この資料は基礎的なサーバー構成の知識について解説しています。
V7の新機能ではありません。

免責事項

当資料は、2008年9月に一般出荷になったWebSphere Application Server for z/OS Version 7.0 を前提として作成したものです。

当資料に含まれている情報は正式なIBMのテストを受けていません。また明記にしろ、暗黙的にしろ、何らの保証もなしに配布されるものです。

この情報の使用またはこれらの技術の実施は、いずれも使用先の責任において行われるべきものであり、それらを評価し実際に使用する環境に統合する使用先の判断に依存しています。

それぞれの項目は、ある特定の状態において正確であることがIBMによって調べられていますが、他のところで同じ、または同様の結果が得られる保証はありません。これらの技術を自身の環境に適用することを試みる使用先は、自己の責任において行う必要があります。

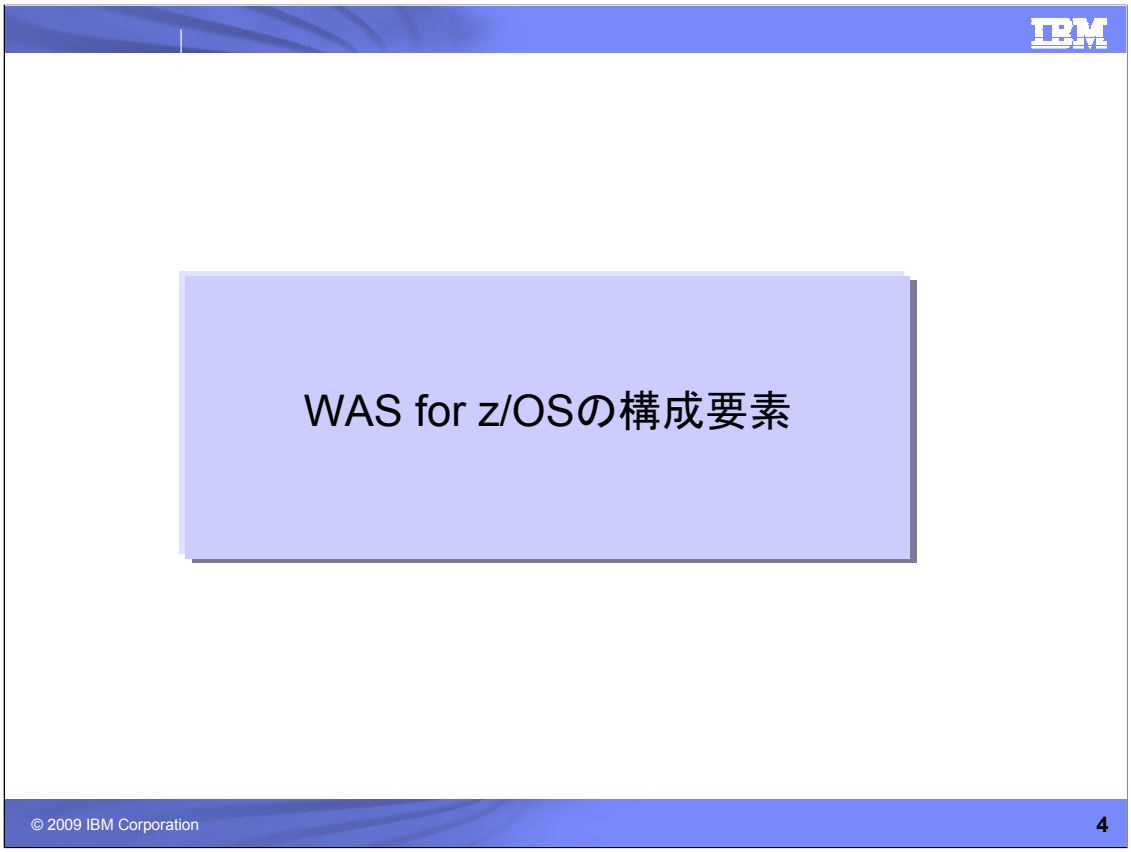
登録商標

1. AIX, CICS, Cloudscape, DB2, IBM, IMS, Language Environment, Lotus, MQSeries, MVS, OS/390, RACF, Redbooks, RMF, Tivoli, WebSphere, z/OS, zSeriesは IBM Corporation の米国およびその他の国における商標です。
2. Microsoft, Windows は Microsoft Corporation の米国およびその他の国における商標です。
3. Java, J2EE, JMX, JSP, EJB は Sun Microsystems, Inc. の米国およびその他の国における商標です。
4. UNIX はThe Open Groupの米国およびその他の国における登録商標です。
5. 他の会社名, 製品名およびサービス名等はそれぞれ各社の商標です。

目次

- WAS for z/OSの構成要素
  - コントローラー/サーバント
  - デーモン
- WASのトポロジー
- HTTPサーバー（IHS）のトポロジー
- アプリケーションの配置

