

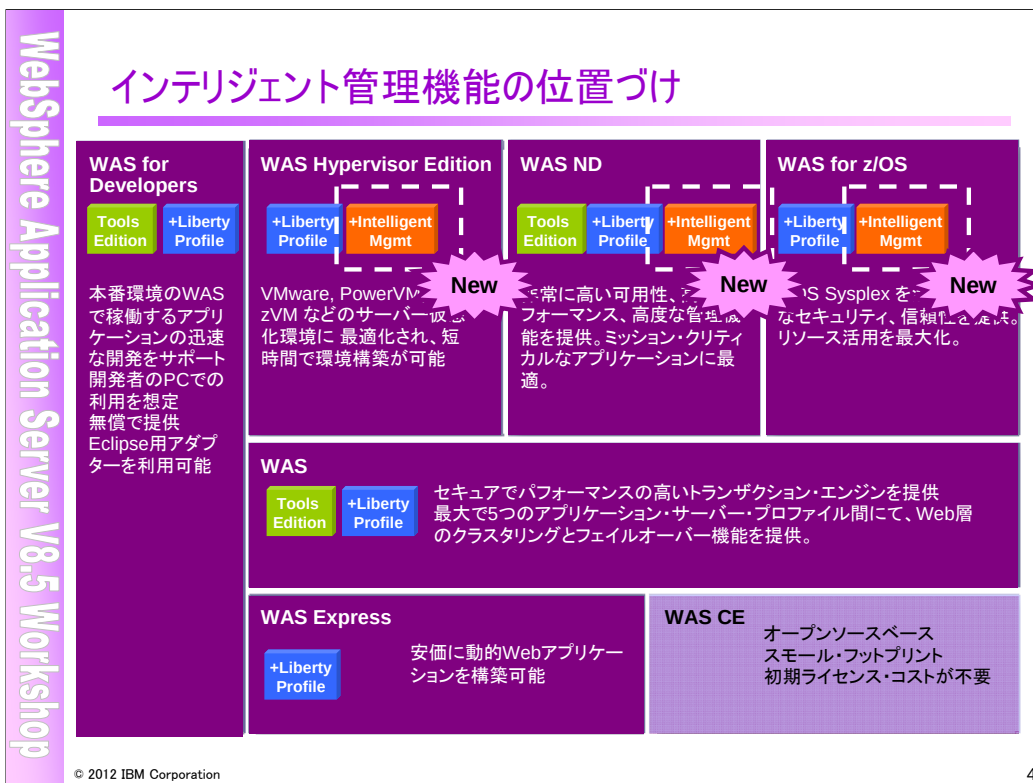
インテリジェント管理(1)

日本アイ・ビー・エム システムズ・エンジニアリング株式会社
Webプラットフォーム
鈴木 淳

Agenda

1. インテリジェント管理の概要
 2. インテリジェント管理トポロジー
 3. インテリジェント・ルーティングとパフォーマンス管理
 4. 可視化機能
 5. まとめ
- 参考文献

1. インテリジェント管理の概要



このセッションと次のセッションの2セッションに分けて、WAS V8.5のもう1つ重要な新機能である「インテリジェント管理」をご紹介します。

WASの上位製品に当たるWebSphere Virtual Enterpriseの機能が「インテリジェント管理」という名前でWAS V8.5に統合されるようになりました。

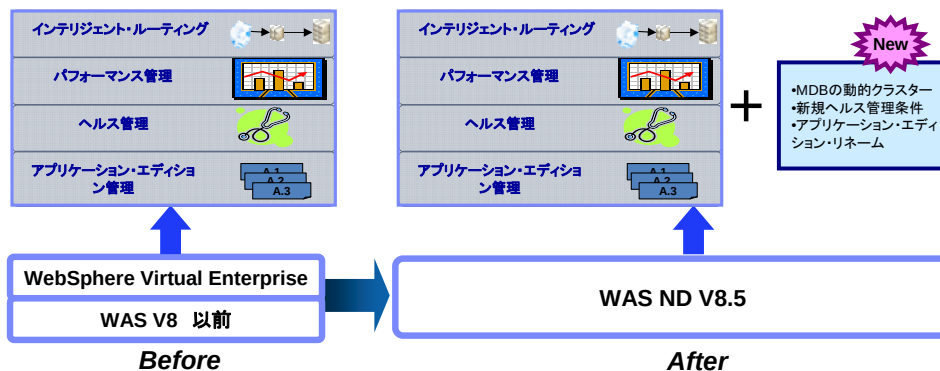
WASのND版を導入すれば、特に追加インストールや、追加ライセンス不要で、インテリジェント管理機能をデフォルトで 사용할 ことができます。

Hypervisor Edition、WAS for z/OSの各パッケージングについても同様です。

インテリジェント管理機能を特に利用せず従来のND版の機能のみ利用し単純にバージョンアップするだけの場合、この機能をオフにすることも可能です。

インテリジェント管理機能の変遷

- WebSphere XD→WebSphere Virtual Enterprise→WAS V8.5
 - ◆ インテリジェント管理はこれまで別製品のWebSphere Virtual Enterpriseで提供
 - ◆ WAS V8.5では一部新規機能を追加
 - ◆ WVEに比べパフォーマンスが向上
 - ◆ WAS ND (Network Deploymentエディション) で利用可能
 - ◆ インテリジェント管理機能の一部を無効化することも可能



© 2012 IBM Corporation

5

インテリジェント管理機能はこれまでWASの上位製品にあたるWebSphere Virtual Enterpriseで(それより以前はWebSphere Application XD版で)提供していました。

WAS V8.5では、従来Virtual Enterpriseのインテリジェント・ルーティング、パフォーマンス管理、ヘルス管理、アプリケーション・エディション管理の4つの大きな機能に加えて、MDBの動的クラスターや新規ヘルス管理条件、アプリケーション・リネームなど一部新規機能を追加されました。

さらに、WebSphere Virtual Enterpriseに比べて、インテリジェント管理関連コンポーネントの起動時間の短縮、スループットの増加、CPUアイドルタイムの短縮などのチューニングも施されます。

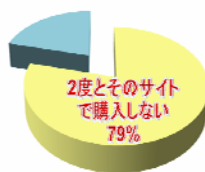
このインテリジェント機能はWASのND版から使用可能で、Base版には含まれていないことが注意点になります。

インテリジェント管理が求められる背景～その1

- 性能要件～サービス・レベルの維持(レスポンス、スループット)
- スケーラビリティ ～ 急増するトランザクションへの対応

トランザクションの急変動や障害が発生しても安定した性能

Webサイトのパフォーマンスに不満を持った買い物客の取る行動



出典: Forrester Research "Every Second Counts: How Website Performance Impacts Shopper Behavior" (2009年)

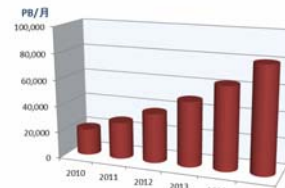
Webアプリケーションの応答が1秒間遅くなった場合の影響



出典: Aberdeen "The Average Impact of a One Second Delay in Response." (2008年)

トランザクション増対応できるスケーラビリティ

猛烈な勢いで増加するトラフィック
モバイル・アクセスの増加が拍車をかける



出典: Cisco® Visual Networking Index (VNI) Forecast (2010-2015)

インテリジェント管理機能が求められている背景として、昨今ITトレンドの変化により、システムの性能要件や運用管理に対して、より厳しい要件が求められるようになったという現実があります。

そのトレンドを非機能要件の観点からまとめます。

■ レスポンスタイム

オンライン・ショッピングサイトにおけるWebアプリケーションの応答が1秒でも遅くなると、顧客満足度が16%低下するという統計データがあります。また、Webサイトのパフォーマンスに不満を持つお客様の約80%が二度とそのサイトで買物をしない、というリサーチ結果があります。ユーザアクセスが混雑している時においても、ユーザ単位、業務単位のサービス・レベル維持が重要な課題となっています。

■ スケーラビリティ

近年、インターネットのトラフィックが年々猛烈な勢いで増加し、過去5年で、世界のIPトラフィックは8倍、今後5年間は更に4倍になる予想されます。また、常に「オン」になっているモバイル・アクセスがさらにこのトランザクションの増加に拍車をかけています。

急激なトランザクション増加にも瞬時に、柔軟に対応できるよう、ITシステムのスケーラビリティが求められています。

インテリジェント管理が求められる背景～その2

- 可用性～「使いたい」時に使えるサービスの確保
- 運用・管理効率化 ～運用・管理コストの削減

24時間365日 サービスを止めない運用

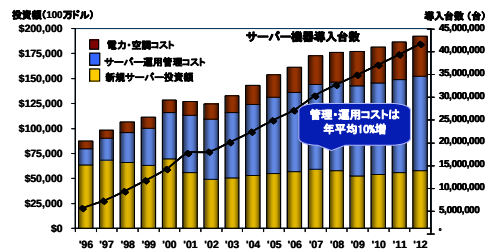
資源利用の効率向上&運用・管理の省力化

オンライン金融サービスを利用する顧客が求める可用性



出典: 2010年 Akamai社 & Forester research

運用・管理コストがIT予算を圧迫



出典: IDC, Transforming the Datacenter: Consolidation, Pervasive Virtualization and Energy Optimization, Doc #DIR2009_T4_MB, Mar 2009.

© 2012 IBM Corporation

7

■ 可用性の観点

24時間365日 サービスを止めない運用はもはや常識になりつつあります。システム障害の未然防止や問題発生時の自動対処機能が求められています。

■ 運用の観点

リニアに増加する運用・管理コストがIT予算を大きく圧迫しています。運用管理者の負担軽減や運用コストの削減が常に大きな課題となります。

サービス継続したままアプリケーションのバージョンアップやメンテナンス作業が常に運用管理者の悩みの種です。



このようなITトレンドの変化に応えるべく、WAS V8.5には「インテリジェント管理」機能をソリューションとして提供しています。

インテリジェント機能の構成要素は大きく分けて4つの部分になります。

1. インテリジェント・ルーティング

この機能は、ユーザ単位、アプリケーション単位で優先順位を設定して、リクエストの優先度に応じて流量制御をしつつ、サーバーに対して自動割り振りを行います。

この機能は重要度の低いリクエストをキューイングし、重要度の高いリクエストを即座に処理させます。これによりリクエストが集中した場合においても、サービス・レベルを維持することを可能です。

インテリジェント・ルーティング機能の中で、中心的な役割を果たしているのがODR(On Demand Router)という新しいサーバータイプです。

2. パフォーマンス管理

このパフォーマンス管理機能は2つの機能を意味します。

①ワークロード管理機能

各ノードのCPU使用率、スループット、同時リクエスト数などのパフォーマンス情報から、リクエスト割り振りのウェイトを動的に計算します。

②リソース管理

処理状況やリソース状況に応じてアプリケーションをどのサーバーで稼働させるのかを動的に決め、JVMの数を動的増減させて制御します。

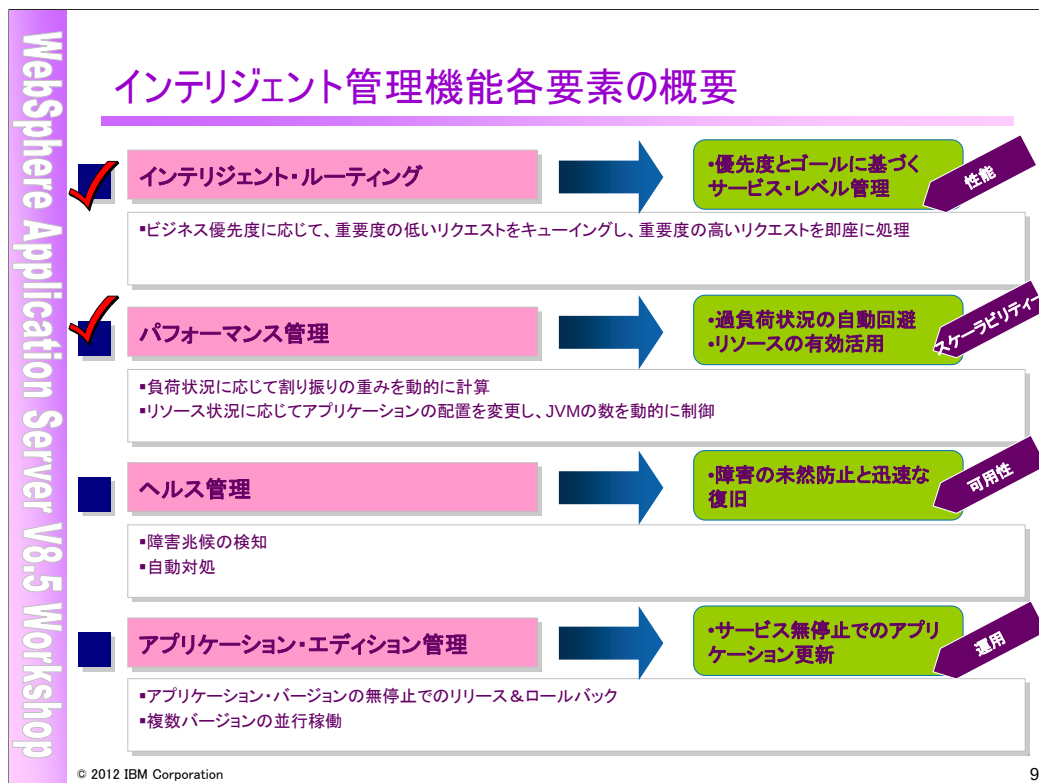
3. ヘルス管理

ヘルス管理とは、障害兆候の検知と自動対処の機能です。例えばアプリケーションの応答状況やJVMヒープ使用量を常に監視して、パフォーマンスが悪化した場合、そのサーバーへの割り振りを停止するようなヘルス・アクションを執行することができます。

ヘルス管理機能を担うコンポーネントはヘルス管理コントローラー(HMC)になります。

4. アプリケーション・エディション管理

アプリケーション管理とは、サービス無停止でのアプリケーション更新・ロールバック、同一アプリケーションの複数バージョンの並行稼働する機能になります。



インテリジェント管理機能の各要素と、新しいITトレンドにおける非機能要件とのマッピングです。

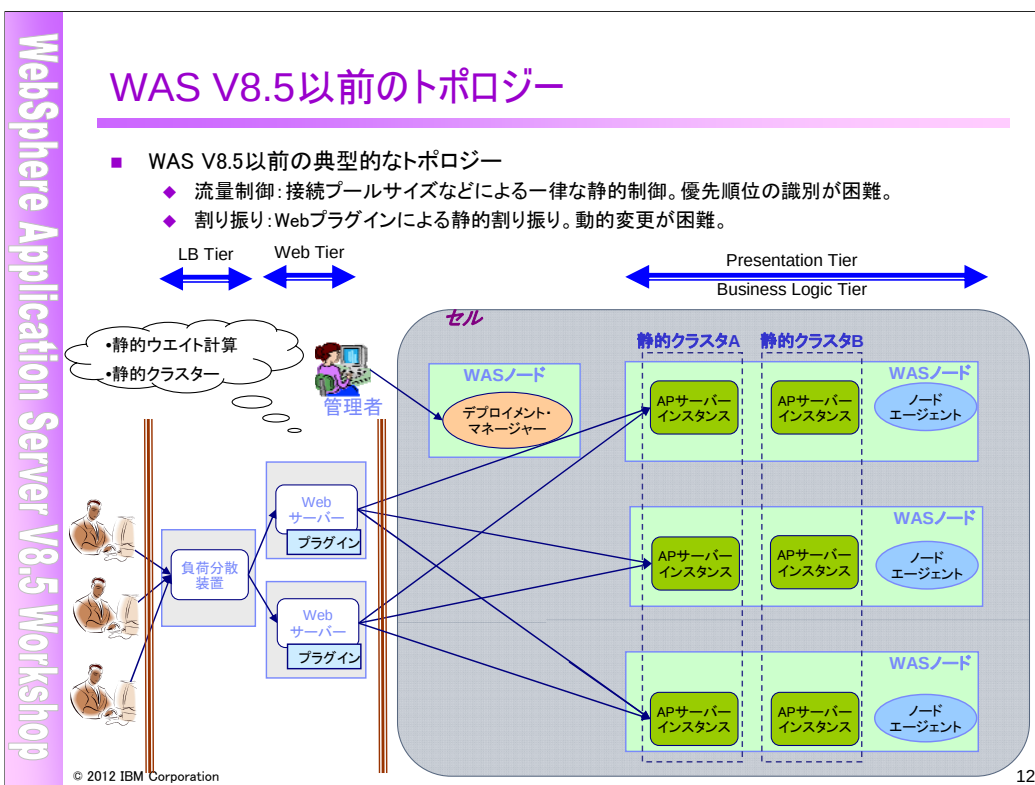
1. インテリジェント・ルーティングが優先順位に応じて流量制御を行い、優先度とビジネス目標に基づくサービス・レベル管理が可能になり、サービス・レベル向上につながります。
2. パフォーマンス管理が本番稼働中突発的なトランザクション増加にも対応可能で、サーバーの過負荷状況を回避しつつ、割り振りの重みを動的に計算して割り振りをします。またリソース状況に応じてアプリケーションの配置を動的変更してJVMの数を動的に制御し、リソースを有効利用しながらスケーラビリティを向上させます。
3. ヘルス管理機能は障害の未然防止と迅速な復旧を可能にし、システムの可用性や耐障害性を向上させます。
4. アプリケーション・エディション管理はユーザーに影響を与えずアプリケーションのバージョンアップや並行稼働を容易に実現し、サービスの継続性を実現しながら、運用負荷の低減を実現します。

