ORACLE

グリッド型システムによる次世代ミドルウェアアーキテクチャのススメ

日本オラクル株式会社

2007年11月

本日の内容

- 今の課題とグリッド型アーキテクチャ
- データグリッド採用パターン
- 先行企業に学ぶグリッド型システムの動向
- データグリッドの特徴的機能
- それでも障害は起こる!

今日、ぜひ覚えていただきたい言葉

データグリッド

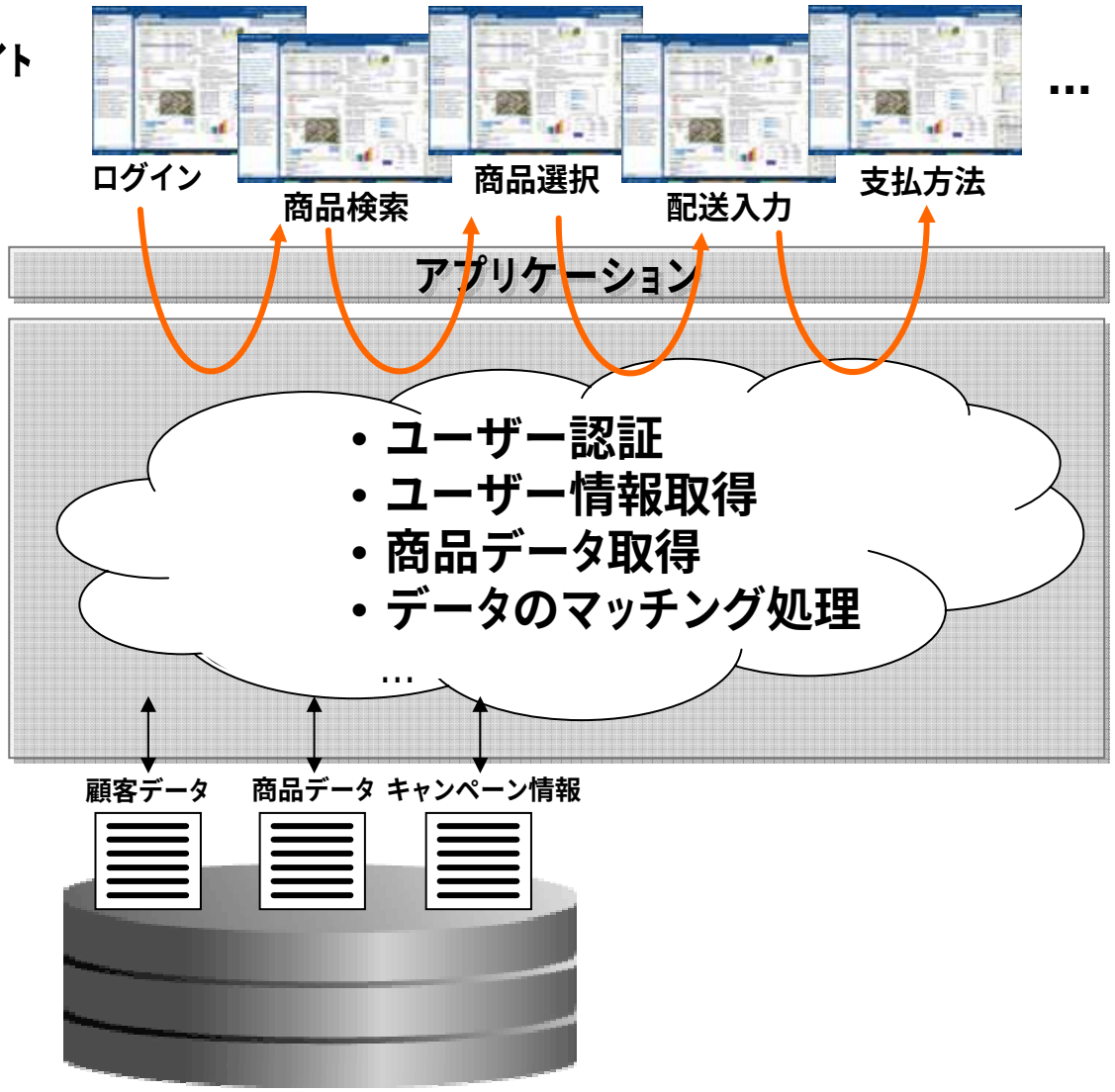
「グリッド」のおさらい

- 「グリッド」の名称の由来は「電力網」（**power grid**）で、電源プラグをコンセントに挿すだけで電力網から送電される電力を簡単に使えるように、グリッドの利用者はそれに対応した端末に自分の望む処理を投入するだけでネットワーク上の計算資源を必要なときに必要なだけ「電気のように」簡単に利用出来る様にすることを目指して命名された。