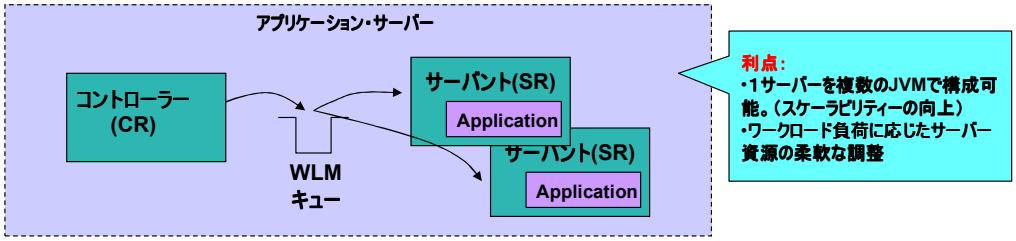
WAS for z/OSの基礎知識として、サーバー構成のバリエーションと一般的な配置について解説します。



z/OS版WASの基礎知識

コントローラー／サーバント

- コントローラー（CR）の役割
  - クライアントとの通信管理
  - トランザクション管理
  - セキュリティー管理
- サーバント（SR）の役割
  - J2EEアプリケーションの稼動環境（JVM）
  - CRとSR間はWLM Queue Managerサービスにより連携。ワークロードに応じて柔軟なSR数の調整が可能。



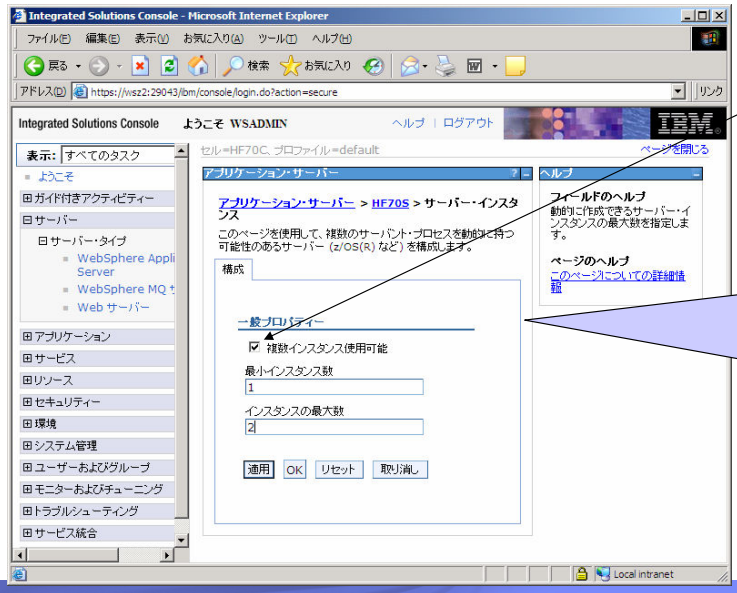
オープン系WASとz/OS版WASの大きな違いは、CR/SR構成です。

z/OS版WASは、特にオプションなしでスケーラビリティを持ち、アプリケーションの処理能力を増やすことができます。

もちろん、必ず複数SRにする必要はなく、1SRが基本です。

複数サーバント構成にする場合

管理コンソール：サーバー＞サーバー・タイプ＞WebSphere Application Server＞サーバー名＞サーバー・インフラストラクチャー＞Java及びプロセス管理＞サーバー・インスタンス
複数インスタンス使用可能　にチェックし、インスタンスの最大数を2以上に設定



New V7

このチェックを、単一サーバント構成の場合（最小・最大は1）でもオンにしておけばコマンドで動的にサーバントを追加できます。

注意！

- SRの起動にはCPU負荷がかかります。このため、SRの起動中はパフォーマンスが劣化する可能性があります。
- 頻繁にSRが起動・停止するような環境の場合は、はじめから最大・最小インスタンス数を揃えておくことをお勧めします。