このセッションでは、インテリジェント・ルーティングとパフォーマンス管理を説明して、この後のセッションでは、ヘルス管理およびアプリケーション・エディション管理を説明します。

2. インテリジェント管理トポロジー

この章の内容

- WAS V8.5以前のトポロジー
- インテリジェント管理トポロジー全体像
- ODR
- 動的クラスター



WASV8.5では、インテリジェント管理機能の導入によって、トポロジーは今までのWASに比べて、大きな変化がありました。

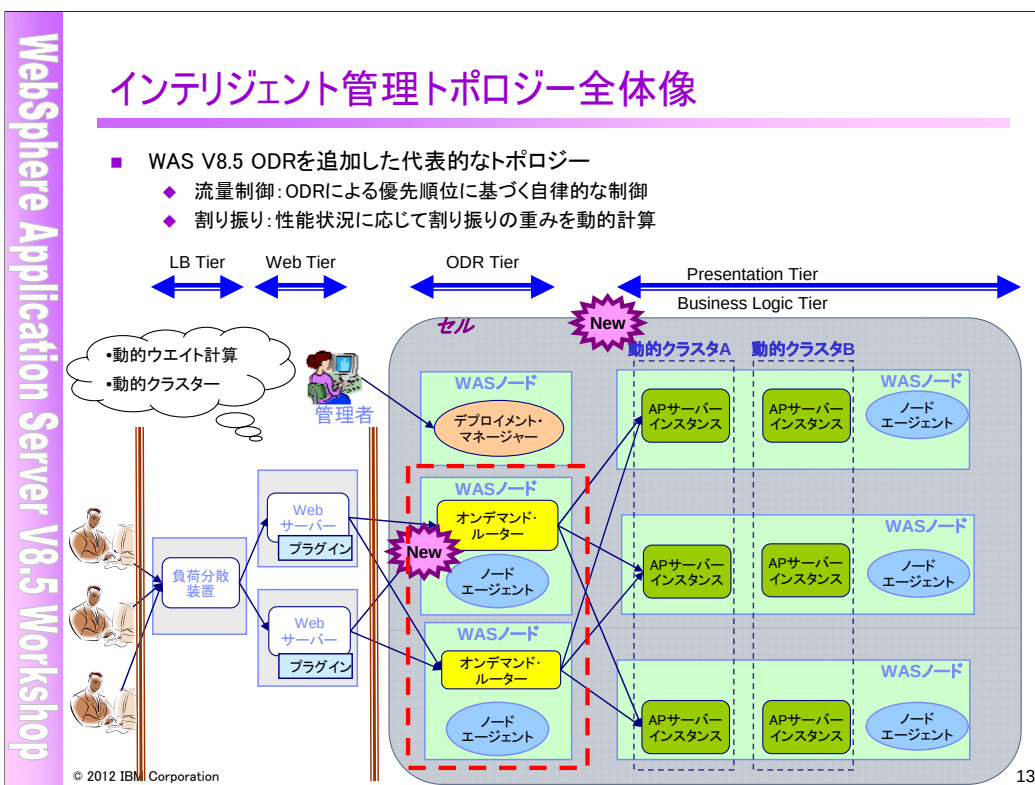
こちらはWAS V8.5以前の典型的なWASトポロジーです。Webサーバープラグインによって、事前に定義した静的なウェイトによって割り振りを実施し、クラスターも静的クラスターであり、サーバーの起動停止は管理者が手動で制御する必要があります。

サービス無停止にアプリケーションを更新したい、もしくは片方のサーバーを止めてメンテナンスしたい場合には管理者にとって負担の大きい作業になります。

割り振りを制御をしたい場合、Webプラグインの構成ファイルを複数用意して、タイミングを見計らって、plugin-cfg.xmlをロードしてあげる必要があります。

また、負荷状況が変化しリソースが余っても、サーバーを勝手に停止することは基本的にできません。

逆に急激にトランザクションが増え、リソースが逼迫した場合、あまったサーバーをすぐに起動してクラスターに加わりたい場合にはかなり困難な作業になります。



こちらはWAS V8.5の新しいトポロジーです。ODR(オンデマンド・ルーター)という新しいタイプのサーバーが追加されました。

Webプラグインからアプリケーション・サーバーに対してではなく、一旦ODRに対して割り振りします。ODRからアプリケーション・サーバーに対して動的割り振りを実施します。

従来Webプラグインの単純なラウンドロビン方式に比べて、ODRがサービス・レベルを遵守するよう、アプリケーションの稼働状況、サーバーの負荷状況に応じて、割り振りの重みを動的計算した上で割り振りを実施します。

また、従来の静的クラスターに対して、WAS V8.5では動的クラスターが登場します。

動的クラスターが、必要に応じてクラスター・メンバーを自動的に始動、停止することができます。

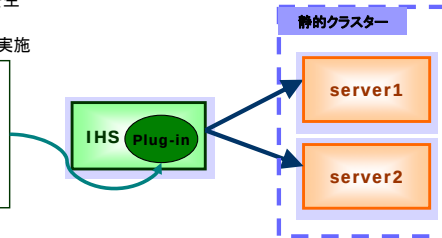
ODRの追加にはODR専用のカスタム・プロファイルの作成が必要です。作成手順は今までのカスタムプロファイルの作成方法と基本的に同様です。

【参考】WebサーバーとODRの連携

■ 従来型の割り振り形態

- ◆ 管理コンソール／手動によるplugin-cfg.xmlを生成
- ◆ プラグインがplugin-cfg.xmlに従い割り振りを実施

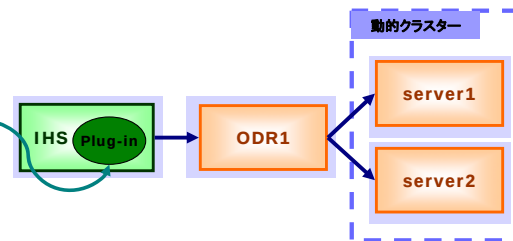
```
<ServerCluster xxx >
<Server CloneID="xxx" MaxConnections="-1" Name="server1">
.....
</Server>
<Server CloneID="xxx" MaxConnections="-1" Name="server2">
.....
</Server>
</ServerCluster>
```



■ ODRを介した割り振り形態

- ◆ ODRによるplugin-cfg.xmlを生成
- ◆ 高可用性プラグイン生成機能によるplugin-cfg.xmlを生成(推奨)

```
<Server Name="was101Cell01_was101Node01_odr1" ConnectTimeout="0" ExtendedHandshake="false" MaxConnections="-1" WaitForContinue="false" LoadBalanceWeight="1">
  <Transport Hostname="was101" Port="81" Protocol="http"/>
  <Transport Hostname="was101" Port="443" Protocol="https">
    <Property Name="keyring" Value="/opt/IBM/WebSphere/AppServer/profiles/Dmgr01/etc/plugin-key.kdb"/>
    <Property Name="stashfile" Value="/opt/IBM/WebSphere/AppServer/profiles/Dmgr01/etc/plugin-key.sth"/>
  </Transport>
</Server>
<PrimaryServers>
  <Server Name="was101Cell01_was101Node01_odr1"/>
</PrimaryServers>
```



【参考】プラグイン構成の動的更新

■ WebサーバーとODRの連携におけるプラグイン構成の動的更新

◆ 方法1: ODRプロセスによる動的更新

- JVM システム・プロパティに ODCPluginXmlDir という名前、plugin-cfg.xmlの生成ディレクトリーの絶対パスを指定
- 生成時に実行するスクリプトを指定することで、リモートのWebサーバーへの伝搬

ただし、起動していないODRは、プラグイン構成を生成できない

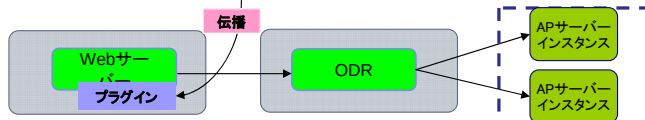
◆ 方法2: WAS高可用性環境内の1つのJVMプロセスによる動的更新(推奨)

- 下記のような一連のカスタム・プロパティをセルレベルで設定
- 生成時に実行するスクリプトを指定することで、リモートのWebサーバーへ伝搬

cp /tmp/plugin-cfg.xml
/Webプラグイン配下/plugin-cfg.xml

JVMプロセスがWebプラグイン導入先と同一ホスト上に存在する場合のスクリプト例

プロパティ名	プロパティ値	説明
ODCPluginCfgOdrList<構成名>	<セル名><ノード名><ODR名>	指定されたODRにルーティングするプラグイン構成を生成
ODCPluginCfgOutputPath<構成名>	定義例: /tmp/plugin-cfg.xml	生成されたプラグイン構成ファイルパス
ODCPluginCfgUpdateScript<構成名>	定義例: /tmp/pluginCfgUpdate	ファイルが更新されるたびに起動されるスクリプトへのパス



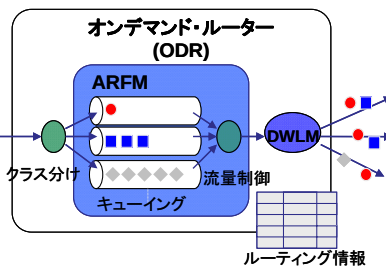
plugin-cfg.xml ファイルの生成はセル内のどのノードでも起こり得るので、環境に応じてスクリプトを作成する必要がある

ODR(オンデマンド・ルーター)とは

- Javaベースのインテリジェントなリバース・プロキシ
 - ◆ プロトコルとしてHTTPとSIP (Session Initiation Protocol) をサポート
 - ◆ サービス・レベル、サーバー状況に応じた流量制御／ルーティングを行う
 - ◆ 新しいサーバータイプとして、管理コンソールにより構成可能
 - ◆ ODR自身のクラスター構成も可能
 - ◆ ODRがインテリジェント管理の各機能をサポートする中心的なコンポーネント

※SIPとはSession Initiation Protocolの略で、IP電話などで採用されているシグナリング・プロトコルのことです。

- ODRが提供するインテリジェント機能
 - ◆ オートノミック要求フロー・マネージャー(ARFM)
 - サービス・ポリシーに基づいてリクエストをキューイング・優先順位付けし、流量制御する
 - 過負荷防止
 - ◆ 動的ワークロード・マネージャー(DWLM)
 - 動的な重み付けによるワークロード管理によりバックエンド・サーバーにルーティングする
- オンデマンド・コンフィグレーション(ODC)
 - ◆ セル内のアプリケーション・サーバーの状況を取得して動的にODRのルーティング情報を更新



© 2012 IBM Corporation

16

ODRはJavaベースのインテリジェントなリバース・プロキシ・サーバーです。
プロトコルとしてはHTTPとSIPをサポートしています。

ODRの実体はJavaプロセスであり、管理コンソールにより構成を行います。ODRはAppServerノード群の前面に位置し、バックエンドのAppServerノード群へのリクエストのフローを制御します。

ODRに送られてきたリクエストはまずサービス・ポリシーごとのキューにキューイングされます。
ここでオートノミック要求フロー・マネージャー (ARFM: Application Request Flow Manager) によって計算されたキューのウェイト値に従って優先順位付けされ、流量制御されます。ARFMにより、どのサービス・ポリシーのリクエストをどれだけ送るかが決定されます。

次に動的ワークロード管理 (DWLM: Dynamic Workload Manager) によって動的に重み付けられたバックエンド・ノードに振り分けられます。DWLMは動的な重み付けによるルーティングの他、セッション・アフィニティ、サーバー回復時のフェイル・オーバー機能も持ちます。

バックエンド・ノードではアプリケーション配置コントローラー (APC: Application Placement Controller) が AppServer を動的に起動したり縮退したりするプロビジョニングを行います。APCはサーバーの負荷状況を把握しており、AppServerの起動要求がある場合にどのノードで起動・停止を行えばよいのかを判断し、実行します。

ODRが提供するインテリジェント機能としては、

■ オートノミック要求フロー・マネージャー(ARFM):

リクエストの運用ポリシー (サービス・クラス) に基づいてリクエストをキューイングし、優先順位付けを行います。

■ 動的ワークロード管理(DWLM):

サービス状況によりバックエンドのAppServerへの振り分けの重みを動的に変更するワークロード管理機能を提供します。

ODRがODC情報に従いリクエストをルーティングします。

ODC: オンデマンド・コンフィグレーション。バックエンドAppServerへのルーティング・テーブルを動的に更新します。

【参考】ODRの作成

- ODR用プロファイルの作成
 - ◆ プロファイルのタイプ: カスタム・プロファイル
 - ◆ 「管理」タイプは使用できない
- 管理コンソール上の新規作成、構成
 - ◆ サーバー・タイプに「オンデマンド・ルーター」を選択
 - ◆ アプリケーション・サーバー作成などのオペレーションとほぼ同様
- 通信ポートの確認
 - ◆ PROXY_HTTP(S)_ADDRESS
 - ◆ この例では、81番が選択される
- 仮想ホストの登録
 - ◆ 上記確認されたポート番号を仮想ホストとして追加

プロファイル作成



管理コンソール>ODR>通信>ポート

DRS_CLIENT_ADDRESS	7873
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	0
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	0
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	0
WC_defaulthost	9081
DCS_UNICAST_ADDRESS	9355
WC_defaulthost_secure	9444
PROXY_HTTP_ADDRESS	81
PROXY_HTTPS_ADDRESS	443
OVERLAY_UDP_LISTENER_ADDRESS	11003
OVERLAY_TCP_LISTENER_ADDRESS	11004
ORB_LISTENER_ADDRESS	0

管理コンソール>環境>仮想ホスト

管理できるリソース:	
☐ admin_host	
☐ default_host	
☐ proxy_host	

追加プロパティ

ホスト別名

MIME タイプ

81

【参考】ODR ルーティング情報 (target.xml)