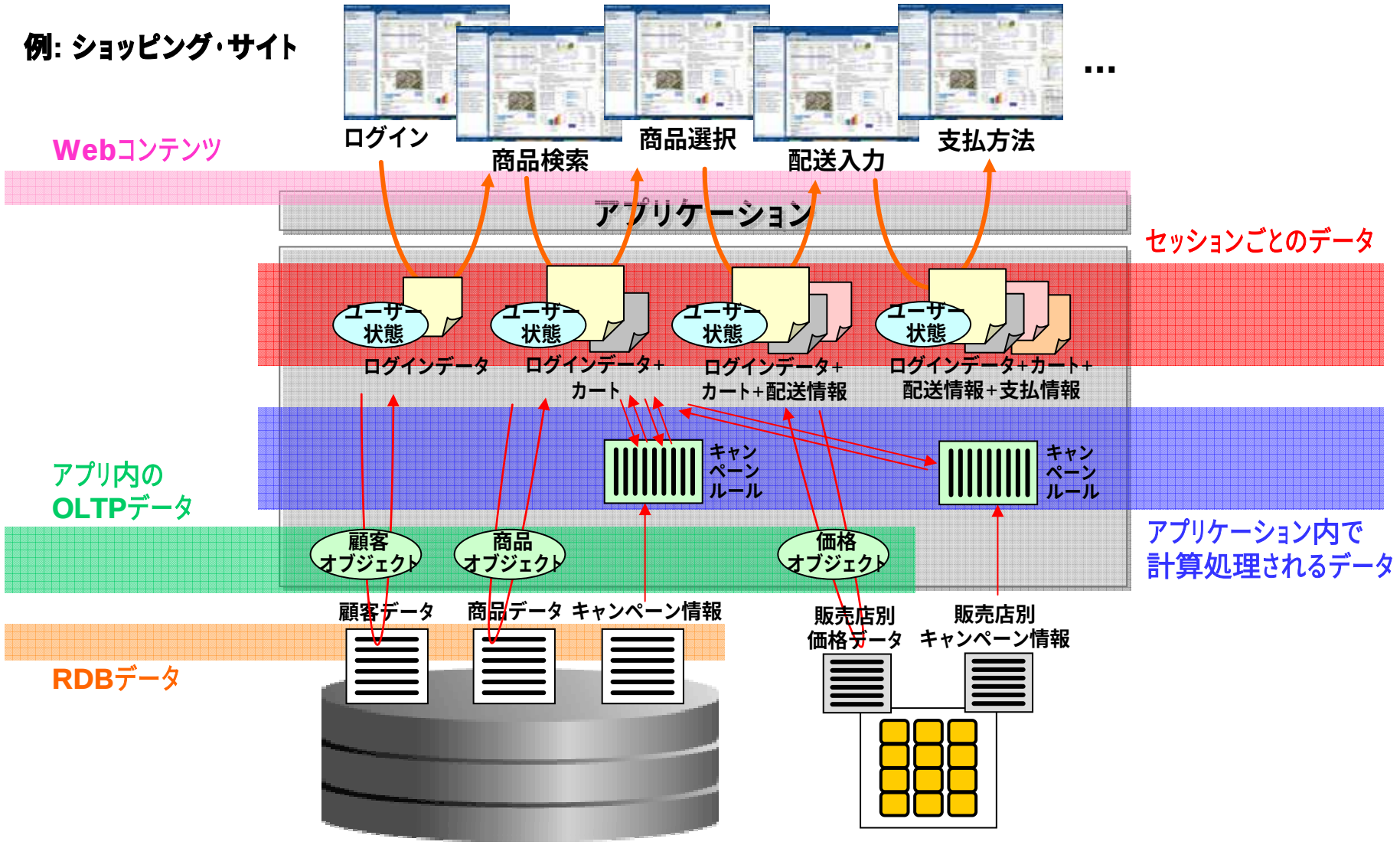
出典: Wikipedia

アプリケーションにおけるデータ処理

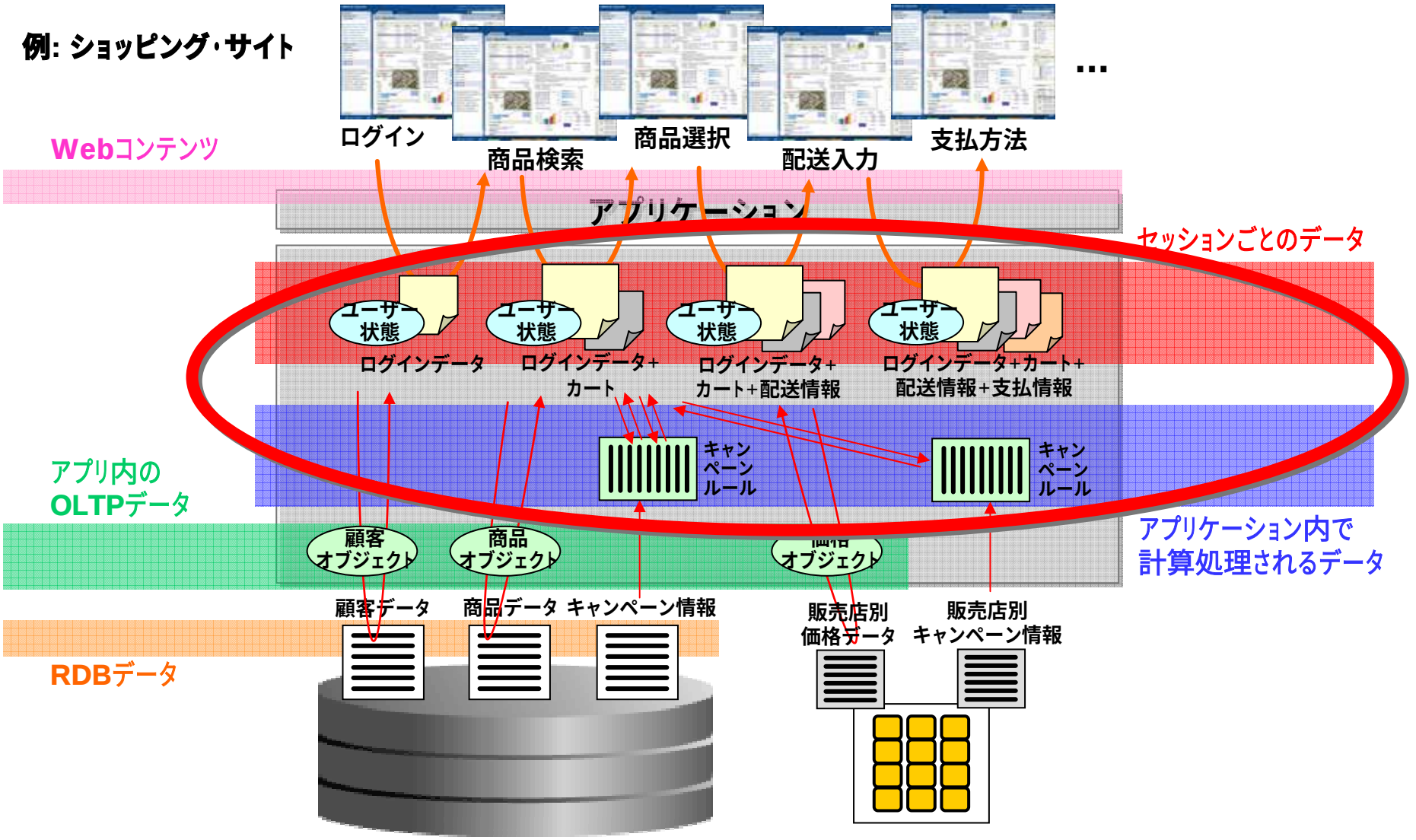
例: ショッピング・サイト



アプリケーションにおけるデータ処理



アプリケーションにおけるデータ処理

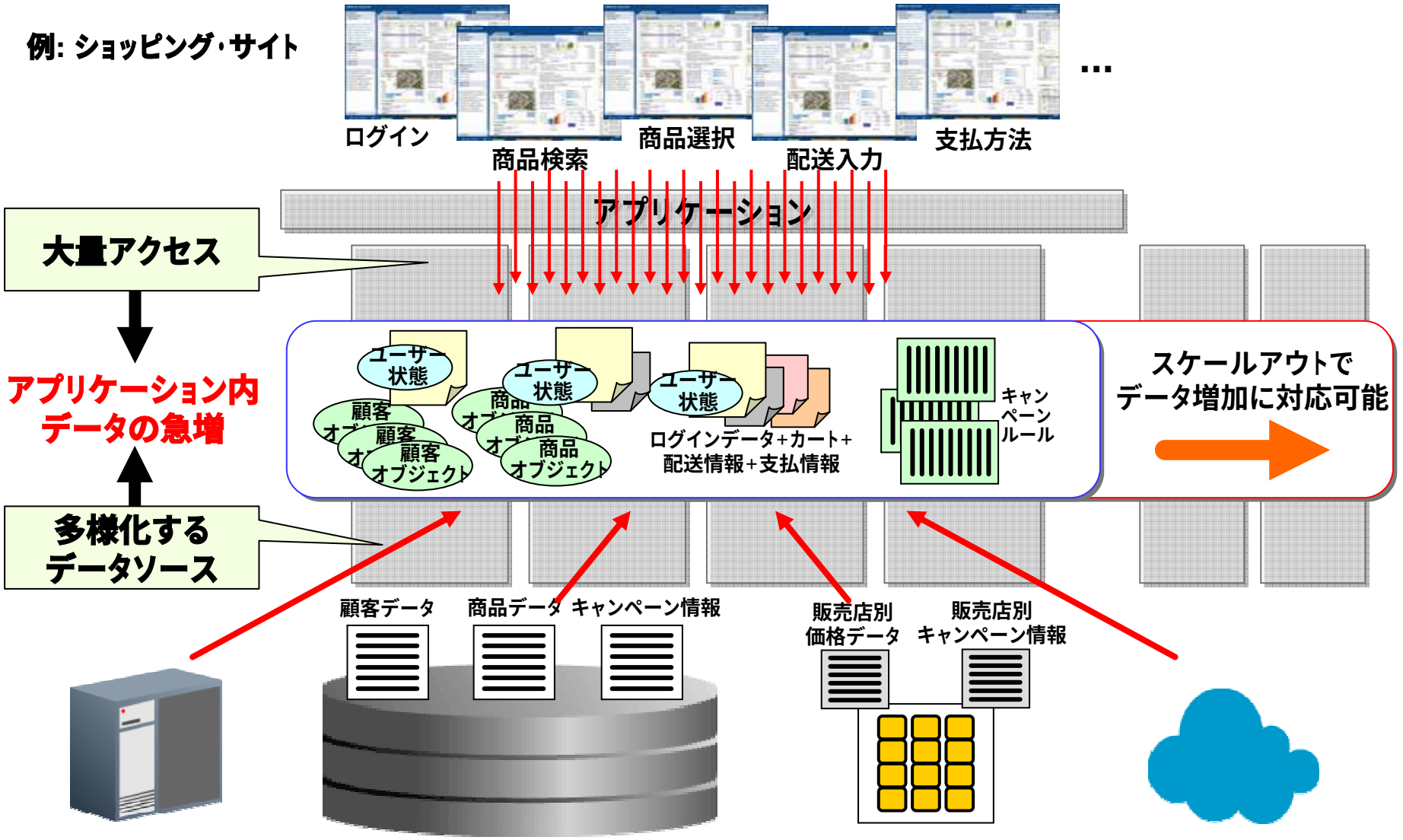


Web 2.0 の波がもたらした影響

- 顧客ごとのパーソナライズ機能
  - 顧客ごとに使いやすい画面配置にカスタマイズ
  - 顧客の利用レベルに見合うサービスの提示
  - 顧客の年齢層や嗜好に適した商品の提案
  - 利用頻度に応じたポイント制度
  - …
- 購入促進施策
  - セット購入、事前予約に対するプレゼント
  - 期間限定、数量限定キャンペーン
  - …
- 関連サイトとの連携
  - パートナー企業の商品の間接販売
  - パートナー企業とのジョイント・キャンペーン
  - 広告ビジネス
  - 口コミ情報との連携
  - …

＋ 信頼性へのニーズ

アプリケーションにおけるデータ増加への対応



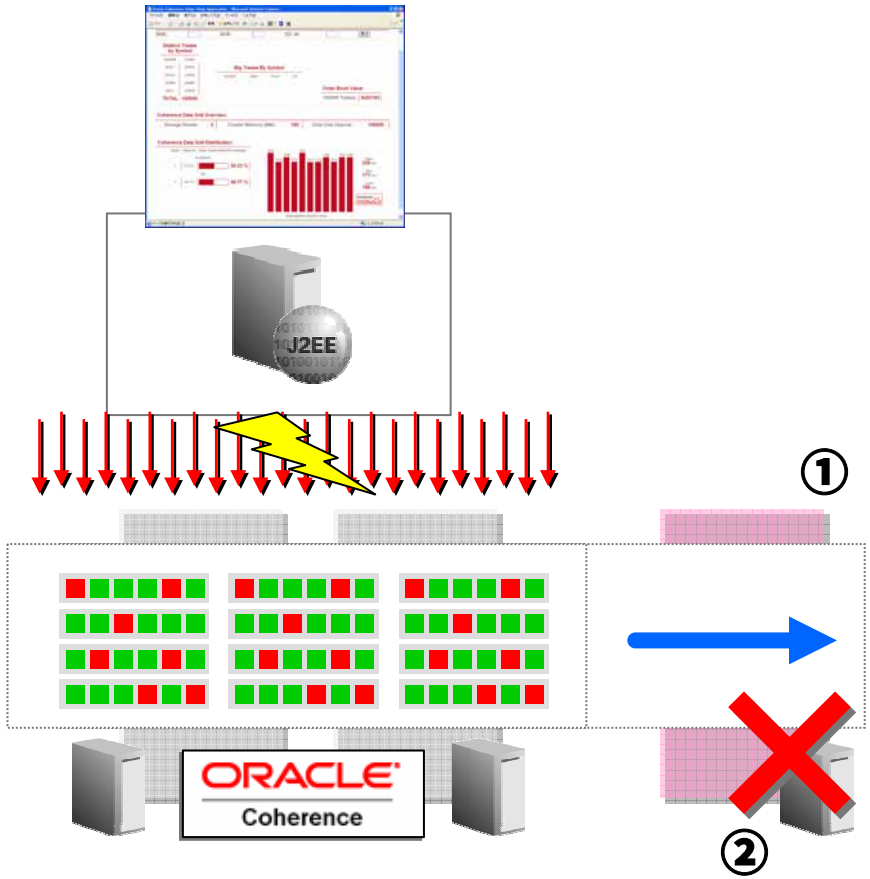
Oracle Coherence

Brand-new since 2007-09-11!

D E M O N S T R A T I O N

Oracle Coherence サーバーの動的な構成

キャッシュ・サーバーの動的な増減

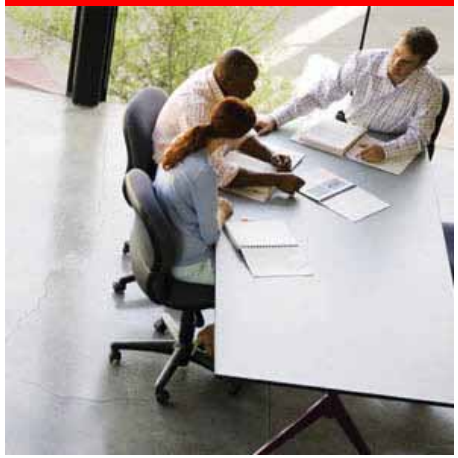


- シナリオ
  - 株取引アプリケーション
  - アプリケーション稼動中

① 取引量の増加に対応するため
メモリ処理可能な上限を
拡大したいが、どうするか?

② 処理途中に **Coherence**
サーバーがひとつダウンした
場合、アプリケーションの処理
にどう影響が出るか?

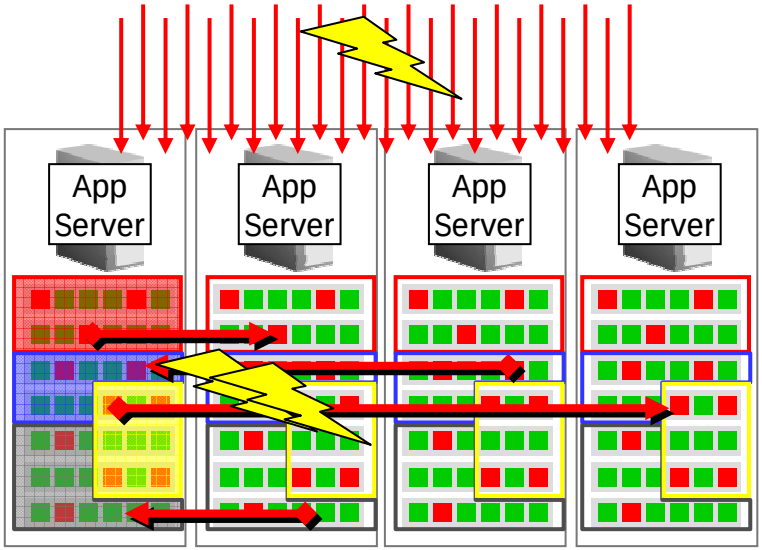
データグリッド採用パターン



従来のアプリケーション・ステート管理手法

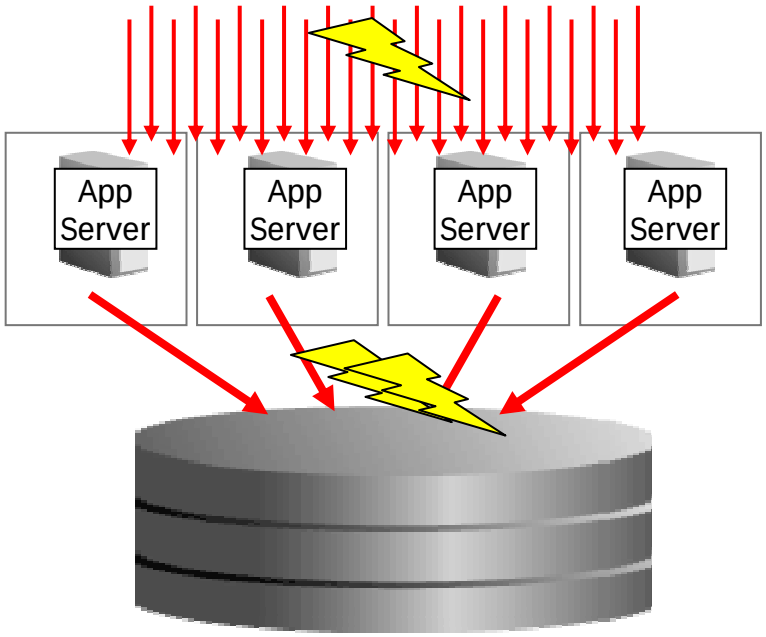
■ アプリケーション・サーバーのクラスタ/レプリケーションの利用

- ✕ HttpSession にオブジェクトをつみすぎると
  - クラスタ時に性能劣化
  - OutOfMemory の要因になりやすい
- △ 設定が煩雑
- △ 運用管理が複雑化

