

ORACLE®
Java/JVM最前線

日本オラクル株式会社 Fusion Middleware 事業統括本部
2010年 10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。

Oracle、PeopleSoft、JD Edwards、及びSiebelは、米国オラクル・コーポレーション及びその子会社、関連会社の登録商標です。その他の名称はそれぞれの会社の商標の可能性があります。

ORACLE®

本日の内容

- **JavaOne および Announcement サマリー**
- **Java/JVM の方向性**
- **Exalogic Elastic Cloud**

JavaOne 2010 @ SF, 9/19-23

- **One City, One Week, Three Conferences**

- Oracle OpenWorld
- JavaOne
- Oracle Develop



ORACLE®

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>

ORACLE®

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>

ORACLE®

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

ORACLE®

Java/JVMの方向性

- Java Core



ORACLE®

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

ORACLE®

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 }
```

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 ]
```

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 ]
```

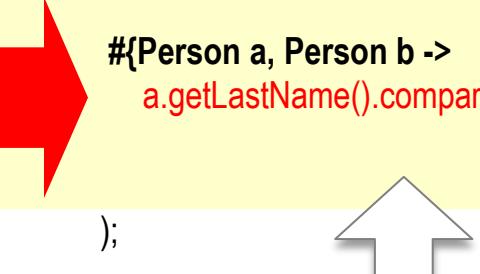
- 生産性が高いスクリプト言語の良い部分を採用
- 同時に、Javaとの親和性向上にも寄与

Project Lambda

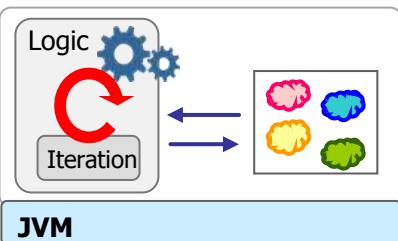
直接の目的は、単一メソッドの内部クラスの記述の簡素化

```
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()) }  
);
```

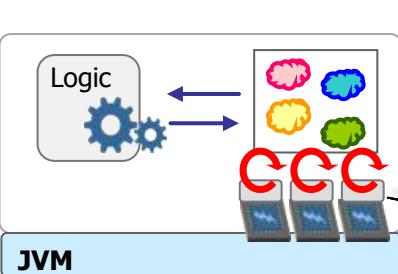
- 
- ① Collections.sort の定義から Comparator<? super T> の表現であることを判別
 - ② Comparator の単一メソッドである compare の実装部分として利用

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();
```

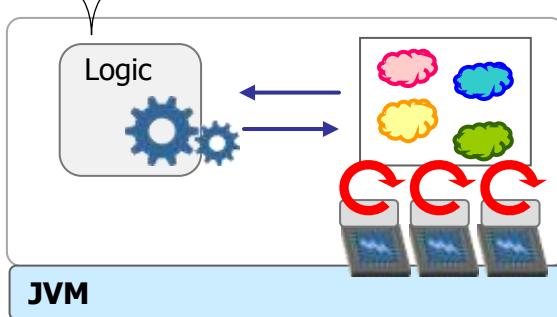
JVMの実装次第で
マルチコア活用型
パラレル処理を容易に
利用可能になる

ORACLE®

Oracle Coherence によるパラレル処理

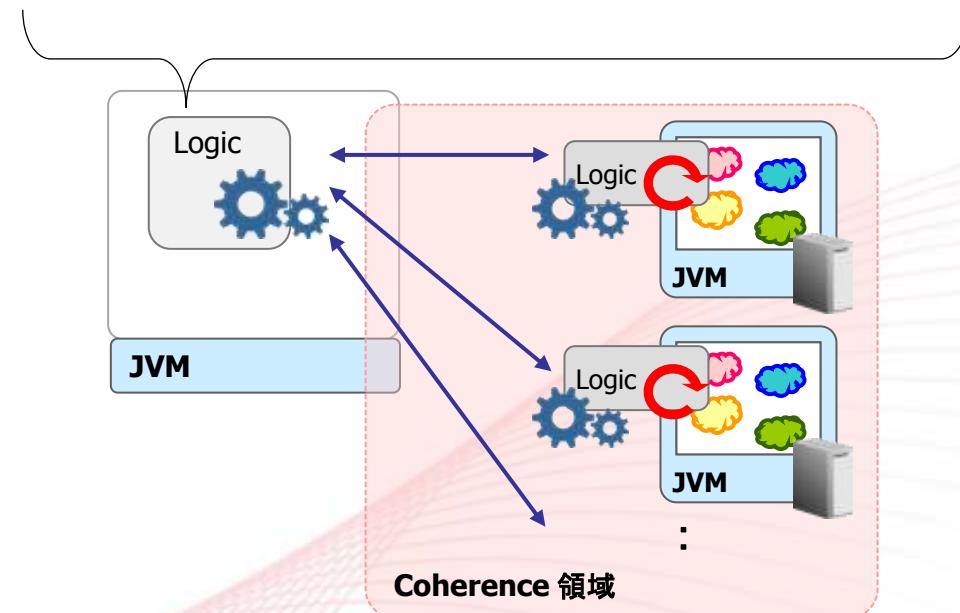
内部イテレーションによるパラレル処理 - マルチコアの活用

```
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" )
  );
```