- ODRのルーティング情報はメモリー上に保持され動的に更新される
- ODRのトレース設定でディスク上に書き出し可能 ⇒ target.xml
 - ◆ ODRのトレースに次のトレース・ストリングを設定 com.ibm.ws.odc.nd.ODCTreeImpl\$Save=all
 - ◆ <ODR_Root>/installedFilters/wlm/<odr_name>/target.xml

```
<server name="DCluster01_jupiterNode01">
  <transient properties="{server=name=solarConfCCell01/jupiterNode01/DCluste
r01_jupiterNode01: weight=20 (/cellGroup/target/cell/solarConfCCell01/node/jupiterNode01/s
erver/DCluster01_jupiterNode01(com.ibm.ws.xd.dwlm.client.XDTargetServerImpl@4be04be0))}" />
  <property name="id" value="12e9dksi7" />
  <property name="type" value="APPLICATION_SERVER" />
  <property name="state" value="STARTED" />
  <property name="ODR" value="false" />
  <property name="reachable" value="true" />
  <property name="weight" value="20" />
  <property name="threadPoolMin" value="5" />
  <property name="threadPoolMax" value="20" />
  <property name="threadPoolsGrowable" value="false" />
  <property name="sessionPersistenceMode" value="DATA_REPLICATION" />
  :
```

サーバーのクローンID

稼働状況

DWLMウェイト値

© 2012 IBM Corporation

18

ODRのルーティング情報は上記トレース・ストリングの設定によりXMLファイルに書き出すことが可能です。

target.xml には以下のようなさまざまな情報が含まれています。

- ノード・グループ情報(ARFMパラメータなど)
- ノード情報(CPU Spec (Speed, CPU数など))
- サーバー情報(クローンID、稼働状況、DWLMウェイト値、エンド・ポイント情報など)
- 仮想ホスト・アプリケーション関連情報
- 動的クラスター情報(マッピングされたノード・グループおよびノード, 導入Webモジュール)
- 運用ポリシー情報
- サービス・ポリシー(ビジネス・ゴールと重要度の定義)
- トランザクション・クラス(ポリシー定義の対象となるURI)

動的クラスター

■ 動的クラスターとは

- ◆ アプリケーションがデプロイされるサーバーのグループ
- ◆ クラスター・メンバーが、自動(又は半自動)の稼働・停止の対象になる点が、静的クラスターとの相違点
- ◆ サーバーの稼働・停止操作に関するモードとして、手動・監視・自動から選択可能

■ メンバーシップ・ポリシー

- ◆ クラスター・メンバーを作成するノードを選択する基準を定義(詳細は後述)
- ◆ メンバーシップ・ポリシーに合致するノードには、動的クラスター作成時、及び、ノード追加時に、クラスター・メンバー(アプリケーション・サーバー)が自動的に作成

■ 動的クラスターの定義例

- ◆ 常に1つのインスタンスを開始済みにしておく
- ◆ 開始できるインスタンスを4つ以内にしておく
- ◆ 同一ノード上2つ以上のインスタンスを同時に開始しない

■ 動的クラスター・サーバータイプ

- ◆ WebSphere Application Server
 - WAS NDノードによる動的クラスター
- ◆ PHPなど他ミドルウェアに関しても定義可能
 - WebSphere Application Serverに比較した場合、管理レベルは異なる

管理コンソール > 動的クラスター

クラスター・インスタンスの最小数

☐ 他の動的クラスターがリソースを必要とする場合は、アクティブなインスタンスを停止するまでに待機する時間:

60 分

☒ 常に 1 つのインスタンスを開始済みにしておく

☐ 常に複数インスタンスを開始済みにしておく

インスタンスの数: 2

クラスター・インスタンスの最大数

☐ 開始できるインスタンス数を制限する

インスタンスの数: 2

☒ 開始できるインスタンス数を制限しない

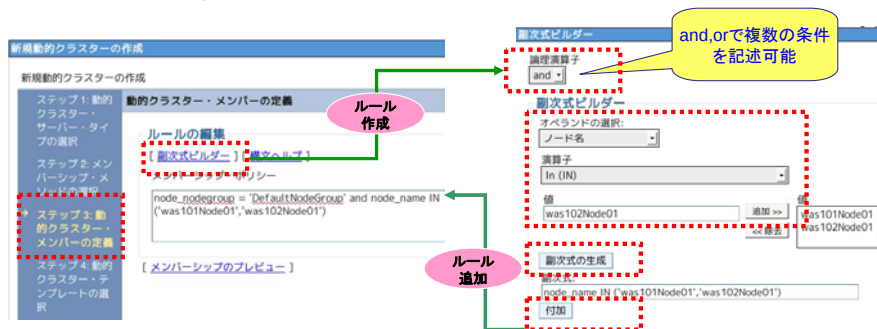
ノード上のインスタンスの垂直スタッキング

☐ 同一ノードで複数のノードを開始することを許可する

インスタンスの数: 2

【参考】動的クラスターメンバーの決定

- 動的クラスター・メンバーは、メンバーシップ・ポリシーによって自動で決定
 - ◆ ルール・ビルダーによりメンバーシップ・ポリシーを定義
- クラスター・メンバーは以下のルールから定義
 - ◆ ノード名、ノード・グループ、ホスト名、ノード・プロパティ
- 動的クラスター・メンバー作成後のノードの追加/削除
 - ◆ ノードの追加/削除を行うとクラスターメンバーであるアプリケーション・サーバーも追加/削除
 - 新規ノード追加時に、メンバーシップ・ポリシーに合致する場合、サーバー・インスタンスを作成して動的クラスターに追加



© 2012 IBM Corporation

20

こちらは動的クラスターのメンバーシップ・ポリシーの定義画面です。

管理コンソール上動的クラスターに所属するノード名、ホスト名などの情報および「AND」や「OR」などの論理演算子を指定することで簡単なステップによって、メンバーシップ・ポリシーを定義することができます。

あとはこのメンバーシップ・ポリシーに合致するように、また前のページで定義した動的クラスターの振る舞いに合致するように、クラスター定義完了時にアプリケーション・サーバーが自動的に作成されます。

【参考】動的クラスターの操作モードの設定

- 手動 :
 - ◆ WAS V8.5以前のクラスター環境と同等
 - ◆ アプリケーション配置コントローラー、およびランタイム・タスク機能は使用できない
 - アプリケーション配置コントローラー: 負荷状況に応じてアプリケーション・サーバー自動起動・停止
 - ランタイム・タスク: サービス・ポリシー違反が検知されるとタスクとして通知される
 - ◆ オートノミック要求フロー・マネージャー(ARFM)、動的ワークロード管理(DWLM)とは連動
- 監視 :
 - ◆ ランタイム・タスクを生成することにより、必要な修正アクションに関する情報が提供される
 - 管理コンソールの[システム管理]-[タスク管理]-[ランタイム・タスク]のパネルで、管理者がオートノミック・マネージャーの推奨を受け入れるか、拒否する
- 自動 :
 - ◆ 自動的に修正処置を実行



手動。手動モードでは、動的クラスターと、静的クラスターを持つ標準のアプリケーション・サーバー環境との違いはありません。手動モードでは、アプリケーション配置、およびランタイム・タスクの候補表示はサポートされません。オートノミック要求フロー・マネージャー、および動的ワークロード管理(DWLM)は、クラスターと連動できます。

監視。監視モードでは、ランタイム・タスクを生成することにより、必要な修正アクションに関する情報が環境で提供されます。管理コンソールの「タスク管理」パネルで、オートノミック・マネージャーの推奨を受け入れるか、拒否することができます。ランタイム・タスクを管理するには、「システム管理」>「タスク管理」>「ランタイム・タスク」とクリックします。

自動。自動モードでは、環境は自動的に修正処置を実行します。

【参考】垂直スタッキング

- 同一ノードにおいて、動的クラスター・メンバーを複数構成することで、より効率的にノードのリソースを利用できるケースへの対応
 - ◆ 動的クラスターごとに、垂直スタッキングの許可とインスタンス数を指定
 - ◆ 水平クラスターで限界に達すると垂直クラスターを構成



クラスター・インスタンスの最小数
☐ 他の動的クラスターがリソースを必要とする場合は、インスタンスを停止するまでに待機する時間:
 60 分

☐ 常に1つのインスタンスを開始済みにしておく
 * 常に複数インスタンスを開始済みにしておく
 インスタンスの数:
 3

クラスター・インスタンスの最大数

☐ 開始できるインスタンス数を制限する
 インスタンスの数:
 2

☒ 開始できるインスタンス数を制限しない

ノード上のインスタンスの垂直スタッキング
☒ 同一ノードで複数のノードを開始することを許可する
 インスタンスの数:
 2

■ ノード上の個別設定

- ◆ 動的クラスター単位の設定のため、異なるスペックのノードが存在する場合は、ノード単位での書き設定を行う

動的クラスター

動的クラスター > DCluster > 動的クラスター・メンバー

動的クラスター・メンバーとは、動的クラスターに属するサーバーです。

* 設定

開始 停止

選択	メンバー名	ノード	バージョン	構成済みウェイト	ランタイム・ウェイト	状況
☐	DCluster_was102Node01	was102Node01	ND 8.5.0.0	2	20	→
☐	DCluster_was102Node01.1	was102Node01	ND 8.5.0.0	2	20	→
☐	DCluster_was103Node01	was103Node01	ND 8.5.0.0	2	20	→
☐	DCluster_was103Node01.1	was103Node01	ND 8.5.0.0	2		→

動的クラスターのカスタム・プロパティ< numVerticalInstances.node_name>

© 2012 IBM Corporation

22

動的クラスターの詳細なオプションについて説明します。

垂直スタッキングは動的クラスターの設定で、一つのノードで複数の動的クラスター・メンバーのアプリケーション・サーバー (JVM) が起動することを可能にします。

設定は動的クラスターに一律設定する方法とノードごとに垂直クラスター数をきめる2つの方法があります。

「垂直スタッキングをサポートする同種ノードの動的クラスターの構成」

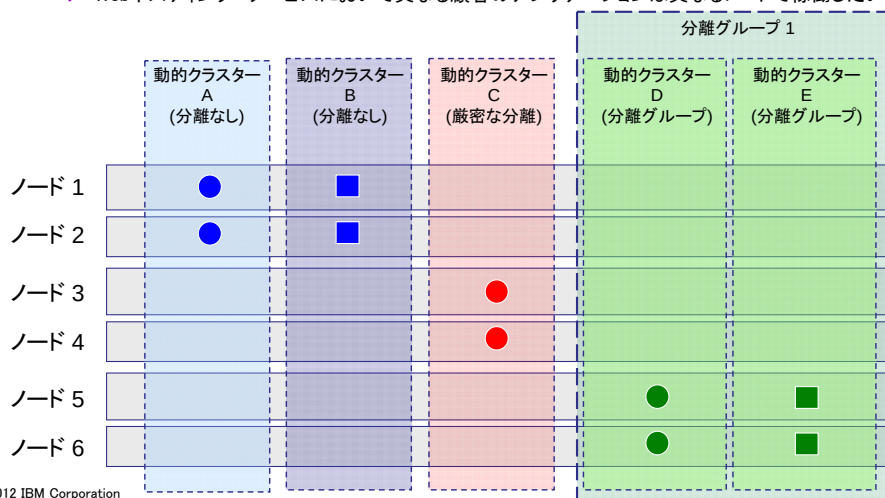
http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.wve.doc/ae/twve_odst_ackhomog.html

「垂直スタッキングをサポートする異種ノードの動的クラスターの構成」

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.wve.doc/ae/twve_odst_ackhetero.html

動的クラスターの分離(Isolation)

- 動的クラスターを分離し、他のクラスター・メンバーは同一ノード上で稼働しないよう制限する機能(以下は利用例)
 - ◆ クリティカル・アプリケーションをノードを占有させて稼働したい
 - ◆ 社内向けアプリケーションは社外向けアプリケーションと異なるノードで稼働したい
 - ◆ Webホスティング・サービスにおいて異なる顧客のアプリケーションは異なるノードで稼働したい



© 2012 IBM Corporation

23

動的クラスターのもう1つの機能はアプリケーション分離です。アプリケーション分離はセキュリティーおよびパフォーマンスなどの理由により、あるアプリケーションを他のアプリケーションと分離して同一ノードで稼働しないように制限をする機能になります。例えば次のような要件がある際に適用することができます。

- 社内向けアプリケーションと社外向けアプリケーションが稼働するシステムにおいて、両者を別々のノードで稼働したい
- セキュリティー／パフォーマンス上クリティカルなアプリケーションについてノードを占有させて稼働したい
- データセンターなどでWebホスティング・サービスを提供し、複数の顧客のアプリケーションを稼働する際、異なる顧客のアプリケーションは異なるノードで稼働したい

図は、アプリケーション分離の設定によるAppServerインスタンス起動の様子を示したものです。ここで“厳密な分離”の設定がしてある動的クラスターCのAppServerインスタンスの起動要求が来た場合、APCはどの動的クラスターのAppServerも稼働していないノード4に起動させます。“厳密な分離”はノードを占有することができるため、クリティカルなアプリケーションの稼働に適します。

“分離要件なし”の動的クラスターAおよびBのAppServerインスタンスの起動要求が来た場合、分離要件なしのAppServerインスタンスとは同一ノードで稼働できるためAPCはノード1、ノード2に起動させます。“分離要件なし”は特に制限のない社内向けアプリケーションの稼働に適しています。

“分離グループ1”のAppServerインスタンスの起動要求が来た場合、APCは“分離要件なし”や“厳密な分離”や他の“分離グループ”のAppServerインスタンスが稼働していないノード5、ノード6に起動させます。同じ分離グループ1の稼働しているノードには同居することができます。“分離グループ”は複数顧客のアプリケーションを顧客ごとにノードを占有させて稼働したい場合に適します。

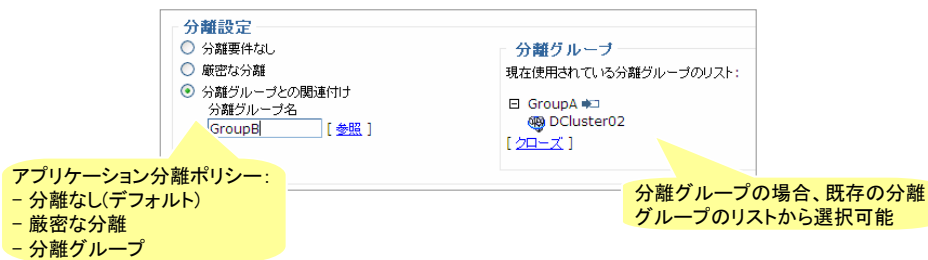
仮想化は稼働場所を意識せずに必要なリソースを割り当てて稼働環境を提供する技術ですが、アプリケーションの性質(セキュリティー、パフォーマンス要件)によって稼働環境を分離して仮想化をある程度制御するソリューションがアプリケーション分離になります。

【参考】動的クラスターの分離

■ 設定方法

[サーバー]-[動的クラスター]-[動的クラスター名]の「分離設定」

- ◆ 厳密な分離: 他の動的クラスターのクラスター・メンバーとは同じノードでクラスター・メンバーを稼働しない
- ◆ 分離グループ: 異なる分離グループの動的クラスターのインスタンスは同じノードで稼働しない



© 2012 IBM Corporation

24

アプリケーション分離の設定は動的クラスターごとに行います。動的クラスターの分離設定において以下の3つの分離ポリシーから選択することができます。

- 分離要件なし(デフォルト): アプリケーション分離を行わない
- 厳密な分離: この動的クラスターのAppServerインスタンスが稼働するノードには他の動的クラスターのAppServerインスタンスは稼働しない
- 分離グループ: この動的クラスターのAppServerインスタンスが稼働するノードには同一の分離グループに属する動的クラスターのAppServerインスタンスのみが稼働できる

