ORACLE 11g FUSION MIDDLEWARE

ORACLE®

全ての挙動は記録される
開発環境でも本番環境でも進化したJVMの現在

日本オラクル株式会社　Fusion Middleware 事業統括本部

2010年 11月

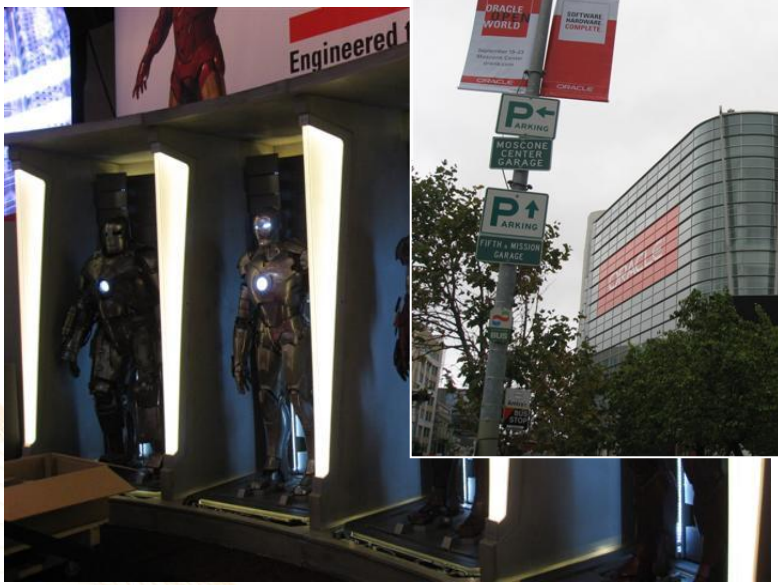
以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。
また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。
オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

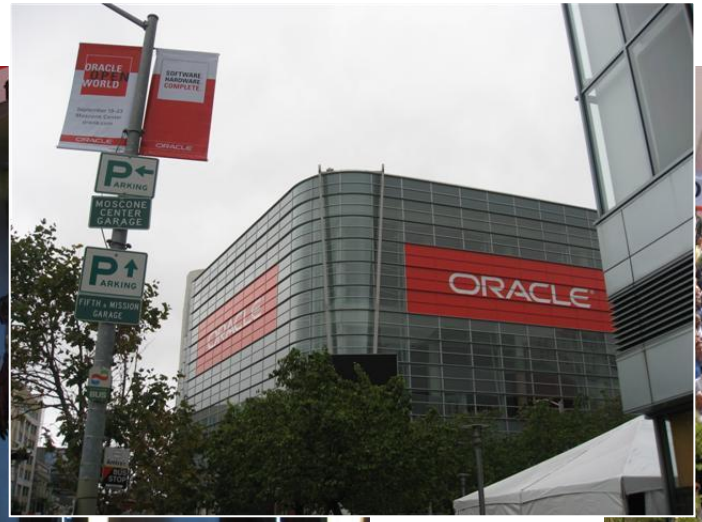
本日の内容

- **JavaOne および Announcement サマリー**
- **Java/JVM の方向性**
  - ロードマップ
  - **Flight Recording**
  - サーバー仮想化対応

JavaOne 2010 @ SF, 9/19-23

- **One City, One Week, Three Conferences**
  - **Oracle OpenWorld**
  - **JavaOne**
  - **Oracle Develop**





Announcement

Plans for Java platform – 2010/09/20

Oracle Outlines Plans for Java Platform

JavaOne 2010 Keynote Address Highlights Oracle's Plans to Grow Java and Showcases Java Technology Innovation

JAVAONE, SAN FRANCISCO – Sept. 20, 2010

News Facts

During the opening keynote of JavaOne 2010, Thomas Kurian, executive vice president, Oracle Product Development outlined plans for the future of the Java platform and showcased product demonstrations illustrating the latest Java technology innovations.

Presentation highlights covered four key areas of Java technology:

- Java Platform, Standard Edition (Java SE) – Oracle is advancing the Java platform and optimizing it for new application models and hardware; including extended support for scripting languages, increased developer productivity and lower operational costs. Kurian discussed the roadmap for JDK 7 and JDK 8, which will be based on OpenJDK, and highlighted some of the key OpenJDK projects.
  - Product demonstration showed Oracle JRockit and Oracle HotSpot Java Virtual Machines (JVMs) working with Oracle JRockit Mission Control to diagnose the root cause of poor response times in applications.
- Java on the Client – Oracle is planning an enhanced programming model that combines the power of Java with the ease of JavaFX, to deliver advanced graphics, high-fidelity media and the best HTML 5, JavaScript and CSS Web capabilities, along with native Java platform support.
  - Product demonstration highlighted the Prism graphics stack delivering 2D and 3D graphics, vector graphics and high-quality media.
- Java Platform, Enterprise Edition (Java EE) – Java EE will continue to evolve making application servers more modular, and programming more efficient with improvements such as dependency injection and reduced configuration requirements.

- **JavaOneキーノート講演の内容を反映した発表**
  - **JDK 7/8 および OpenJDKについて**
  - **JavaFX、HTML5/JavaScript連携**
  - **Java EE 6、ライトウェイト化**
  - **Java のデバイス対応**

http://www.oracle.com/us/corporate/press/173712

Announcement

JDK roadmap for Advancing Java SE – 2010/09/21

Oracle Announces JDK Roadmap for Advancing Java SE

JAVAONE, SAN FRANCISCO – Sept.21, 2010

News Facts

- Oracle is announcing its plans for advancing the Java Platform, Standard Edition (Java SE) and optimizing it for new application models and hardware, including extended support for scripting languages, increased developer productivity and lower operational costs.
- The announced roadmap for the OpenJDK accelerates the availability of Java SE with two releases, one in 2011 and one in 2012. These OpenJDK releases will continue to serve as the basis for the Oracle Java Development Kit (JDK) 7 and JDK 8.
- The decisions regarding the features to be included in the JDK 7 and JDK 8 releases were made with active participation of the Java community.
- The OpenJDK project continues to thrive with contributions from Oracle, as well as other companies, researchers and individuals. The OpenJDK licensing model remains the same.
- Oracle is currently working to merge the Oracle Java HotSpot Java Virtual Machine (JVM) and the Oracle JRockit JVM into a converged offering that leverages the best features of each of these market-leading implementations.
- Oracle plans to contribute the results of the combined Oracle Java HotSpot and Oracle JRockit JVMs to the OpenJDK project.
- The Oracle JDK and Java Runtime Environment (JRE) will continue to be available as free downloads, with no changes to the existing licensing models.
- Premium offerings such as JRockit Mission Control, JRockit Real Time, Java for Business and Enterprise Support will continue to be made available for an additional charge.

Proposed Features of JDK 7 and JDK 8

- Proposed JDK 7 Features

- **OpenJDKの継続のコミット**
  - **JDK 7/8 のベースとして開発**
  - **ライセンス形態の継続**
- **JDK7/8 のプラン**
- **HotSpot と JRockit の融合**

http://www.oracle.com/us/corporate/press/173782

JVM 推奨事項

- 現行
  - HotSpot
    - 全てのSolaris上のJavaアプリケーション
    - Windows/Linux上のJavaクライアント
  - JRockit
    - Windows/Linux上のJavaサーバー・システム
      特に、オラクル製品はJRockitに最適
  - あくまで推奨。どちらを使ってもよい。
- 将来
  - コンバージェンス?

Announcement

JavaFX, Java Communities など

JavaFX ロードマップ - 9/21

- HTML、JRuby、Groovy、JavaScript連携
- JREとの連携強化
- 2D、3D レンダリングでのHWアクセラレータ活用

Java/オープンソースへのコミットメント - 9/22

- GlassFish Serverの継続
- Eclipse、NetBeansの継続
- OTNを通じてのコミュニティ支援

OpenJDK における IBM の参画 - 10/11

- IBM とともにロードマップを策定
- 開発/実装の協力


Announcing:
The Exalogic Elastic Cloud
Cloud in a Box
ORACLE
Sun
ORACLE
EXALOGIC
ELASTIC CLOUD
X

Java/JVMの方向性

- Java Core

Java Core – JDK のロードマップ

開発生産性/保守性向上、最適化/モジュール化、
最新ハードウェアへの対応(マルチコア、大容量メモリ、InfiniBand)

- **JDK7 – 2011年内?**
  - マルチコアを利用した並列処理(Folk-Join)
  - Project Coin の一部実装
  - その他
- **JDK8 – 2012年後半?**
  - Project Coin　　: 開発生産性向上のための仕様追加
  - Project Lambda : ラムダ式、クロージャ
  - Project Jigsaw　: モジュール化構造

ORACLE

Project Coin

開発簡易化、動的な型判定などの小さな言語仕様の追加

- **switch の条件での String の利用**

```
switch(s) {
    case “foo”: processFoo(s); break;
    case “bar”: processBar(s); break; …
```

- **コレクション（ListやMap）の初期化の簡素化**

```
final Map<Integer, String>  platonicSolids = { 4 : "tetrahedron", 6 : "cube", 8 : "octahedron‟ };
```

- **可読性の高い整数リテラル表現**

```
final int num = 1_345_428_906;
```

- **try-catch における自動的なリソース解放**
  （例: InputStream の自動close）

さらなる多くの拡張仕様に関し、JDK8を目指して議論継続中

スクリプト言語の状況

JavaScript

- **switch**

```
switch ( x ) {
    case "foo": result = "It's Foo!"
    case "bar" : result += "Bar" …
```

- **配列**

```
scores = { "Brett":100, "Pete":-1, "Andrew":86 }
```

Groovy

- **switch**

```
switch ( x ) {
    case "foo": result = "It's Foo!"
    case "bar" : result += "Bar" …
```

- コレクション

```
scores = [ "Brett":100, "Pete":-1, "Andrew":86 ]
```

Ruby

- **case**

```
case x
when "foo"
    puts "It's Foo!"
when "bar"
    puts "It's Bar"
```

- コレクション

```
scores = Hash[ "Brett",100, "Pete",-1, "Andrew",86 ]
```

- 生産性が高いスクリプト言語の良い部分を採用
- 同時に、Javaとの親和性向上にも寄与

Project Lambda

直接の目的は、単一メソッドの内部クラスの記述の簡素化

```
Collections.sort(people,
    new Comparator<Person> () {
        public int compare(Person a, Person b) {
            return a.getLastName().compareTo(b.getLastName());
        }
    }
);
```