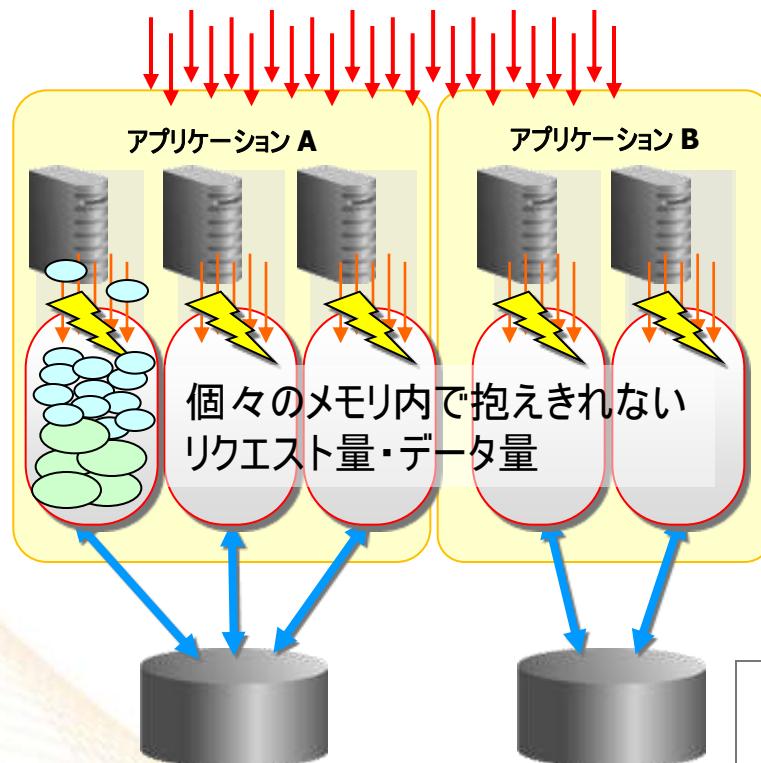


Oracle Coherence

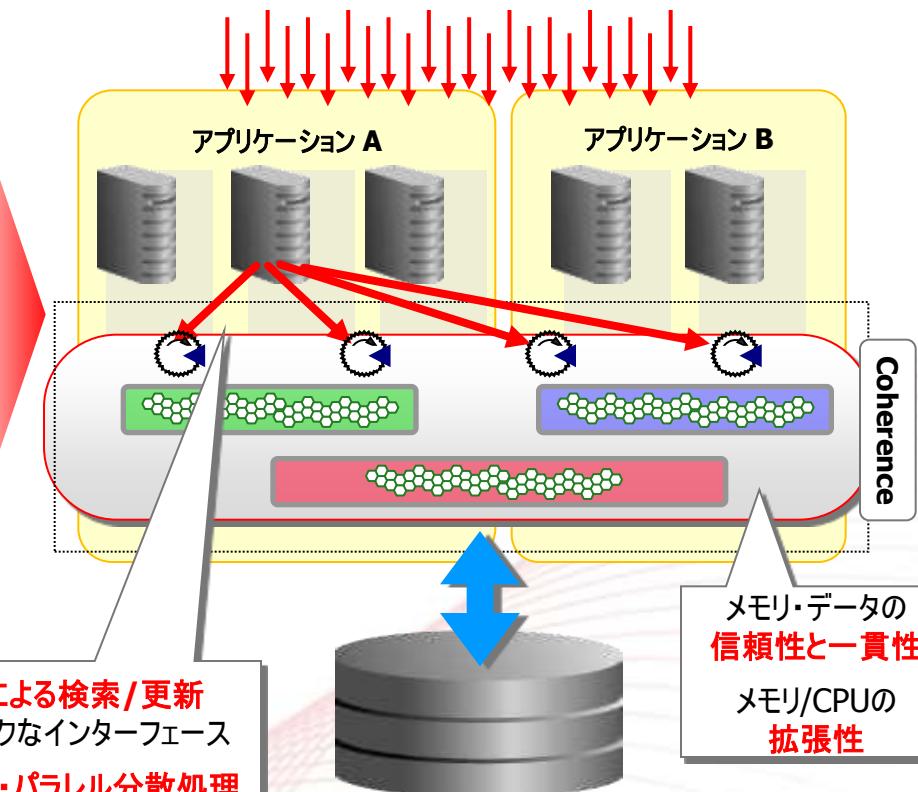
情報爆発の時代に応える企業向けインメモリ分散キーバリュー・ストア(**KVS**)

- HW横断型の分散共有メモリ領域
- メモリ・データの信頼性向上
- インメモリ・イベント処理基盤

- スケールアウト型でHWコストを最適化
- ディスクIOの最小化(高速化)
- サブシステム間連携を最適化(高速化・効率化)



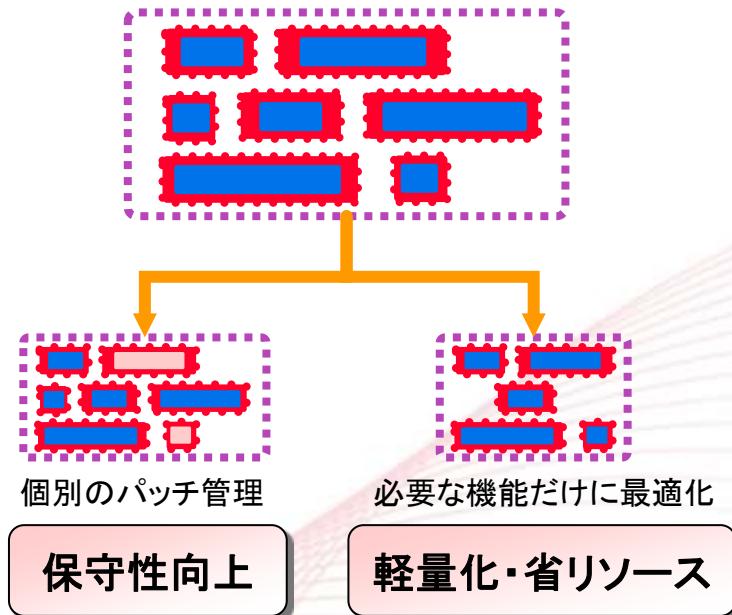
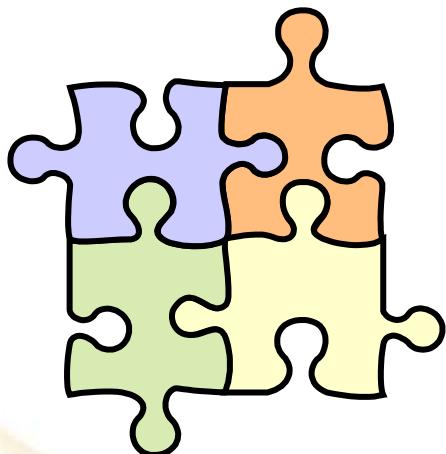
属性による検索/更新
SQLライクなインターフェース
インメモリ・パラレル分散処理



ORACLE®

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



ORACLE®

Java/JVMの方向性

- Flight Recording



ORACLE®

JRockit

Java SE 完全準拠の高機能 JVM

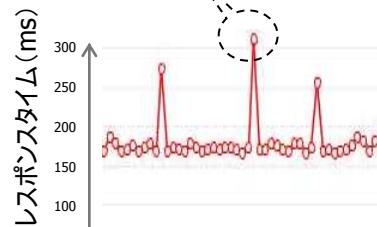
- **高速**：サーバーサイド用途に最適化。公式ベンチマークで実証済み。
- **安定**：GCによる遅延時間の平準化。ミリ秒単位の応答時間の維持を実現。
- **全てを記録**：稼動状況を常に記録。予測外の事態でも過去に遡って分析可能。

#1 パフォーマンス(高速)

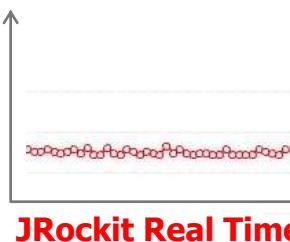
公式ベンチマーク(SPECjbb)

- ・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)