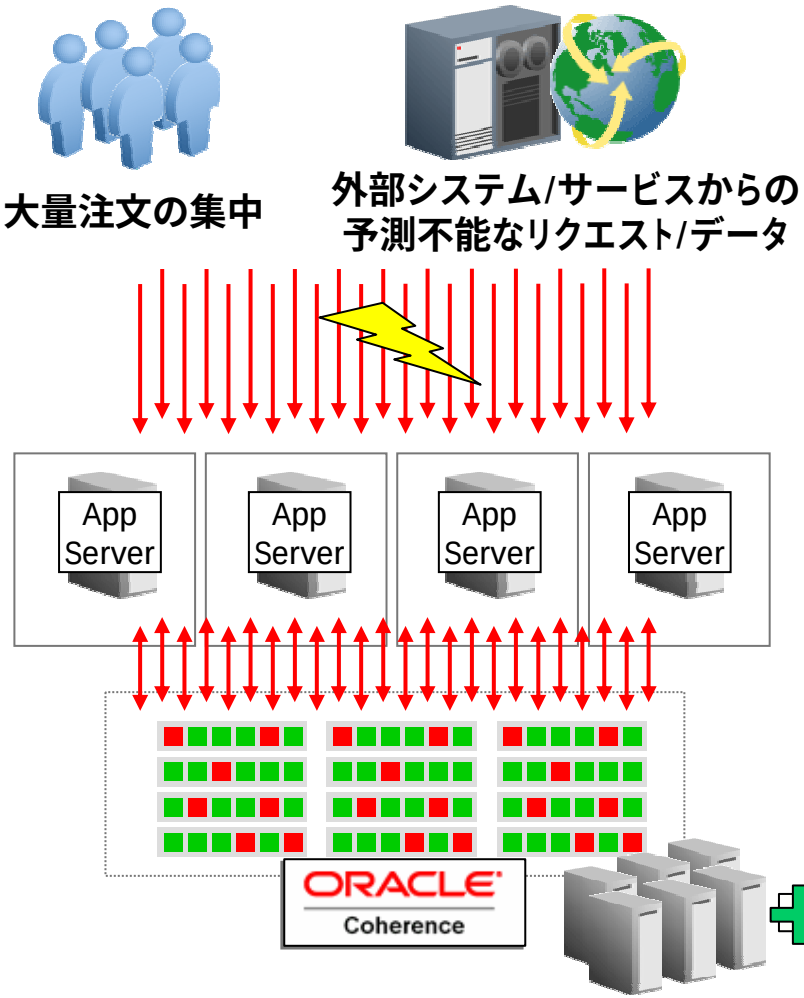


■ アプリケーション・ステートをDBに格納

- ○ 設定/通常運用はそれほど複雑ではない
- ✕ 都度のDBアクセスが発生しパフォーマンスに大きく影響
- ✕ DBへの負荷が高い



高速かつスケーラブルなアプリケーションのステート管理



- アプリケーション内での処理負荷が高いコンシューマ向けサービス
- ロング・トランザクションのシステム
- 高トランザクションSOAシステム
- **Web2.0型マッシュアップ・サーバー**

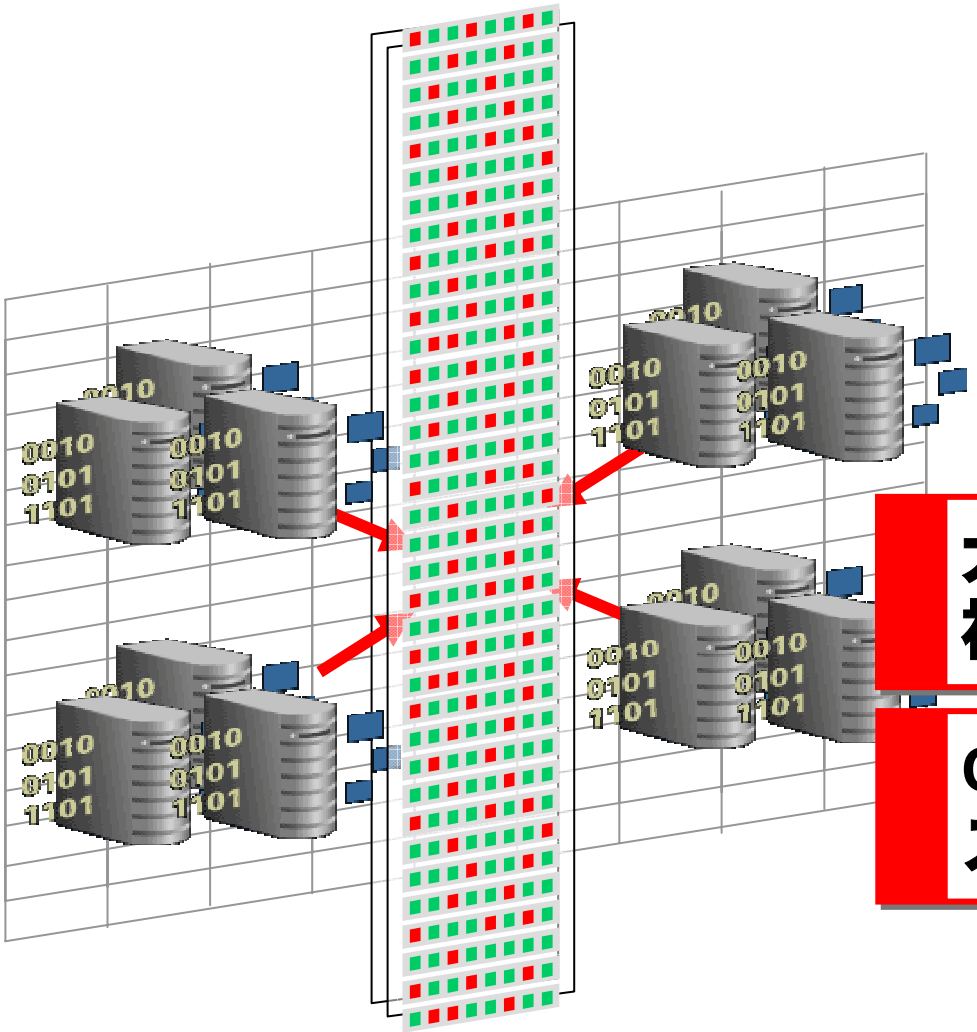
データ量に応じてデータグリッドサーバーをスケールアウト

Appサーバー障害でもステート管理を保証 (ビジネスを継続)

事例: ネットによるコンシューマ向けサービス展開

- **FedEx**
  - クリスマスシーズンに**980**万の配送、毎秒**1,000**のオンライン問合せ
  - 今後も増加していくことが予想される
  - オンデマンドで拡張可能なデータ処理の仕組みが必要
- **Betfair**：オンラインオークション/ゲーム・サイト
  - ユーザー・ステートを**J2EE**サーバーの**HTTP**セッションに格納
    → セッションの肥大化によるパフォーマンス劣化が発生
  - ピーク時に**1,000**を超えるトランザクションが発生
  - **Coherence**を導入してステート情報をアプリケーションから分離し、パフォーマンスとスケーラビリティを確保
  - **1,500％**のキャッシング効率改善
- **Delta Airlines**
  - オンライン予約システム
  - 増加するトランザクションにスケールアウトで対応したい
- **Macy's**（**macys.com**）
  - オンライン・ショッピング・サイト
  - **Spring Framework + Coherence** で構築しスケールアウトに対応

超高速計算処理: コンピューティング・グリッド



- 金融リスク計算
- デリバティブ分析
- シミュレーション計算
- バイオ、天文などのデータ解析

大規模データに対する計算を複数サーバーで並行処理

Coherence がリニアなスケーラビリティと性能を保証

事例: 複雑な計算処理の高速化

WACHOVIA

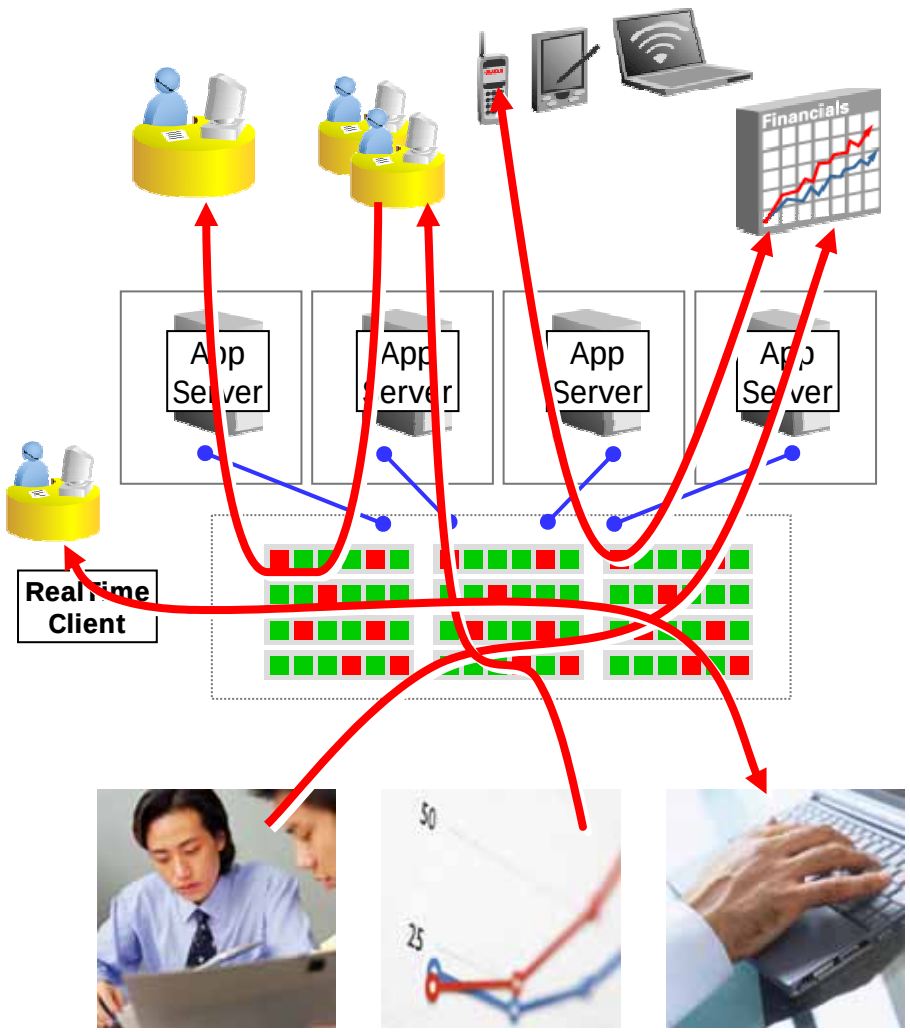
- **Wachovia Bank**
  - データ・マネジメントおよび並列処理のデータ基盤（データの仮想化）としてCoherence を採用
  - Credit Risk プロジェクトでリスク計算の仕組みを構築
  - 効果: データを仮想化することによって、データの位置を意識せずにアプリケーションが記述できるようになった

    パフォーマンスを100倍以上改善
    （50日かかっていたリスク計算処理が、同じHW環境で1時間で完了）

Hotwire.com

- **Hotwire.com：ネット予約サイト**
  - 予約料金をルールベースで動的に計算
    → ユーザーによる 1回のクエリに対して1,300～1,500 のトランザクションが発生
  - Coherenceの活用でレスポンス時間を1秒にまで短縮

イベントに対するリアルタイム・アクション



- オンライン・トレーディング
- 不正検知システム
- **NGN基盤**
- オンライン・ゲーム、チャット

オンメモリで複数システム、デバイスをリアルタイム連携

共有メモリを介したリアルタイムコミュニケーション

事例: リアルタイム処理

- **FXDirectDealer：オンライン外貨取引サービス**
  - **FOREX**（外国為替証拠金取引）のインターネット・トレーディングの仕組みを構築
  - クライアント・ツール「**FXDD Trader**」（**Pure Java**）から**Coherence**のキャッシュ・サーバーとコミュニケーションすることでリアルタイムのトレーディングに対応

- **INCERNO**
  - リモートで機材/デバイスの情報を収集・監視する**Web**ソリューション
  - 大量データのリアルタイム処理（**3,000tx/秒**）が必須
  - 効果：**5,000tx/秒**を実現（設計時の想定の**167%**）
    当初見積もりの**DB**の**HW**コスト（**30**万ドル）を**6**万ドルに削減