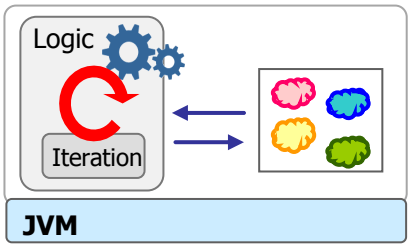
```
Collections.sort(people,

    #{Person a, Person b ->
        a.getLastName().compareTo(b.getLastName()) }

);
```

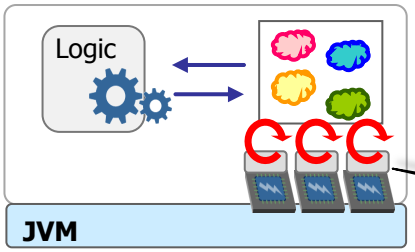
① Collections.sort の定義から Comparator<? super T> の表現であることを判別
② Comparator の単一メソッドである compare の実装部分として利用

Project Lambda




```
double max = 0.0;
for (Student s : students) {
    if (s.getGradYear == 2010) {
        if (s.getScore > max) {
            max = s.getScore;
        }
    }
}
```

ラムダ式表現によって
通常の繰り返し処理から
内部イテレーションの利用に
対する敷居が下がる




```
double max =
    students.filter(
        new Predicate<Student>() {
            public boolean op(Student s) {
                return s.getGradYear == 2010;
            }
        }
    ).map(
        new Extractor<Student, Double>() {
            public Double extract(Student s) {
                return s.getScore;
            }
        }
    ).max();
```

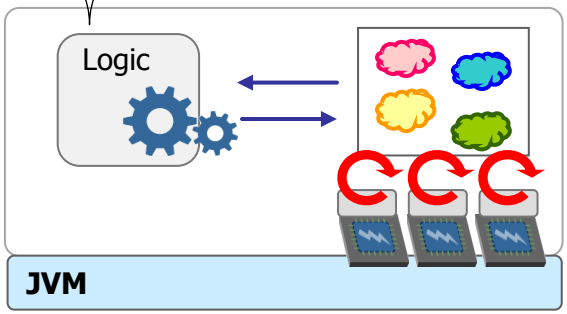
```
double max =
    students.filter( #{ s -> s.getGradYear==2010 }
    ).map(           #{ s -> s.getScore }
    ).max();
```

JVMの実装次第で
マルチコア活用型
パラレル処理を容易に
利用可能になる

Oracle Coherence によるパラレル処理

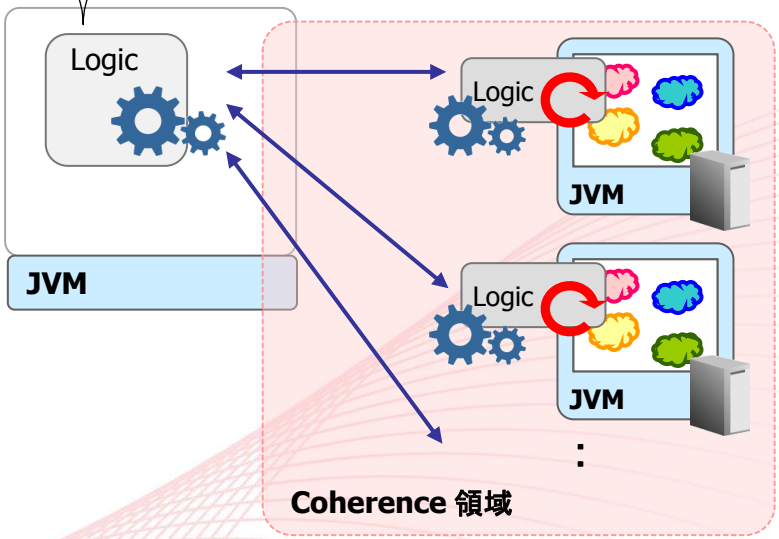
内部イテレーションによるパラレル処理 - マルチコアの活用

```
double max =
    students.filter(
        # { s -> s.getGradYear == 2010 }
    ).map(
        # { s -> s.getScore }
    ).max();
```



Oracle Coherenceによるパラレル処理 - マルチサーバー・リソースの活用

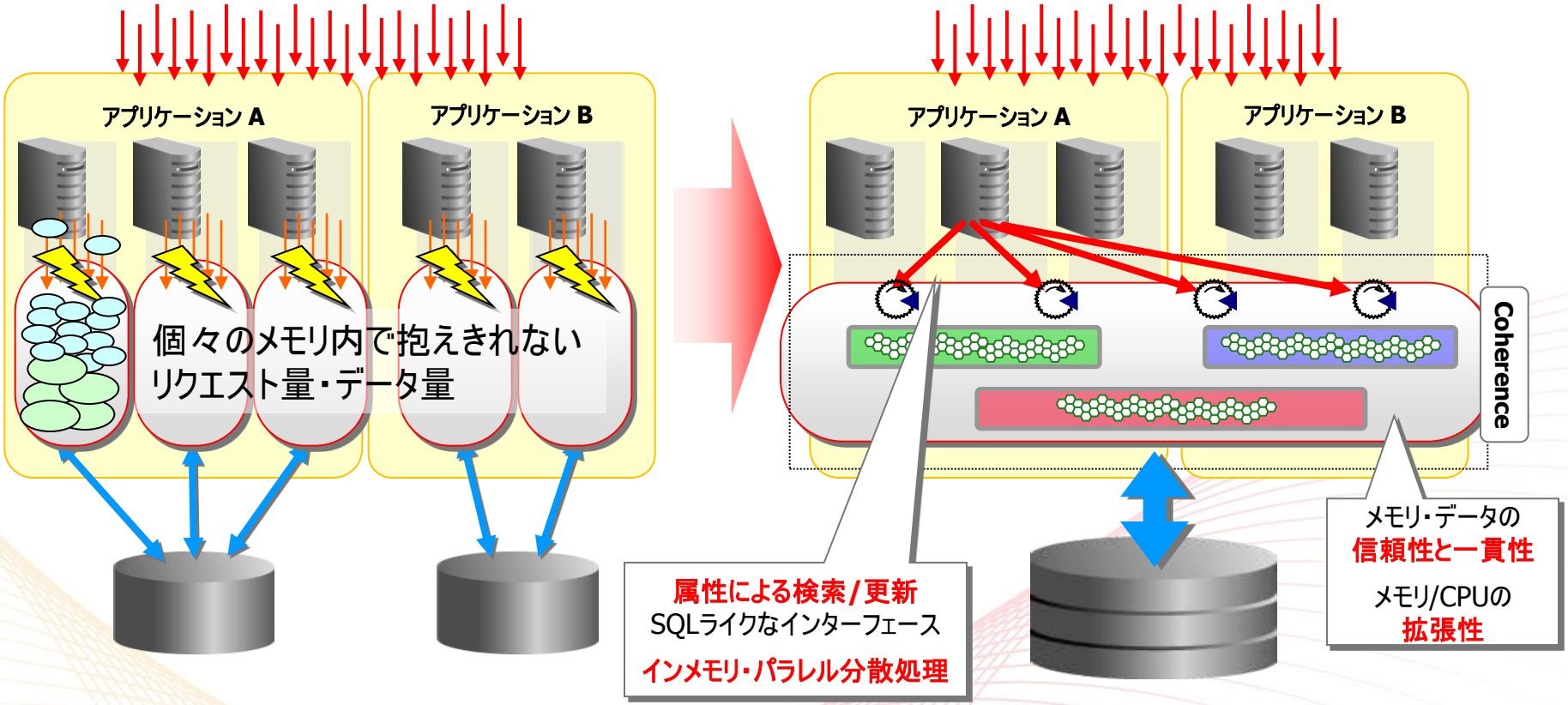
```
double max =
    students.aggregate(
        QueryHelper.createFilter( "gradYear = 2010" ),
        new DoubleMax ( "getScore")
    );
```



Oracle Coherence

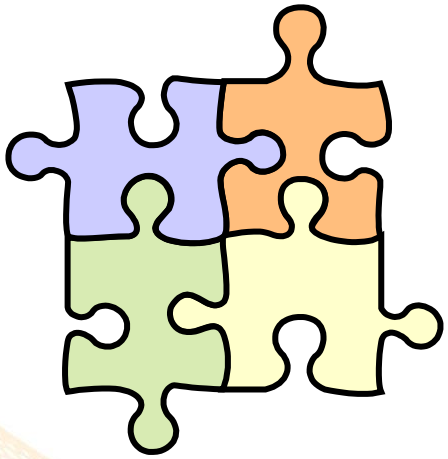
情報爆発の時代に応える企業向けインメモリ分散キーバリュー・ストア（KVS）

- HW横断型の分散共有メモリ領域 → スケールアウト型でHWコストを最適化
- メモリ・データの信頼性向上 → ディスクIOの最小化（高速化）
- インメモリ・イベント処理基盤 → サブシステム間連携を最適化（高速化・効率化）