過去に遡って
トラブル解析

性能を犠牲にすることなく
挙動を記録



時間の経過

Java アプリ



WebLogic
Server

Oracle
JRockit

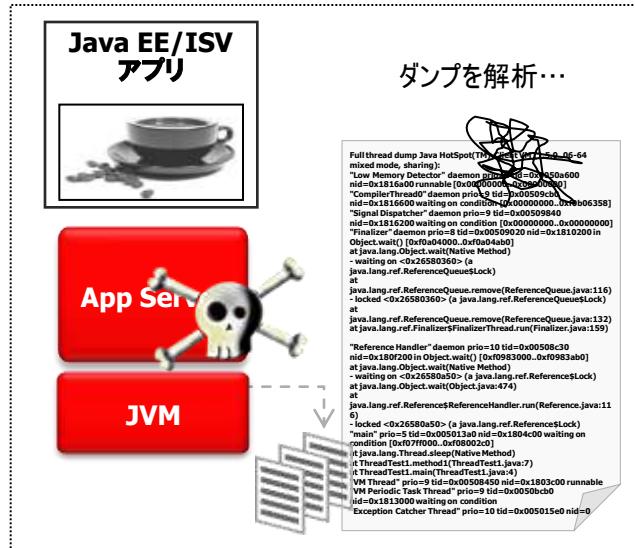
自動記録

自動ファイル出力

挙動を常時記録
"録りっぱなし"

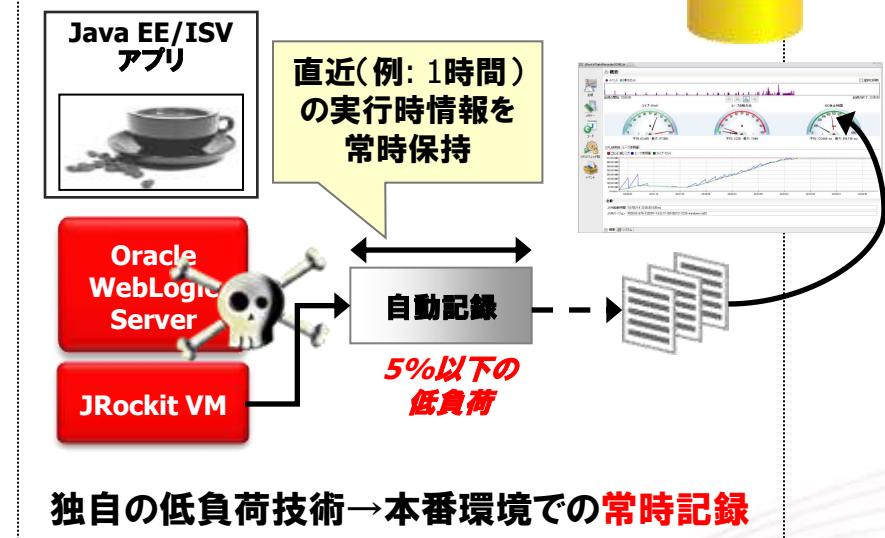
JRockit Flight Recorder

• 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



- ✓ 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

ORACLE®

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

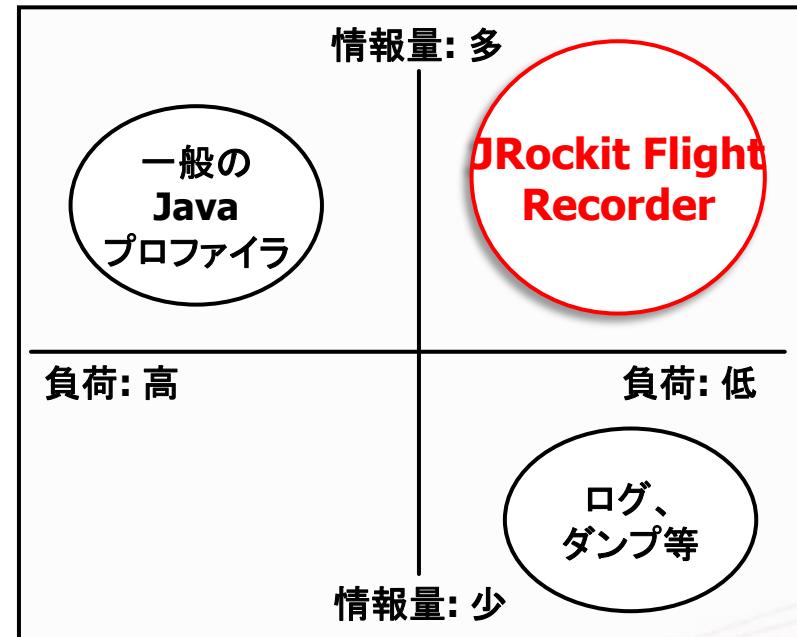
- JRockitの内部機構を利用した
本番環境を邪魔しない低負荷記録

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

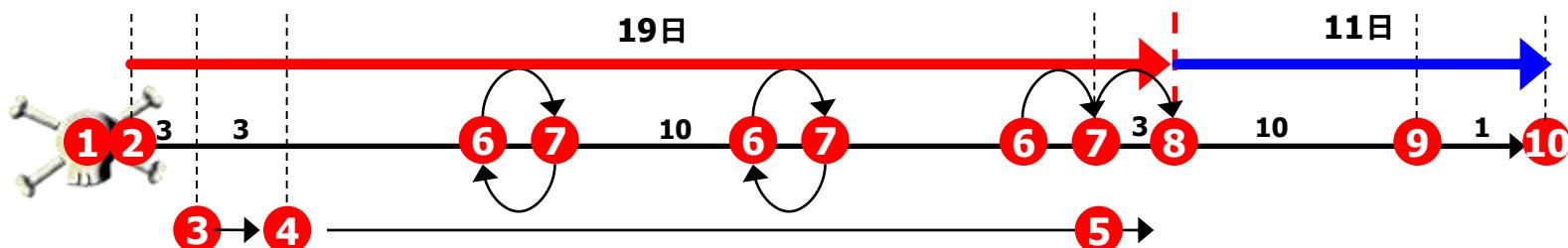
対象: WebLogic Server上のWebアプリ

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

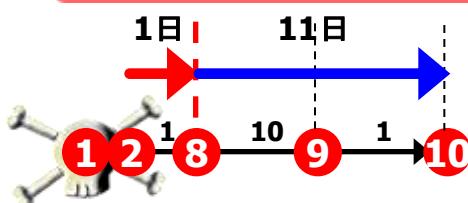
これまでのJava

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)



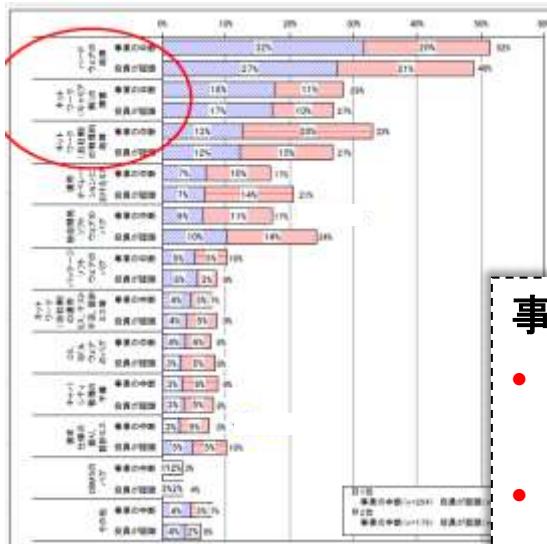
迅速かつ確実な問題解決

期間短縮
の理由

再現作業不要
再発待ち不要

- 問題発生、対処のため再起動
 - 解析開始
 - 再現環境の準備開始
 - 再現環境にて再現を試行
 - 再現せず
 - 本番環境で再発
 - 情報収集した後再起動
- 原因判明
 - 対処、テスト
 - 本番に適用
- JRockit Flight Recorderを利用することで省略できる作業

