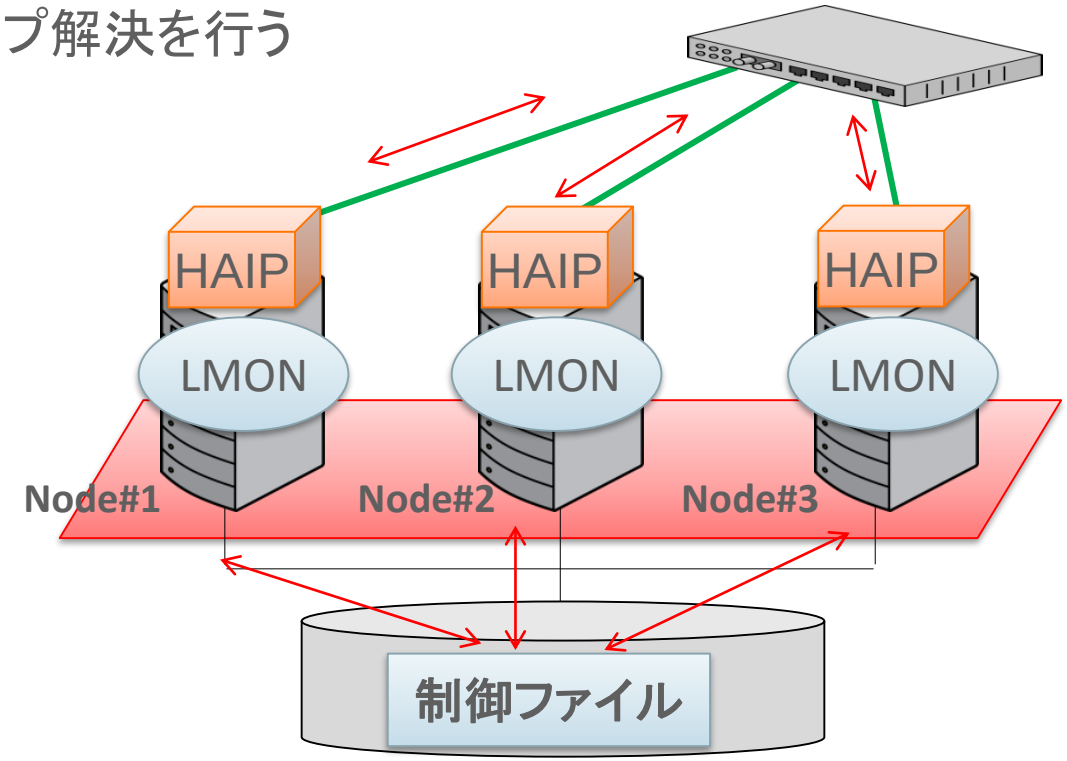


- IPMIによる停止命令が完了するタイミングはH/W依存(どの状態をサーバーの停止と捉えるか)

インスタンス・メンバーシップ管理

LMONによるハートビート

- 各ノードの LMON がメンバーシップを監視
  - インスタンスや回線障害の検知時にはメンバーシップ解決を行う
- 2種類のハートビートを行っている
  - ネットワーク・ハートビート
    - cluster_interconnect で指定されたネットワーク（11.2.0.2からは通常HAIP）
    - 一定時間ハートビートがない場合はタイムアウトし、メンバーシップの解決が行われる
  - ディスク・ハートビート（制御ファイル）

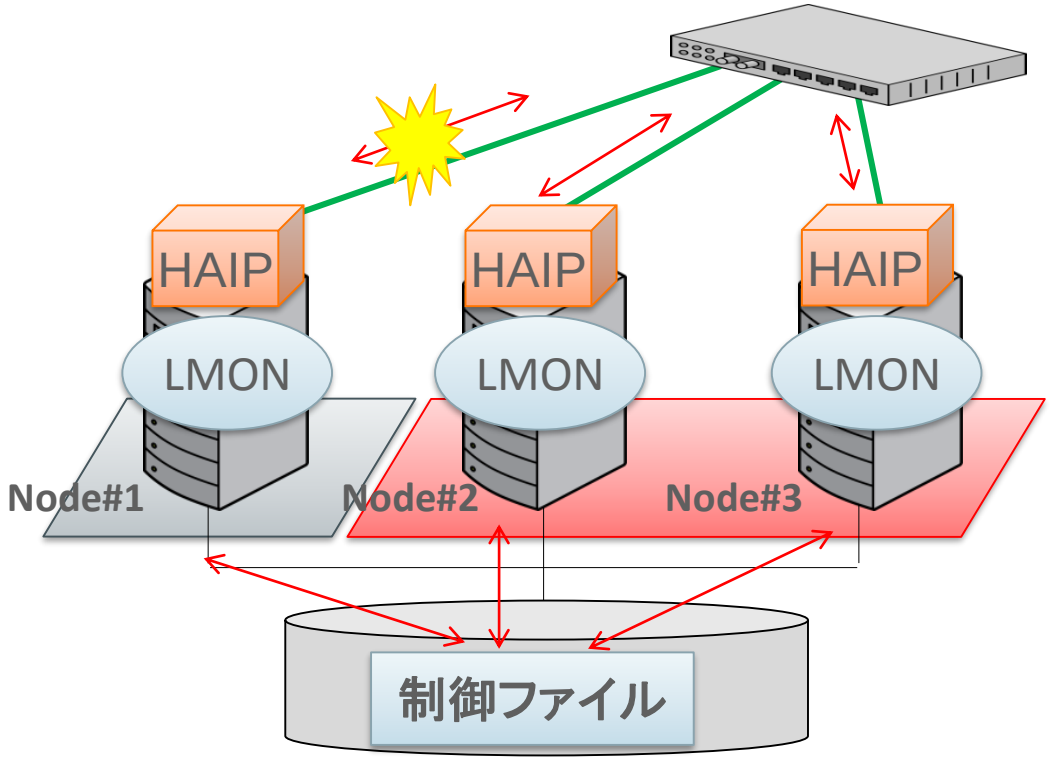


インスタンス・メンバーシップ管理

排除インスタンスの決定：IMR(Instance Membership Recovery)