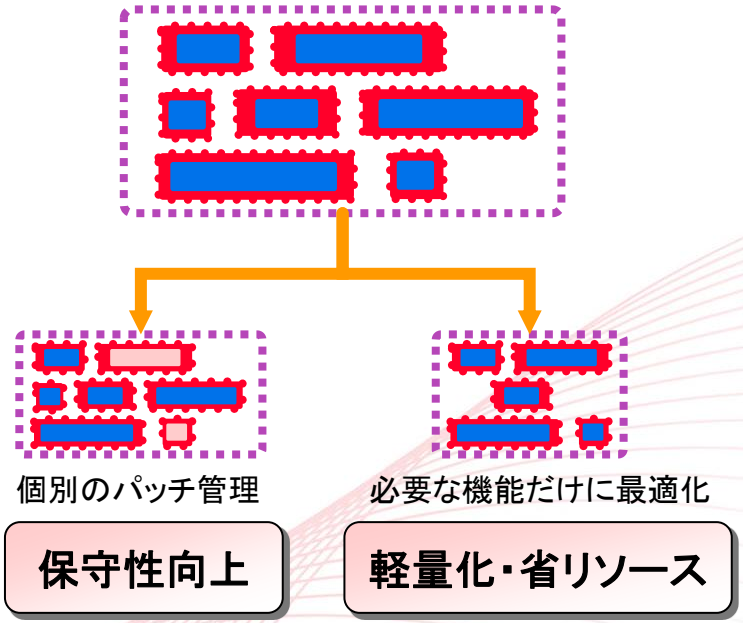


Project Jigsaw

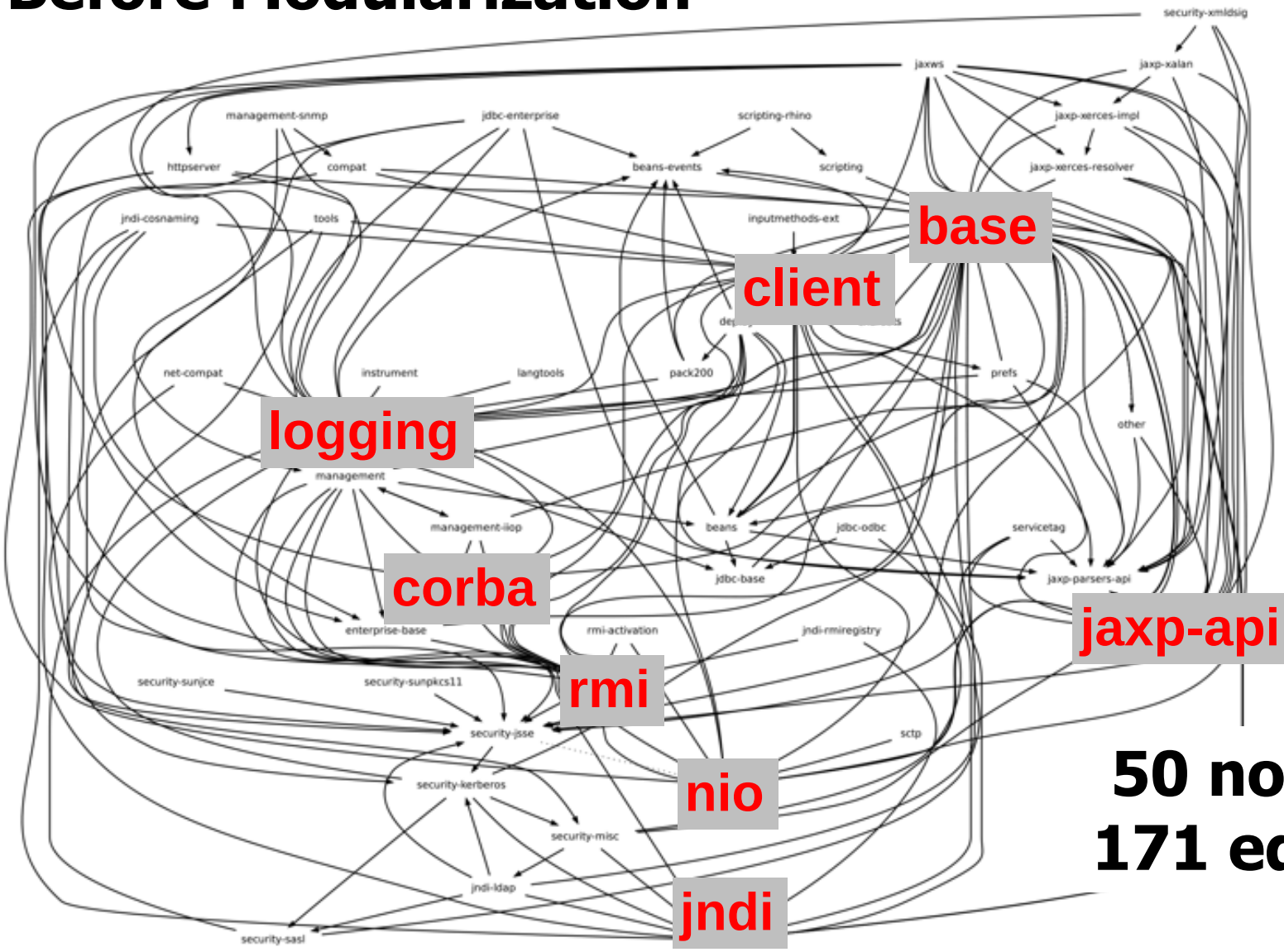
- 現行の Jar によるパッケージングをより高精度にモジュール化し、世代管理を組み込む
  - 依存性の解決・簡素化
  - 起動時間のスピードアップ
  - リソース節約・最適化



- JDK自身も対象
- 先行する OSGi との相互運用
- デバイス向け、クラウド環境向け

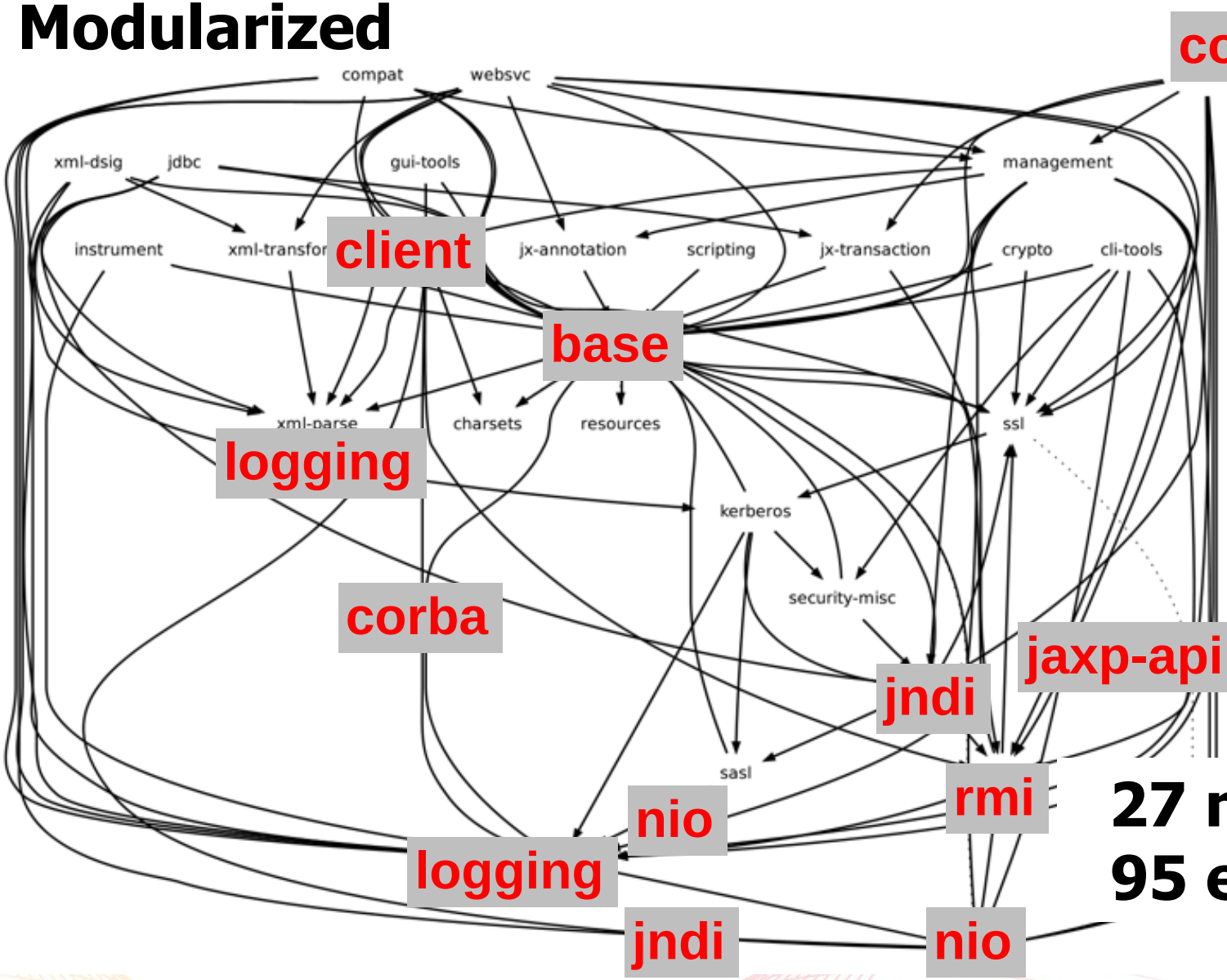


Before Modularization



50 nodes
171 edges

Modularized



Java/JVMの方向性
- Flight Recording

JRockit

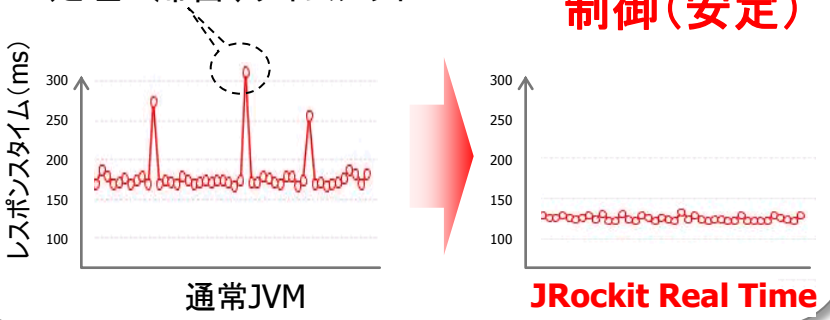
Java SE 完全準拠の高機能 JVM

- **高速：**サーバーサイド用途に最適化。公式ベンチマークで実証済み。
- **安定：**GCによる遅延時間の平準化。ミリ秒単位の応答時間の維持を実現。
- **全てを記録：**稼動状況を常に記録。予測外の事態でも過去に遡って分析可能。