JRockit Flight Recorder が有効な範囲



社団法人日本情報システム・ユーザー協会

「第16回企業IT動向調査2010」
-「障害の原因」に関する調査より、
「事業の中断」の要因の一位および二位として
あげられた項目の割合の合計値の平均から算出

事業中断に至った障害の種別内訳

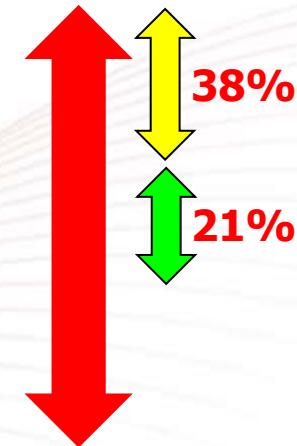
- ハードウェア(**HW**)障害(26%)
 - **HW**の故障
 - ネットワーク(**NW**)障害(34.5%)
 - **NW**(キャリア側)の障害、**NW**(自社側)の物理的故障、**NW**(自社側)の設定/運用ミス

HW障害、NW障害を
除く範囲に限定した場合

- ソフトウェア障害(15%)
 - パッケージソフト、OS、ミドルウェア、DBMS、の不具合
 - キャパシティ管理等設定ミス
 - アプリケーション障害(8.5%)
 - 独自開発ソフトウェアの不具合
 - 人的ミス(12.5%)
 - 運用オペミス、要件/仕様ミス
 - その他(3.5%)

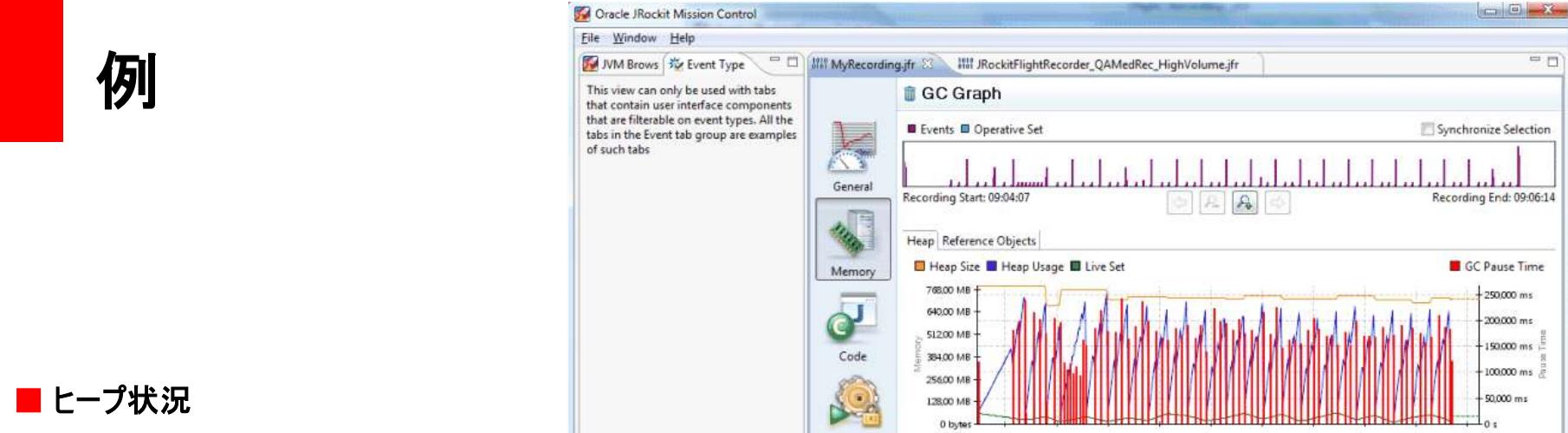
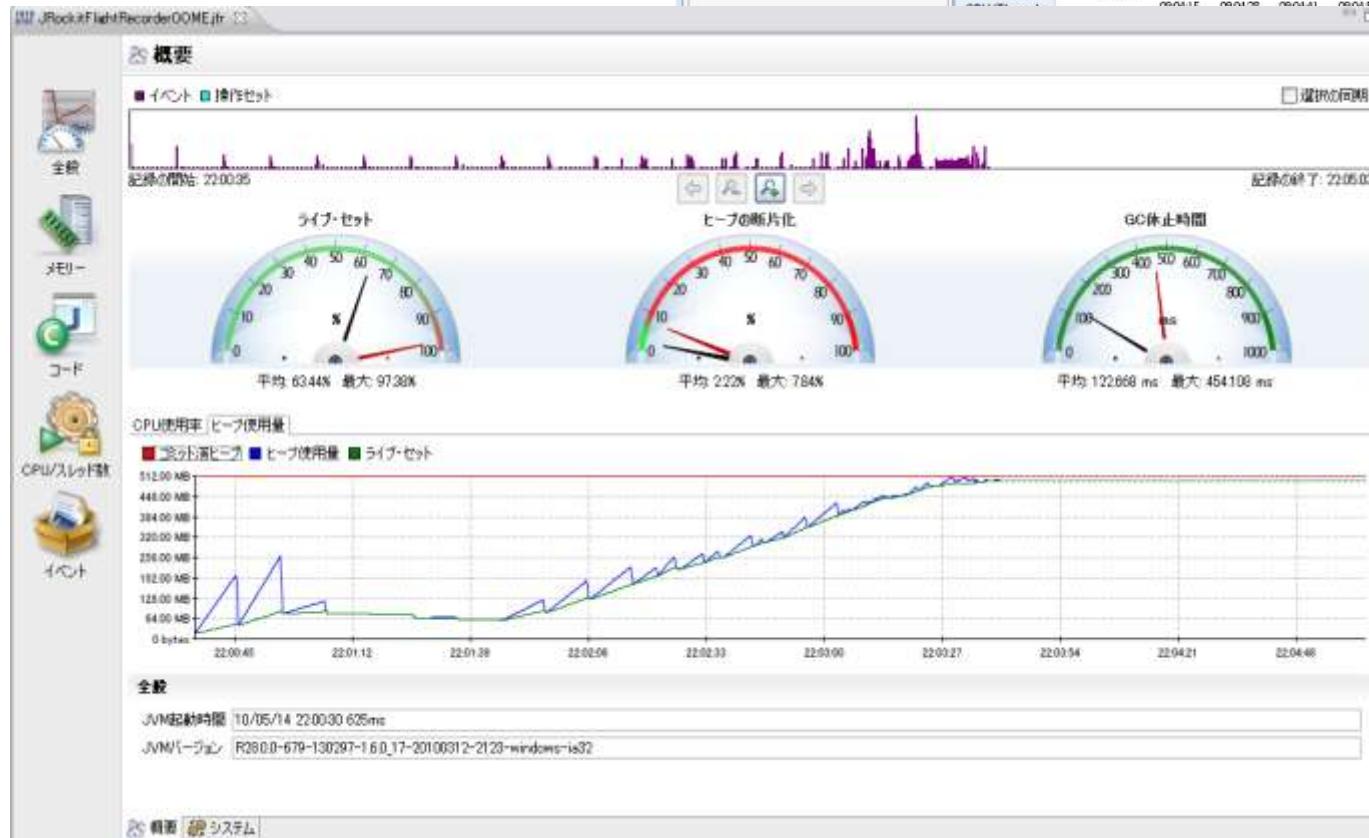
Flight Recorderが 障害原因の切り分けに 有効だと考えられる範囲