- LMON のハートビートは通常、CSS によるハートビートと同一の
ネットワークを通して行われる
  - その場合はCSSが先に障害を検知、解決する
  - 通常、ネットワーク障害でLMONのハートビートがタイムアウトするケースはない

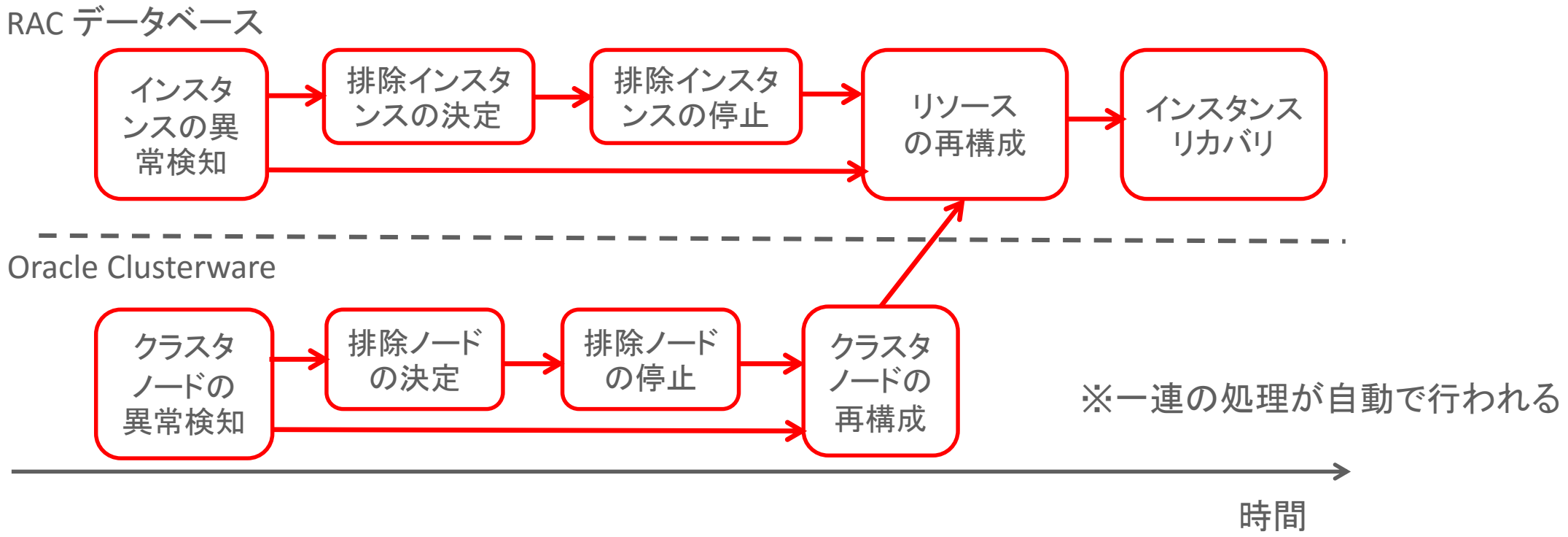
- ハートビートにより異常を検知した場合は
制御ファイルを用いて、生き残りインスタンスの解決を行う
  - IMR（Instance Membership Recovery）



障害検知後の動作

DB層の障害かサーバー層での障害かを分けて考えることが重要

- 出力されるログも異なる（DBのアラートログか、クラスタウェアのログか）



その他の監視

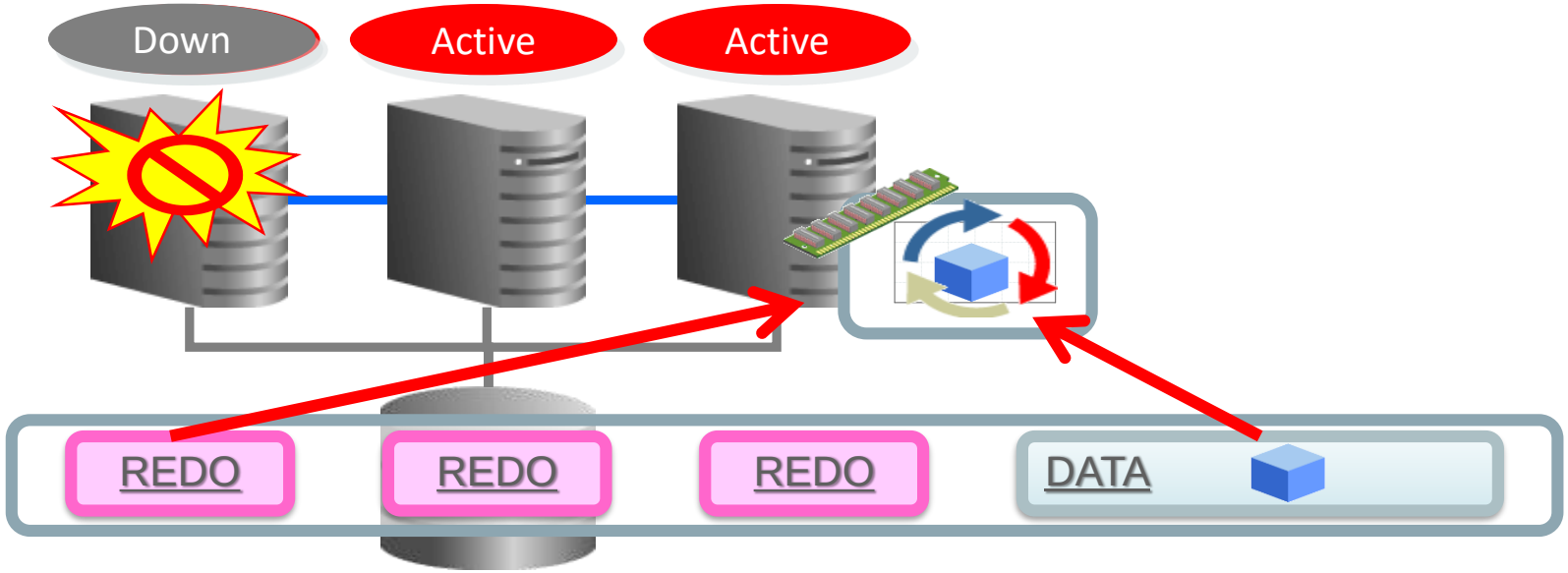
- ノード内での監視については本日のセミナーでは割愛
- 最終的にはノード停止、インスタンス再起動、プロセス再起動などの対処が行われる
- 対処後の動作は、前頁の通り