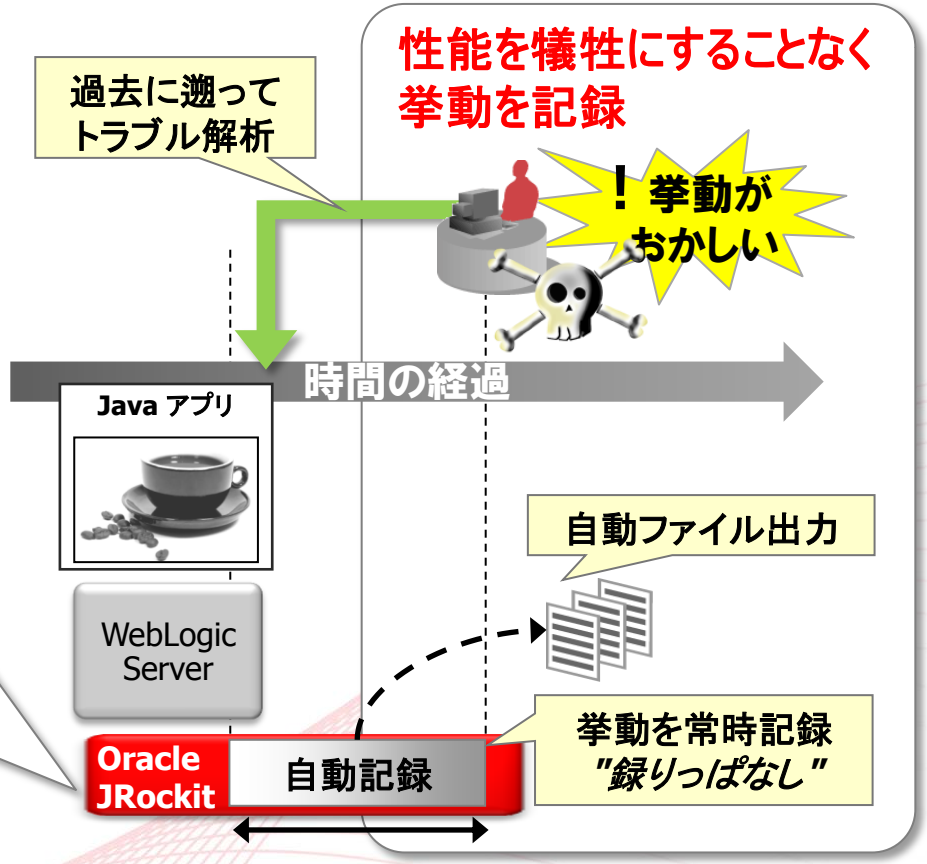
#1 パフォーマンス（高速）

公式ベンチマーク（SPECjbb）
- 処理性能数値の世界最高値
- JVMあたりの性能数値の世界最高値

不測のGC処理
→処理の滞留、タイムアウト…

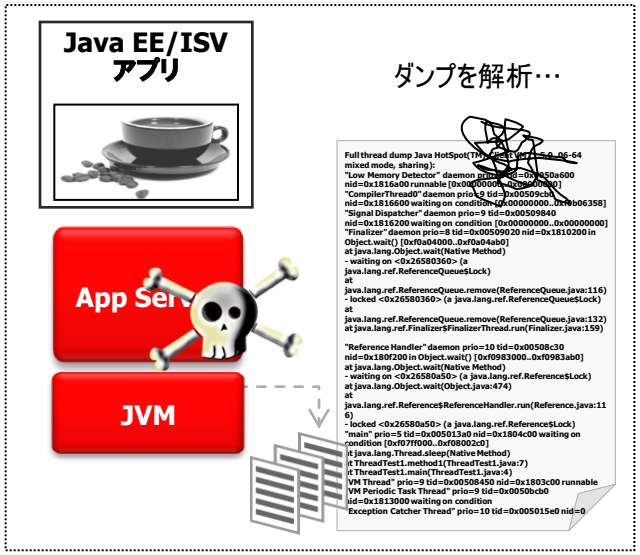


GCをJVMで制御（安定）



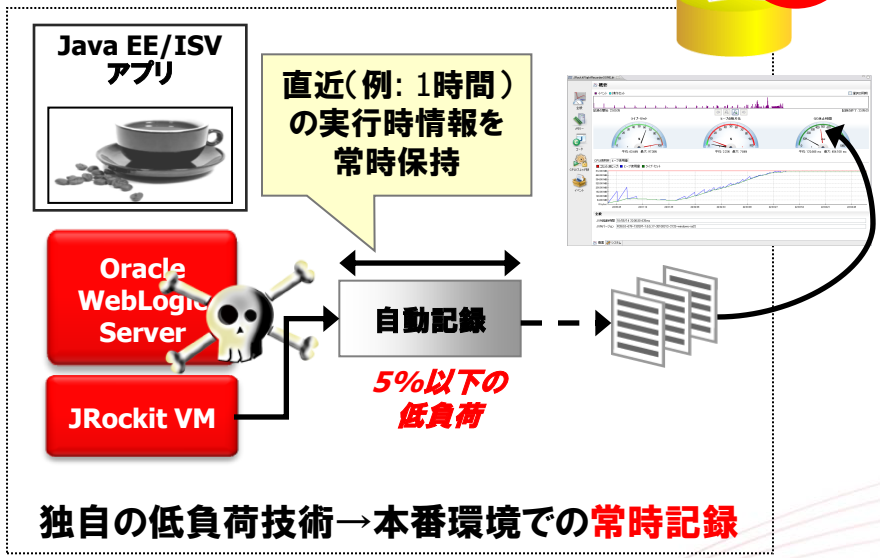
JRockit Flight Recorder

• これまでのJava



- ✓ 問題解決に必要な情報の多くは消失
- ✓ 原因究明に多くの時間が必要

• JRockit Flight Recorder



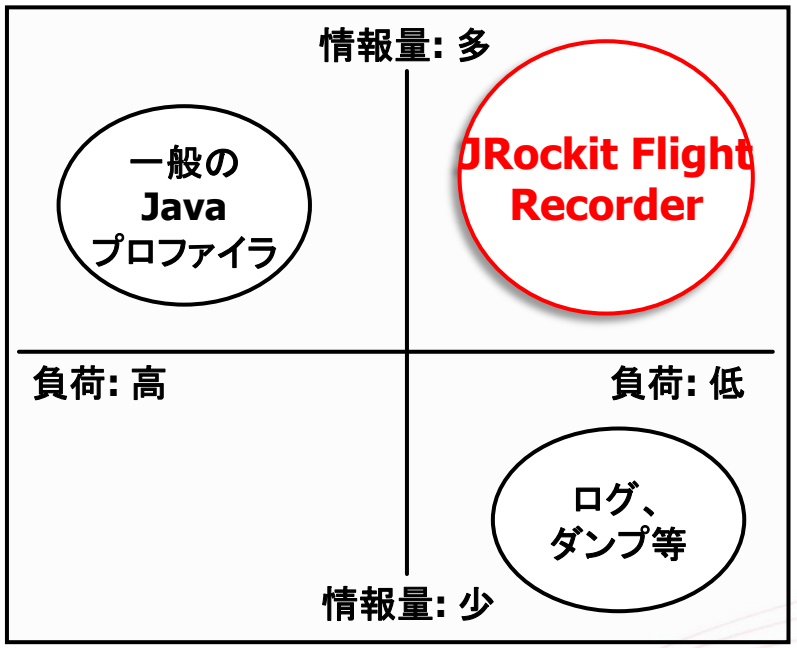
独自の低負荷技術→本番環境での**常時記録**

- ✓ 履歴を含めた情報を記録し自動出力
- ✓ 迅速な原因究明を支援するGUIツール

1. トラブルの確実な原因追究を「後から」実施可能
2. 障害発生→改善のサイクルと手間を大きく短縮化

これまでのトラブル対応ソリューションとの違い

1. “録りっぱなし” の実現
   - JRockitの内部機構を利用した
     本番環境を邪魔しない低負荷記録
2. 高精度の情報と分析ツール
   - JVM、Javaアプリのイベントを
     高精度に記録
   - 優れた解析ツール
3. 既存環境への適用が容易
   - インストール不要
     アプリケーション改変不要



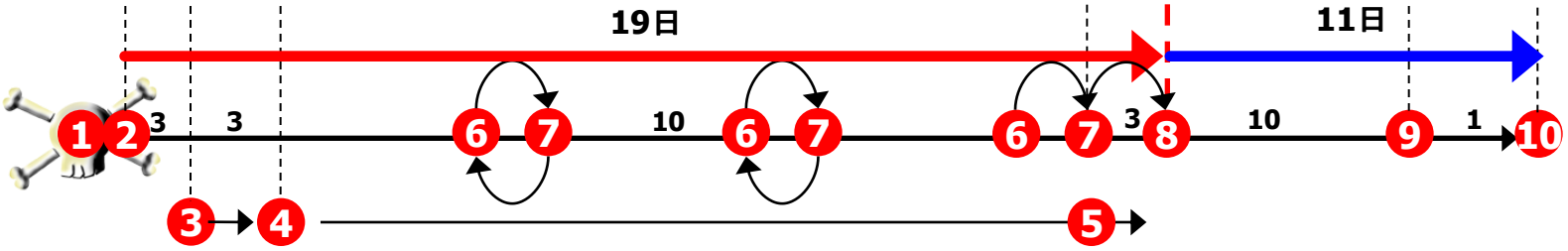
- 本番環境への適用が可能
  - ✓ 機会損失の最小化
  - ✓ 障害対応コストの最小化
- 社会的信用の維持
  顧客からの信頼度向上

JRockit Flight Recorder の効果例

- 過去に実際にあった問題（以下）の対応プロセスを元に比較
  - –対象: WebLogic Server上のWebアプリ
  - –事象: アクセスが増える毎にレスポンスタイムが悪化

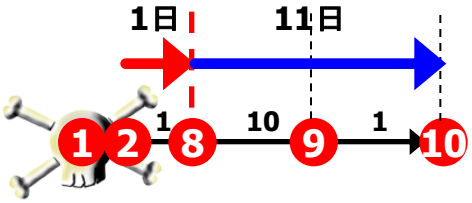
（※）JRockit Flight Recorderの場合の日数=当該事象を元に推測

これまでのJava　**問題解決までの期間: 30日（原因判明まで: 19日）**



JRockit Flight Recorder　**問題解決までの期間: 12日（原因判明まで: 1日）**

迅速かつ確実な問題解決



1. 問題発生、対処のため再起動
2. 解析開始
3. 再現環境の準備開始
4. 再現環境にて再現を試行
5. 再現せず
6. 本番環境で再発
7. 情報収集した後再起動
8. 原因判明
9. 対処、テスト
10. 本番に適用