分離ポリシーで分離グループを選択した際は、既存の分離グループのリストが表示されるので、そこから選択するか新規の分離グループを設定します。

【参考】アプリケーションの遅延(lazy)スタート

- 時間指定によるサーバー停止準備とリクエストによる自動スタートの仕組み
 - ◆ 管理コンソールの動的クラスターの設定

クラスター・インスタンスの最小数

他の動的クラスターがリソースを必要とする場合は、アクティビティがない期間にこのクラスターのすべてのインスタンスを停止する

インスタンスを停止するまで待機する時間:

60 分

☐ 常に 1 つのインスタンスを開始済みしておく

☐ 常に複数インスタンスを開始済みしておく

インスタンスの数:

2

- ◆ 待機時間を経過し、別の動的クラスターでリソースが必要になった時点で全クラスター・メンバーを停止

- クラスター・メンバーを個別停止するより、効率的にリソースを解放
- カスタム・プロパティ(proactiveIdleStop)を設定することで、リソース要求の有無に関わらず待機時間経過後停止することも可能

動的クラスター > DCluster > カスタム・プロパティ

データの任意の名前と値の付与を繰り返します。名前またはキー、値はストリングで、これらにより特定のシステム構成プロパティを設定できます。

※ 設定

名前	値	説明
proactiveIdleStop	true	

合計: 1

- ◆ 停止時にリクエストを受けた場合に、少なくとも1つのクラスター・メンバーを開始
 - サーバーが開始するまでに受信したリクエストに対してはHTTPコード503を応答(カスタム・エラー・ページやカスタム・エラー・アプリケーションで任意の画面を表示させることも可能です)

© 2012 IBM Corporation

25

アプリケーションの遅延(lazy)スタートはリクエストがきてからJVMがスタートする仕組みです。

この設定は動的クラスターに対するクラスター・インスタンスの最小数の設定により行います。

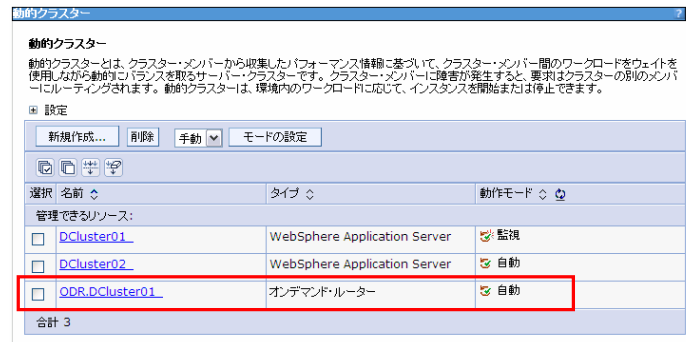
デフォルトでは、リクエストがない待機時間をオーバーした場合に自動停止するわけではなく、他のアプリケーション・サーバーの開始等のリソースの不足により停止されます。

"proactiveIdleStop=true"のカスタム・プロパティを設定することで、リソース要求の有無に関わらず、待機時間経過後にクラスター・メンバーを停止することも可能です。

なお、動的クラスターの停止時にリクエストを受けた場合には、クラスター・メンバーが開始しますが、始動中のリクエストに対しては503を返します。開発、テスト環境において、複数の動的クラスターが定義されている場合に、リソースを有効活用することができます。

【参考】ODRの動的クラスター

- ODRの動的クラスターが構成可能になり、より自律的な管理が可能に
 - ◆ 最低稼働数をポリシーとして設定することで、自動的に最低稼働数を維持するようにODRを自動で起動させることが可能

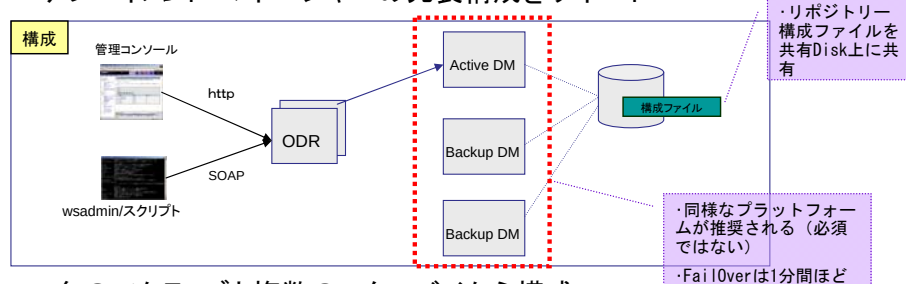


アプリケーション配置コントローラー(APC)が最小数のODRを開始するのに最適なノードを選択し、ODRの障害時にも、自動的に最低稼働数を維持するようにODRを自動で起動させることが可能です。

なお、ODRの動的クラスターを構成場合には、“APC.predictor = CPU” カスタム・プロパティを設定して、アプリケーション配置コントローラー(APC)は、ODRの起動時に、オートノミック要求フロー・マネージャー(ARFM)からデータを取り出さず、CPU使用率のみにもとづいてODRの起動ノードを決定するようにします。

【参考】高可用性デプロイメント・マネージャー

■ デプロイメント・マネージャーの冗長構成をサポート



■ 1台のアクティブと複数のスタンバイから構成

- ◆ スタンバイでは管理コンソールへのログイン、構成の変更などを禁止
- ◆ フェールオーバーはHAマネージャーが実施
- ◆ ODRはアクティブ・サーバーを検知してルーティング

■ 以下の機能の高可用性が求められる場合に使用を検討

- 可視化データ・サービス
- レポート
- 監視モード

© 2012 IBM Corporation

27

高可用性デプロイメント・マネージャーとは、デプロイメント・マネージャーの冗長化を行う機能です。セル内に1台のアクティブなデプロイメント・マネージャーと複数のスタンバイ・デプロイメント・マネージャーの設置をサポートします。スタンバイのデプロイメント・マネージャーはプロセスは起動していますが、管理コンソールへのログイン含め、構成などへの変更は一切禁止されており、スタンバイ経由でこれらの操作はできません。この機能を使用するためには、どのデプロイメント・マネージャーからもアクセス可能な共有ディスク上などの場所に構成ファイルを保管する必要があります。アクティブなデプロイメント・マネージャーの状態はHAマネージャーによって監視され、アクティブ・デプロイメント・マネージャーの障害を検知するとHAマネージャーはスタンバイのいずれかをアクティブにすることでフェールオーバーを実施します。また、ODRはどのサーバーがアクティブかを常に把握しているため、管理コンソールやwsadminスクリプトからのJMXのリクエストはODRを介することで常にアクティブなサーバーにルーティングされます。なお、ODRを介してデプロイメント・マネージャーにアクセスする場合はSOAPアクセスのみのサポートになります。

通常のNDのデプロイメント・マネージャーが実施する機能の加えて、以下の機能を実施しますので、これらの機能の可用性が求められる場合は、高可用性デプロイメント・マネージャーの導入を検討します。

・可視化データサービス：デプロイメント・マネージャーが統計情報を収集してログに記録します。高可用性デプロイメント・マネージャーを使用する場合はログファイルも共有ディスクに保管するようにします。

・Visualizationエンジン：管理コンソールの報告書機能で表示されるデータはデプロイメント・マネージャーが収集し、報告書のグラフ表示のためのエンジンもデプロイメント・マネージャーで実行されます。

・監視モード：監視モードでは、アクションの実行を管理者が管理コンソールで受諾するまでアクションが実行されません。デプロイメント・マネージャーが起動していないとアクションが実行できないため、監視モードでの運用を行う場合は検討してください。

高可用性デプロイメント・マネージャーは、hadmgrConfig コマンド・ユーティリティを使って構成します。詳細はInfoCenterを参照してください。

http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/topic/com.ibm.websphere.wve.doc/ae/cwve_xdso dmgr.html

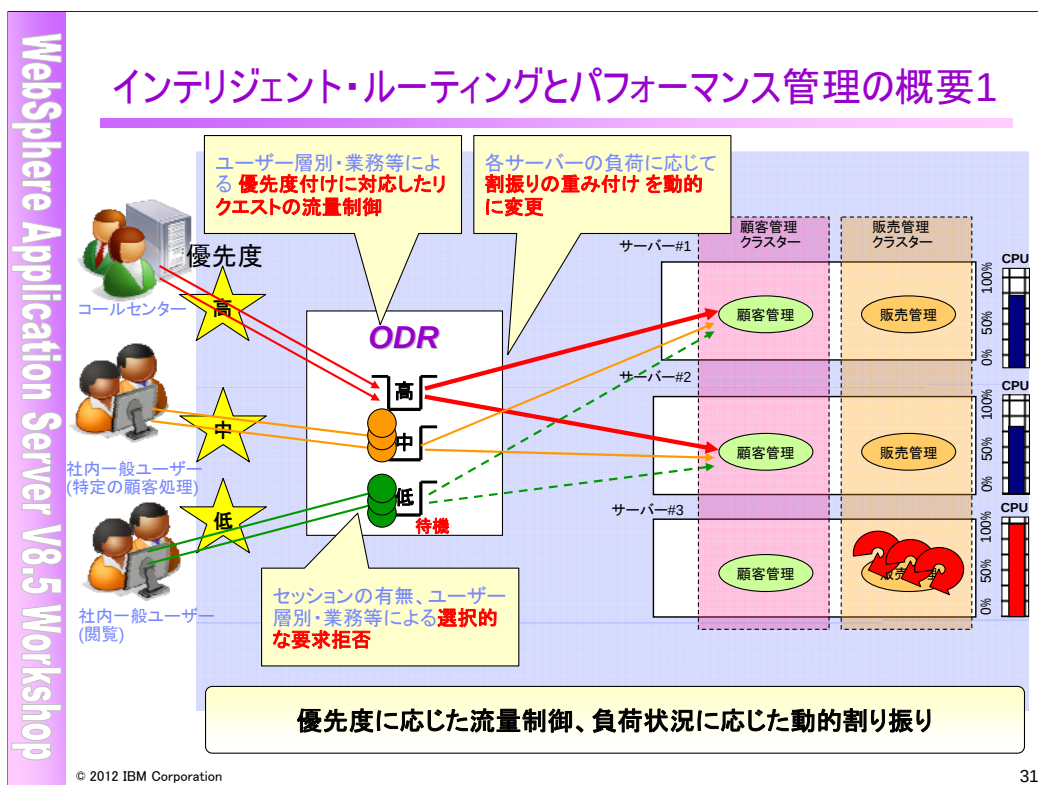
インテリジェント管理トポロジー:まとめ

- インテリジェント管理トポロジー
 - ◆ 新しいサーバー・タイプODRおよび動的クラスターが新規追加された
- ODR
 - ◆ Javaベースのリバース・プロキシ
 - ◆ リクエストの優先順位付け、流量制御、過負荷防止 (ARFM)
 - ◆ 動的ワークロード管理 (DWLM)
- 動的クラスター
 - ◆ 負荷状況、リソース状況に応じてアプリケーションを動的配置
 - ◆ 自律的に拡張・縮小 (JVMの自動起動・停止)

3. インテリジェント・ルーティングとパフォーマンス管理

この章の内容

- インテリジェント・ルーティングとパフォーマンス管理の概要
- サービス・ポリシー
- インテリジェント管理関連コンポーネント

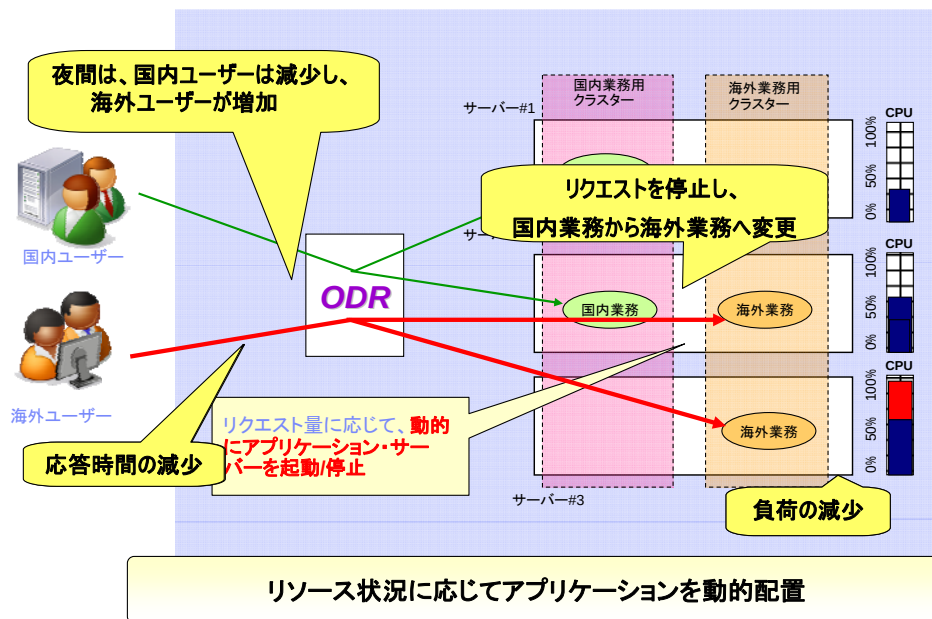


インテリジェント・ルーティングとパフォーマンス管理のコンセプトの1つ目は、優先度に応じた流量制御と負荷状況に応じた動的割振りです。

アプリケーション・サーバーが一時的に高負荷になると考えられる場合に、前段のODRにおいて、リクエストを優先度に応じてキューイングを行い、バックエンドの負荷状況に合わせて、優先度の高いリクエストを優先的にバックエンドに割振ります。また、ODRでは、優先度に応じて、リクエストのキューイングを拒否したり、バックエンドの各サーバーの負荷状況に応じて動的に割振りの重み付けを変更して割振りを行います。

これにより、バックエンドの過負荷を防止しつつ、優先度に応じたサービス・レベル管理が実現されます。

インテリジェント・ルーティングとパフォーマンス管理の概要2



32

インテリジェント・ルーティングとパフォーマンス管理のコンセプトの2つ目は、リソース状況に応じてアプリケーションを動的に配置することです。

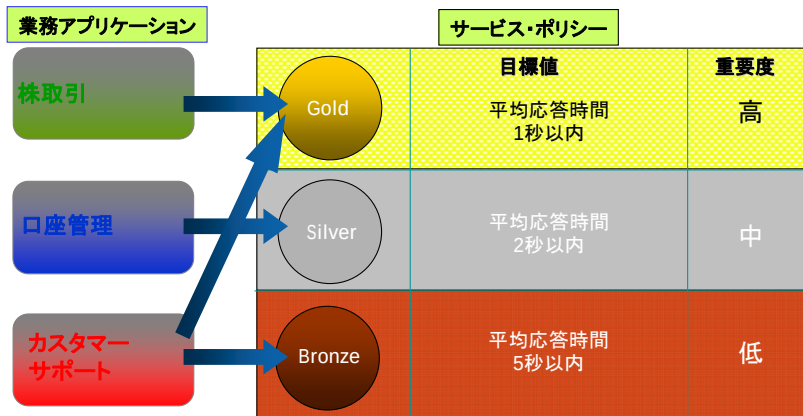
特定のアプリケーションのリソース不足とリソースがあまっているアプリケーションを検知し、動的にアプリケーション・サーバーの起動/停止を行います。

これにより、限りあるリソースを有効活用できます。

サービス・ポリシー

■ サービス・ポリシーとは

- ◆ アプリケーションのサービス・レベル(ビジネス目標)を定義
- ◆ 応答時間目標値と重要度の組合せ(管理者より簡単に指定可能)
- ◆ サービス・ポリシーに従って、優先度に応じた流量制御、アプリケーション配置、リソース管理を自動的に実施