- 各監視内容については下記資料を参照ください
  - Oracle DBA & Developer Days 2012
    “Oracle Real Application Clusters/ Oracle Clusterware の高可用性機能”
    - http://otndnld.oracle.co.jp/ondemand/ddd/PDF/MA-2_print_c.pdf

データ・ブロックの自動リカバリ

インスタンスリカバリ

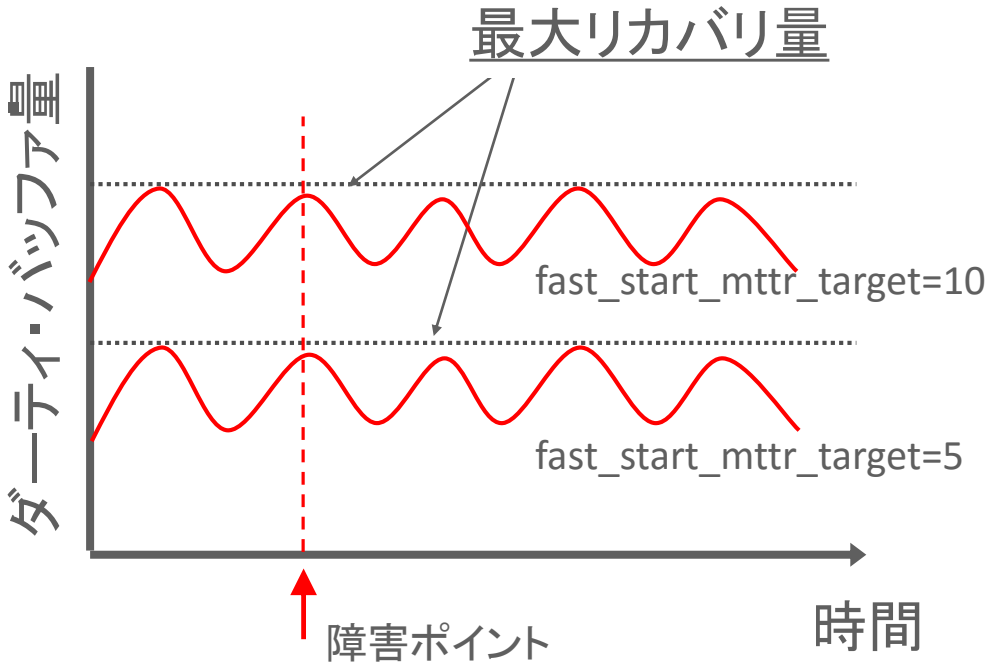
- 障害が発生したインスタンスのキャッシュ上にしか存在しなかったデータ・ブロックは、生存する他インスタンスが自動的にリカバリを実施
  - 正常インスタンスが障害インスタンスのREDOログを読み込み自動リカバリ
    - 障害インスタンスの再起動を待つ必要はない



インスタンスリカバリに要する時間

FAST_START_MTTR_TARGET パラメータ

- リカバリ時間はインスタンス障害発生時のダーティ・バッファの量に依存する
  - バッファ・キャッシュ上のデータとデータファイル上のデータの差
- ダーティ・バッファは DBWR によって、定期的に書き出されている（ファストスタート・チェックポイント）



リカバリ目標時間をパラメータ「FAST_START_MTTR_TARGET」に設定することで上記動作を自動制御可能

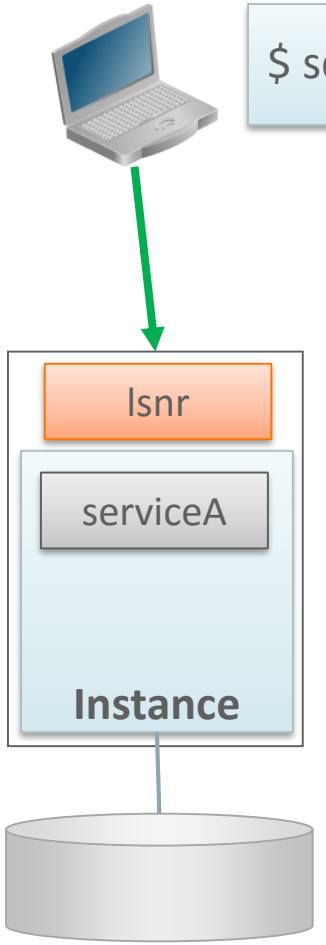
クラスタ・データベースへの接続

求められること

- 障害ノードをクラスタから除外する動作をしていても、アプリケーションから正常なサーバーに接続を切り替えなければシステムとして動作しない

- RACに対する接続に求められることと提供するRACの機能
  1. RAC側のノード数を意識せずに接続が可能なこと → SERVICE_NAME と SCAN
  2. 正常ノードとの間で接続が確立されること → 接続時フェイルオーバー
  3. 障害ノードとの間の無効な接続が破棄されること → FANイベントとFCF