代表的な採用パターン

複数JVM間での高速データ共有
- アプリケーション・ステートの共有 (アプリケーションの高信頼性)
- 出荷予約管理/確認
- 電子マネー処理

データの高速突合処理
- 証券取引における需給のマッチング
- 最適料金計算
- 電話呼び出し/モバイル・メッセージのルーティング

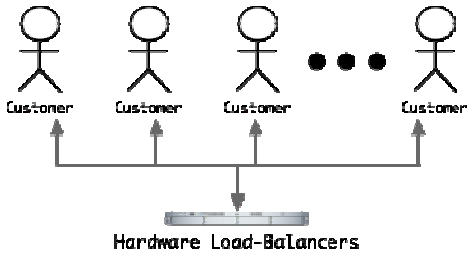
コンピューティング・グリッド
- シミュレーション計算
- リスク計算

リアルタイム・イベント処理
- FX 自動取引システム
- オンラインゲーム (ピア・ツー・ピア)
- 双方向TV投票システム
- 認証システム/不正検知

複数データソースの仮想化統合
- 企業の統合データサービス層
- パートナー企業の商品情報のキャッシュ

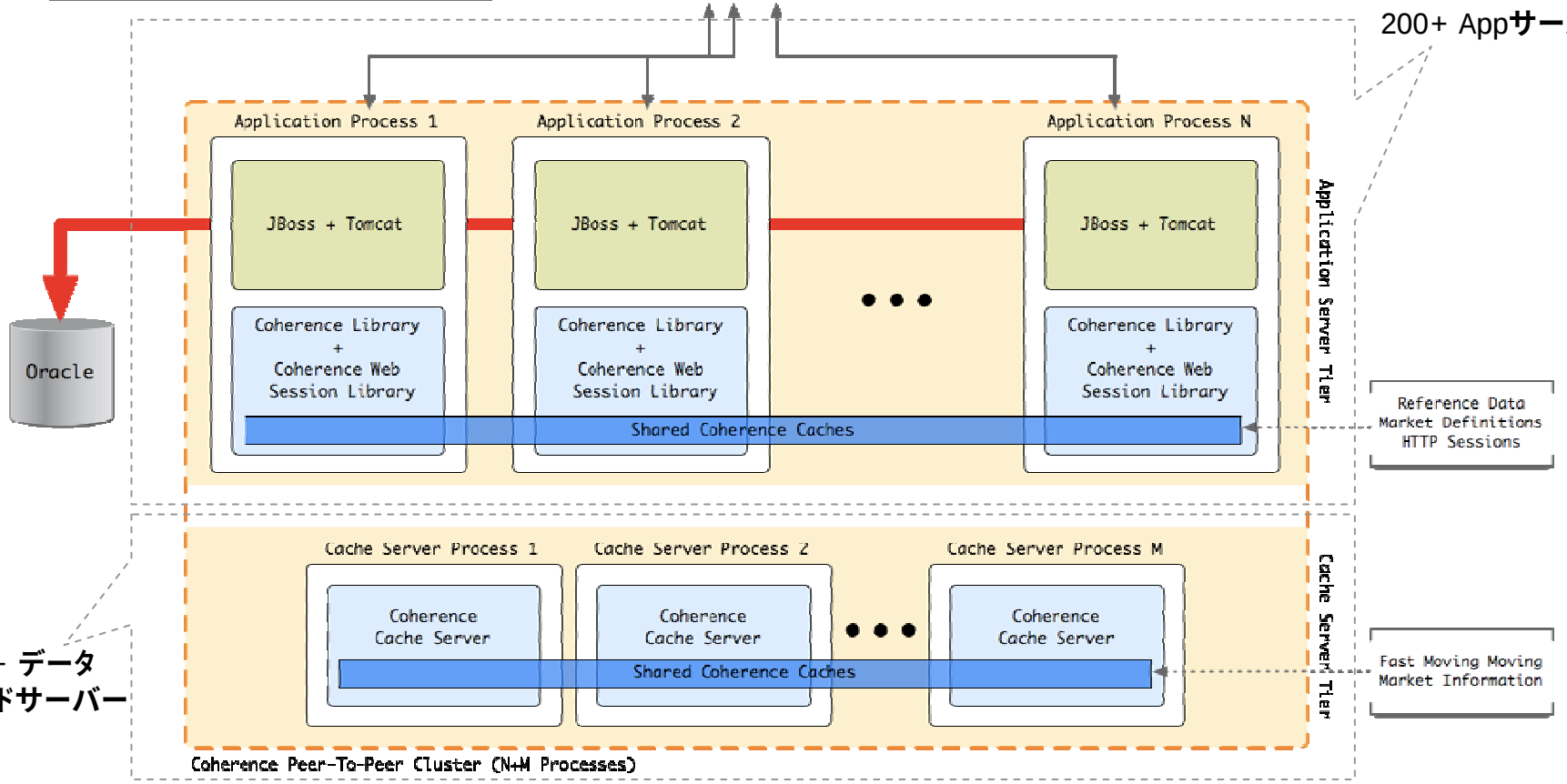
海外オンラインゲーム企業における採用例

- 数千のカテゴリの賭け市場を提供
- 24 x 7 の高可用性
- 15ヶ国語言語対応
- 掲示板、コミュニティ機能を併設
  (Coherenceを活用)



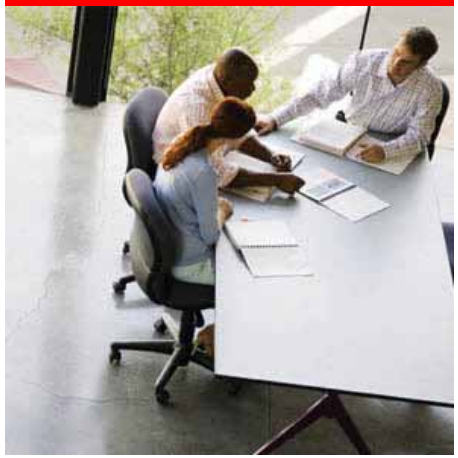
30,000+ **同時セッション**
(Coherenceで**セッション管理**)

200+ App**サーバー**



50+ **データ グリッドサーバー**

データグリッドの特徴的機能

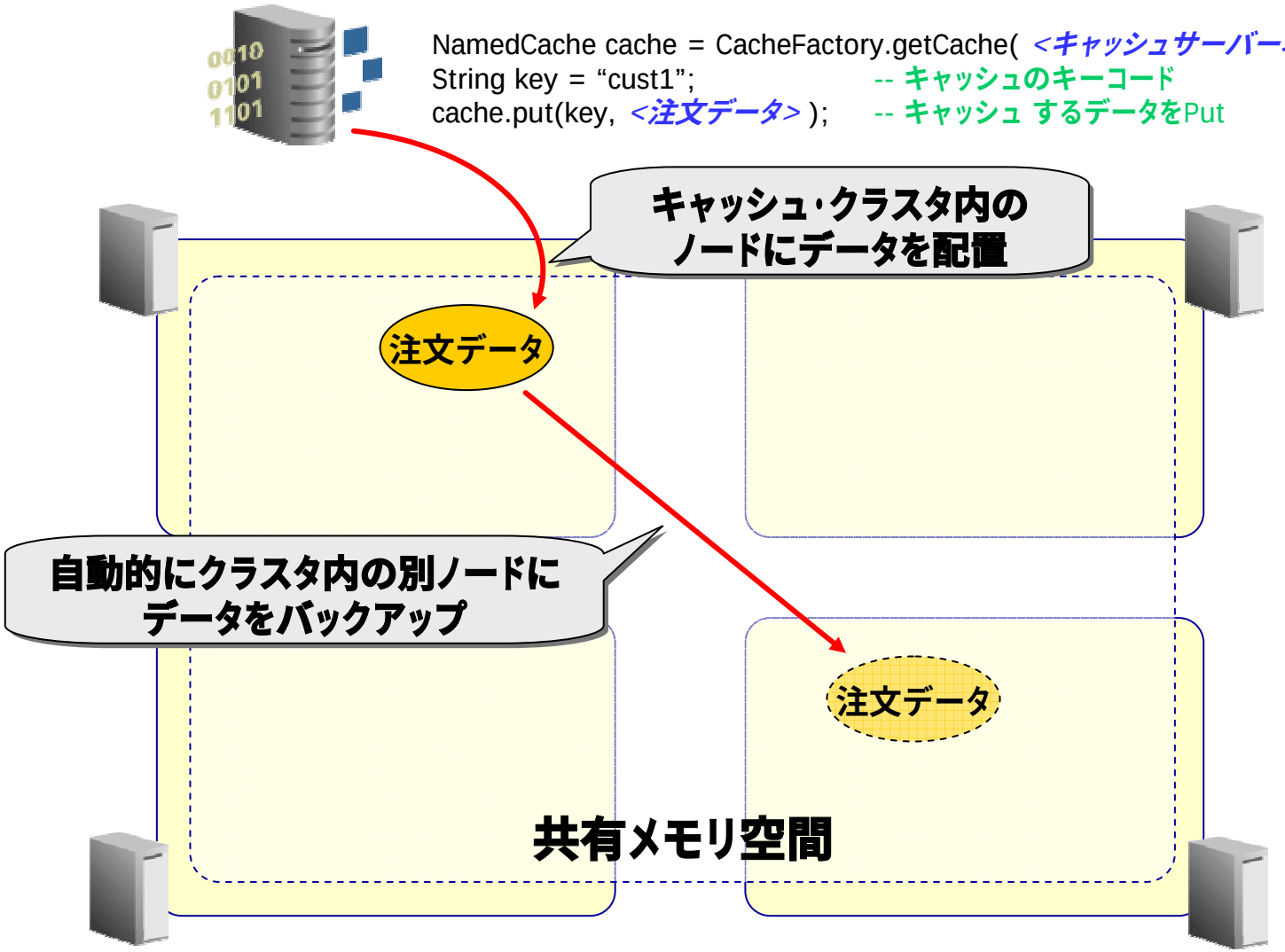


Javaアプリの高速化のために

パフォーマンスのボトルネック	データグリッドによる効果
• DBアクセス、ディスクIO	• **大量データでも**オンメモリ展開可能
• **大量データのオブジェクト化コスト**	• **オブジェクトとしてキャッシュ** **RDBやメッセージング・データからの変換コストを排除**
• GC	• **複数JVMで一つのメモリ空間を構成** **GCが細分化され、影響が最小化**