© 2012 IBM Corporation

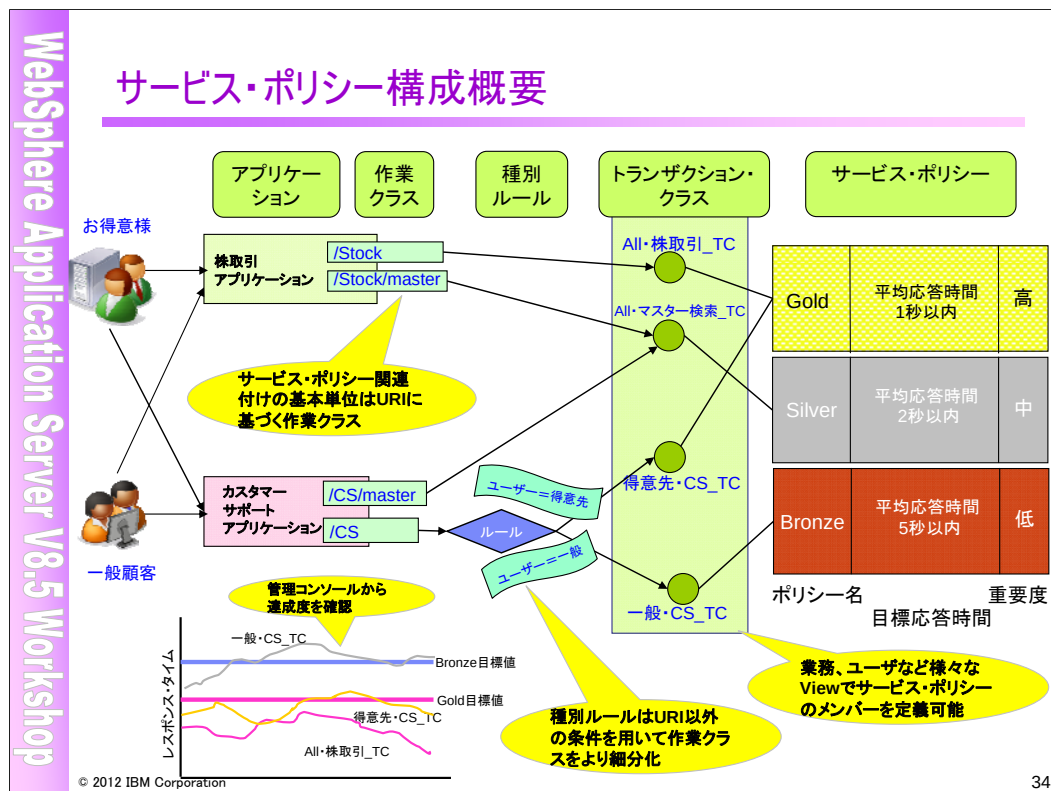
33

インテリジェント・ルーティングを実現するため、サービス・ポリシーを設定します。

ビジネス目標としてレスポンスタイムの目標値と重要度を管理者が簡単に指定可能です。管理者が設定した目標値と重要度を遵守するために、オンデマンド・ルーターにて流量制御を行い、サーバーリソースプール内のリソースを調整し、リクエストに応じてアプリケーション(JVM)を動的に起動・停止します。これらのビジネス目標はサービス・ポリシーとして定義します。

サービス・ポリシーは業務アプリケーションと結びつけることができるため、業務単位でSLAを設定することができます。さらに、一つのアプリケーション(業務)でも複数のサービス・ポリシーを定義することも可能です。こちらは、ユーザーによって優先順位の差を付けたい場合などに有効です。

SLAを満たしているかどうかを管理者はグラフでモニタリングが可能です。この機能は管理コンソールより提供されます。



サービス・ポリシーの設定は複数設定項目がありますが、設定の目的はどのユーザー、アプリケーションをどのサービス・ポリシーに結びつけるかということです。

株取引アプリケーションは、お得意様からでも一般顧客からでもユーザーに限らず全てGoldサービス・ポリシーで1秒以内にレスポンスタイムを返すようにSLAを結びます。

カスタマーサポートアプリケーションは、一般顧客からの場合はBronzeサービス・ポリシーで5秒以内です。しかし、ユーザーがお得意様の場合は、Goldサービス・ポリシーという定義が可能です。このユーザーごとのポリシーの振り分けは、種別ルールという定義の中で行うことが可能です。

さらに、両アプリケーションともにマスター検索を行いたい場合は、Silverサービス・ポリシーに遵守するよう定義しています。異なるアプリケーションでも同じサービス・ポリシーを設定することが可能です。これは、アプリケーション内で設定する作業クラスのURIをわけることにより可能となります。

このように、サービス・ポリシーは業務アプリケーション単位、業務アプリケーション＋ユーザー単位、異なるアプリケーションに跨った単位など様々なViewで定義することができます。これらのViewがトランザクション・クラスとなります。SLA管理者はSLAを遵守しているか否かはトランザクション・クラスをベースに確認を行います。下のグラフはトランザクション・クラスがサービス・ポリシーの目標値を達成しているかを確認できるグラフです。このグラフは管理コンソールで提供されています。インテリジェント管理機能これらの目標値を超えないようリクエストの優先順位付けや、リクエスト数に応じたアプリケーションの動的起動・停止を行います。

サービス・ポリシーの定義

- サービス・ポリシー = **応答時間目標値と重要度**の組合せ
 - ◆ 応答時間目標値: 平均値、百分位応答時間、任意(=設定しない)
 - ◆ 重要度: 7段階
 - ◆ サービス・ポリシーは、トランザクション・クラスを介して、アプリケーションごとのリクエストと結び付ける
- 動作
 - ◆ 重要度のより高いサービス・ポリシーを満たすように以下を実施
 - 重要度に基づくリクエストの優先度別流量制御
 - サービス・ポリシー単位キューイングを行う
 - バックエンドの過負荷を検知した場合キューイング拒否
 - アプリケーションの動的配置(高優先アプリケーションをホストするAPサーバーの稼働、必要に応じてAPサーバーの停止)



© 2012 IBM Corporation

35

サービス・ポリシーでは遵守するビジネス目標値をレスポンス・タイムと重要度の組み合わせで指定します。

サービス・ポリシーは、アプリケーションではなく、[動作ポリシー]-[サービス・ポリシー]より設定します。ここでは、サービス・ポリシー名を任意に決定し、重要度は7タイプから選択します。目標タイプはレスポンス・タイムあるいは、リクエストの何%中が何秒以内という設定を行います。

サービス・ポリシー決定にあたり、動的クラスター内でプールするサーバー・キャパシティ以上のパフォーマンスは出せないということ念頭においてください。決定前に本番機と同じハードウェア・スペックのマシンで本番アプリケーションを使用したパフォーマンス・テストを実施後サービス・ポリシーの定義を行ってください。

【参考】トランザクション・クラス

- トランザクション・クラス
 - ◆ サービス・ポリシーの別名
 - ◆ 各アプリケーションの作業クラスにてトランザクション・クラスを指定
- トランザクション・クラスを対象に管理者はモニタリング
 - ◆ サービス・レベルを監視

選択	名前	重要度	目標
☐	Bronze_SP	低	平均応答 3 秒
☐	Default_SP		任意
☐	Gold_SP	高	平均応答 1500 ミリ秒
☐	Platinum_SP	最高	平均応答 1250 ミリ秒
☐	Silver_SP	中	平均応答 2 秒

トランザクション・クラスの作成

[サービス・ポリシー](#) > Bronze_SP

サービス・ポリシーのプロパティを定義し、トラ

報告書

トランザクション・クラス

このサービス・ポリシーに関連付けるトランザクション・クラスを指す

☒ 作業クラスによるアプリケーション要求の分類

Bronze_SP のメンバー:

General_CS_TC

作業クラスと
トランザクション・クラスを
関連付け

アプリケーション定義 サービス・ポリシー

エンタープライズ・アプリケーション > XDSTrack

このページを使用して、エンタープライズ・アプリケーションを構成します。リンクをクリックする
 ことできます。

概要表 オペレーション 構成 サービス・ポリシー ルーティング・ポリシー

StockTrade_WC

HTTP 更新が一致する場合
 HTTP パターン1

HTTP パターンの編集

次の個別ルールを適用

ルール	操作	ルール名
1		ルールエディタ

ルールエディタ

ルールを編集

ルールを削除

適用するルール名:

priorityCustomer = 'PriorityCustomer'

次に、トランザクションクラスに結合します Priority_CS_TC (Gold, SP)

ルールを検証 取り消す

ルールを適用する場合は、次のトランザクションクラスに分割する
 トランザクションクラスの選択

General CS_TC (Bronze SP)

サービス・レベルを定義ためのビューが、トランザクション・クラスになります。

トランザクション・クラスはサービス・ポリシーと作業クラスをマッピングします。

設定はサービス・ポリシー定義内で行います。

[動作ポリシー]-[サービス・ポリシー]-[サービス・ポリシー名] にてトランザクション・クラス名を指定します。

ここで設定後、アプリケーションのサービス・ポリシー定義において、作業クラス内でトランザクション・クラスを指定することで、作業クラスとサービス・ポリシーをマッピングします。

【参考】作業クラス

- アプリケーション単位・各プロトコル用に作成され、そこにルールによるポリシーとのマッピング(サービス・ポリシー/ルーティング・ポリシー)を設定
 - ◆ サービス・ポリシー: HTTP, SOAP, SIP, IIOP, JMS / ルーティング・ポリシー: HTTP, SOAP
- サービス・ポリシーのHTTP要求用作業クラスでの設定
 - ◆ URLごとに作成
 - ◆ アプリケーション導入時にコンテキスト・ルートのURL用デフォルト・作業クラスを自動作成
 - ◆ ワーク種別ルールによるトランザクション・クラスへのマップを定義

アプリケーション設定

エンタープライズ・アプリケーション > XDStock

このページを使用して、エンタープライズ・アプリケーションを構成します。リンクをクリックするとアクセスできます。

設定書 オペレーション 構成 サービス・ポリシー ルーティング・ポリシー

すべてのサービス・ポリシーに紐着するすべてのアプリケーション作業クラスのマッピングを表示

サービス・ポリシーとアプリケーション作業クラスの関連付け

適用 OK リセット 取り消し

HTTP 要求用の作業クラス

SOAP 要求用の作業クラス

IIOP 要求用の作業クラス

JMS 要求用の作業クラス

HTTP 要求が一括する場合

HTTP パターン:

/StockTrade/IOBound (StockTrade.war)

/StockTrade/memBound (StockTrade.war)

/StockTrade/CPUBound (StockTrade.war)

/StockTrade/sleepBound (StockTrade.war)

/StockTrade/CpuAndSleepBound (StockTrade.war)

トランザクション・クラスを指定

URLを指定

割り振り先を指定

エンタープライズ・アプリケーション > XDStock

このページを使用して、エンタープライズ・アプリケーションを構成します。リンクをクリックするとアクセスできます。

設定書 オペレーション 構成 サービス・ポリシー ルーティング・ポリシー

アプリケーションのルーティング・ポリシーとアプリケーションのすべてのエディションの定義

適用 OK リセット 取り消し

HTTP 要求用の作業クラス

Default_HTTP_WC

StockQuery_WC

HTTP 要求が一括する場合

HTTP パターン:

/StockQuery/IOBound (StockQuery.war)

/StockQuery/memBound (StockQuery.war)

/StockQuery/CPUBound (StockQuery.war)

/StockQuery/sleepBound (StockQuery.war)

/StockQuery/CpuAndSleepBound (StockQuery.war)

割り振り先を指定

© 2012 IBM Corporation

37

作業クラスはトランザクション・クラスあるいは種別ルールにマッピングを行うための設定です。

設定はアプリケーションの定義内で行います。管理コンソールの日本語表示は作業クラスです。

[エンタープライズ・アプリケーション]-[アプリケーション名]の[サービス・ポリシー]タブより、サービス・ポリシーがサポートしているプロトコル(HTTP,SOAP,SIP,IIOP,JMS)単位で作業クラスの作成が可能です。

HTTPリクエスト用の作業クラスであれば、アプリケーション内のURLを指定します。SOAPリクエストの場合、WSDLオペレーションを指定します。IIOP、JMSプロトコルの場合、EJBメソッドを指定します。

ルーティング・ポリシーはオンデマンド・ルーターがルーティング・ポリシーで定義されたルールに従って割り振り先を変更します。ルーティング・ポリシーを使用する場合もアプリケーション定義内で作業クラスを作成します。ルーティング・ポリシーはHTTPリクエストかSOAPリクエストに対してのみ設定可能です。

【参考】種別ルール

- サービス・ポリシー、あるいはルーティング・ポリシーにおける条件別ルールの設定
 - ◆ サービス・ポリシー: トランザクション・クラスの選択
 - ◆ ルーティング・ポリシー: アクション、アプリケーション・エディションの選択

アプリケーション設定

エンタープライズ・アプリケーション >> XDStock

このページを使用して、エンタープライズ・アプリケーションを構成します。リンクをクリックして、リンク先を開きます。

種別ルールの条件

オペランドの選択:

- ユーザー ID
- グループ ID
- ロール
- 要求照会パラメーター名
- Cookie ヘッダー名
- MIME タイプ
- HTTP メソッド
- クライアント・ホスト
- クライアント IP V4
- クライアント IP V6
- サーバー ホスト
- サーバー IP V4
- サーバー IP V6
- ポート
- プロトコル
- 仮想ホスト
- 時間

トランザクション・クラスを選択

条件によりトランザクション・クラスを変更

条件によりルーティング・アクションを変更

アクションを選択

ルールは複数指定可能

条件、トランザクション・クラスを指定

条件によりトランザクション・クラスを変更

条件によりルーティング・アクションを変更

アクションを選択

条件にヒットしない場合のトランザクション・クラス

© 2012 IBM Corporation

38

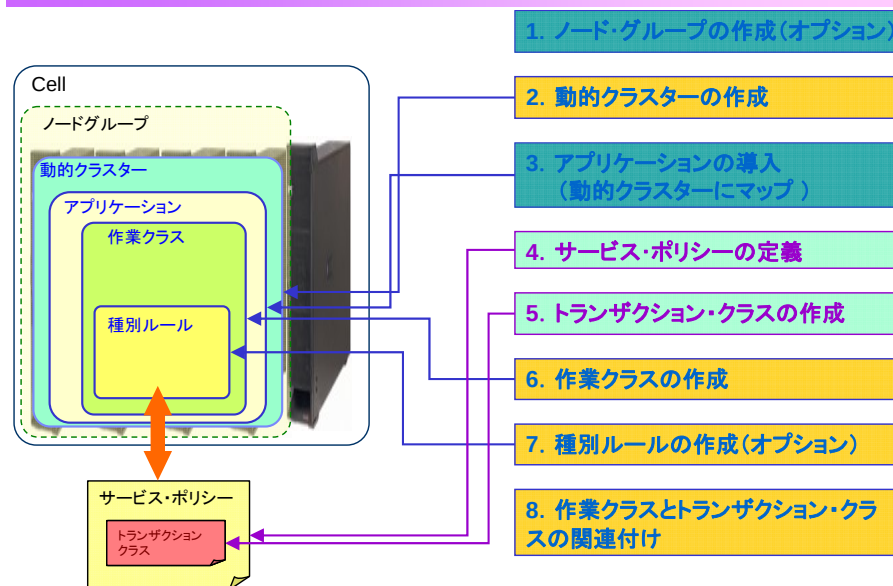
種別ルールは、ルール条件によってトランザクション・クラスを変更する、あるいはオンデマンド・ルーターからの割り振り先を変更する場合に設定します。種別ルールの設定は必須ではありません。