JRockit Flight Recorderを利用することで省略できる作業

期間短縮の理由

- 再現作業不要
- 再発待ち不要

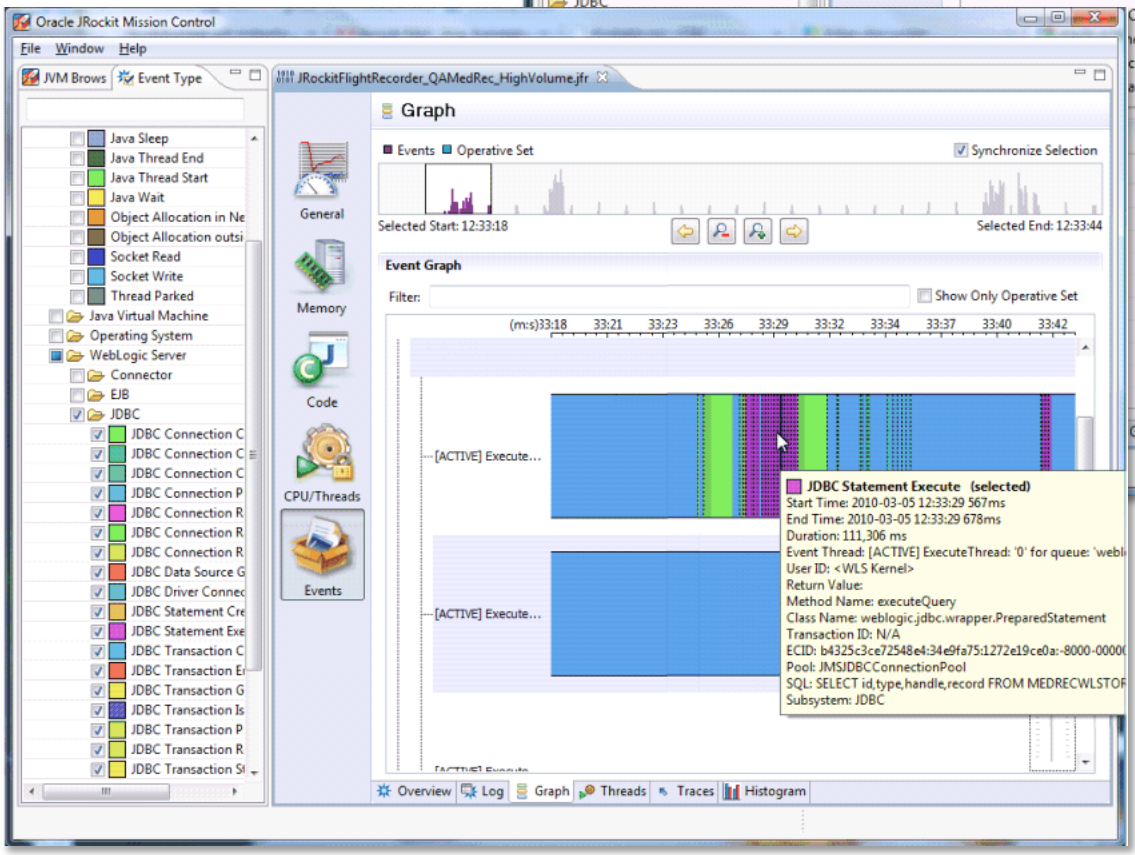
例

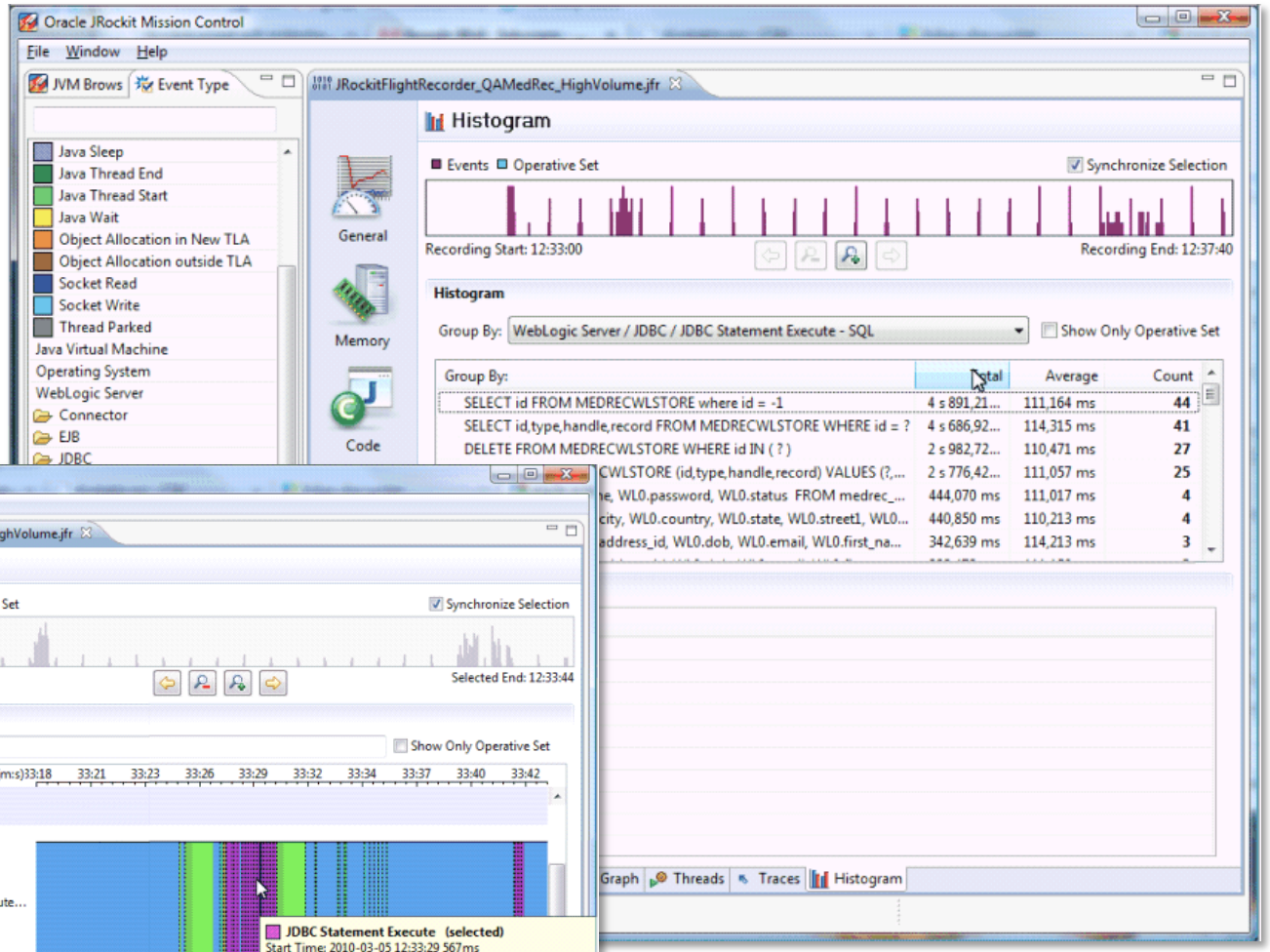
■ ヒープ状況

■ GC状況の可視化

例

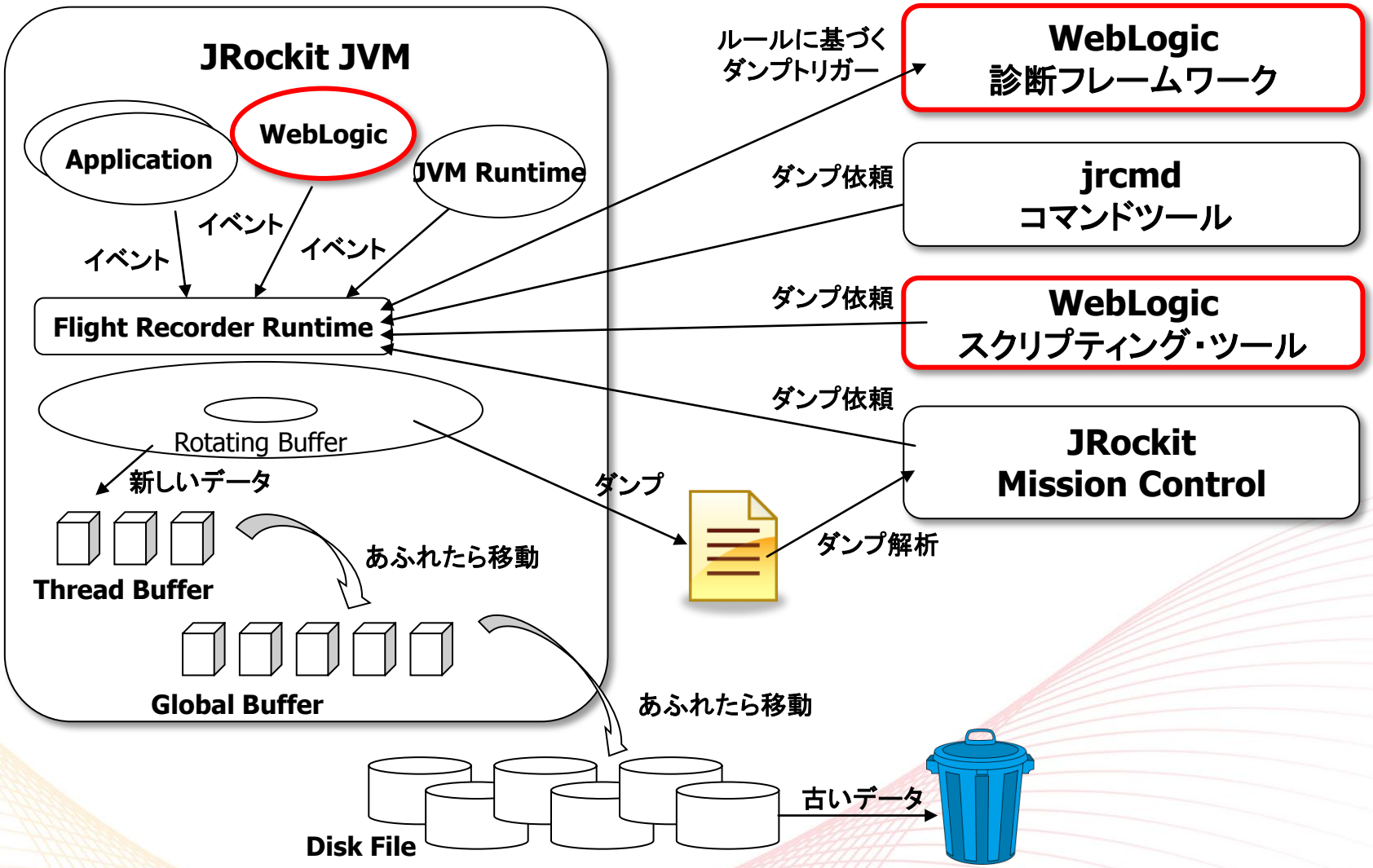
■ JDBCコール部分の分析





■ 実際にコールされたSQLの識別

JRockit Flight Recorder のアーキテクチャ



Java/JVMの方向性

- Flight Recording

with Application Server

診断ボリューム設定（WebLogic Server 10.3.3以降）

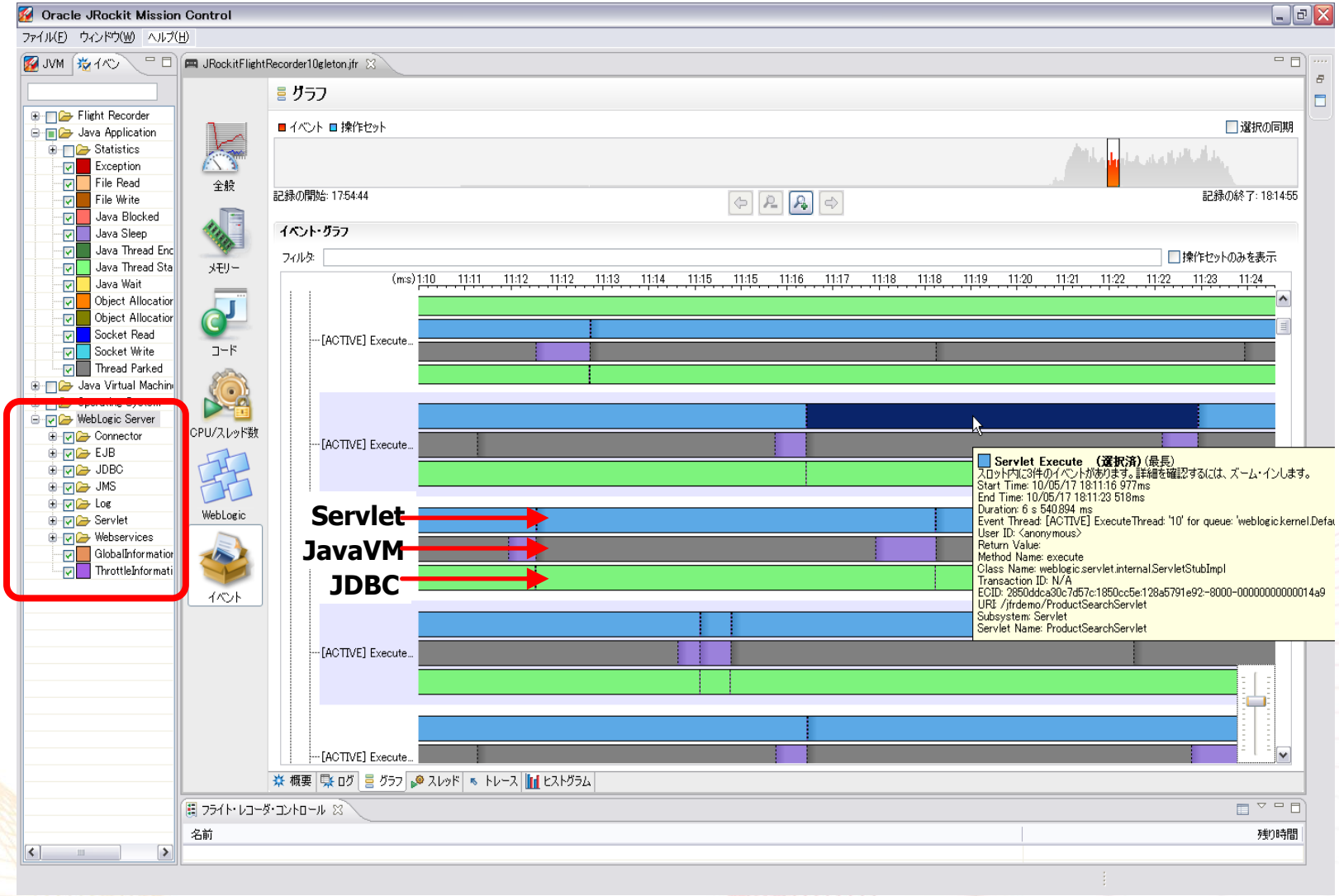
- **WebLogic Server が実行時に JRockit Flight Recorder用に生成するイベントの量を制御可能**
  - オフ、低、中、高のいずれかを指定

■ **WebLogic 管理コンソールでの設定**

JRockit Flight Recorder で記録できる情報

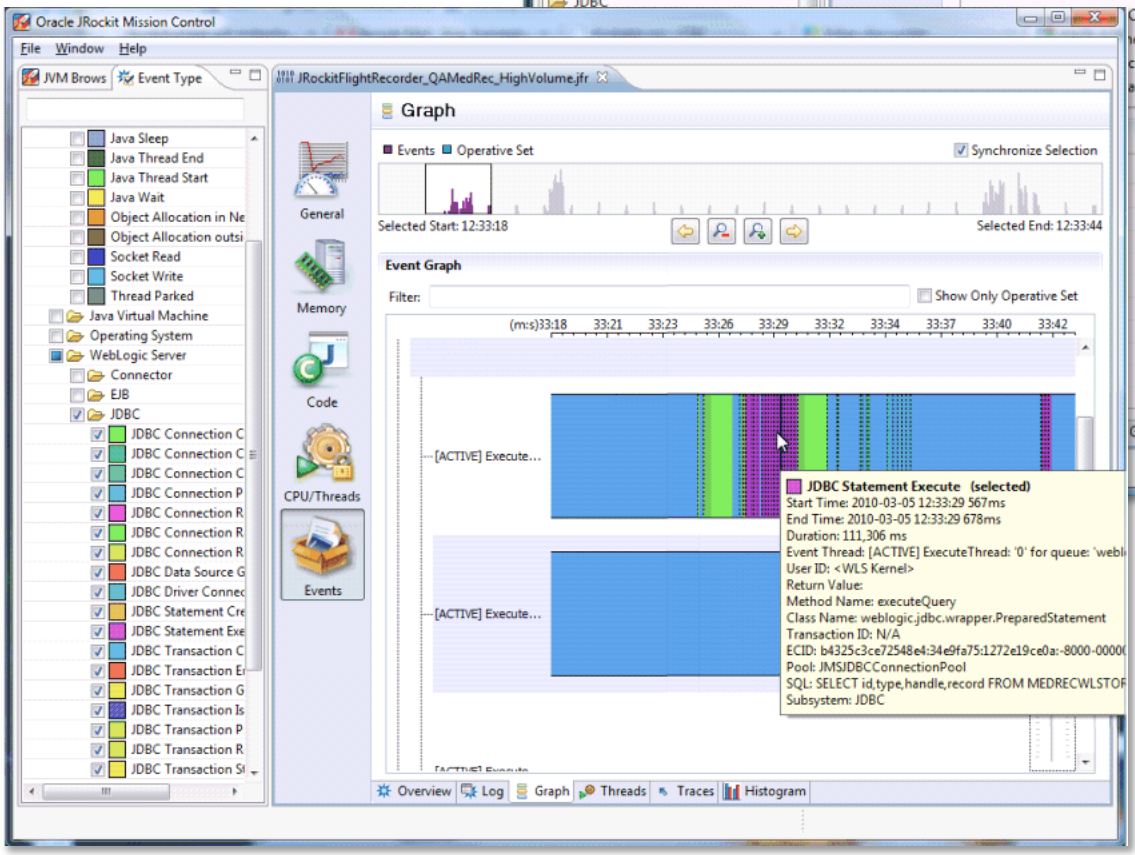
分類	取得項目	JRockitでの指定					WLSでの指定			