Flight Recorderが 障害原因の切り分け+ 原因解決まで有効だと 考えられる範囲



例

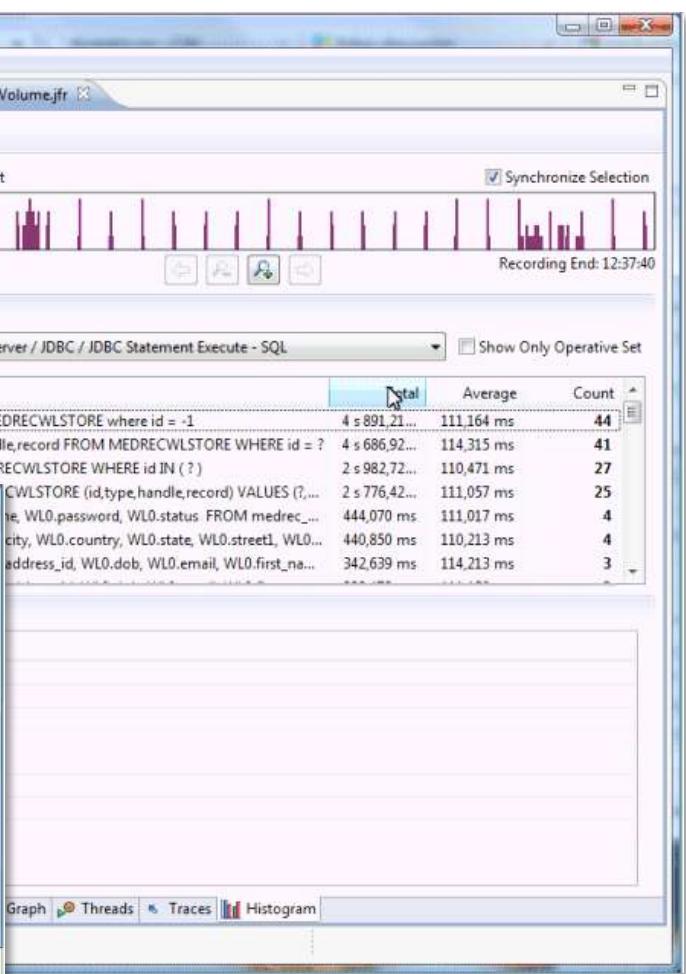
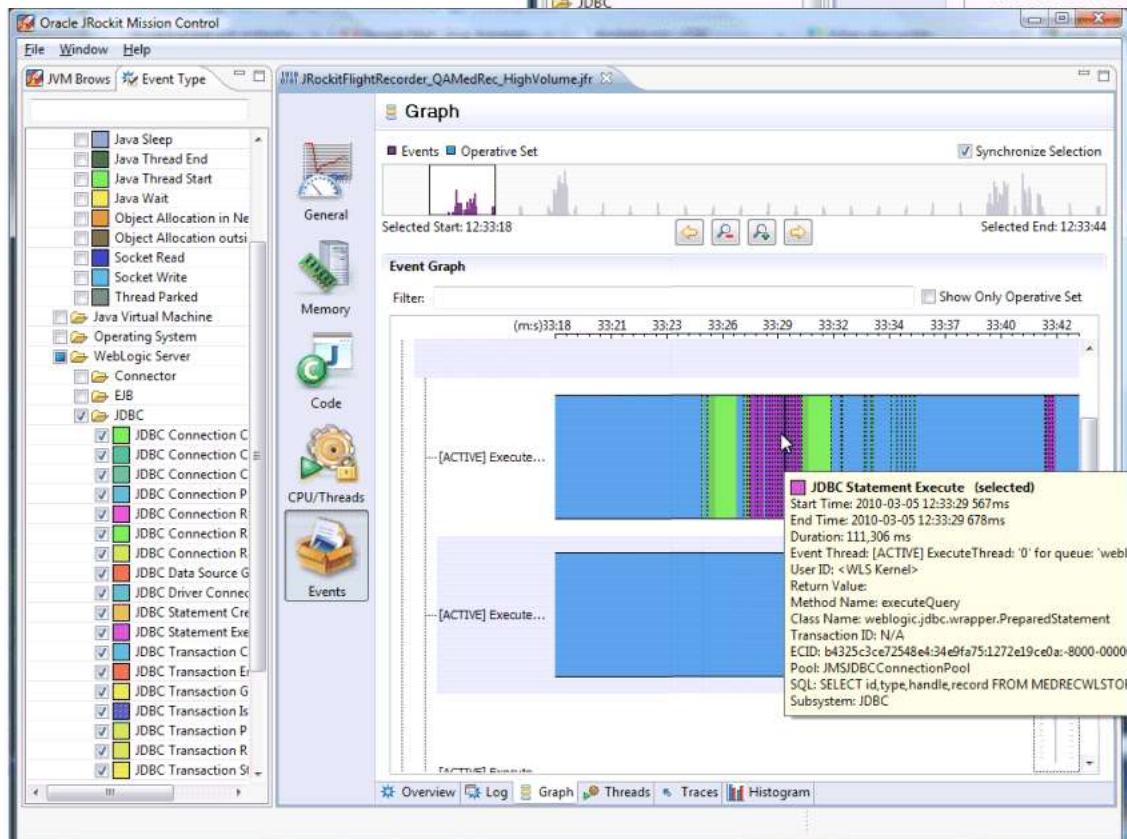
■ ヒープ状況