アプリケーション設定内のサービス・ポリシーあるいはルーティング・ポリシーの作業クラス内で定義します。

種別ルールを作成するには、ルール・ビルダー(ビルド副次式)を利用します。ルール条件としてはプロトコルによっても異なりますが、HTTPリクエストの場合チャートの真ん中にある「種別ルールの条件」より選択可能です。サービス・ポリシーの場合は、ルールに応じたトランザクション・クラスを指定します。ルールは複数作成することができ、ルール順序に従ってルールが適用されます。ルールにマッチしない場合のトランザクション・クラスも選択します。

ルーティング・ポリシーの場合はアクションを選択します。アクションでは、リクエストを転送、アフィニティーを効かせたリクエスト転送、転送先の変更、エラー・コードなどをクライアントに返しリクエストを拒否の中から選択が可能です。

インテリジェント・ルーティング構成手順



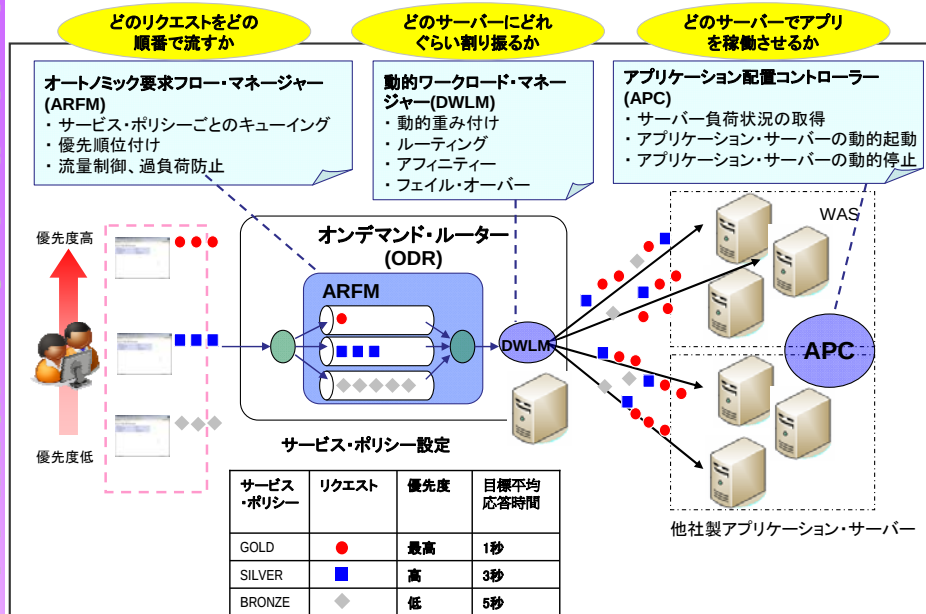
© 2012 IBM Corporation

39

インテリジェント・ルーティング環境を構築する手順は全部で8ステップあります。

1. セル内にノードグループの作成(オプション)
2. ノードグループ内に動的クラスターを作成
3. アプリケーションをインストールし、動的クラスターにマッピング
4. アプリケーションとは別にサービス・ポリシーを定義
5. サービス・ポリシー定義内でトランザクション・クラスを作成
6. アプリケーション定義内でプロトコル単位の作業クラスを作成
7. 作業クラス内で必要に応じ種別ルールを作成
8. 最後に作業クラス定義内あるいは種別ルール定義内でトランザクション・クラスを指定することで関連付けを行います。

インテリジェント管理関連コンポーネント



© 2012 IBM Corporation

40

インテリジェント管理関連コンポーネントの概要について説明します。

オンデマンド・ルーター (ODR: On-Demand Router) に送られてきたリクエストは先ずサービス・ポリシーごとのキューにキューイングされます。ここでオートノミック要求フロー・マネージャー (ARFM: Application Request Flow Manager) によって計算されたキューのウェイト値に従って優先順位付けされ、流量制御されます。ARFMにより、どの作業クラスのリクエストをどれだけ送るかが決定されます。

次に動的ワークロード管理 (DWLM: Dynamic Workload Manager) によって動的に重み付けられたバックエンド・ノードに振り分けられます。DWLMは動的な重み付けによるルーティングの他、セッション・アフィニティー、サーバー回復時のフェイル・オーバー機能も持ちます。

バックエンド・ノードではアプリケーション配置コントローラー (APC: Application Placement Controller) がAppServerを動的に起動したり縮退したりするプロビジョニングを行います。APCはサーバーの負荷状況を把握しており、AppServerの起動要求がある場合にどこのノードで起動・停止を行えばよいのかを判断し、実行します。

このように、ARFM, DWLM, そしてAPCの大きく3つのコンポーネントによって、動作ポリシーで設定したパフォーマンス・ゴールおよびビジネス・ゴールを達成するように自動的にフローやリソースが調整されます。

【参考】インテリジェント管理関連コンポーネント

- **オートノミック要求フロー・マネージャー (ARFM)**
 - ◆ オンデマンド・ルーター(ODR)に到着したリクエストは、サービス・ポリシー(重要度 & 目標応答時間)ごとのキューに入れられる(長さはデフォルト1000)
 - ◆ 各サービス・ポリシーごとに、リクエスト処理に要したCPUリソース量を計測
 - ◆ 到着したリクエストの必要リソース量(CPU/メモリー等)を計算し、不足が想定される場合にはキューで保持(直ちに拒否するオプションもあり)
 - ◆ サービス・ポリシーに基づき、目標値を達成するようにキューからサーバーへディスパッチ
 - ◆ 高い重要度のサービス・ポリシーを優先
- **動的ワークロード・マネージャー (DWLM)**
 - ◆ サーバーの負荷状況を分析して、各サーバーへの割振りの重み付けを決定
 - ◆ 重み付けラウンド・ロビン方式で、リクエストをルーティング
 - ◆ オン/オフが可能(デフォルトでは、動的クラスターではオン、静的クラスターではオフ)
- **アプリケーション配置コントローラー (APC)**
 - ◆ ARFMからのCPUリソース不足の通知を受けて、必要ならアプリケーションの配置を変更
 - ◆ 配置変更の決定に基づき、動的クラスター・メンバーの起動/停止を実行

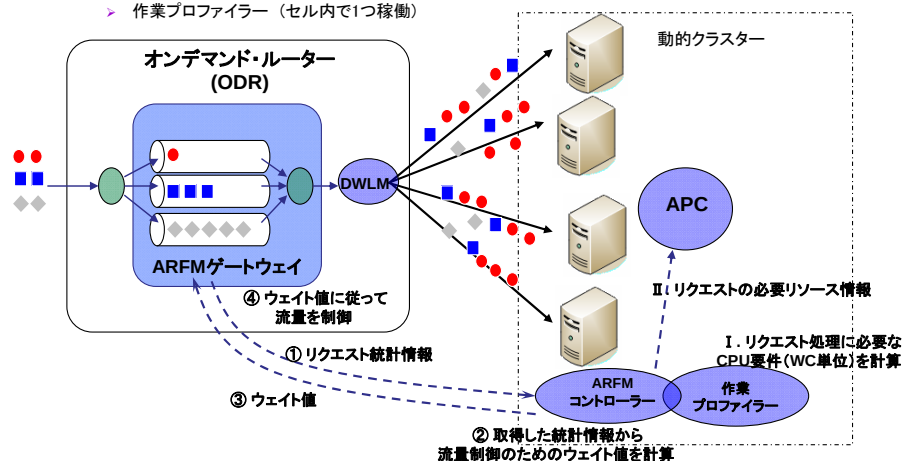
上記3つのコンポーネントが「オートノミック・マネージャー」とも呼ばれます。

オートノミック要求フロー・マネージャー (ARFM)

- サービス・ポリシーに基づいてリクエストをキューイングし、流量制御を行うオートノミック・マネージャー

- ◆ オートノミック要求フロー・マネージャー(ARFM)を構成するコンポーネント

- ゲートウェイ (すべてのODRで稼働)
- コントローラー (セル内で1つ稼働)
- 作業プロファイラー (セル内で1つ稼働)



© 2012 IBM Corporation

42

オートノミック要求フロー・マネージャー(ARFM)について説明します。ARFMはサービス・ポリシーに基づいて、各リクエストをキューイングし、優先順位付けし流量制御を行うコンポーネントです。ARFMはゲートウェイ、コントローラー、作業プロファイラーという3つのコンポーネントから構成されています。

ゲートウェイはODR内に存在するコンポーネントです。機能としてはサービス・ポリシーごとに自動生成したキューにリクエストをキューイングすることです。リクエストはサービス・ポリシーごとにキューに入り、処理を待ちます。ゲートウェイはキュー上のリクエストをコントローラーが計算したウェイト値(重み)に応じて優先順位付けし、流量を制御します。

コントローラーは、ゲートウェイによってサービス・ポリシーごとにキューイングされたリクエスト・フローの最適化を担うコンポーネントです。コントローラーは設定されたサービス・ポリシーのゴールと取得した実際のリクエストの統計情報からキューのウェイト値を計算し、BBSON(Bulletin Board Service Overlay Network: WAS内部でルーティング情報を交換するための電子掲示板)経由でゲートウェイに通知します。コントローラーが利用する統計情報とはリクエストの到達率、サービス・タイム(リクエストがODRからアプリケーション・サーバーに送信されて処理を終えて帰ってくるまでの時間)、レスポンス・タイム(リクエストがODRに入ってきてからアプリケーション・サーバーに送信されて処理を終えて帰ってくるまでの時間)のことを言います。

また、作業プロファイラーは常にリクエストの処理に必要なCPU要件(作業クラス単位)を計算しており、必要なリソース情報をコントローラー経由でアプリケーション配置コントローラー(APC)に配信しています。

APCはこの情報をもとにアプリケーション・サーバーの起動/停止を実行することになります。

過負荷防止機能

■ ARFMの過負荷防止機能

◆ メモリー過負荷保護

- サーバーのヒープ使用率が設定値以上になる場合にリクエストのキューイングを拒否
 - サーバー・アフィニティを持つ場合はキューイングされ、拒否はされない

◆ CPU過負荷保護

- サーバーのCPU過負荷を検知し、サービス・ポリシー違反が発生する可能性がある場合の対応を次の3つのポリシーから設定可能
 - 要求拒否せず、全てのリクエストをキューイング(デフォルト)
 - リクエストをキューイングせずに、即時に拒否
 - 設定した割合を越えるリクエストの応答時間が目標値を超える場合に拒否
- 要求拒否した場合、既定ではHTTPコード503を応答(コードは変更可能)
- 過負荷の判断のしきい値となるCPU使用率を設定可能(デフォルトは90%)

【参考】過負荷防止設定

オートノミック要求フロー・マネージャー

オートノミック要求フロー・マネージャーは、メッセージがドルウェア・サーバーによってサービスできるかどうかとその時間を判断して、アプリケーションの通信メッセージを管理します。この管理アクションは、HTTP および Session Initiation Protocol (SIP) トラフィック用のオンデマンド・ルーターと、Java Message Service (JMS) および Inter-ORB Protocol (IIOP) メッセージ用のアプリケーション・サーバーで行われます。

構成

一般プロパティ

- ※ 集約期間: 5 秒
- ※ コントロール・サイクルの最小長: 59 秒
- ※ 平滑化ウィンドウ: 12 集約期間の数
- ※ 最大キュー長: 1000 要求
- ※ メモリー過負荷防止: WebSphere Application Server ヒープ・サイズの使用率: 100 %

メモリ使用率の過負荷条件

CPU 過負荷防止

ノードは、メッセージをキューイングおよび経路指定することによって、CPU の過負荷を防止します。指定された最大 CPU 使用率を超えると、CPU は過負荷とみなされます。

- ※ 最大 CPU 使用率: 90 %

CPU使用率の過負荷条件

拒否ポリシーで、HTTP または SIP トラフィックの既存のダイアログまたはセッションの一部でない通信 HTTP または SIP メッセージを拒否することにより、CPU が過負荷になることを防止します。

拒否ポリシー:

- ☒ 通信メッセージの拒否なしでは、タイムアウトになったリソース・ポリシー目標の応答時間しきい値に違反する可能性があることには関係なく、すべてのメッセージをキューに入れることができます。
- ☐ メッセージの予測応答時間がそのサービス・ポリシー目標の応答時間しきい値を超える場合、既存のダイアログまたはセッションの一部でない通信メッセージを拒否します。
- ☐ メッセージの予測応答時間がそのサービス・ポリシー目標の応答時間しきい値を次のパーセンテージよりも多く超える場合、既存のダイアログまたはセッションの一部でない通信メッセージを拒否します:

拒否しきい値: -1 %

CPU過負荷防止のための要求拒否ポリシー:

- 要求拒否しない
- 即時拒否
- 応答時間しきい値設定

© 2012 IBM Corporation

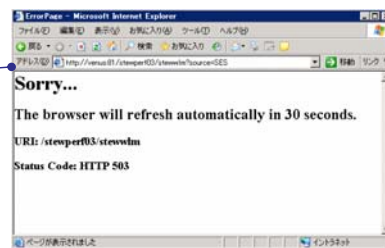
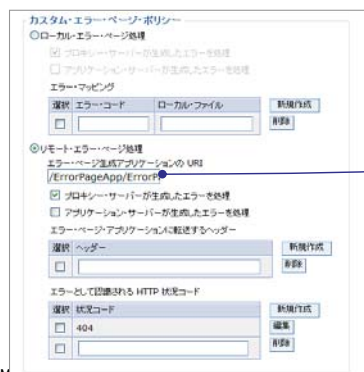
44

過負荷防止のARFMの設定画面です。

管理コンソールで、[動作ポリシー]-[オートノミック・マネージャー]-[オートノミック要求フロー・マネージャー]で設定します。

【参考】ODR カスタム・エラーページ

- 要求拒否などの場合にエラー・ページへ転送
- エラー発生時のエラーページをカスタマイズ
 - ◆ ローカル・エラー・ページ処理：ローカル・ファイルを応答
 - ◆ リモート・エラー・ページ：エラーページ生成アプリケーションに転送
 - エラー・ページ生成アプリケーションは、ODR自身にデプロイすることも可能
 - エラー・ページ・アプリケーションのサンプル：<WAS_HOME>/installableApps/HttpErrorHandler.ear
- カスタム・エラー・ページの構成方法：
 - ◆ [サーバー]-[オンデマンド・ルーター]-[オンデマンド・ルーター名]-[カスタム・エラー・ページ・ポリシー]



© 2012 IBM

45

要求拒否の際に転送するエラー・ページ・アプリケーションのテンプレートとして以下のEARが用意されています。

<WAS_Home>/bin/installableApps/HttpErrorHandler.ear

ODRへのアプリケーション導入／起動は以下のwsadminコマンドで実行することができます。

```
wsadmin> $AdminApp install <WAS_HOME>/installableApps/HttpErrorHandler.ear {-server myServer -node myNode}
```

```
wsadmin> $AdminConfig save
```

```
wsadmin> set appManager [$AdminControl queryNames  
cell=mycell,node=mynode,type=ApplicationManager,process=odr_server,*]
```

```
wsadmin> $AdminControl invoke $appManager startApplication HttpErrorHandler
```

【参考】オートノミック要求フロー・マネージャー (ARFM)の調整

管理コンソール: [動作ポリシー]-[オートノミック・マネージャー]-[オートノミック要求フロー・マネージャー]

オートノミック要求フロー・マネージャー

オートノミック要求フロー・マネージャーは、メッセージがミドルウェア・サーバーによってサービスできるかどうかとその時期を判断して、アプリケーションへの着信メッセージを管理します。この管理アクションは、HTTP および Session Initiation Protocol (SIP) トラフィック用のオンデマンド・ルーターと、Java Message Service (JMS) および Inter-ORB Protocol (IIOP) メッセージ用のアプリケーション・サーバーで行われます。

構成

一般プロパティ

- * 集約期間: 5 秒
- * コントロール・サイクルの最小長: 59 秒
- * 平滑化ウィンドウ: 12 集約期間の値
- * 最大キュー長: 1000 要求