拡張可能・高信頼性・高可用性のワケ

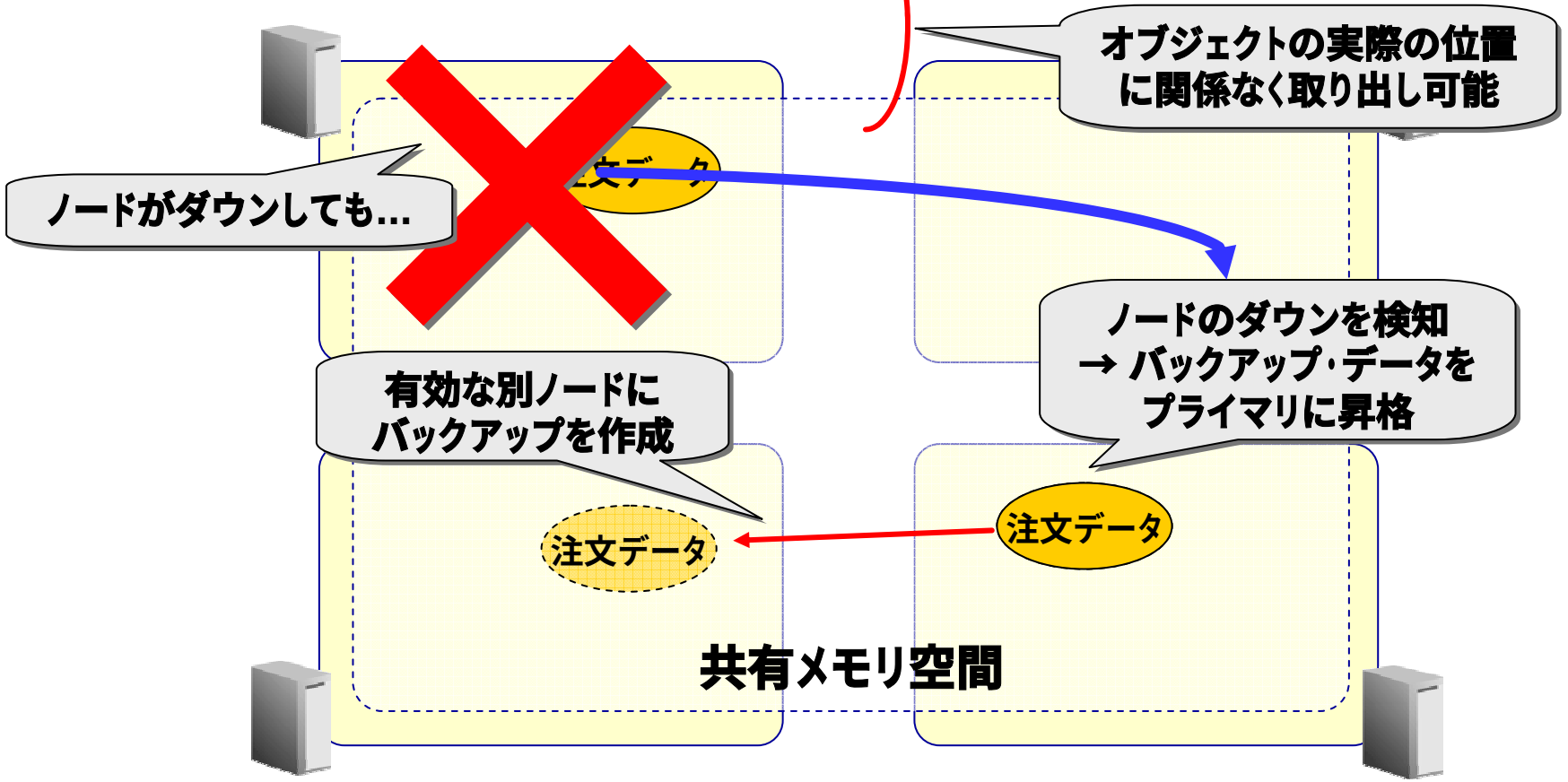
Partitioned Cache



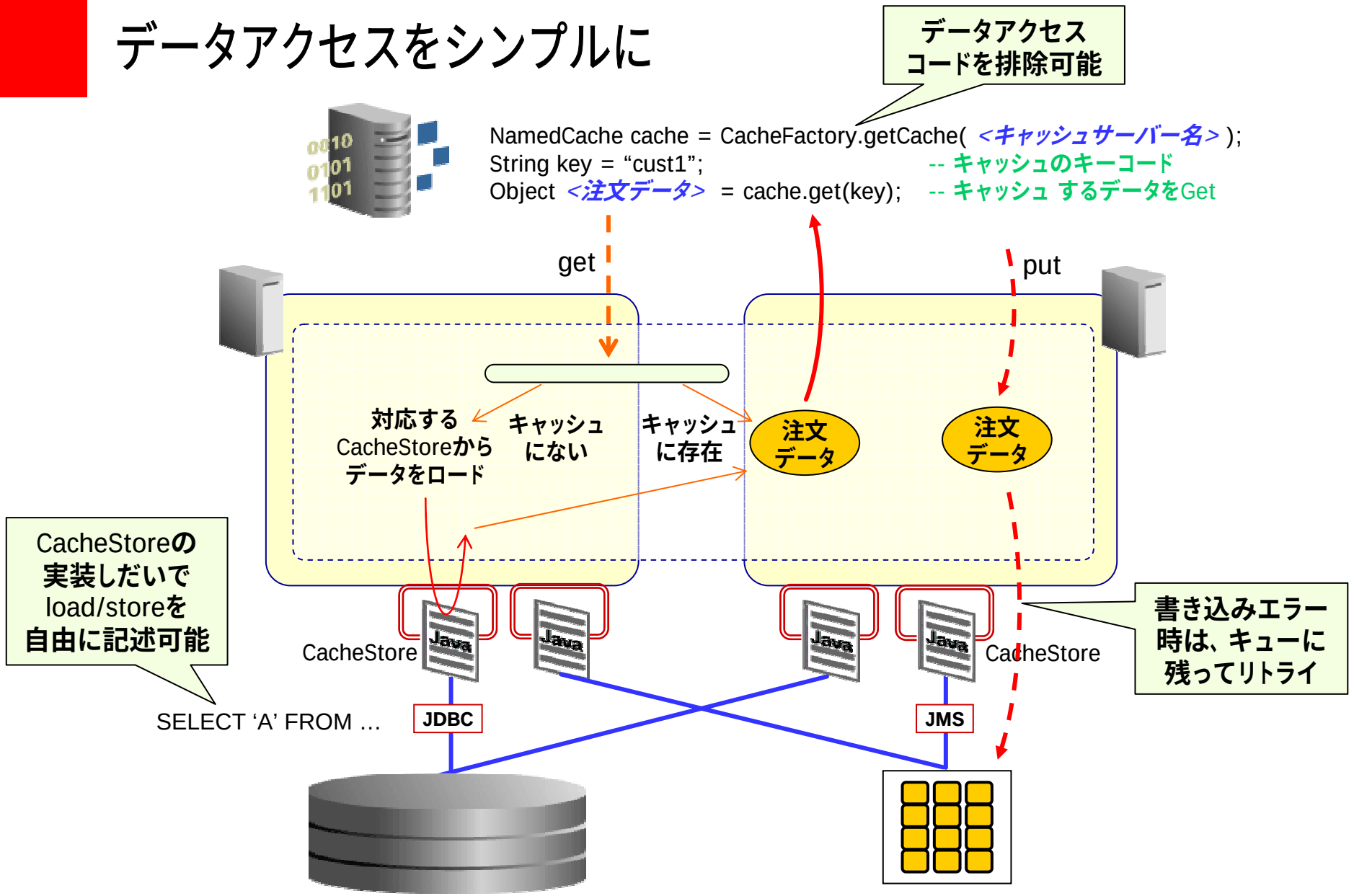
拡張可能・高信頼性・高可用性のワケ

Partitioned Cache

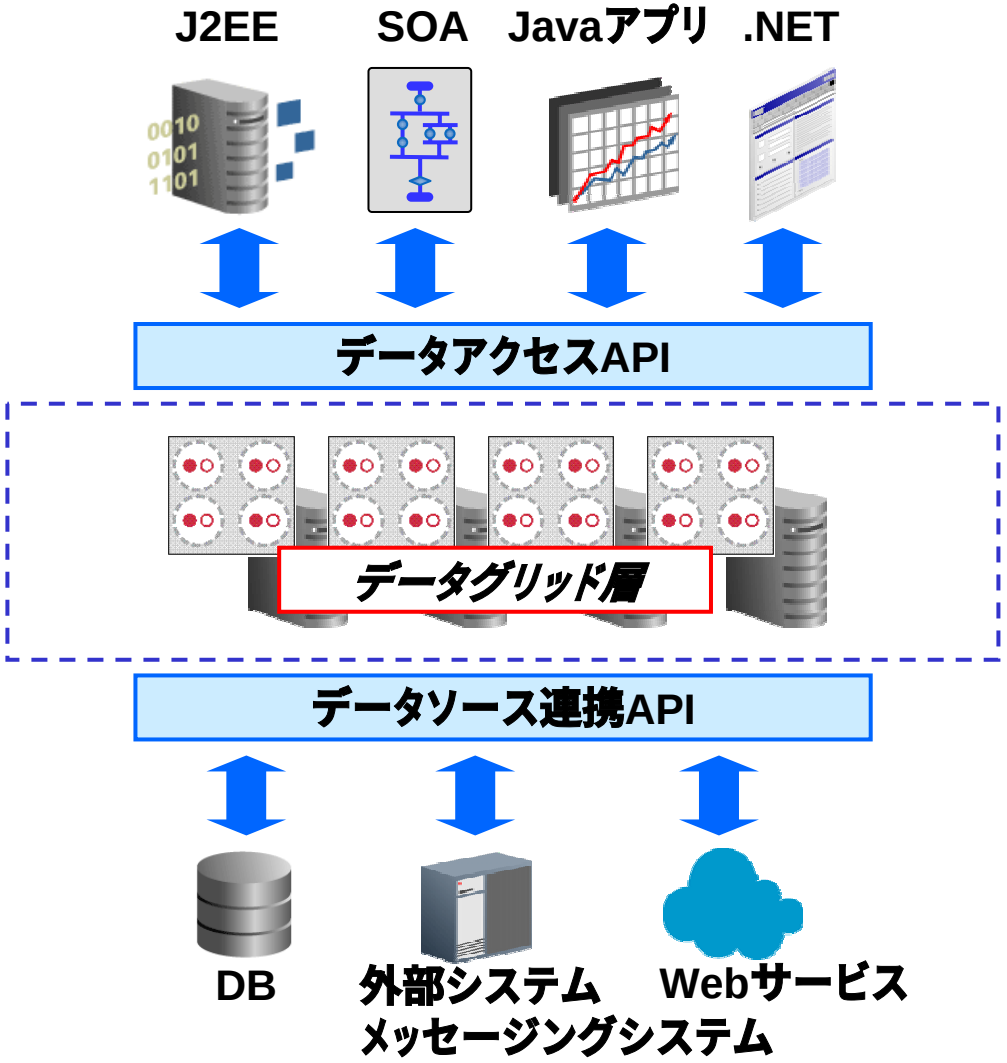
```
NamedCache cache = CacheFactory.getCache( <キャッシュサーバー名> );
String key = “cust1”;                          -- キャッシュのキーコード
Object <注文データ> = cache.get(key);          -- キャッシュ するデータをGet
```



データアクセスをシンプルに



データグリッドによるデータの仮想化

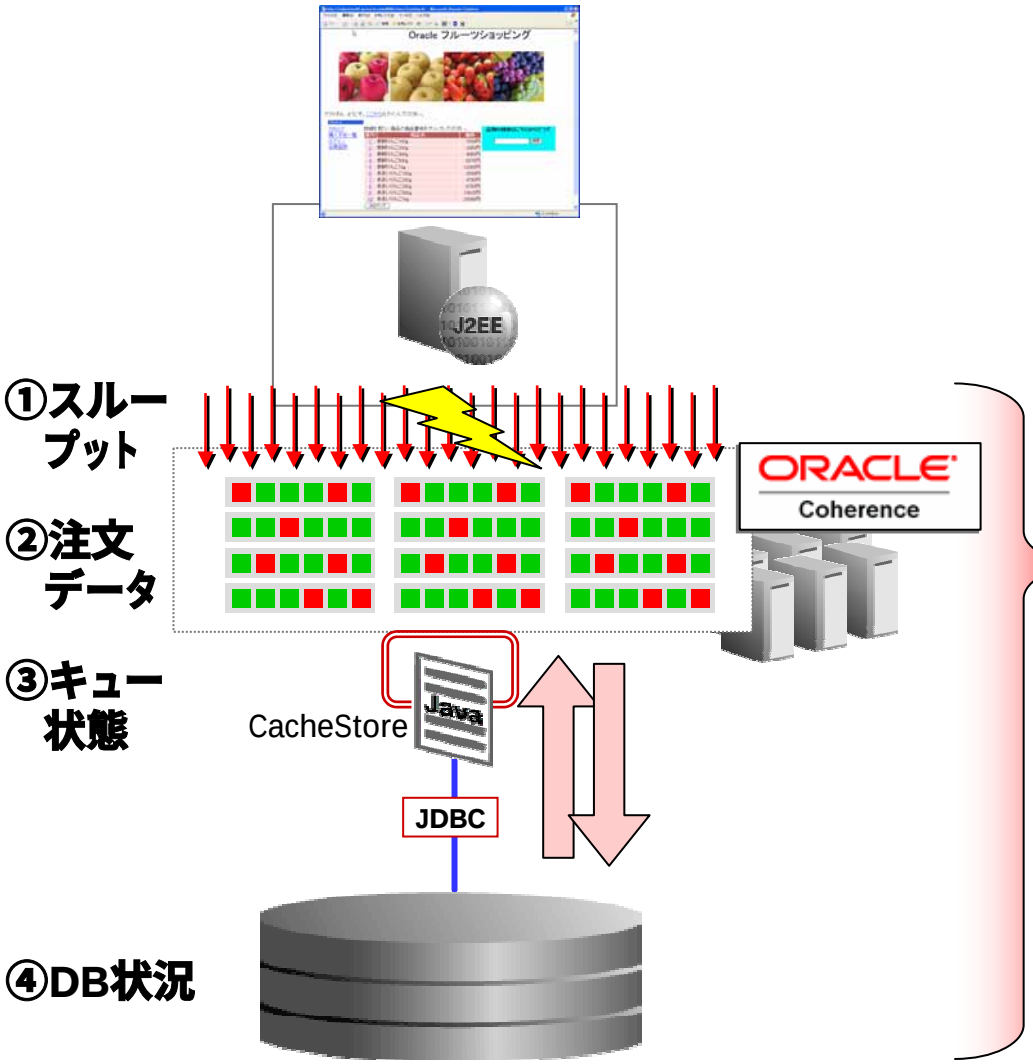


- **アプリケーション・コードをシンプル化**
  - put/get でデータを取得
  - 追加のアプリ開発も容易
- **データ処理の高速化**
  - データの急増にも動的なスケールアウトで対応可能
  - GCの細分化により、パフォーマンスへの影響を最小化
  - データ問合せをパラレル処理
- **データソースへの依存度を最小化**
  - データソースの障害による影響を最小化
  - データソースの入れ替えや修正も容易