■ GC状況の可視化

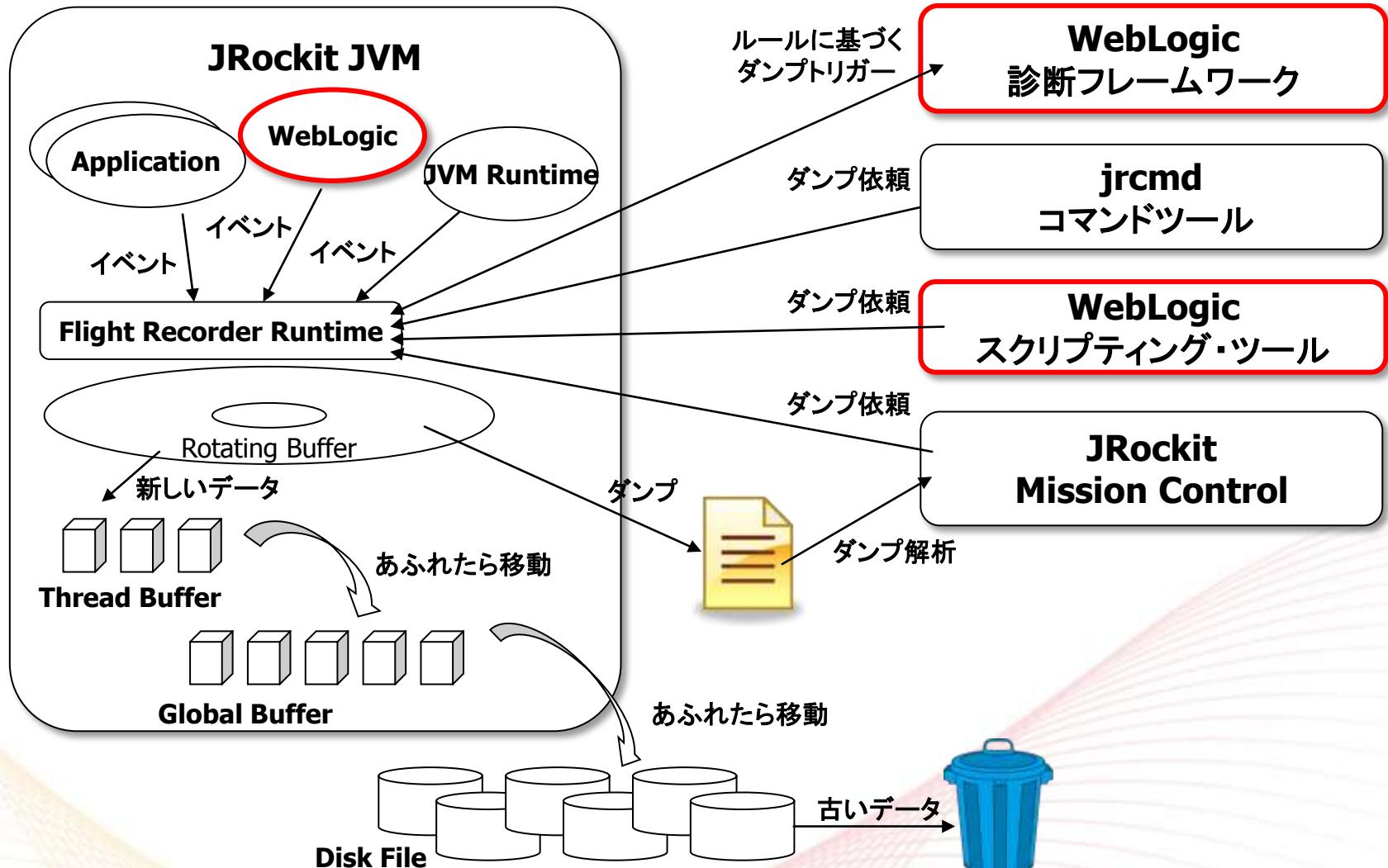
例

■ 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



ORACLE®

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

The screenshot shows the WebLogic Management Console interface. On the left, the navigation tree is expanded to show the 'サーバー' (Server) node, which is highlighted with a red box. A red arrow points from this node to the 'AdminServer(管理サーバー)' entry in the list of servers on the right, also highlighted with a red box. The 'AdminServer(管理サーバー)' entry is selected, and its details are displayed on the right side of the screen. The '診断ボリューム' (Diagnostic Volume) dropdown menu is open, showing the '中' (Medium) option selected, also highlighted with a red box. Other configuration options visible include listening port 7001, SSL listening port 7002, and Java compiler javac.

ドメイン構造

jfr_base_domain

- 環境
- サーバー
- クライアント
- 仮想ホスト
- 移行可能なターゲット
- マシン
- ワーク・マネージャ
- 起動クラスと停止クラス
- デプロイメント
- サービス
- セキュリティ・ルーム
- 相互運用性
- 診断

操作ガイド

このページでは、現在のWebLogic Serverドメイン内に

この表のカスタマイズ

サーバー (フィルタ処理されました - 列がさらに存在)

このページのすべてのボタンをアクティブ化するには、

新規 クローンの作成 削除

名前
AdminServer(管理サーバー)