Oracle Database への接続




```
$ sqlplus username/password@tns_alias
```

```
tns_alias=
  (DESCRIPTION =
    (ADDRESS =(PROTOCOL=TCP)(HOST=hostname)(PORT=port))
    (CONNECT_DATA = (SERVICE_NAME = serviceA ))
  )
```

Oracle クライアントは、ホスト名、ポート番号、サービス名を
指定してリスナーに接続リクエストを送る

データベース・サービスの構成

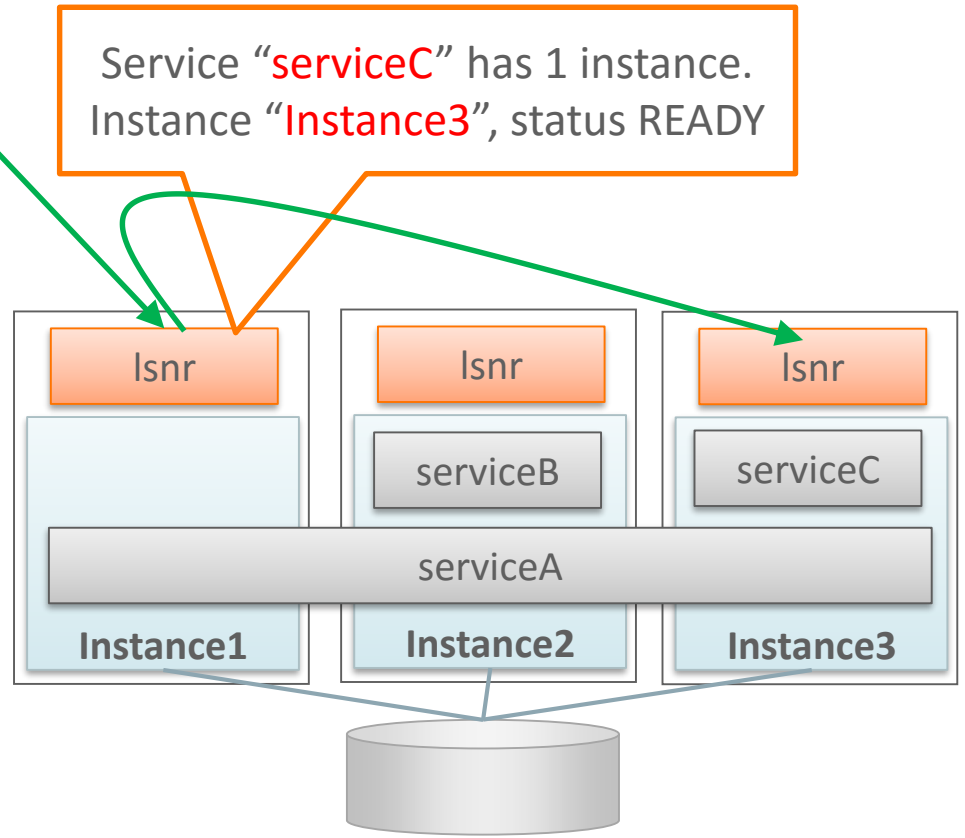
「サービス」はアプリやワークロードの
単位に相当する論理的なグループ

```
(DESCRIPTION =
  (ADDRESS_LIST=
    (ADDRESS = (PROTOCOL = TCP)(HOST = VIP1)(PORT = port))
    (ADDRESS = (PROTOCOL = TCP)(HOST = VIP2)(PORT = port))
    (ADDRESS = (PROTOCOL = TCP)(HOST = VIP3)(PORT = port))
    (CONNECT_DATA = (SERVICE_NAME = serviceC ))))
```