複数SRにするには、管理コンソールで設定するだけです。

1SR構成の場合でも、デフォルトではオフの「複数インスタンス使用可能」チェックを入れておくことをお勧めします。

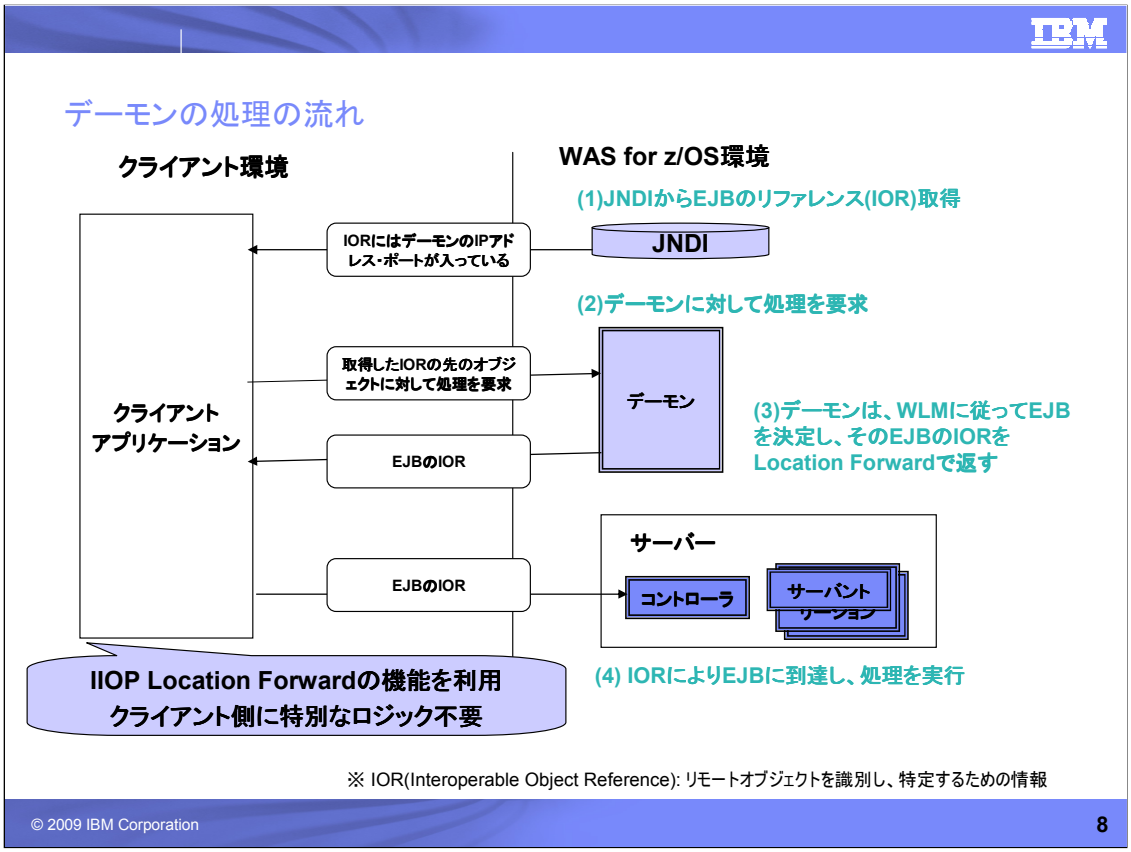
後から動的にSRを増やすことができます。

デーモン（ロケーション・サービス・デーモン）

- EJBを利用するクライアント・アプリケーション(J2EEクライアント、サーブレットなど)が利用する。
- セル内のサーバーの稼動状況をダイナミックに把握し、クライアントの要求を適切なサーバーにリダイレクトする機能。
- クライアントはJNDIからデーモンへのオブジェクト・リファレンスを入手する。デーモンは実サーバーのオブジェクト・リファレンスを通知する。
- アプリケーションに特別なコーディング不要。
- スタンドアロン構成ではセル単位に1つ必要。

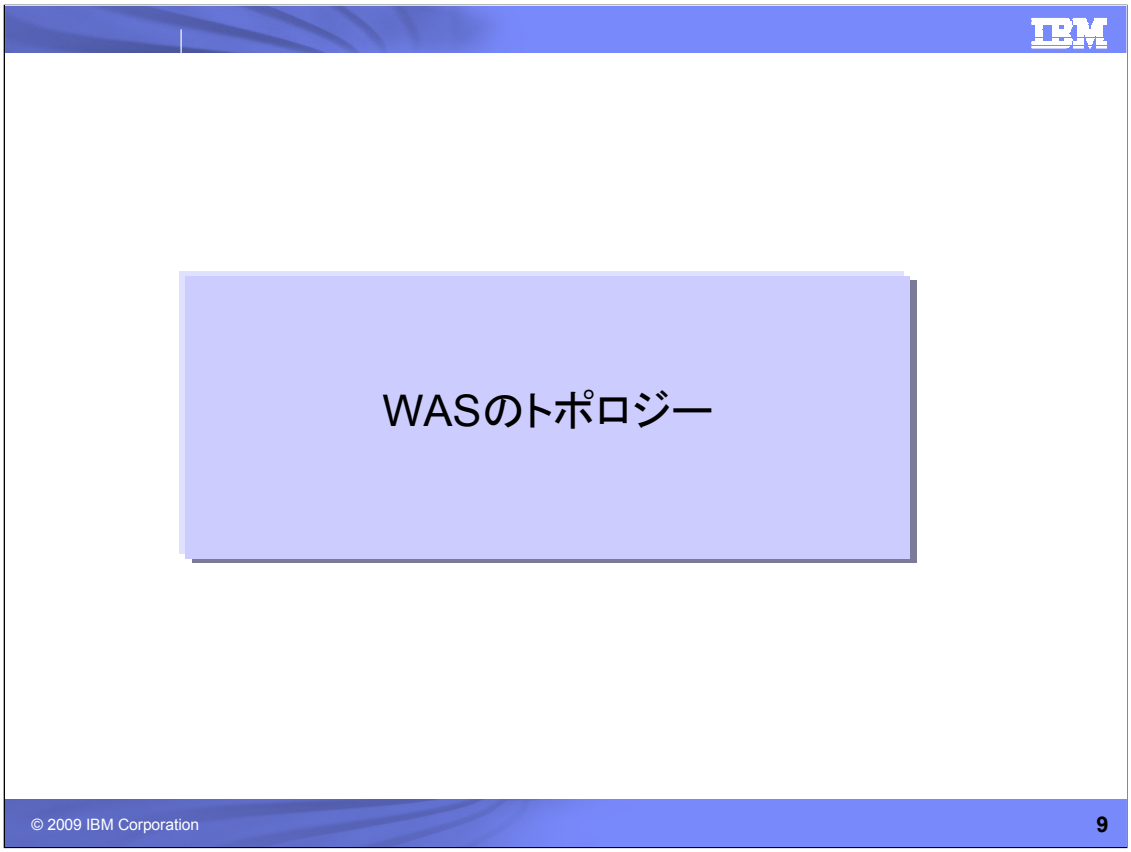
メリット：
クライアントはサーバーの稼動場所（IPアドレス、ポート番号）を直接知らなくてもよい。
クラスタリング機能の実現。（ND構成）