D E M O N S T R A T I O N

Oracle Coherence 高可用性と信頼性

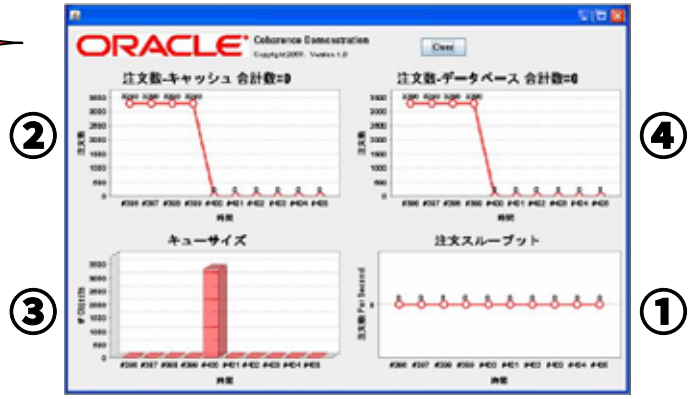
可用性を高める Coherence データグリッド



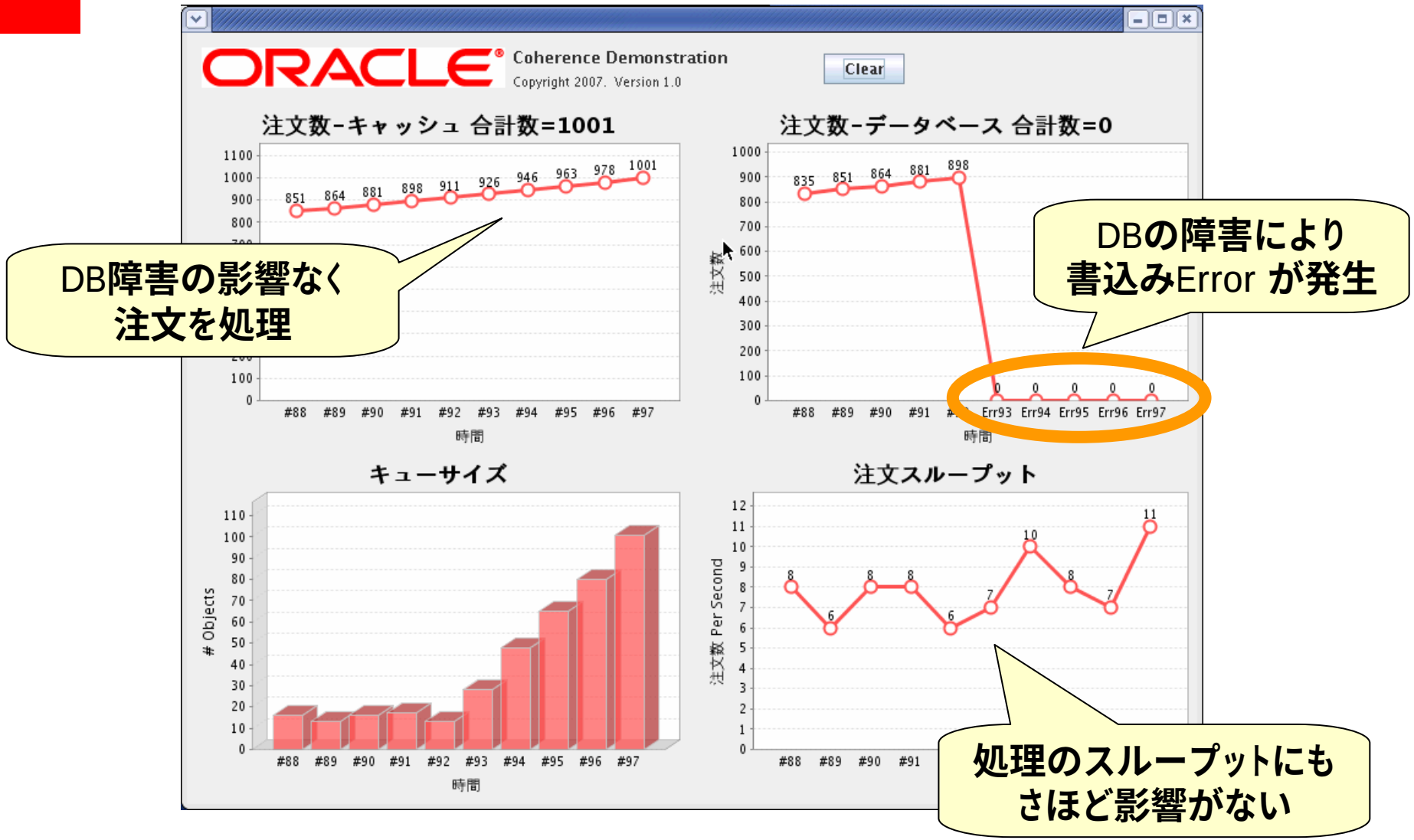
- シナリオ
  - ショッピング・サイト
    - 商品選択、個数決定、購入…
  - 絶えず発注が発生

  処理途中に**DBがダウン**したら…

トランザクション監視用
カスタム・アプリケーション



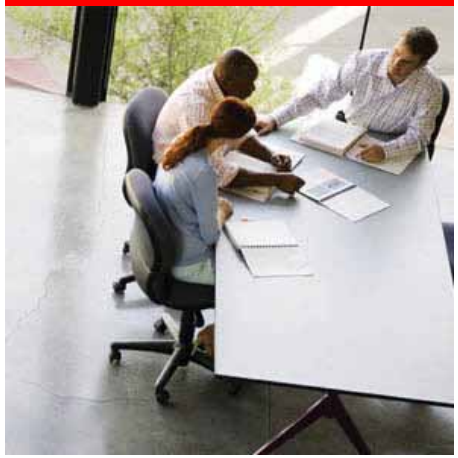
以下の進捗とスループットをモニタリング
- 発注トランザクション状況
- DBに対するトランザクション状況



事例: データグリッド基盤としての採用

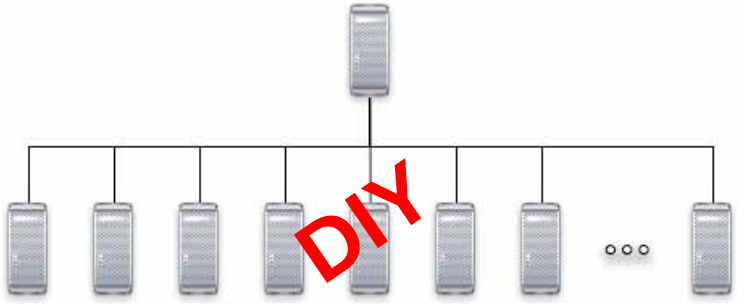
- **Wells Fargo Bank**
  - 背景: * 富裕層の個人顧客を多く抱える (80万以上)
    * 個々の顧客が複数の口座を所有しているため、財務データがいくつものデータベース、データマートに散在
    * 統合したビューを作りたい
  - データの集約として、最初はJMSの利用を検討するが、WebLogicのJMSでは信頼性は確保できても遅い
    → 仮想データサービス層としてCoherenceを採用

- **Starwood Hotel グループ**
  - 予約システムをはじめとするITシステムのレガシーモダナイゼーションを検討 (急増するインターネット経由での予約、サービス拡充のためにSOAを検討)
  - 評価ポイント: パフォーマンス、信頼性、スケーラビリティ、コストの予測性
    →企業のデータグリッド標準としての採用を決定



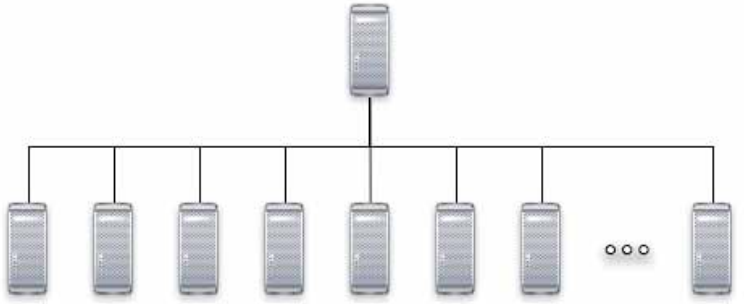
先行企業に学ぶ
グリッド型システムの動向

1st Generation = HPC + 自社開発



- 自社開発
- 特定のアプリケーション向け
  - バッチ処理
  - 典型的な料金計算エンジン
- アーキテクチャ
  - マスター/ワーカー・パターン、クライアント/サーバー、ブローカ/エージェント
  - + メッセージ・バス
- エンジニアがすべき課題
  - アルゴリズム設計 (同時接続、並行処理)
  - タスクの分散化、協調化
  - ネットワーク・インフラと連携プロトコルの設計

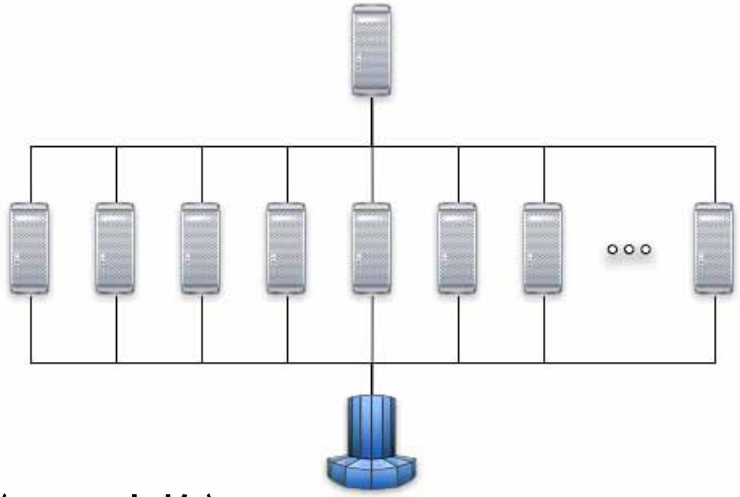
2nd Generation = ユーティリティ



- ベンダーによる協調の仕組み
- 多数のアプリケーションで利用
  - 料金計算エンジン
  - データの参照
  - OSの仮想化
- アーキテクチャ
  - ベンダー製品の利用：ブローカ/エージェント（XML変換）
  - カスタム開発 + メッセージ・バス
- エンジニアがすべき課題
  - ベンダーのツールの利用（コミュニケーション、スケジューリング、プロビジョニング、SLA定義、API、管理/監視コンソール）
  - 分散コンピューティング設計

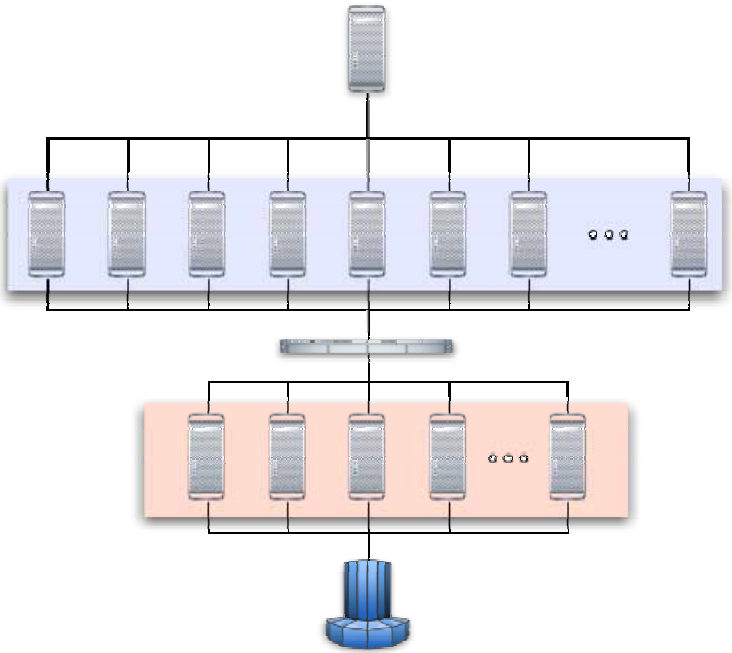
3rd Generation = データ・ボトルネック

- データが課題となる!
- 読み取りキャッシュの実装
  - 自社開発 or グリッド対応製品の利用
  - タスク用のローカル・キャッシュ
  - データソースの保護
- アーキテクチャ
  - ベンダー製品の利用：ブローカ/エージェント (XML変換)
  - カスタム開発 + メッセージ・バス
- エンジニアがすべき課題
  - 分散コンピューティング設計
  - キャッシュの有効化 (データアクセスへの理解)
  - データと処理の親和性



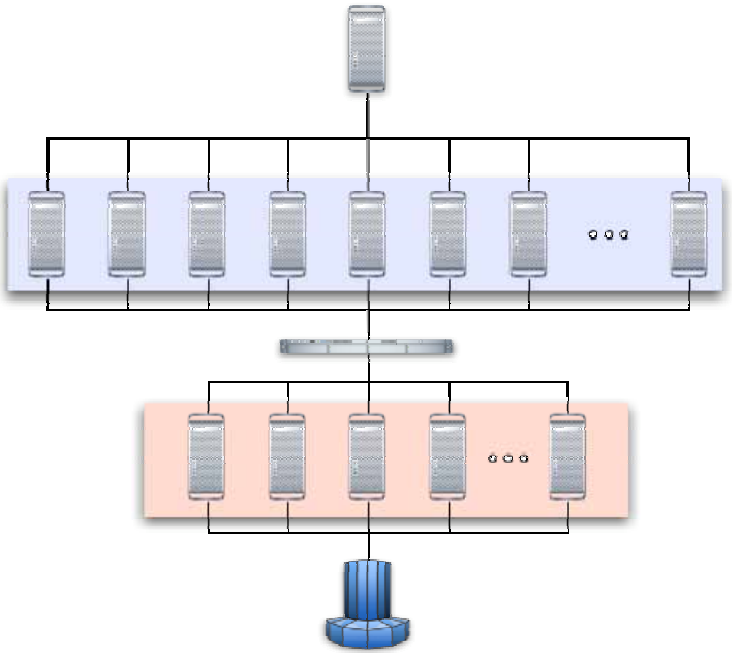
4th Generation = データグリッド (読取専用)