クライアントは接続先は「サービス」に対して接続

RACのノード数、起動しているか、停止しているかも
意識不要

リスナーは、指定されたサービスが
どのインスタンスで利用可能かを把握



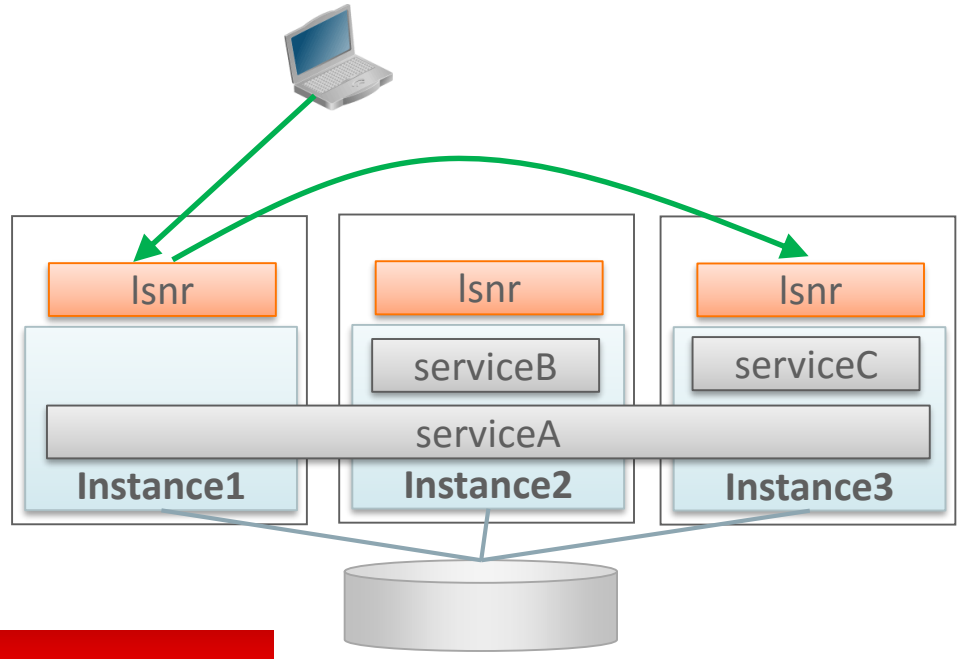
※図は便宜上SCAN_LSNR 1つだが
複数起動する

Oracle リスナーが担う2つの役割

1. 接続リクエストのリダイレクト
2. Oracle インスタンスとセッション確立

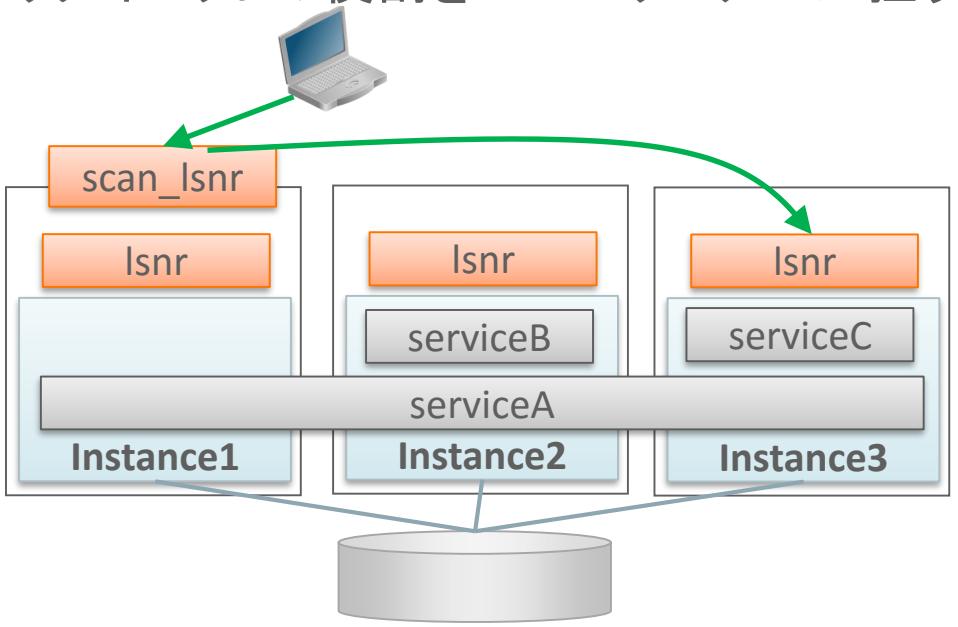
Oracle RAC 11g Release 1 まで

すべてのリスナーが 2つの役割を担う



Oracle RAC 11g Release 2 から

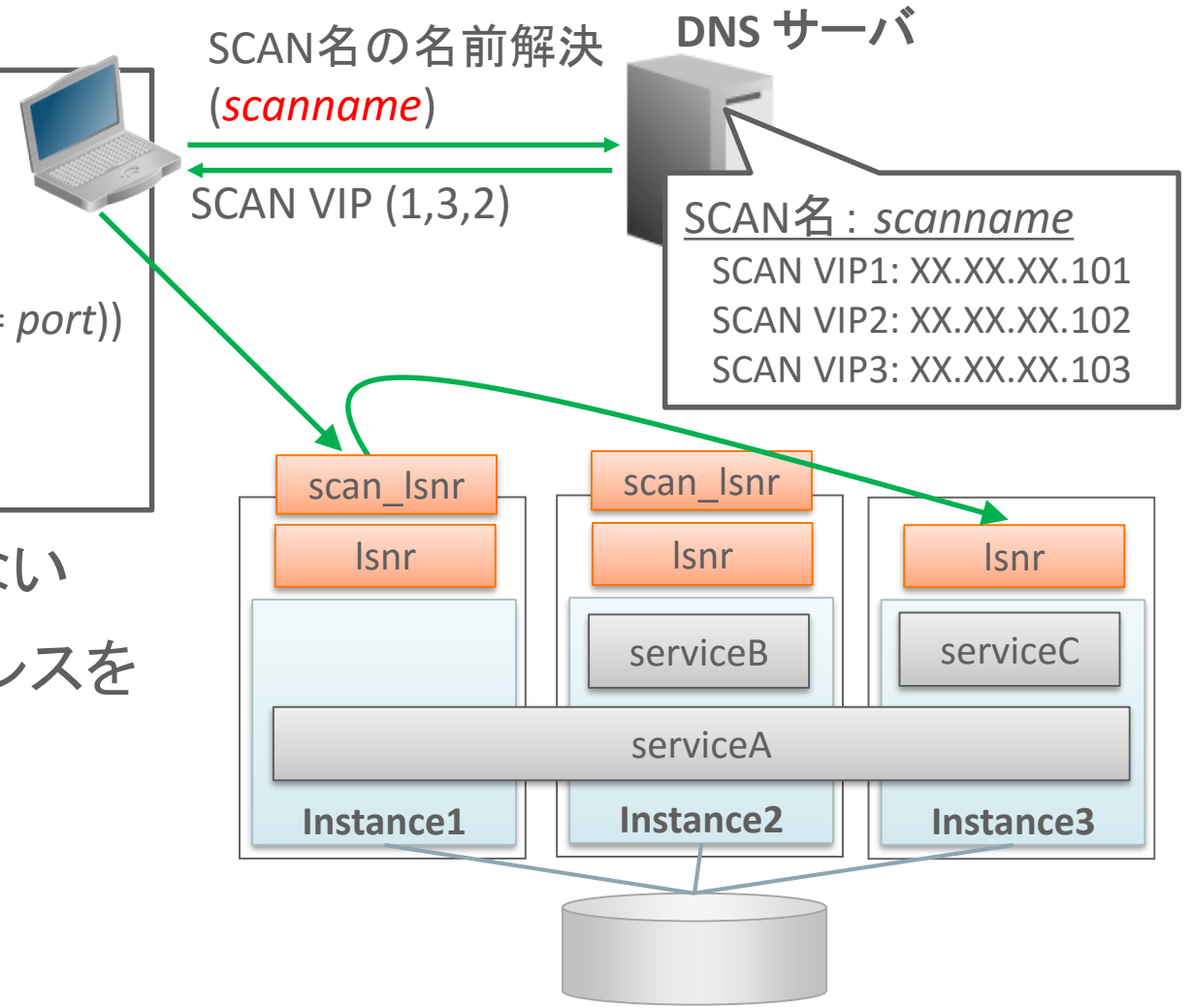
2つの役割を異なるリスナーが担う
リダイレクトの役割をSCANリスナーが担う



※図は便宜上SCAN_LSNR 1つだが
複数起動する