デーモンもまた、z/OS版固有のプロセスですが、これは設計上の名残（Component Broker）と言ってもよいでしょう。オープン系の場合は各サーバーやNode agentにロケーションサービスデーモンがあります。

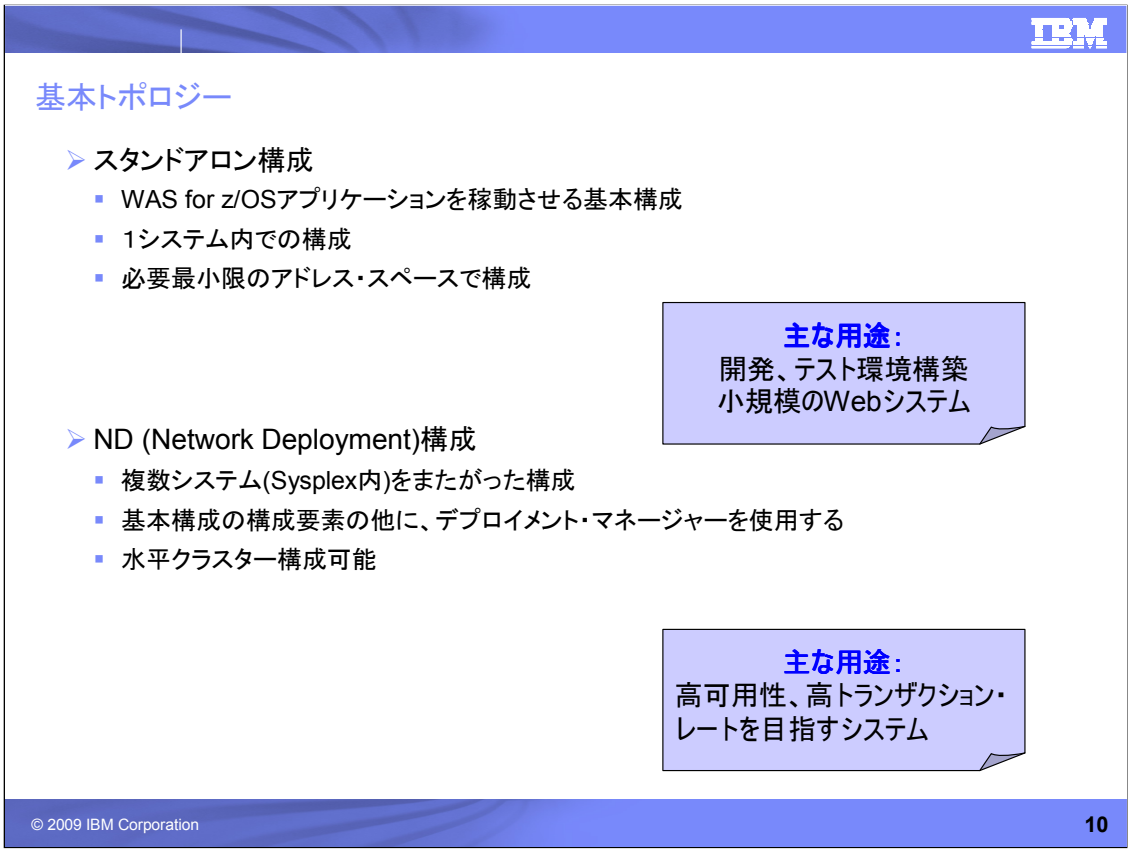


サンプルプログラム

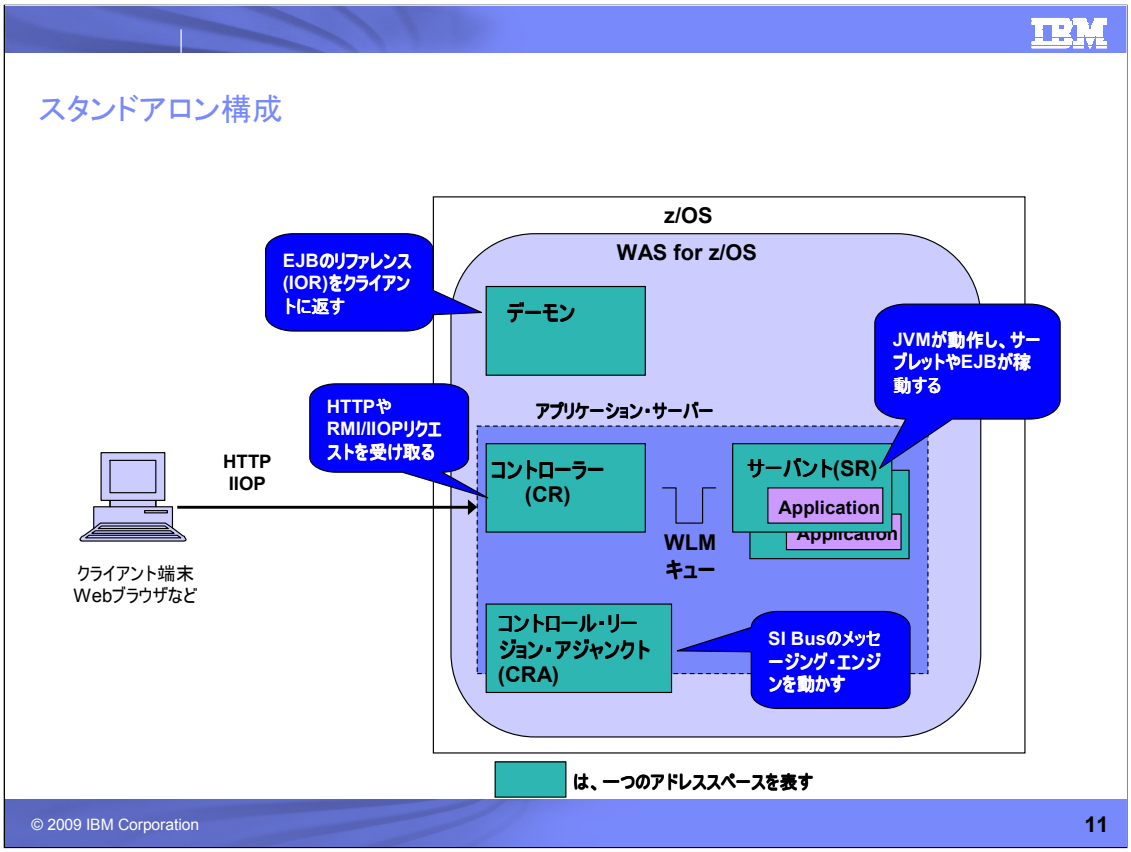
```
InitialContext ic = new InitialContext();

Object obj  = ic.lookup( "java:comp/env/ejb/testEJB");

EJBHome ejbhome = (EJBHome)
javax.rmi.PortableRemoteObject.narrow(obj, EJBHome.class);

EJB ejb = ejbhome.create() ;
```