		組込	通常	ロック含む	例外含む	カスタム指定	Off	Low	Medium	High
OS	コンテキスト・スイッチ	○	○	○	○			○	○	○
	CPUビジー	○	○	○	○			○	○	○
	物理メモリ使用状況	○	○	○	○			○	○	○
	環境変数の情報	○	○	○	○			○	○	○
	自JVM以外のプロセスの情報					○				
Java VM	オブジェクト・アロケーション	○	◎	◎	◎			○	○	○
	ガベージ・コレクション	○	◎	◎	◎			○	○	○
	空きメモリー	○	◎	◎	◎			○	○	○
	Nativeコード最適化	○	○	○	○			○	○	○
	自JVMのCPUビジー	○	○	○	○			○	○	○
	システム・プロパティ	○	○	○	○			○	○	○
	クラス・ロード		○	○	○					
	JITコンパイル		○	○	○					
	クラス別ヒープ利用状況		○	○	○					
Java App	ファイル/ソケットの読み書き	○	○	○	○			○	○	○
	ロック獲得/待機、イベント待機	○	○	◎	○			○	○	○
	例外				○					
WebLogic	Connector							○	○	○
	Servlet							○	◎	◎+
	EJB							○	◎	◎+
	JDBC							○	◎	◎+
	Web Service							○	○	○
	JTA									○