- データのための独立した層
- データグリッドの実装
  - 計算処理のための共有キャッシュ
  - タスク用のローカル・キャッシュ
  - 参照データおよび利用頻度の高いデータのキャッシュ
- アーキテクチャ
  - ベンダー製品の利用
    - ブローカ/エージェント
    - クラスタ対応キャッシュ・インフラ
  - カスタム開発 + メッセージ・バス
- エンジニアがすべき課題
  - 並行処理、データアクセスのためのアルゴリズムの設計
  - データと処理の親和性 (追加API、デプロイ)



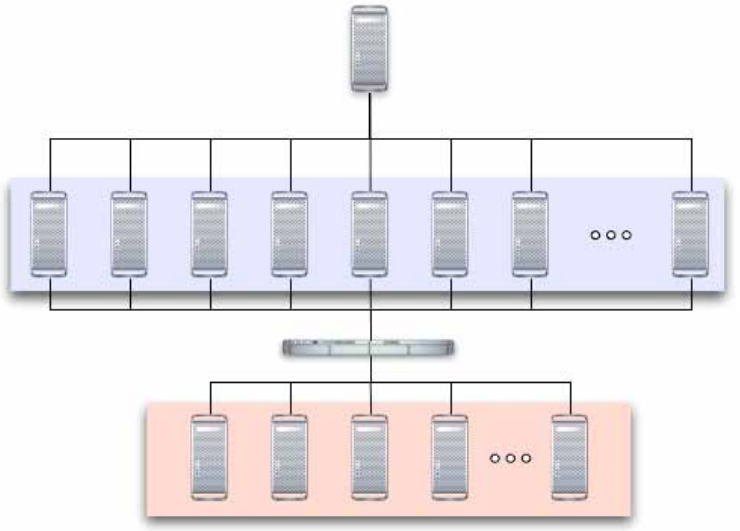
5th Generation = データグリッド (読み書き)

- 信頼性データグリッドの実装
  - データ記録の一次システム
  - 現行の処理結果に責任を持つ
  - ステートフル・アプリケーションでの利用
  - データソース = レポート/リカバリ目的
  - リアルタイム・イベントの提供
  - グリッド内のデータを通しての問合せ
- アーキテクチャ
  - ベンダー製品の利用
    - ブローカ/エージェント
    - 信頼性データグリッド・インフラストラクチャ
- エンジニアがすべき課題
  - データグリッド対応アプリケーションの設計 (システム・モデリング)

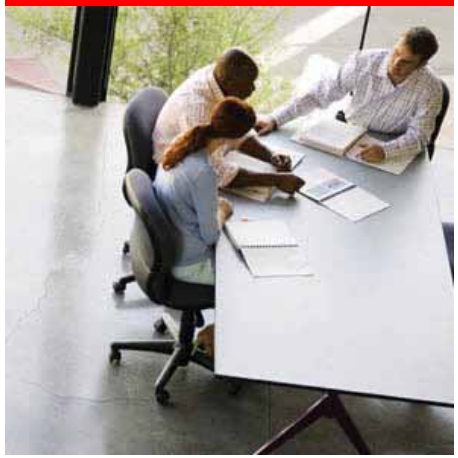


6th Generation = データグリッド (XTP)

- 信頼性データグリッドの実装
  - –データ記録の主システム
  - –トランザクションはデータグリッド上でのみ発生
  - –リアルタイム・イベントが必須
  - –データソースはリカバリ用途のみ
- アーキテクチャ
  - –ベンダー製品の利用
    - ブローカ/エージェント
    - 高信頼性データグリッド・インフラストラクチャ
- エンジニアがすべき課題
  - –データグリッド対応アプリケーションの設計
  - –トランザクションに縛られない分散コンピューティング
  - –高品質インフラストラクチャ



それでも
障害は起きる!

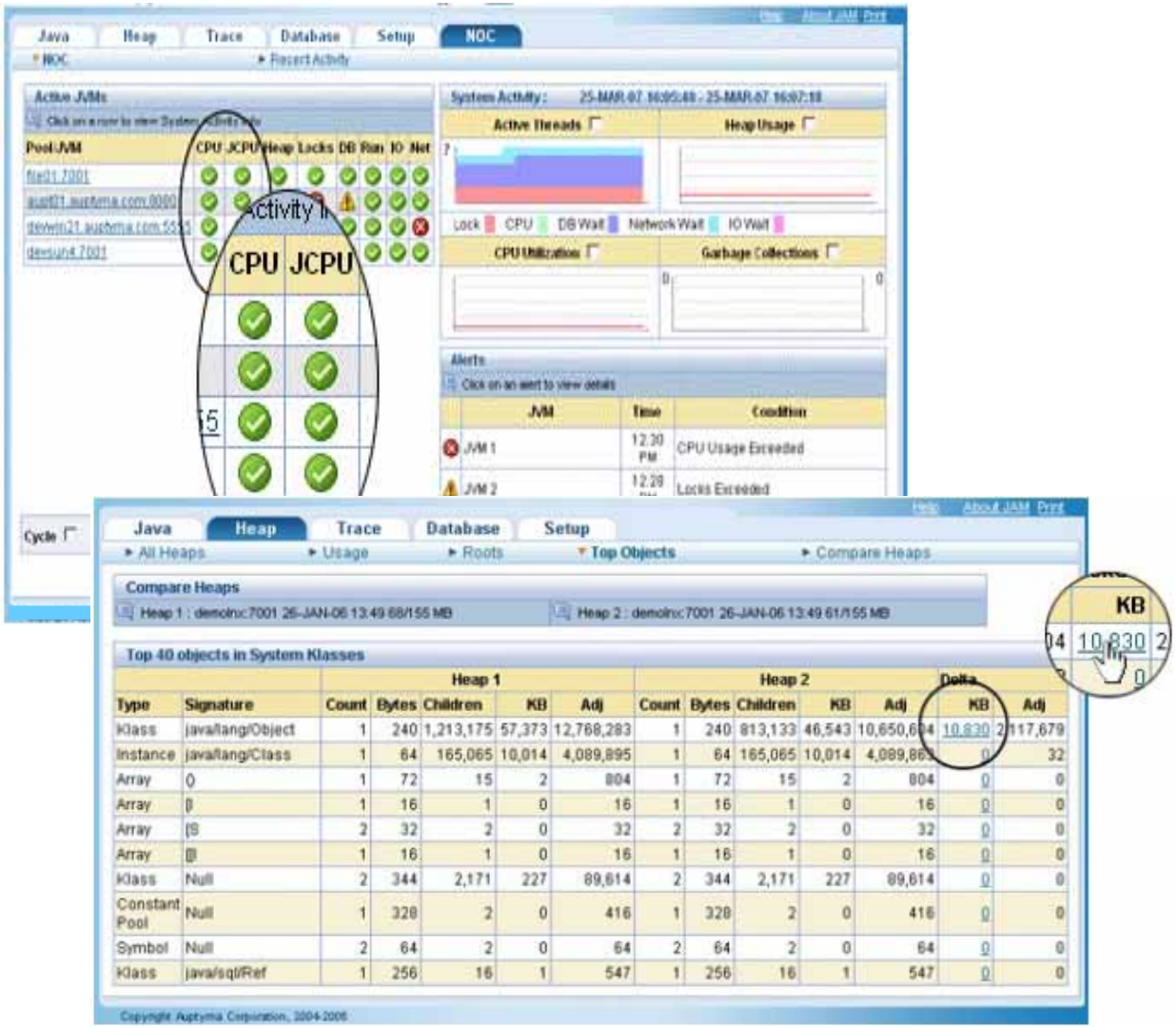


障害は起きる、問題は如何に原因にたどりつくか

- アプリケーションのモニタリングと診断の実施
  - 極力、オーバーヘッドのないトレース
  - **Java ←→ DB** のドリルダウン・トレース
  - 既存アプリケーションへの容易な組み込み

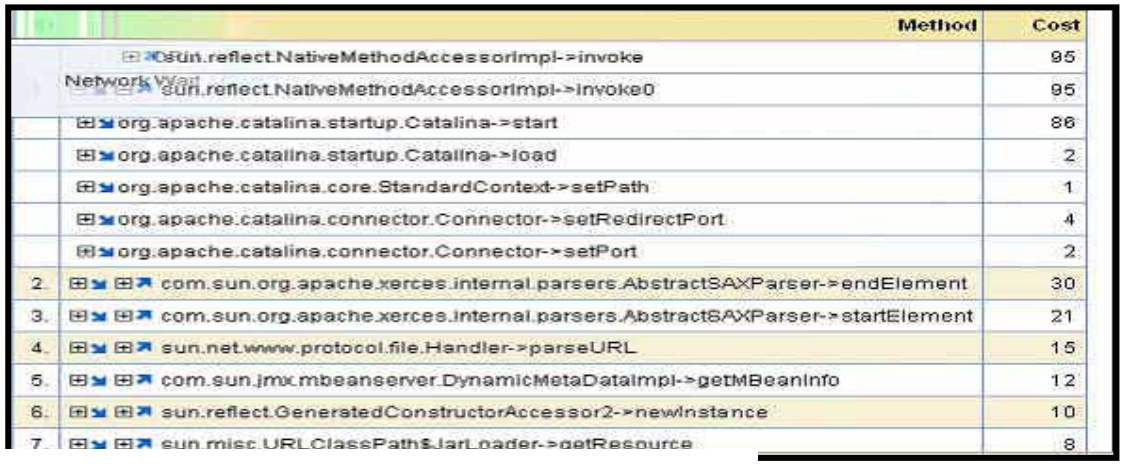
JVMモニタリング、ヒープの差分分析

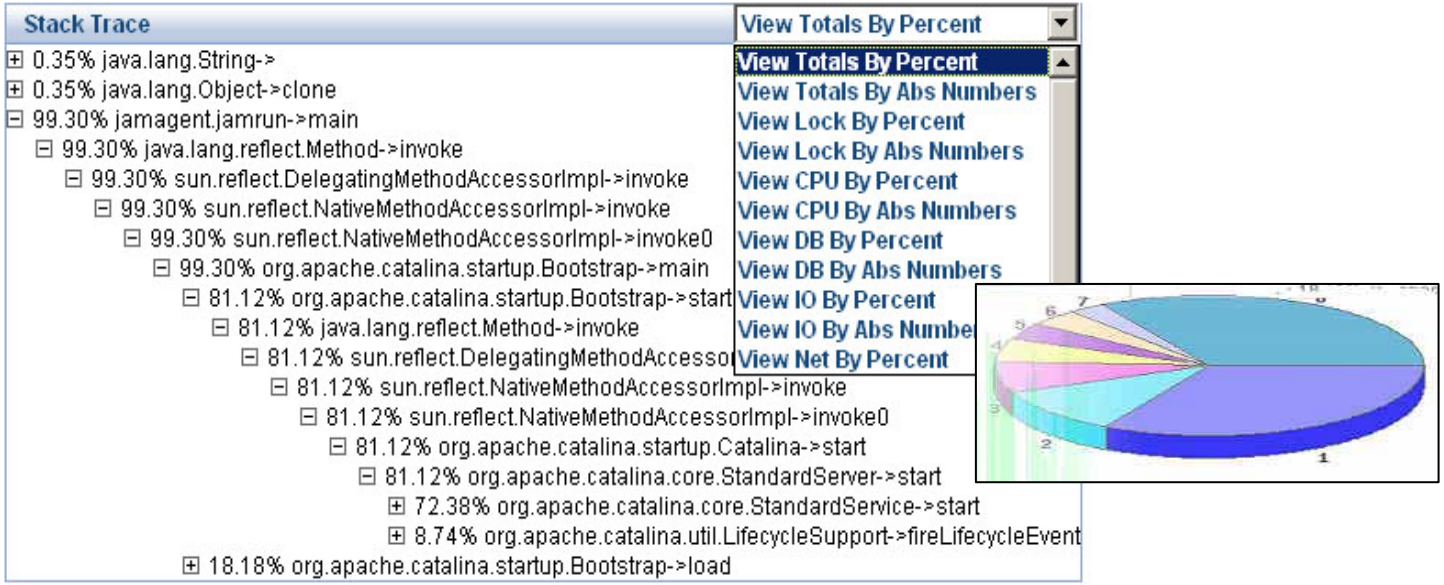
- CPU
- ヒープ使用
- GC挙動
- 時間コストの監視
  - オブジェクトのロック
  - DBコール
  - ディスクIO
  - ネットワーク
- 最小限オーバーヘッドでヒープの状況を取得
  - 定期的に取得したヒープ状態を比較してメモリリークを検出