新規 クローンの作成 削除

リスニング・ポート: 7001

SSLリスニング・ポートの有効化

SSLリスニング・ポート: 7002

クライアント証明書プロキシの有効化

Javaコンパイラ: javac

診断ボリューム: 中

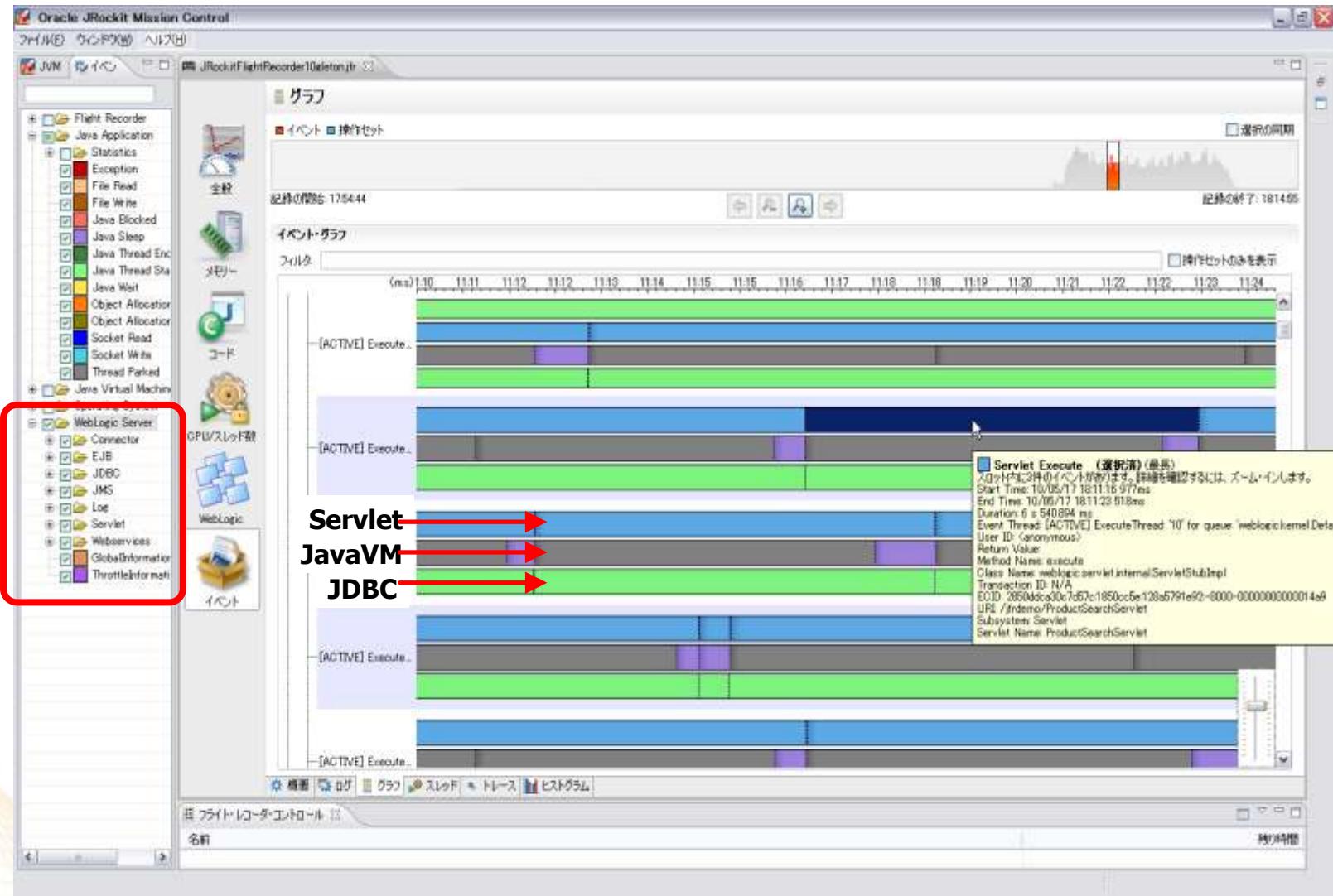
ORACLE®

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	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	CPUビジー	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	物理メモリ使用状況	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	環境変数の情報	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	自JVM以外のプロセスの情報					<input type="radio"/>				
Java VM	オブジェクト・アロケーション	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	ガベージ・コレクション	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	空きメモリー	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Nativeコード最適化	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	自JVMのCPUビジー	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	システム・プロパティ	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	クラス・ロード		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>					
	JITコンパイル		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>					
	クラス別ヒープ利用状況		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>					
Java App	ファイル/ソケットの読み書き	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	ロック獲得/待機、イベント待機	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>		<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	例外				<input type="radio"/>					
WebLogic	Connector						<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Servlet						<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	EJB						<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	JDBC						<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Web Service						<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	JTA									<input type="radio"/>

ORACLE®

WebLogic イベントの可視化



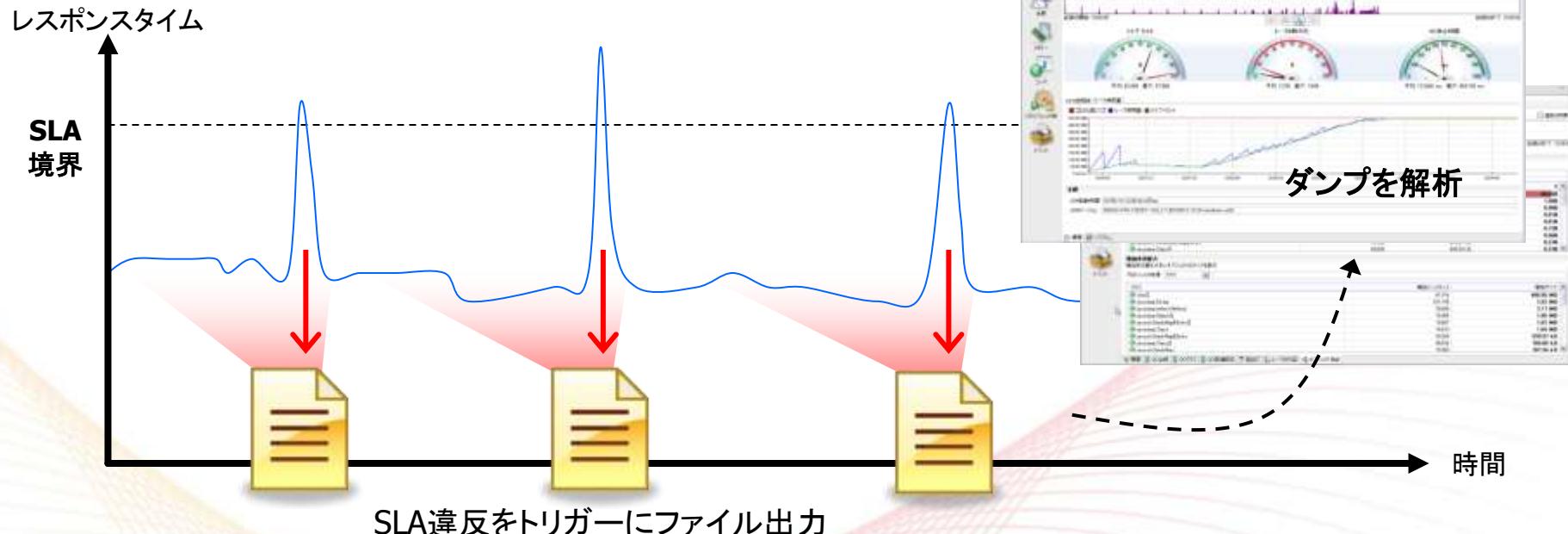
ORACLE®

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

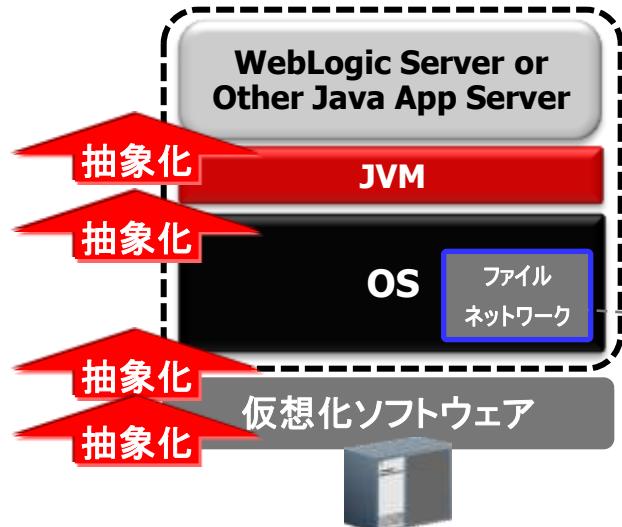


ORACLE®

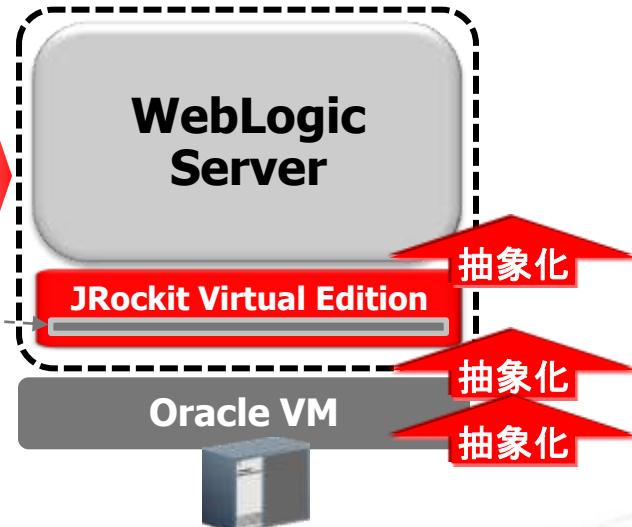
JRockit Virtual Edition

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

標準ゲストOSとJVM



WebLogic Virtualization



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
JVM層までの必要容量	1GB以上
JVM層までの構成ファイル数	膨大なOS設定
OS保守(パッチ、入れ替え)	必要
OSセキュリティ・ホール/不正ログイン	可能性あり

V.S.