SCAN(Single Client Access Name)を用いた接続

```
(DESCRIPTION =
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=TCP)(HOST= scanname)(PORT = port))
    (CONNECT_DATA =  (SERVICE_NAME = serviceC ))))
```

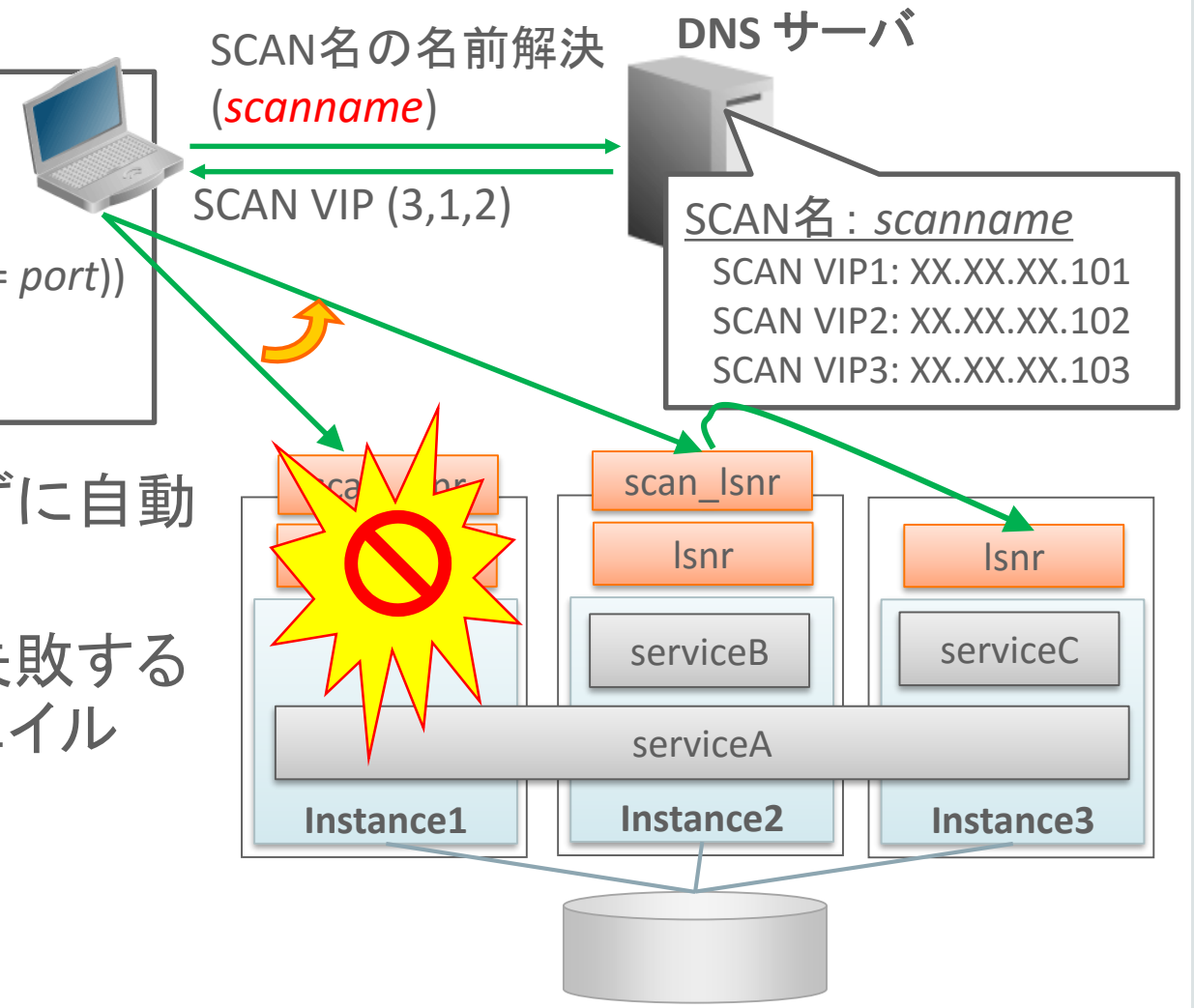


接続リクエストを送る先を意識する必要がない

ノードが増えた場合にADDRESSリストにアドレスを追加する必要がない

接続リクエストが正常なノードに割り振られる

```
(DESCRIPTION =
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=TCP)(HOST= scanname)(PORT = port))
    (CONNECT_DATA =  (SERVICE_NAME = serviceC ))))
```