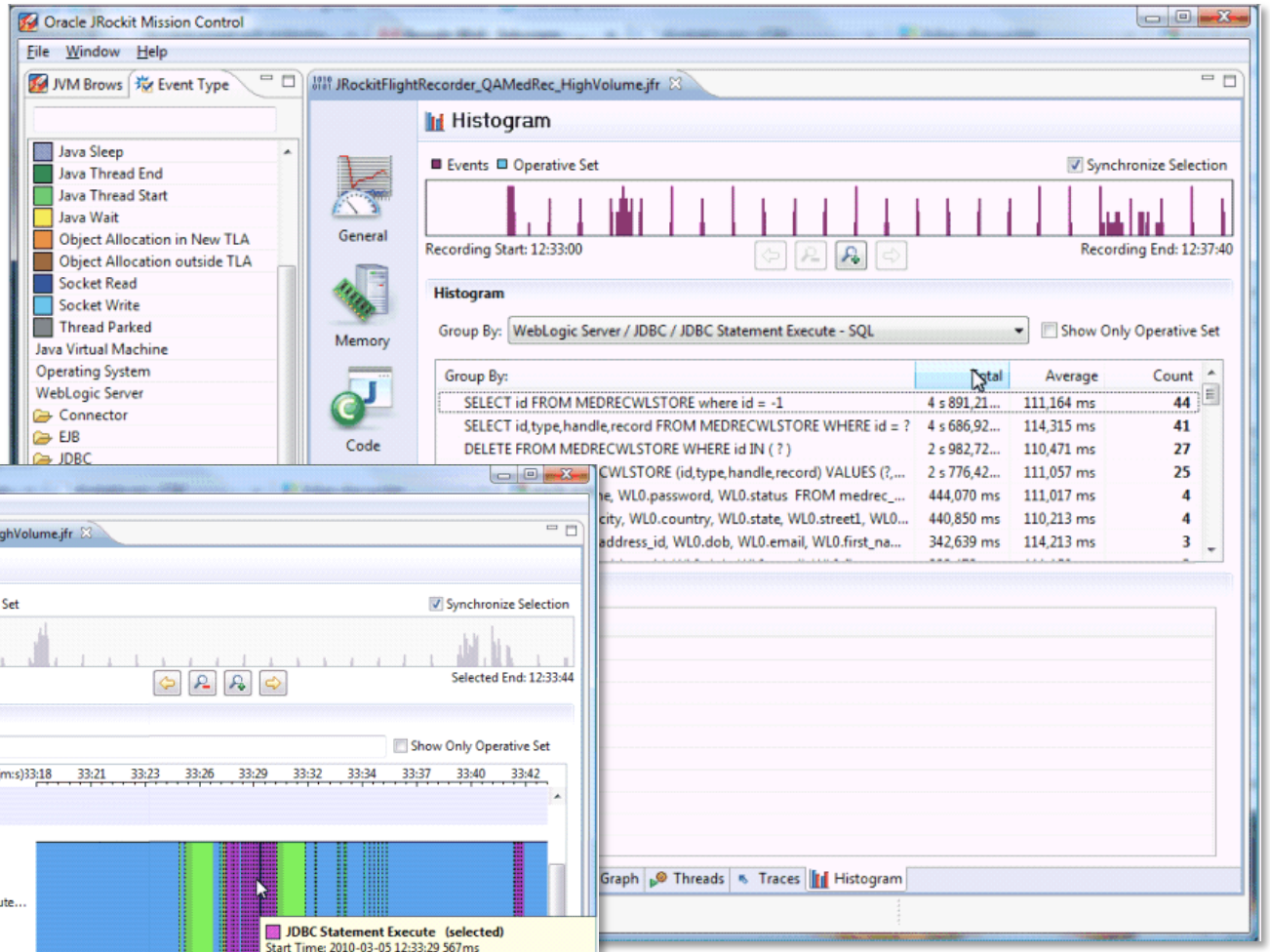
WebLogic イベントの可視化



例

■ JDBCコール部分の分析





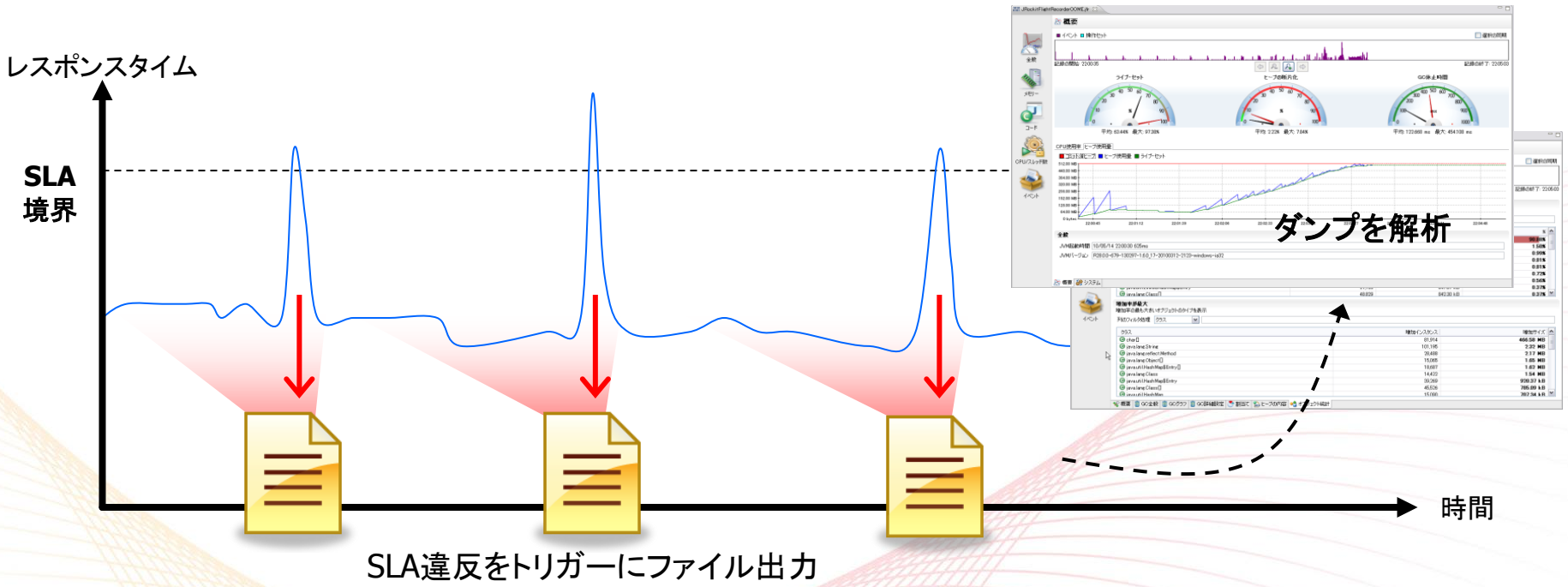
■ 実際にコールされたSQLの識別

ダンプのタイミングの制御

適用例

- 通常時: 常時記録(一定時間分をメモリ上で循環記録)
- SLA違反時に自動的にファイルにダンプ
  - ✓ エラーや例外発生
  - ✓ 性能劣化(レスポンス低下)

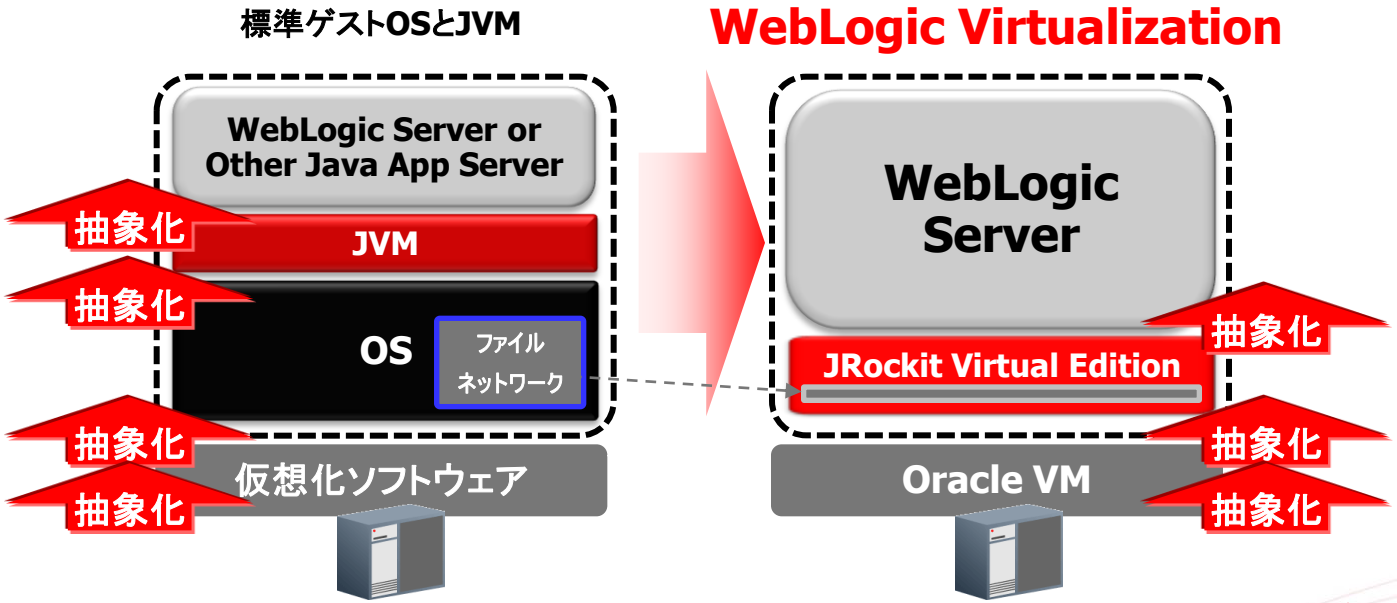
WebLogic 診断フレームワークで設定可能



Java/JVMの方向性
- Virtualization

JRockit Virtual Edition

HW利用の効率性を高める仮想化環境向けソリューション



仮想化環境での処理性能	通常OS環境より 20-40%のオーバーヘッド		チューンされた環境より 20%程度高速
JVM層までの必要容量	1GB以上		50MB程度
JVM層までの構成ファイル数	膨大なOS設定	V.S.	1つ
OS保守（パッチ、入れ替え）	必要		不要
OSセキュリティ・ホール/不正ログイン	可能性あり		なし

高速

シンプル

セキュア

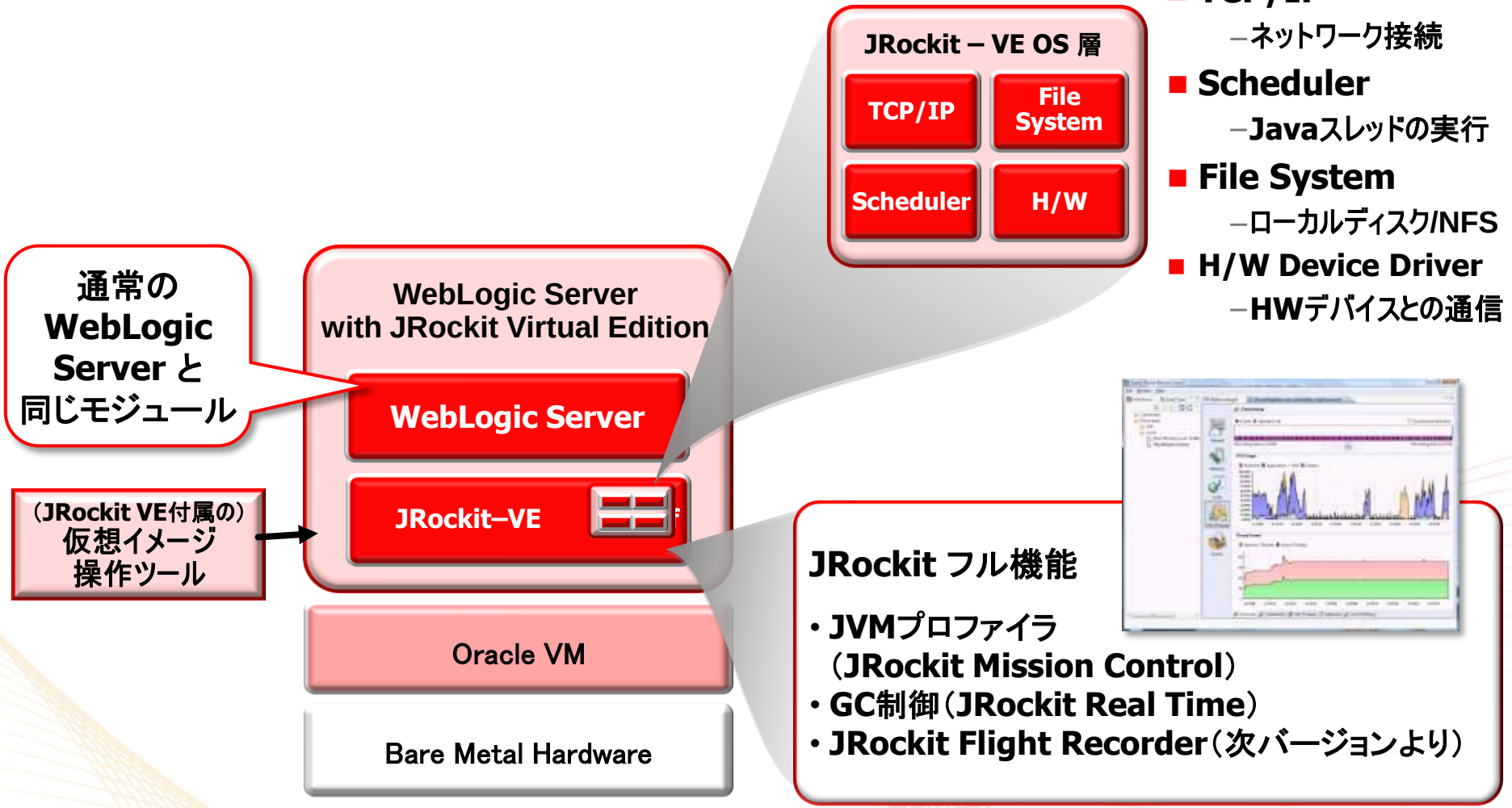
WebLogic Virtualization

仮想化環境におけるパフォーマンス

WebLogic Server 11*g* R1 PatchSet 1、CPUクロック数 2.8GHz

構成	物理環境	on JRockit Virtual Edition	仮想化環境（ゲストOS）
搭載メモリ容量	24GB	4GB	6GB
OS	OEL5.4	OVM/JRVE	OVM/OEL5.4 PV
JRockit バージョン	R27.6.2-20	R27.6.3-40	R27.6.2-20
JVMヒープサイズ（GB）	3.6	3.3	3.6
秒間のオペレーション数	300	299	225