ゲートウェイがリクエストの統計情報を取得し公開する間隔

コントローラーがサービス状況を収集しキューのウェイト計算などを実行する間隔

コントローラーがゲートウェイから受け取る最後のいくつかの統計データから計算するかを指定

ゲートウェイのキューの最大キュー長。キューが最大値に達すると要求はリジェクトされる

統計情報取得/配信

ARFMゲートウェイ

集約期間

ARFMコントローラー

活動化ウェイト値計算

コントロール・サイクル

平滑化ウィンドウ

活動化ウェイト値計算

© 2012 IBM Corporation

46

管理コンソールにおいて、[動作ポリシー]-[オートノミック・マネージャー]-[オートノミック要求フロー・マネージャー]を選択するとARFMの動作をパラメータで調整することができます。以下が、調整可能なパラメータです。

集約期間:ゲートウェイの役割のページで記述したようにゲートウェイは定期的にリクエスト統計情報を収集しBSONに公開しています。集約期間はこの間隔を指定します。集約期間を小さくしすぎるとサンプルとなるリクエストの数が少なく、パフォーマンスの測定基準が信頼性の低いものになります。また、集約期間を長くしすぎると制御設定の再計算の回数が少なくなり、リクエストのトラフィック量の急な変更に対応できなくなる可能性があります。集約期間はデフォルトでは5秒となっています。

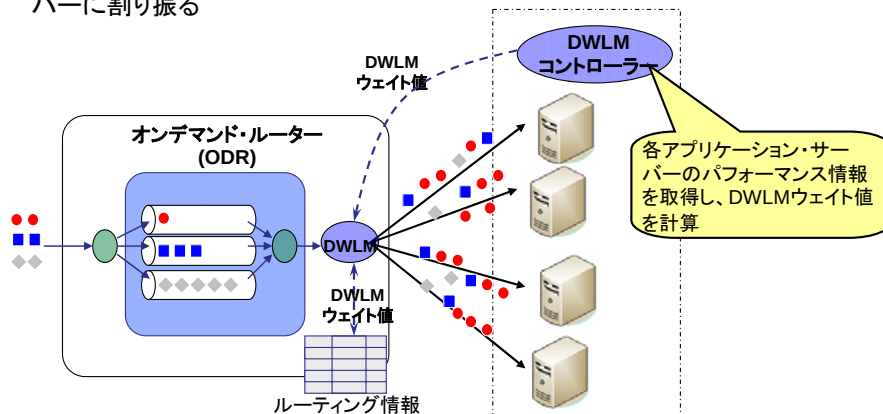
コントロール・サイクルの最小長:コントローラーはサービス状況を監視し、ゲートウェイからのリクエスト統計情報を取り込み、定期的にサービス・ポリシーごとのキューのウェイトを計算して公開します。このパラメータはその実行を行う時間間隔の下限値を指定します。このパラメーターも集約期間同様、小さくするとパフォーマンス測定基準の信頼性が低いものになる可能性があり、大きすぎると急なトラフィック変更に対応できなくなる可能性があります。デフォルトは59秒となっています。

平滑化ウィンドウ:コントローラーが定期的にサービス・ポリシーのキューのウェイト値を計算する際、ゲートウェイからのリクエスト統計レポートの最後の数個の実行平均を基に算出します。平滑化ウィンドウはそれをいくつまで含めるかを指定します。これを大きくすると急激なトラフィック変更は平滑化され、小さくすると急激なトラフィック変更が速く反映されることになります。平滑化ウィンドウと集約期間の積がコントロール・サイクルの間隔となるように(つまりコントロール・サイクル間隔内のすべてのリクエスト統計を計算に含めるように)設定することが推奨されています。

最大キュー長:ゲートウェイのキューの最大キュー長を表します。キューが最大値に達するとリクエストは拒否されます。拒否された要求はHTTP状況コード500で返るので、これをODRのカスタム・エラー・ページ・ポリシー(後述)で設定することにより、エラー・ページにリダイレクトすることも可能です。デフォルトは1000となっています。

動的ワークロード・マネージャー (DWLM)

- DWLMコントローラーは、リクエスト割振りのDWLMウェイト値を各ノードのパフォーマンス情報(CPU使用率、スループット、同時リクエスト数など)から動的に計算
- DWLMは、コントローラーからの結果に従い、動的にルーティング・テーブルのDWLMウェイト値を更新
- DWLMは、ルーティング・テーブルのDWLMウェイト値に従ってリクエストを各サーバーに割り振る



© 2012 IBM Corporation

47

動的ワークロード管理(DWLM)について説明します。実際の割り振りを行うDWLMの実体はODR内に存在しますが、この割り振りのウェイトを計算しDWLMにレポートするコンポーネントがDWLMコントローラーです。

DWLMコントローラーはパフォーマンス情報(ノードのCPU使用率、アプリケーション・サーバーのスループット、アプリケーション・サーバーの同時リクエスト数など)からアプリケーション・サーバー毎の重み(ウェイト)を自動的に計算し、DWLMに定期的(20秒毎)にレポートします。DWLMはこのウェイト値にしたがってリクエストの各アプリケーション・サーバーへの割り振りを行います。DWLMコントローラーはどこかのノード・エージェントまたはアプリケーション・サーバー上でHA Managerの管理のもとで稼働しています。したがって、DWLMコントローラーが稼働しているプロセスが落ちたとしてもHA Managerによってフェイル・オーバーされます。DWLMは動的クラスターではデフォルトで利用可能に設定されています。通常の静的クラスターに対してもDWLMを動作させることができますが、デフォルトでは利用可能にはなっていません。

動的クラスターの使用可能の有無を設定するには管理コンソールから [サーバー]-[クラスター]-[動的クラスター]-[動的クラスター名]-[追加プロパティ: 動的ワークロード管理(DWLM)] を選択します。

【参考】セッション・リバランス機能 (DWLM)

- DWLMはセッション数がDWLMウェイト値と比例するようにリバランスを行う
- セッション移動の必要なリクエストのクローンIDを振りなおして転送する

Cookie: WSJSESSIONID
= CacheID:Session24:P2:C1

- 特定少数ユーザーが長時間利用するようなシステムにおいて有効
- セッション・アフィニティがオンで、かつ、セッション・パーシスタントを使用している場合に使用可能
- 動的クラスターに関して、デフォルトでオン(カスタムプロパティ HttpSessionRebalanceOffをtrueとすることでオフに設定可能)



© 2012 IBM Corporation

48

次にセッション・リバランス機能について説明します。

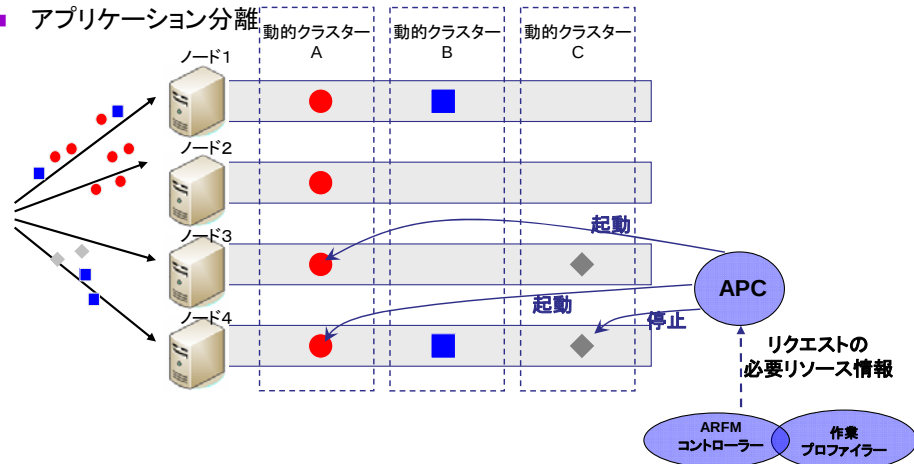
図において、サーバー1が何らかの理由(障害／縮退など)により停止している場合、リクエストはサーバー2～4に振られてそれぞれのサーバーにアフィニティされます。つまり、各リクエストにはクローンID:C2, C3, C4が振られています。ここでサーバー1が回復あるいはプロビジョニングにより起動したとします。このとき仮にアフィニティが効き続けるとすると既存のリクエストは相変わらずサーバー2～4に振られ続けることになります。不特定多数のユーザーが利用するシステムで新規ユーザーが次々とリクエストして来る場合はそのリクエストがサーバー1に振られていずれバランスされますが、特定少数のユーザーが比較的長時間利用するようなシステムの場合、セッションのアンバランスが継続することになります。このような状況を解決する機能がセッション・リバランス機能になります。

DWLMは各サーバーのセッション数がDWLMウェイト値と比例するように各サーバーでのセッション移動数を決定します。セッションの移動が決まった場合、DWLMはアフィニティされているリクエストのクローンIDをC2, C3, C4からC1に振り直します。これによりリクエストは必要な数だけサーバー1に振り直され、セッション・リバランスが実現されます。

セッション・リバランスはURL rewritingやSSL IDによるセッション管理をしている場合はサポートされません。また、POSTリクエストやODRを介さずダイレクトにAppServerにアクセスするリクエストに対しては機能しません。

アプリケーション配置コントローラー (APC)

- ARFMコントローラーの必要リソース情報に従い、アプリケーション・サーバー・インスタンスを動的に起動・停止
- ノードの負荷状況(メモリーやCPU負荷など)から、どのインスタンスを起動するかを決定
- アプリケーション分離



© 2012 IBM Corporation

49

3つめのオートノミック・マネージャーがアプリケーション配置コントローラー (APC) です。APCはARFMコントローラー／作業プロファイラーからの必要リソース情報よりその要件を満たすためにAppServerインスタンスを自動的に起動したり停止したりします。APCはノードの負荷状況(メモリーやCPUの使用率)をみて、どのノードにインスタンスを起動すればよいかを判断し実行します。

上の図において動的クラスターAのアプリケーションのリクエストが増加し、リソース追加要求が来たとき、APCはサーバー状況からノード3での始動が最適と判断し実行します。さらに、このリクエストが増加した際、始動できるのはノード4になりますが、このノードは既にB, CのAppServerが稼働しておりメモリー上、逼迫しています。APCはA, B, Cのサービス・ポリシーの達成状況からCのAppServerを停止(縮退)し、AのAppServerの起動することを判断し実行します。

APCが起動・停止の優先順位を判断する際、まずサービス・ポリシーのパフォーマンス・ゴールが達成できているかどうか優先されます。ゴールを達成していないアプリケーションを処理するAppServerが優先的に起動され、その他のアプリケーションはそのために縮退される可能性があります。もし、A,B,Cいずれも逼迫しておりパフォーマンス・ゴールを達成できていない場合は、それぞれのサービス・ポリシーの重要度の高い方が優先的にAppServerの増加が行われ、低い方はそのために縮退される可能性があります。

WebSphere Application Server V8.5 Workshop

【参考】アプリケーション配置コントローラー (APC) 動作モード

- 動的クラスターごとにAPCの動作モードを指定することができる
- 設定方法
サーバー > 動的クラスター > モードの設定

動作モードの選択
 - 手動: インスタンスの起動・停止は手動で行う
 - 監視: APCによる起動／停止要求はランタイム・タスクにリストされ、管理者による受諾、否認を待機する
 - 自動: APCは起動・停止を自動的に行う

ランタイム・タスク
 監視モードのとき、APCによるインスタンスの起動・停止要求がリストされる。受諾すると要求された起動・停止命令が実行される。

© 2012 IBM Corporation
50

動的クラスターごとに**APC**の動作モードを設定することができます。動作モードは手動、監視、自動の3モードから動的に設定できるようになっています。

手動モード: AppServerインスタンスの起動・停止は管理者によって手動で行う。(APCは何もしない)

監視モード: APCによるAppServerインスタンスの起動／停止要求は管理コンソールのランタイム・タスクにリストされます。管理者はその要求を受諾するか拒否するかを判断します。

自動モード: APCはAppServerインスタンスの起動・停止を自動的に行います。起動・停止要求のメッセージだけはランタイム・タスクにリストされます。

監視モードのときは下の画面にあるように、ランタイム・タスクに起動／停止要求が届き、管理者による受諾／拒否を待ちます。

【参考】アプリケーション配置コントローラー (APC) 設定

[動作ポリシー]-[オートノミック・マネージャー]-[アプリケーション配置コントローラー]

アプリケーション配置コントローラー

このページは、アプリケーション配置コントローラーを構成する場合に使用します。アプリケーション配置コントローラーによって、動的クラスターが監視モードまたは自動モードで管理されます。

構成: **ランタイム**

APC機能を使用(チェック)を外した場合は、全ての動的クラスターは静的クラスターと同等になる

一般プロパティ

☒ 使用可能

承認タイムアウト: 10 分

監視モード時、この時間を過ぎても承認が得られない場合、ランタイム・タスクは期限切れとなる

サーバー操作タイムアウト: 5 分

サーバーの起動(停止)を指示し、この時間を過ぎても起動(停止)しない場合その操作を失敗とみなす

配置変更間の最小時間: 15 分

Elasticity モード

☐ Elasticity オペレーションを使用可能にする

自動

前回のサーバー操作あるいはタイムアウト後、次の操作を開始するまでの待機時間

Elasticity オペレーションのタイムアウト: 10 分

© 2012 IBM Corporation

51

アプリケーション配置コントローラーの設定は、[動作ポリシー]-[オートノミック・マネージャー]-[アプリケーション配置コントローラー]で設定します。

インテリジェント・ルーティングとパフォーマンス管理:まとめ

- サービス・ポリシー
 - ◆ サービス・レベル(ビジネス目標)を定義
 - ◆ 応答時間と重要度の組み合わせ
 - ◆ 設定単位はURIに基づく作業クラス
- インテリジェント管理関連コンポーネント
 - ◆ オートノミック要求フロー・マネージャー (ARFM)
 - リクエストの優先順位付け、流量制御、過負荷防止
 - ◆ 動的ワークロード・マネージャー (DWLM)
 - パフォーマンスに応じた重みを動的計算
 - セッション・リバランス機能
 - ◆ アプリケーション配置コントローラー
 - サービス・ポリシーおよびリソース状況に応じてアプリケーションを動的配置し、JVMを自動起動・停止

4. 可視化機能

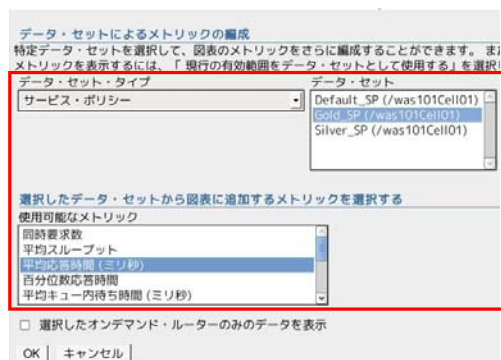
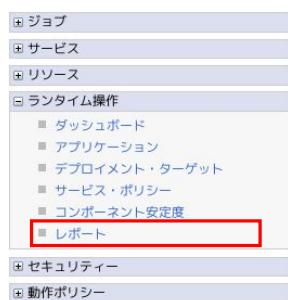
この章の内容

- レポート
- オペレーション・アラート
- ランタイム・タスク

レポート(1/2)