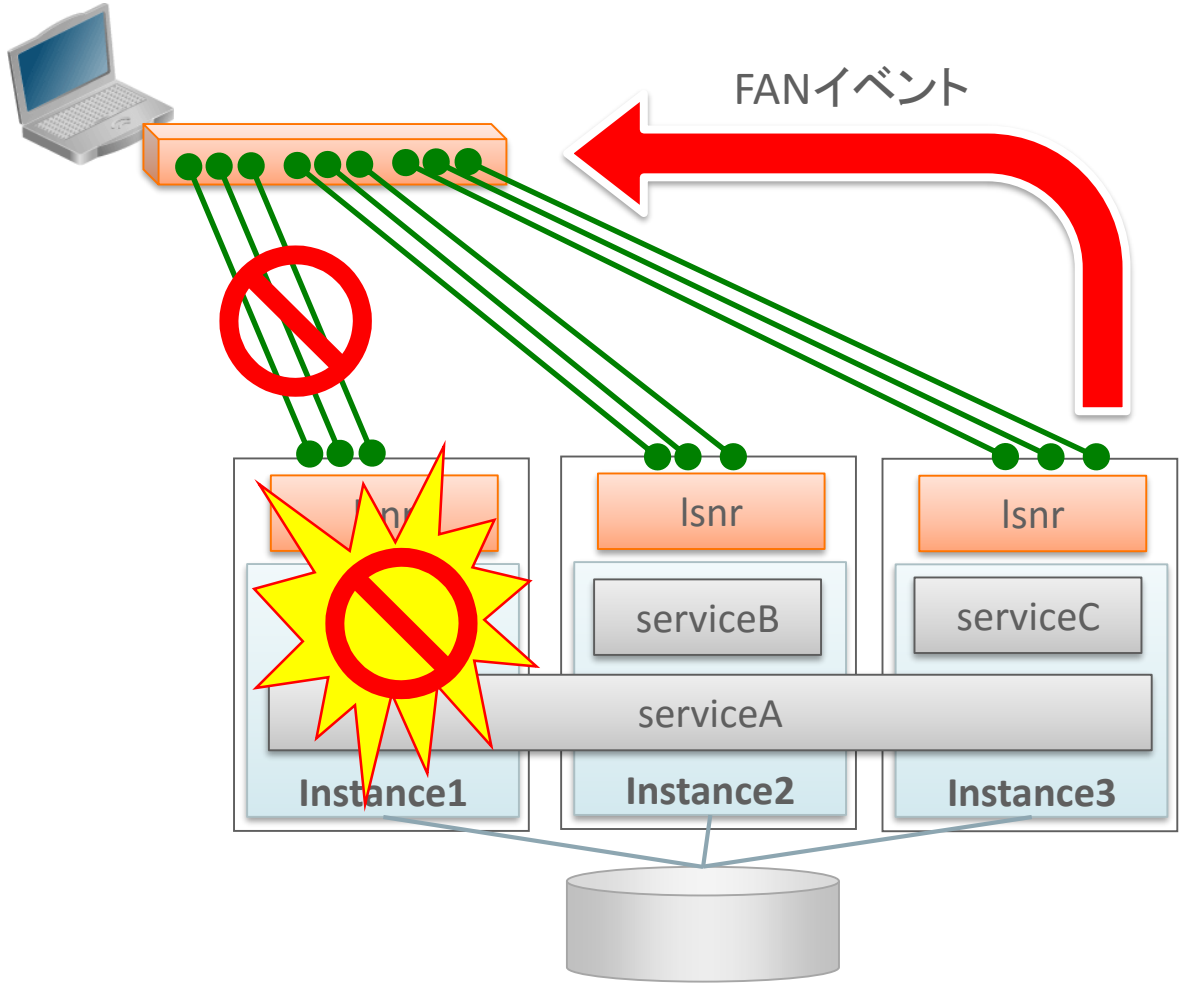


- 接続確立に失敗した場合、エラーを返さずに自動でアドレスリスト内の別の接続先へ試行
- 最初のリスナー（リスナー1）への接続に失敗すると、別のリスナー（例えばリスナー2）にフェイルオーバーする
- 「接続時フェイルオーバー」と呼ぶ

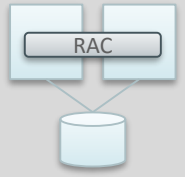
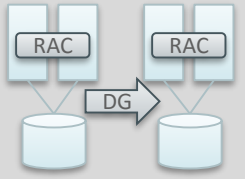
不要な接続を破棄する

Fast Connection Failover（FCF）

- コネクションプールを利用している場合、障害ノードとの間で確立している無効な接続を能動的にすべて破棄する機能
  - SQL結果待機中のコネクションにエラーを返す
  - ノード障害発生後、コネクションプール内に残っている、障害ノードにつながるコネクションを能動的に破棄する機能
    - コネクションプールから無効なコネクションを取ってしまうのを防ぐ
    - 接続時フェイルオーバーによって新規接続は障害ノードに行かない

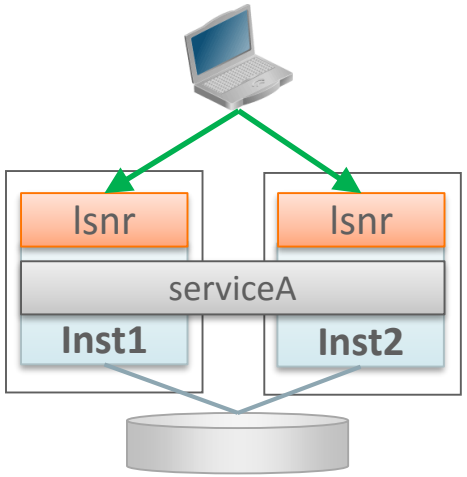


データベース構成と計画停止時間の目安

適用対象	パッチの種類	RAC Rolling 可否	Single	HA	RAC	RAC + DG
						
データベース	BP/PSU/CPU	Yes	DB停止後適用 (数分～数十分)	フェイルオーバーで 交互適用(数分 x 2回)	RACローリング適用(ゼロ)	
	PSR	No	DB停止後適用(数十分～数時間)			スイッチオーバーで 交互適用(5分未満 x 2回)
Grid Infrastructure (OCW/ASM)	BP/PSU/CPU	Yes	DB停止後適用 (数分～数十分)	フェイルオーバーで 交互適用(数分 x 2回)	RACローリング適用(ゼロ)	
	PSR	Yes	DB停止後適用 (数十分	フェイルオーバーで 交互適用(数分 x 2回)	RACローリング適用(ゼロ)	
OS	-	Yes	DB停止後適用 (数分～数時間)	フェイルオーバーで 交互適用(数分 x 2回)	RACローリング適用(ゼロ)	