Oracle WebLogic Server with JRockit Virtual Edition



- **TCP/IP**
  - ネットワーク接続
- **Scheduler**
  - Javaスレッドの実行
- **File System**
  - ローカルディスク/NFS
- **H/W Device Driver**
  - HWデバイスとの通信

まとめ

Java 実行環境のさらなる発展へ

- 開発/保守時
  - 開発生産性/保守性向上、スクリプト言語対応
  - プロファイラ機能強化
- 実行時
  - 最適化/モジュール化
  - 最新ハードウェア対応（マルチコア、大容量メモリ、InfiniBand）
  - サーバー仮想化への対応
- 監視・障害対応
  - 常時監視・常時記録

おまけ

- **JRockit は WebLogic Server や Coherence とではなく、単独でも利用できるライセンスがあります**
  - 詳しくは日本オラクルまでお問い合わせください

- **「WebLogic Server 11g 構築・運用ガイド」でます!** →

2010年12月1日 刊行予定

Oracle Exalogic Elastic Cloud

Oracle Exalogic Elastic Cloud

- 100万 Webリクエスト/秒
- 180万 メッセージ送信/秒
- 200万 SQLオペレーション/秒

超高速 Java プラットフォーム

10倍

までレスポンス改善

ミッションクリティカル クラウド

集約化→運用コストを

60%

削減

事前統合された システム

展開までの時間を

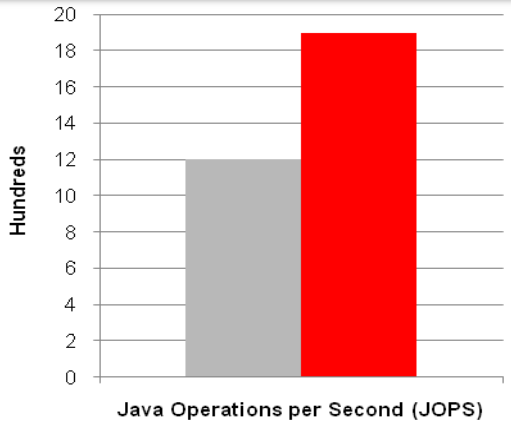
90%

削減

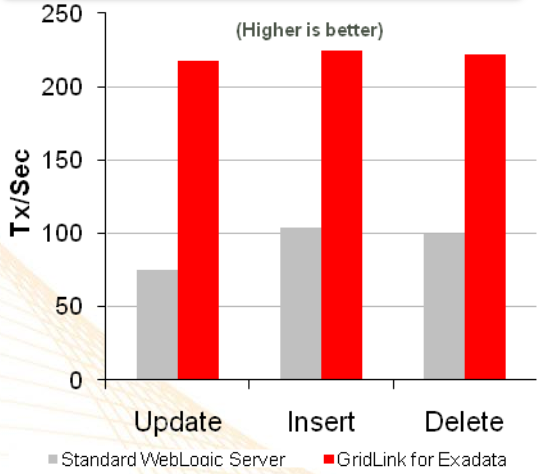
超高速Javaプラットフォーム

従来型との比較

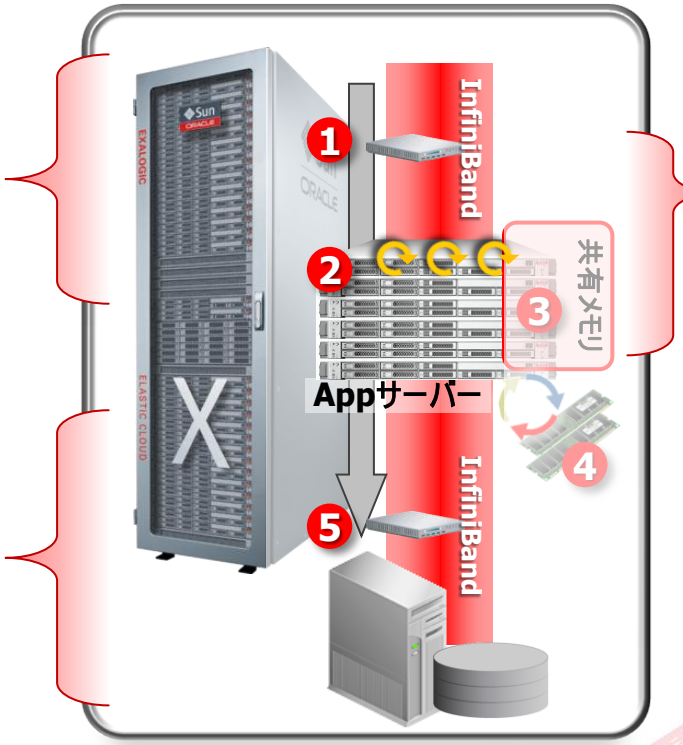
Javaオペレーション：60% 高速



DBトランザクション：2-3倍高速

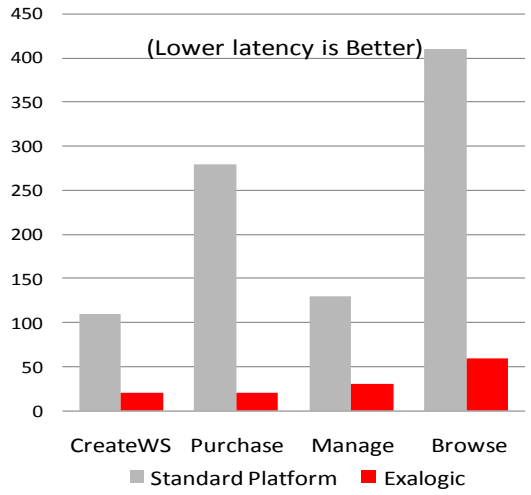


- InfiniBand
- IO/Protocol Performance Accelerator
  - Java NIOによるIO処理最適化
  - InfiniBand向けチューニング



- 動的なDBノード・バランシング
- JDBC over SDP + InfiniBand

対話型処理：3-10倍高速



比較対象
同スペックCPUのIAサーバー
＋ 1GbE
＋ 通常版ソフトウェア
【Linux、WebLogic、JRockit、JDBC】
（Exalogic Elastic Cloud SW 未適用、
Coherence および Flash SSD 未使用）

したがって、③および④の効果は含まれない

ミッションクリティカル・クラウド

高密度なアプリケーションの統合/集約基盤

オープンスタンダード基盤

- 様々なアプリケーションを展開可能

ネットワーク・パーティショニング

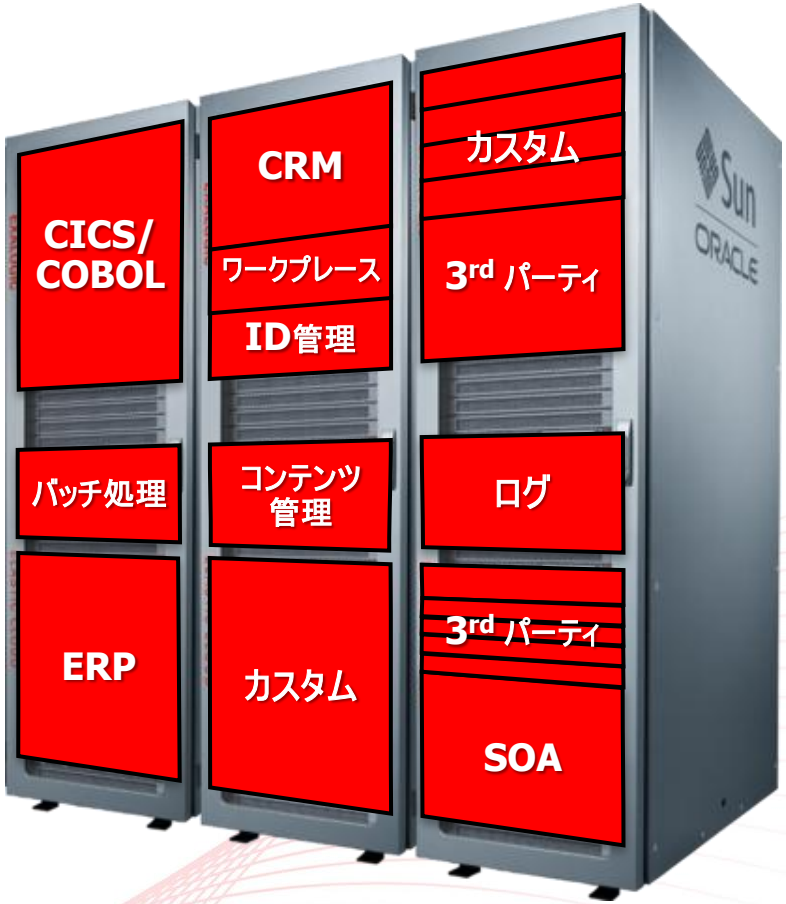
- 可用性やセキュリティ要件に応じて
  パーティション化

リソース割り当て制御

- 性能要件に応じてCPU/メモリの占有化や
  共有リソース化が可能

全体運用の最適化

- 基盤メンテナンスを集約・標準化
- 管理・監視作業を集約化



ORACLE

スペシャル・サイトのご案内

■ アプリケーション・グリッド / Coherence

http://www.oracle.co.jp/appgrid/

- 業種や課題別の解決策のご紹介
- 効果をわかりやすくアニメーションで紹介
- 事例動画（字幕付き）
- その他、関連ライブラリ

■ Oracle WebLogic Server マニアックス

http://www.oracle.co.jp/weblogic/

【3つの迷信】

【伝説のエキスパートが語る】

【ライブラリ】他

■ CIO for Tomorrow

http://www.oracle.co.jp/campaign/cio/

【連載コラム】
- CIOのミッション
- アナリストの提言

【ソリューション】他


ORACLE®