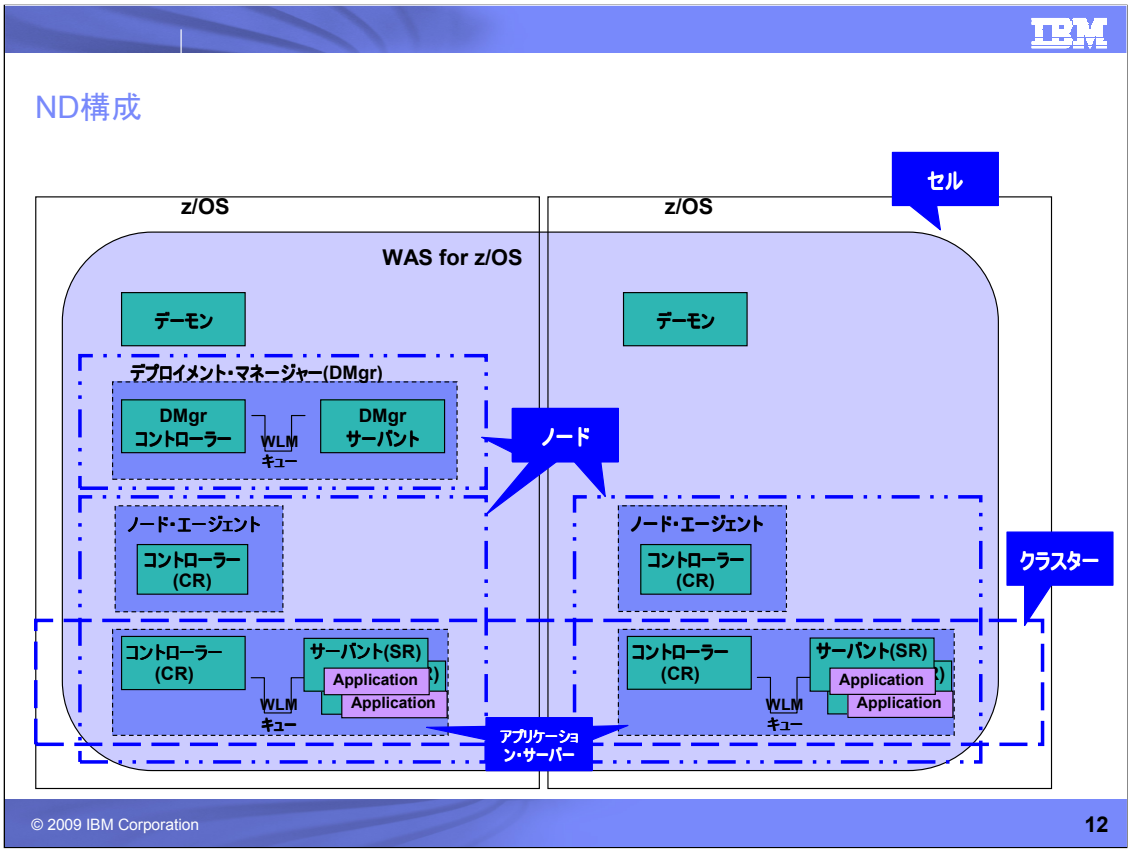
サーバー構成の基礎知識



可用性を意識する場合、スタンドアロン構成ではSRの冗長性のみですので、やはりSysplex+ND構成がお勧めとなります。



スタンドアロン構成でもSRの可用性は確保できます。



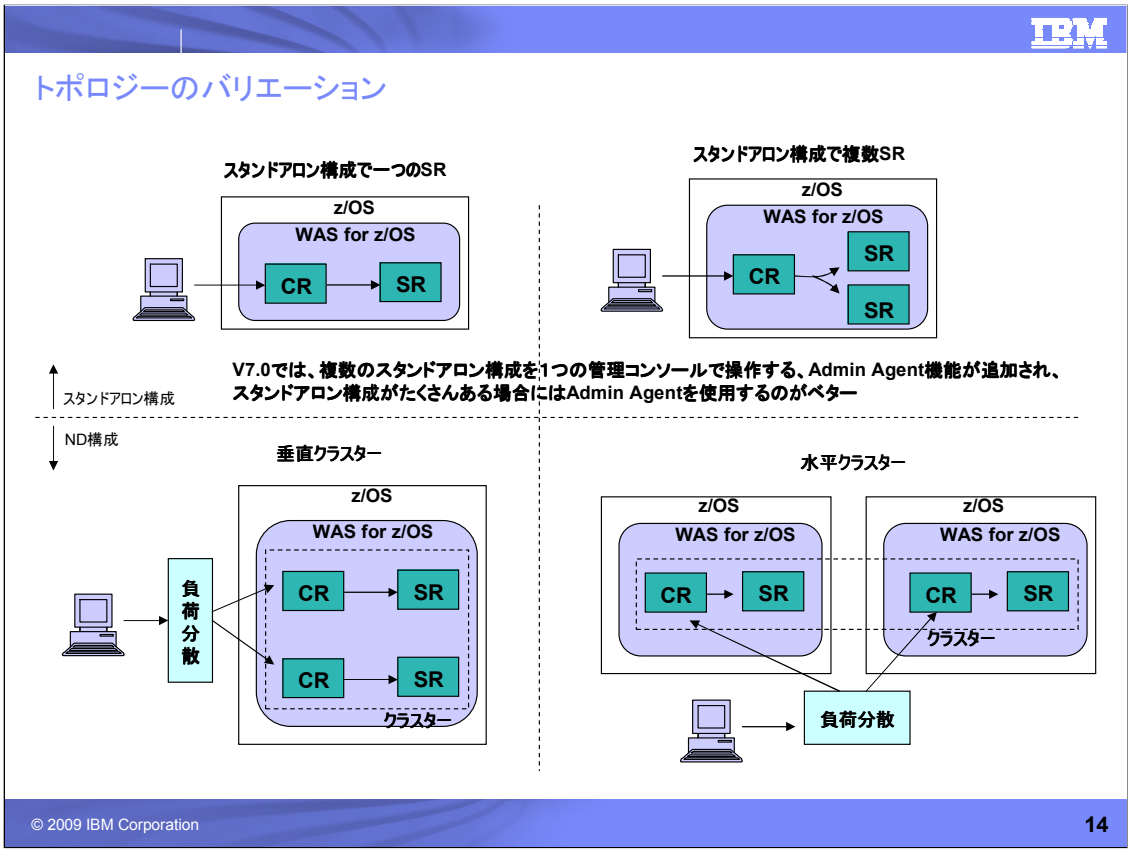
ND構成の場合でも、CR/SRが各サーバーにあることは変わりません。

スタンドアロン構成・ND構成の比較

	スタンドアロン	ND
構成方法	セルの初期構成はzPMTで生成したJCLを実行 サーバントの数を管理コンソールで定義	セルの初期構成はzPMTで生成したJCLを実行 ノードは生成したJCLかaddNode.shコマンドによりセルに追加 サーバーの追加、クラスタリングは管理コンソールで実施
アドレス・スペース構成	- デーモン - AppServerのCR - AppServerのSR - AppServerのCRA	- デーモン - デプロイメント・マネージャーのCR - デプロイメント・マネージャーのSR - ノード・エージェント - AppServerのCR - AppServerのSR - AppServerのCRA
構成操作の単位	それぞれのスタンドアロン・サーバー	それぞれのNDセル
サーバー（CR/SR）構成	1つ	複数
クラスタリング	不可	可能