ボトルネック分析

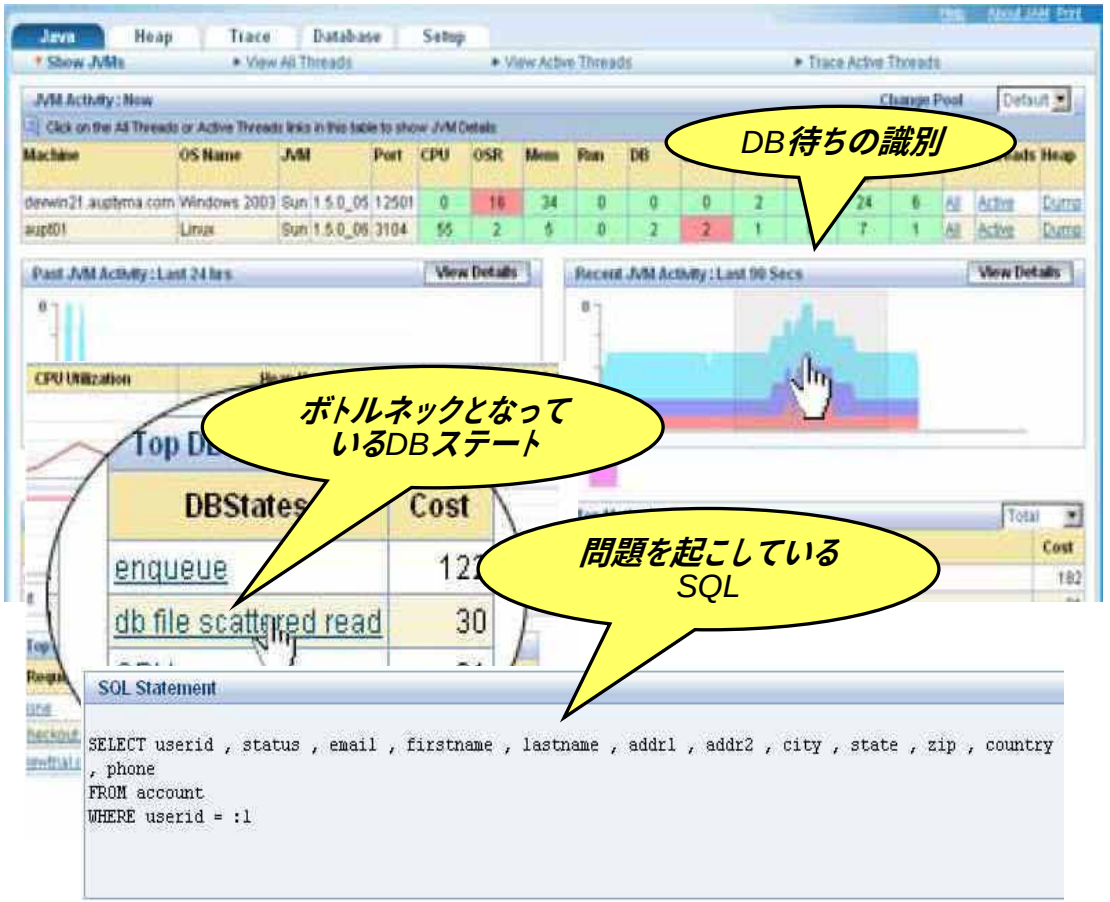
- メソッド/リクエスト分析
  - 完全なコールスタック
  - スタック内の各処理のコスト
- 実行中のスレッドの
  リアルタイム・トレース
- ボトルネックリソースの表示





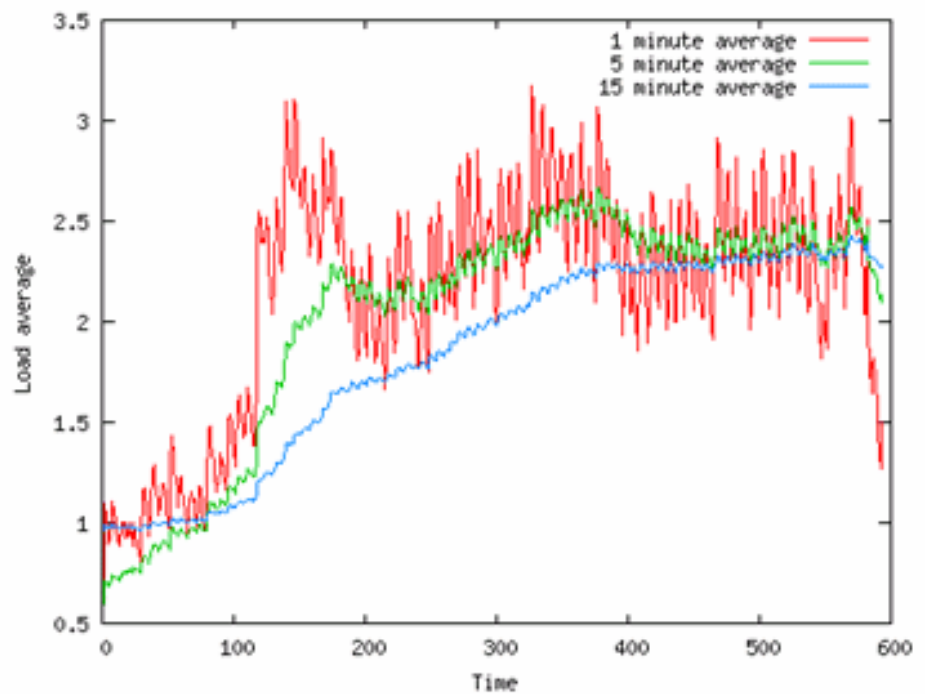
Oracle DBにまたがるトレース

- JavaスレッドからDBセッションへのトレース
  - DBリソース待ちの実行中Javaスレッドの識別
  - SQLへのドリルダウン
- DBセッションからJavaスレッドへのトレース
  - 待ち状態/ロック状態のDBセッションの表示
  - DBセッションを保持しているJavaスレッドの識別

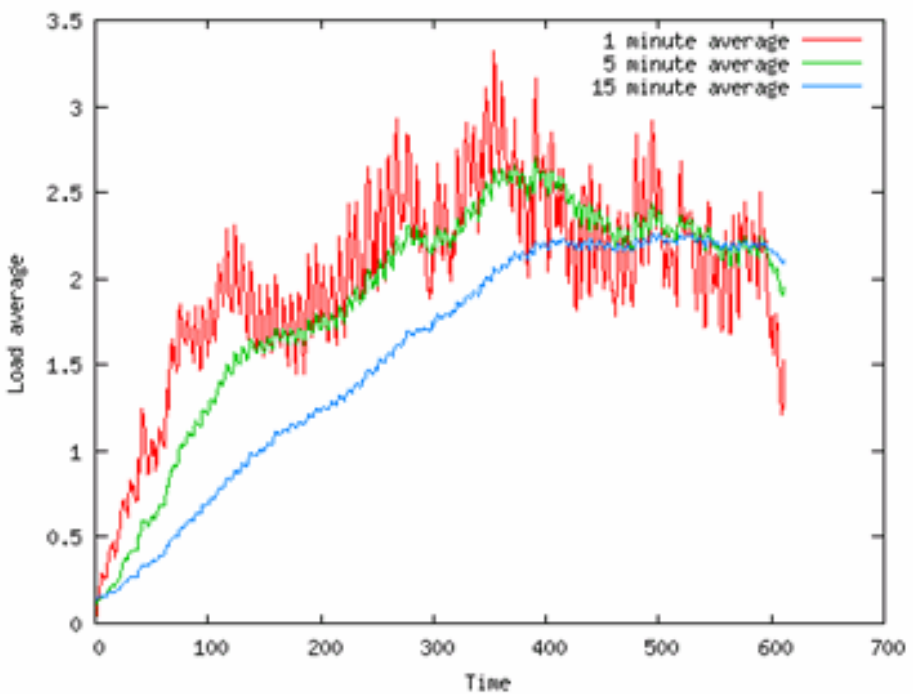


これらを最小限のオーバーヘッドで!

Java Application Monitor なし



Java Application Monitor あり

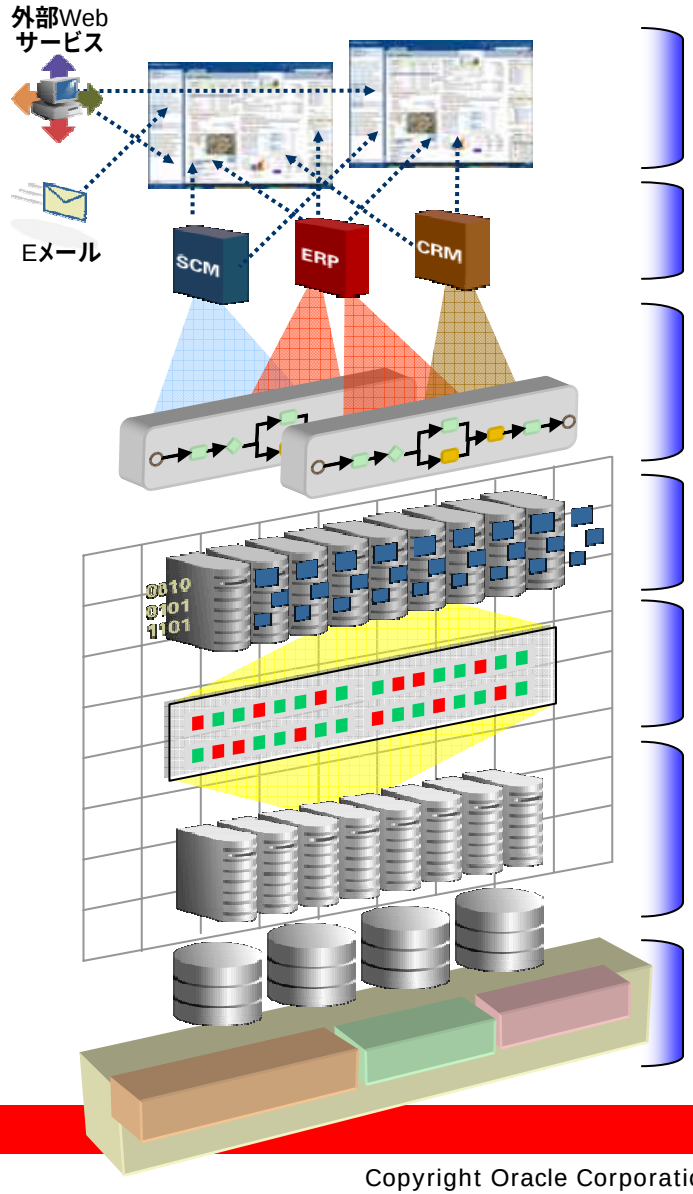


最後に製品情報

- **Oracle Coherence**
  http://www.oracle.co.jp/products/middleware/coherence/
  - **2007年9月11日より 出荷中**
  - 金融、通信、ネットビジネスのシステム向け
  - ホスト基幹システムのオープン化

- **Oracle Enterprise Manager**
  Java Application Diagnostic Expert (JADE)
  2008年出荷予定

オラクルのトータル・グリッド/SOAソリューション



Web 2.0型 UIマッシュアップ
- Oracle WebCenter：
  業務の流れに従って連動する統合UI の実現

ビジネス・アプリケーション
- Fusion Applications、Application Integration Architecture
  真にSOA対応のアプリケーション・アーキテクチャ

SOA/イベント駆動アーキテクチャ (EDA)
- Oracle SOA Suite：SOAをトータルでカバーするスイートセット
- Oracle EDA Suite：Oracle SOA Suite と同じテクノロジースタックを使って構成された、イベント駆動の連携基盤 (SOA+RFID)

アプリケーション・サーバー・グリッド
- Oracle Application Server：Webキャッシュ層、HTTPサーバー層、Java EE層を個別にクラスタ化し、DBとの優れた連携統合を実現

ミドルウェア・データグリッド
- **Oracle Coherence**：
  ミドルウェアにおけるメモリ・データの可用性とスケーラビリティを確保

データベース・グリッド
- 実績あるOracle Database および Oracle Real Application Clusters がグリッドの特徴そのままに、データベースの可用性と信頼性、スケーラビリティを確保

ハードウェア仮想化技術
- Oracle GRID Center：ハードウェアベンダー各社とともにグリッド基盤のベスト・プラクティスを構築


ORACLE IS THE INFORMATION COMPANY