チューンされた環境より 20%程度高速	高速
50MB程度	シンプル
1つ	セキュア
不要	
なし	

ORACLE®

WebLogic Virtualization

仮想化環境におけるパフォーマンス

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

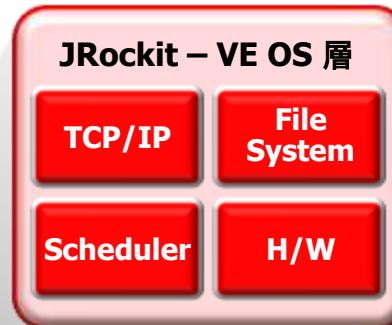
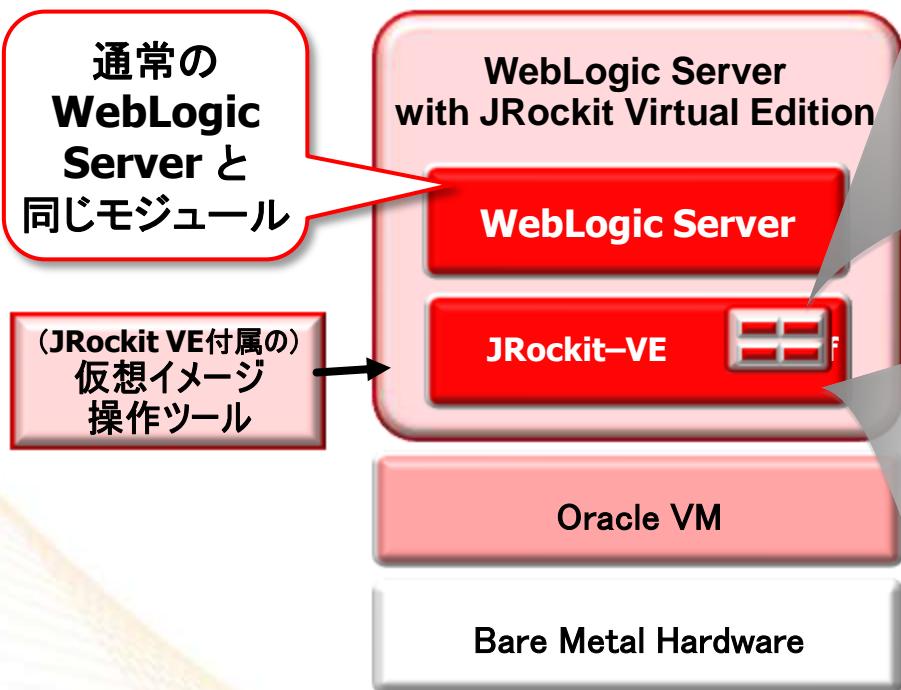
構成	物理環境	on JRockit Virtual Edition	仮想化環境 (ゲストOS)
搭載メモリ容量	24GB	4GB	6GB
OS	OEL5.4	OVM/JRVE	OVM/OEL5.4 PV
JRockit バージョン	R27.6.2-20	R27.6.3-40	R27.6.2-20
JVMヒープサイズ (GB)	3.6	3.3	3.6
秒間のオペレーション数	300	299	225



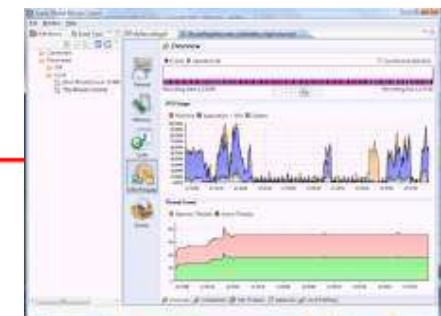
通常の
仮想化環境より
32% 性能向上

ORACLE®

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



- JRockit フル機能**
- JVMプロファイラ
(JRockit Mission Control)
 - GC制御(JRockit Real Time)
 - JRockit Flight Recorder(次バージョンより)

ORACLE®

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

Oracle Exalogic Elastic Cloud



ORACLE®

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

ORACLE®

Oracle Elastic Cloud Foundation

Application Grid, OS, Enterprise Manager + ハードウェア

Enterprise Manager

WebLogic Server
No.1 パフォーマンスAppサーバー

Coherence
No.1 データグリッド

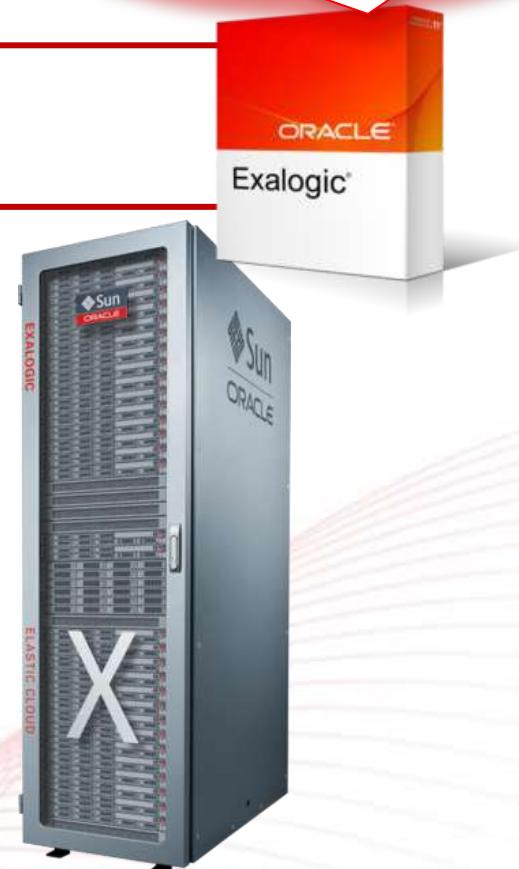
JRockit / HotSpot
高速JVM + GC制御 + Flight Recording

Oracle Linux / Solaris
ネットワーク処理最適化

Exalogic Elastic Cloud Hardware

InfiniBand Fabric	40Gb/s
2.93 GHz Cores	360
1333 MHz RAM	2.8 TB
FlashFire SSD	960 GB
SAS Disk Storage	40TB

最適化
高密度
オープン
スタンダード
※ X2-2 Full Rack



ORACLE®

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



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®