ND構成はDeployment ManagerとNode Agentがあるので、それだけアドレス空間は増えます。

これを考慮に入れて必要リソースを検討してください。



SRの可用性とCR/SRの可用性は、別々のメトリックがあります。

CRはアプリケーションが動作しないので、一般的に安定性が高い。

SRはアプリケーションの書き方次第で、OufOfMemoryなどのサーバーダウンを起こすこともある。

一方、クラスターを選択する基準は可用性だけでなく、サーバー処理能力や負荷分散といった点もあります。

スタンドアロン構成のバリエーション

- スタンドアロン構成の場合、水平クラスター構成はできないため、筐体やOS自身に障害が発生した場合はサービスが停止する。
- 複数のSRで運用する場合、リクエストを処理するSRはWLMに従って決定される。SRを指定してリクエストを出すことはできない。

	メリット	デメリット
スタンドアロン構成 SRは1つ	最も単純な形式	可用性の最も低い構成 スレッド数の上限以上は同時に動かない
スタンドアロン構成 SRは複数 （垂直分散）	負荷に応じてSRの数を調節可能 JVMに障害発生時には、他のSRでサービス続行 CPUには余裕があるが、1SRでこれ以上スレッドを増やせない場合	以下の状況ではサービス停止 - OS障害・筐体障害時 - アプリケーション入れ替え時 特定のSRを指定してリクエストを出すことはできない ログ出力等でHFSへファイルを出力する時には、個々のSRの出力先が重ならないように注意する

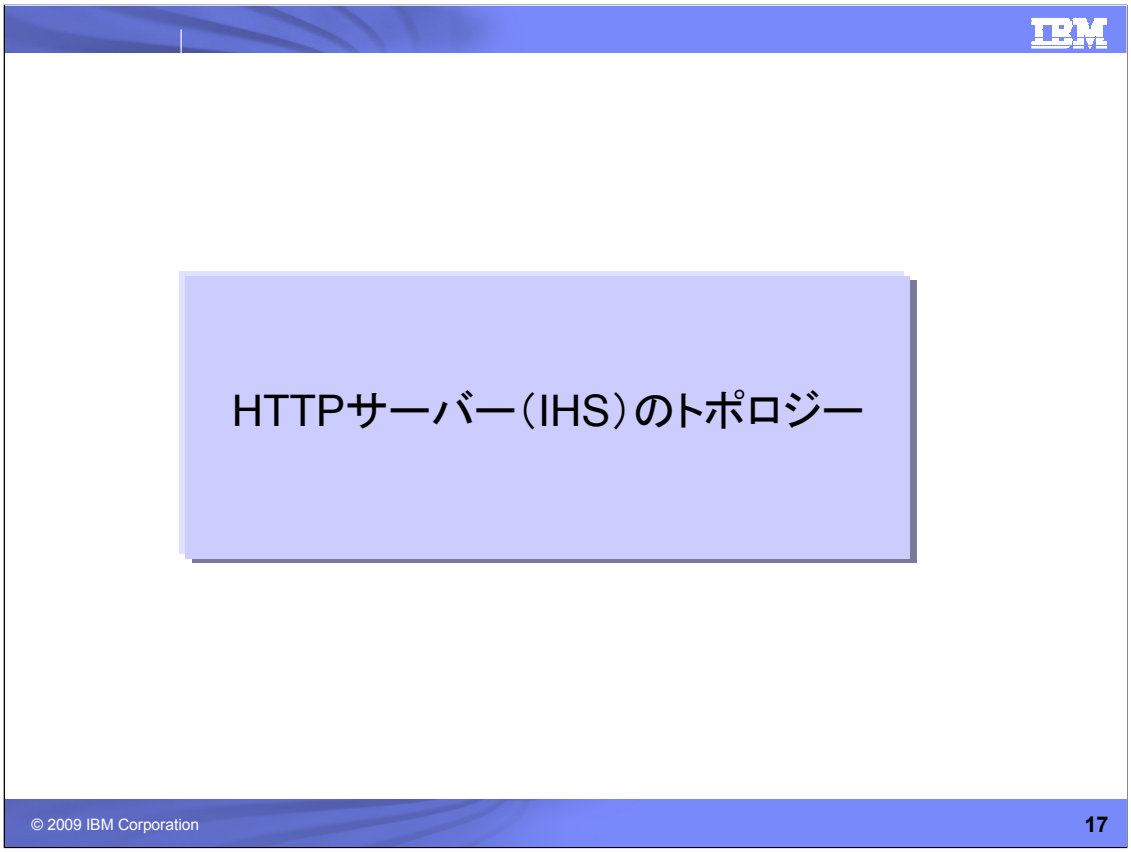
複数SRで動的にSRを増減させると、そのSRを起動するための負荷が付きまとうため、現実にはあまり採用されていません。