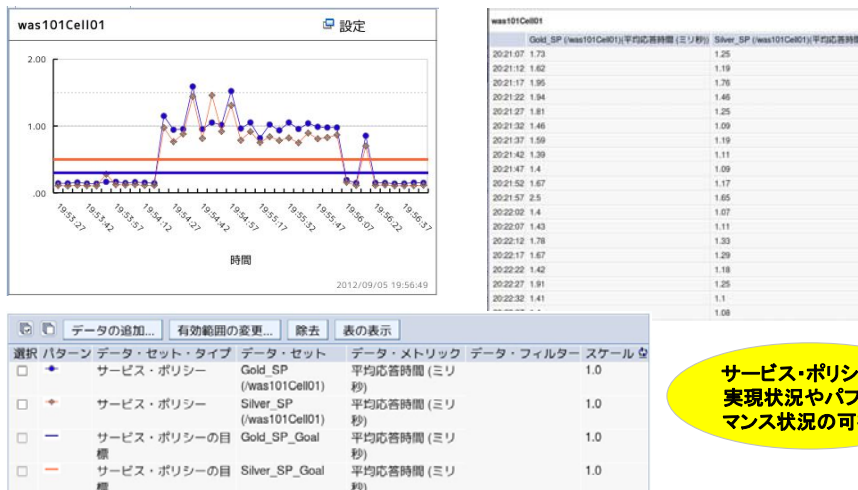
- 管理コンソール: 「ランタイム操作 > レポート > データの追加」で取得したいデータ項目を指定
 - ◆ データ・セット・タイプ
 - 取得データの項目名 (サービス・ポリシーや動的クラスターなど)
 - ◆ データ・セット
 - サービス・ポリシーなど実際に設定されている名前
 - ◆ 使用可能なメトリック
 - 上記で選択したデータ・セットで取得できる統計情報
 - 「平均応答時間」、「平均スループット」、ノードの「CPU使用率」、「合計メモリー」など

サービス・ポリシーを含め多くのパフォーマンス情報を取得可能



レポート(2/2)

- 管理コンソールからパフォーマンス状況を図表でリアルタイムに確認
 - ◆ 管理コンソール: 「ランタイム操作 > レポート」から参照
 - ◆ サービス・ポリシーごとの平均応答時間やノードごとのCPU使用率など様々な情報を見ることができる



サービス・ポリシーの実現状況やパフォーマンス状況の可視化

オペレーション・アラート

- 発生している問題を報告書/サマリービューで表示することで管理者に通知
 - ◆ ランタイム操作以下の各メニュー
 - ◆ 動的クラスター、アプリケーション、サーバーなどのオペレーションタブ
- 問題の種類に応じて
 - ◆ 状況を知
 - ◆ アクションプランを提示

「ランタイム操作」でのオペレーションアラート

オペレーション・アラート

- サーバー DCluster_was102Node01 (/was101Cell01/was102Node01): 注意が必要な未解決のタスクがあります。3の未解決タスクのうち、検出された最高の重大度レベルは「致命的」であり、これが3タスクに当てはまります。詳しくは、ランタイム・タスク・パネルを参照してください。
- サーバー DCluster_was103Node01 (/was101Cell01/was103Node01): 注意が必要な未解決のタスクがあります。3の未解決タスクのうち、検出された最高の重大度レベルは「致命的」であり、これが3タスクに当てはまります。詳しくは、ランタイム・タスク・パネルを参照してください。

「動的クラスター」でのオペレーションアラート

動的クラスター > DCluster

動的クラスターとは、クラスター・メンバーから収集したパフォーマンス情報、メンバー間のワークロードをウェイトを使用しながら動的にバランスを取るサー、クラスター・メンバーに障害が発生すると、要求はクラスターの別のメンバーに動的クラスターは、環境内のワークロードに応じて、インスタンスを開始または

レポート オペレーション 構成

オペレーション・アラート

- クラスター DCluster (/was101Cell01): このサーバーには、あるサービス・ポリシーにアクティブに分類されている作業がありますが、現在そのサービス・ポリシーは、その目標である Gold_SP サービス・ポリシーに違反しています。構成された目標しきい値は3ミリ秒、現行値は12.012ミリ秒です。
- クラスター DCluster (/was101Cell01): このサーバーには、あるサービス・ポリシーにアクティブに分類されている作業がありますが、現在そのサービス・ポリシーは、その目標である Silver_SP サービス・ポリシーに違反しています。構成された目標しきい値は5ミリ秒、現行値は11.426ミリ秒です。

© 2012 IBM Corporation

57

オペレーション・アラートは、発生している問題を通知し、必要に応じてアクションを実行できるようにアドバイスをしてくれる機能です。通知する問題に応じて、状況を知する場合と、アクションプランと一緒に提示するケースがあります。例えば起動していないODRの存在や重大なタスクがあがっていることを通知したり、不安定なサーバー/ノードに対して保守モードを設定するように、といったアドバイスが通知されます。アラートは、管理コンソール上「ランタイム操作」以下の各メニューや、動的クラスターの「オペレーションタブ」などで参照可能です。管理者は管理コンソールに表示されるこのオペレーション・アラートに注意することでも、現在の稼働環境で起きている可能性がある問題を早期に発見し、対応をとることが可能となります。

ランタイム・タスク (1/2)

- ランタイム・コンポーネントによって生成されるイベントを管理
 - ◆ オートノミック・マネージャーによるポリシー違反の検知が記録される
- 監視モードの場合はアクションの受諾を受付
- 管理コンソール「システム管理 > タスク管理 > ランタイム・タスク」で設定

自動モードの場合:
自動アクション実行し通知

選択	アクション	タスク ID	状態	重大度	発生時刻
☐	クローズ	1299763152	新規	クリティカル	07/07/19 22:15:48
	ARFM0261E: セル solarConfCell01 のサービス・ポリシー Platinum_SP が、7/19/07 10:11 PM から重大な違反を受けています。				
☒		3482764260	成功	重大	07/07/19 22:13:28
	DCPC0305I: アプリケーション配置コントローラが、CPU 使用率の値がサービで指定されたしきい値を超えたことを検出しました。 "[mercuryNode01]"。動的クラスター・インスタンス				

監視モードの場合:
サービス・ポリシー違反の通知

DCPC0305I: アプリケーション配置コントローラが、CPU 使用率の値がサービで指定されたしきい値を超えたことを検出しました。
"[mercuryNode01]"。動的クラスター・インスタンス

タスク詳細情報の補足表示

状態を診断するために使用するデータの概要

ターゲット・オブジェクト	ターゲット・タイプ	重大度	ターゲット・モニター
Platinum_SP	serviceclass	クリティカル	Platinum_SP の平均応答時間回表を表示します。

監視モードの場合、ランタイムタスクで管理者がプランされたアクションを受諾してはじめてアクションが実行される

© 2012 IBM Corporation

58

WAS V8.5環境内では、WAS V8.5ランタイム・コンポーネントによってさまざまなタスクが生成されます。主には、オートノミック・マネージャーが検知するポリシー違反などが含まれます。オートノミック・マネージャーによって検知されたポリシー違反は、各オートノミック・マネージャーによってタスクとして生成され、デプロイメント・マネージャーに送信されます。生成されたタスクは管理コンソールのランタイム・タスクから確認できます。ランタイム・タスクはWAS V8.5内で生成されるイベントを管理するツールの位置づけになります。タスクはランタイム・タスクで一覧表示され、個々のタスクの詳細画面へのリンクを含みます。

ランタイム・タスクには主にサービス・ポリシー違反およびヘルス・ポリシー違反がタスクとして通知されます。自動モードに設定されている場合は、ポリシー違反のイベントがまず通知され、続いて実行されたアクションの実行結果が通知されます。管理者は定期的にランタイム・タスクを見てタスクがあがっているかを確認することができます。

また、監視モードに設定されている場合は、前述のとおりアクションは管理者がタスクを受諾するまで実行されません。管理者はタスクの中身を確認し、WAS V8.5によって提案されたアクションの実行を承認する場合は受諾を選択し、サブミットを実行します。

タスク管理では、さらに通知機能を設定することでランタイム・タスクにタスクがあがったことをトリガーに指定したアドレスにe-mailを送信することも可能です。特に監視モードの際に設定することで、アクション実行の受諾までの操作をスムーズに進めることができます。

ランタイム・タスク (2/2)

- ランタイム・タスクの内容を管理コンソールを経由せずに確認する方法として「イベント・ロガー」、「E メール通知」の2種類がある
- イベント・ロガー
 - ◆ ランタイム・タスクで確認できるイベントをログに書き出すことが可能。
 - ◆ セルレベルのカスタム・プロパティで出力設定をする。

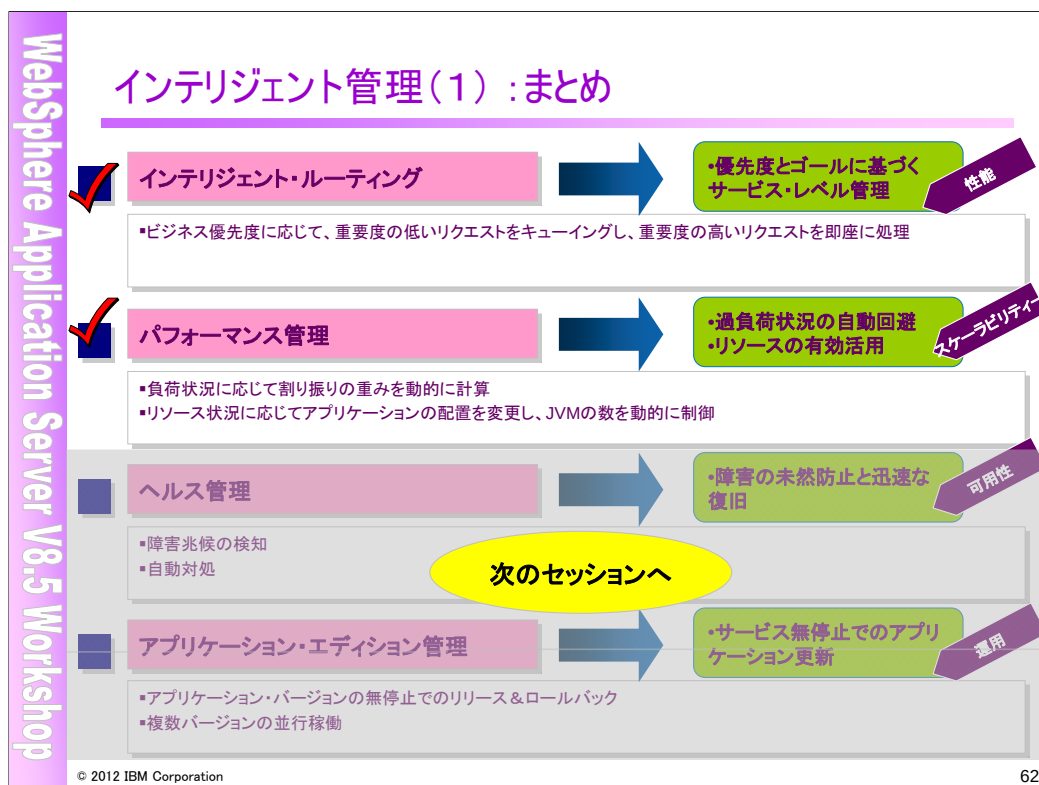
プロパティ名	プロパティ値	説明
tmslog.enable	false	タスク管理サービス・イベントのログギングを有効にするかどうかを指定します。
tmslog.fileLocation	\$(LOG_ROOT)	ログ・ファイルを書き込む場所を指定します。

- E メール通知
 - ◆ ランタイムタスクの内容をEメールで通知することが可能
 - ◆ 管理コンソール「システム管理 > タスク管理 > 通知」で設定

可視化機能:まとめ

- レポート
 - ◆ サービス・ポリシーの実現状況やパフォーマンス状況の可視化
- オペレーション・アラート
 - ◆ 発生している問題を迅速に把握
- ランタイム・タスク
 - ◆ サービス・ポリシー違反およびヘルス・ポリシー違反の検知および自動対処
 - ◆ 監視モードの場合管理者にアクションを提示

5. まとめ



インテリジェント管理(1)のまとめです。

1. インテリジェント・ルーティングが優先順位に応じて流量制御を行い、優先度とビジネス目標に基づくサービス・レベル管理が可能になり、サービス・レベル向上につながります。
2. パフォーマンス管理が本番稼働中突発的なトランザクション増加にも対応可能で、サーバーの過負荷状況を回避しつつ、割り振りの重みを動的に計算して割り振りをします。またリソース状況に応じてアプリケーションの配置を動的変更してJVMの数を動的に制御し、リソースを有効利用しながらスケーラビリティを向上させます。

ヘルス管理機能およびアプリケーション・エディション管理次のセッションにてご説明致します。

参考文献

- ◆ 「WebSphere Application Server V8.0 アナウンスメント・ワークショップ資料」
 - http://www.ibm.com/developerworks/jp/websphere/library/was/was8_ws/
- ◆ 「WebSphere Application Server V8.0によるWebシステム基盤設計ワークショップ資料」
 - http://www.ibm.com/developerworks/jp/websphere/library/was/was8_guide/
- ◆ 「WebSphere Virtual Enterprise ソリューション・ワークショップ資料」
 - http://www.ibm.com/developerworks/jp/websphere/library/wxd/wve7_ws/
- ◆ 「WebSphere Virtual EnterpriseによるWebシステム運用負荷軽減のヒント」
 - http://www.ibm.com/developerworks/jp/websphere/library/wxd/wve61_usecase/
- ◆ 「WebSphere Virtual Enterprise を複雑な WebSphere Application Server のトポロジーに組み込む」
 - http://www.ibm.com/developerworks/jp/websphere/library/wxd/wve_was_topology/

ワークショップ、セッション、および資料は、IBMまたはセッション発表者によって準備され、それぞれ独自の見解を反映したものです。それらは情報提供の目的のみで提供されており、いかなる参加者に対しても法律的またはその他の指導や助言を意図したのではなく、またそのような結果を生むものでもありません。本講演資料に含まれている情報については、完全性と正確性を期するよう努力しましたが、「現状のまま」提供され、明示または暗示にかかわらずいかなる保証も伴わないものとします。本講演資料またはその他の資料の使用によって、あるいはその他の関連によって、いかなる損害が生じた場合も、IBMは責任を負わないものとします。本講演資料に含まれている内容は、IBMまたはそのサプライヤーやライセンス交付者からいかなる保証または表明を引き出すことを意図したもので、IBMソフトウェアの使用を規定する適用ライセンス契約の条項を変更することを意図したものでもなく、またそのような結果を生むものでもありません。

本講演資料でIBM製品、プログラム、またはサービスに言及していても、IBMが営業活動を行っているすべての国でそれらが使用可能であることを暗示するものではありません。本講演資料で言及している製品リリース日付や製品機能は、市場機会またはその他の要因に基づいてIBM独自の決定権をもっていつでも変更できるものとし、いかなる方法においても将来の製品または機能が使用可能になると確約することを意図したものではありません。本講演資料に含まれている内容は、参加者が開始する活動によって特定の販売、売上高の向上、またはその他の結果が生じると述べる、または暗示することを意図したもので、またそのような結果を生むものでもありません。パフォーマンスは、管理された環境において標準的なIBMベンチマークを使用した測定と予測に基づいています。ユーザーが経験する実際のスループットやパフォーマンスは、ユーザーのジョブ・ストリームにおけるマルチプログラミングの量、入出力構成、ストレージ構成、および処理されるワークロードなどの考慮事項を含む、数多くの要因に応じて変化します。したがって、個々のユーザーがここで述べられているものと同様の結果を得られると確約するものではありません。

記述されているすべてのお客様事例は、それらのお客様がどのようにIBM製品を使用したか、またそれらのお客様が達成した結果の実例として示されたものです。実際の環境コストおよびパフォーマンス特性は、お客様ごとに異なる場合があります。

IBM、IBM ロゴ、ibm.com、WebSphere、およびz/OSは、世界の多くの国で登録されたInternational Business Machines Corporationの商標です。他の製品名およびサービス名等は、それぞれIBMまたは各社の商標である場合があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtmlをご覧ください。

JavaおよびすべてのJava関連の商標およびロゴは Oracleやその関連会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。