段階的な接続切り替えがポイント

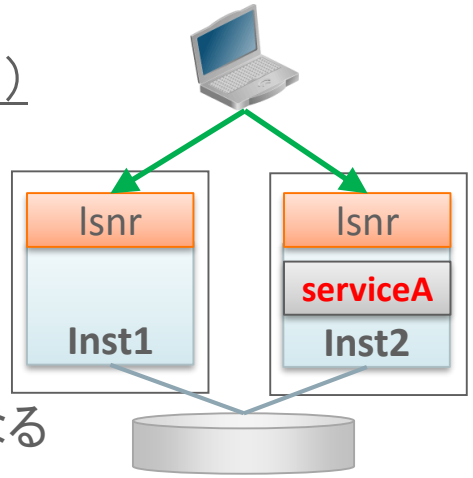
1) 通常動作



2) サービスの移動（正常移動）

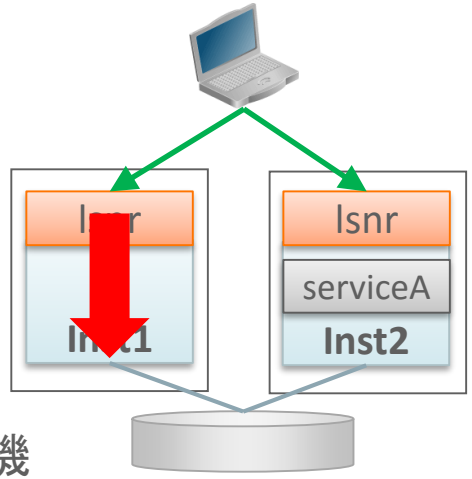
- 新規物理接続はInst2へ
- サービス強制移動をすると、既存セッションが強制切断

→ トランザクションロールバック
→ 再接続して再実行が必要になる



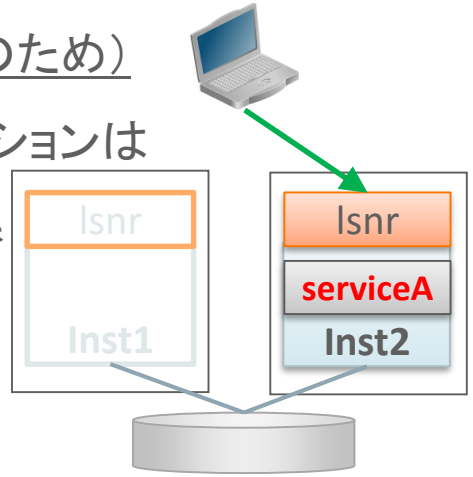
3) DBインスタンス正常停止

- OSやクラスタを停止すると、misscount やクラスタ再構成が発生するので、OS停止が必要な場合でも先にDBを停止
- 正常停止すれば、インスタンスリカバリは発生しない
- RAC Reconfiguration の間は待機



4) 縮退運転（サービス移動済みのため）

- トランザクションが終了したセッションはInst2へ再接続
- 次のトランザクションからInst2で実行される



RACリソース再構成

- 前ページの手順だとアプリケーションの停止はRACリソース再構成の間
- RACリソース再構成は次に依存
  - GCSリソース（バッファキャッシュのサイズ）
  - GESリソース（獲得されているエンキューの数など）
- 大きなバッファキャッシュを持つシステムに対しては、MOS Document ID 1619155.1のチューニング（パラメータにチューニングとパッチ適用）の効果は期待できる
  - お客様環境を模したあるテストではRACリソース再構成時間が約20%短縮された

まとめ

- RACを利用することで高可用性と拡張性が得られる
- RACもシングルインスタンスも基本的な考え方は同じ

参考資料

- Oracle DBA & Developer Days 2011 “実はシンプル！RACチューニングの考え方”
  - http://www.oracle.com/technetwork/jp/ondemand/b-10-ractuning-1448392-ja.pdf
- Oracle DBA & Developer Days 2011 “実践!!高可用性システム構築~RAC基本編~ “
  - http://www.oracle.com/technetwork/jp/ondemand/db-technique/d-4-rac11gr2-1448379-ja.pdf
- Oracle DBA & Developer Days 2012 “Oracle Real Application Clusters/ Oracle Clusterware の高可用性機能”
  - http://otndnld.oracle.co.jp/ondemand/ddd/PDF/MA-2_print_c.pdf
- Oracle DBA & Developer Days 2011 “解説！Oracle RACへの接続ロードバランスと接続フェイルオーバー”
  - http://www.oracle.com/technetwork/jp/ondemand/db-technique/b-3-rac-1484714-ja.pdf
- Core Tech Seminar Database 12c Release 2 RAC Stack
  - http://otndnld.oracle.co.jp/ondemand/od12c-oct2016/04_DB12201_coretech_RACStack_forOTN_v1.3.pdf

- 以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

～ みなさまの投稿をお待ちしております ～

Twitter

#OracleTechNight

こんな時、かけこむ会社が増えています。

Oracle Digitalは、オラクル製品の導入をご検討いただく際の総合窓口。
電話とインターネットによるダイレクトなコニュニケーションで、どんなお問い合わせにもすばやく対応します。
もちろん、無償。どんなことでも、ご相談ください。

Oracle Digital

お問い合わせは電話またはWebフォーム

0120-155-096

受付時間:月~金9:00~12:00 / 13:00~18:00(祝日・年末年始休業日を除く)

http://www.oracle.com/jp/contact-us

ORACLE

Integrated Cloud

Applications & Platform Services


ORACLE®