ND構成のバリエーション

- ND構成の場合、通常は水平クラスター構成になる。Sysplexの可用性を最大限生かすために、本番環境では2ノード以上のND構成を推奨。
- 負荷分散の仕組みは別途必要。

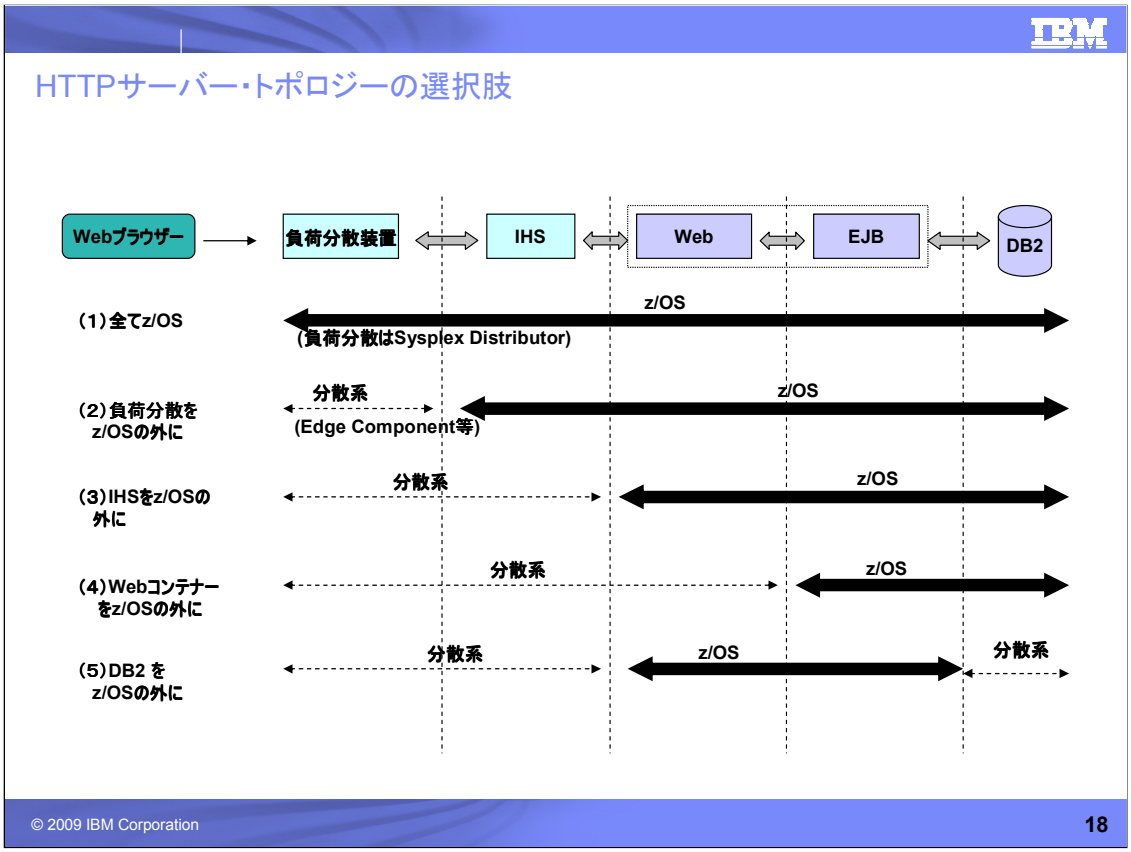
	メリット	デメリット
ND構成 各サーバーのSRは1つ	OS障害・筐体障害等でも、他のクラスターメンバーでサービス可能 アプリケーションの稼動するJVMを特定可能	
ND構成 一つのOSに複数サーバー （垂直クラスター）	CR障害時に他への影響を少なくできる	通常はスタンドアロン構成の複数SRを選択してよい
ND構成 各サーバーのSRは複数	OS障害・筐体障害等でも、他のクラスターメンバーでサービス可能 負荷に応じてSRの数を調節可能 JVMに障害発生時には、他のSRでサービス続行 CPUには余裕があるが、1SRでこれ以上スレッドを増やせない場合	ログ出力等でHFSへファイルを出力する時には、個々のSRの出力先が重ならないように注意する

可用性・スケーラビリティのバランスとして、最も多く採用されているのは
ND構成の水平クラスター・単一SR構成
です。



IHSはWASに付属する、Webフロントサーバーです。

Webの受け口を1つにして、バックエンドに複数WASに割り振るために必要なコンポーネントです。



WAS以外のコンポーネントは同一筐体に置く場合、外に置く場合でいくつかの選択肢があります。

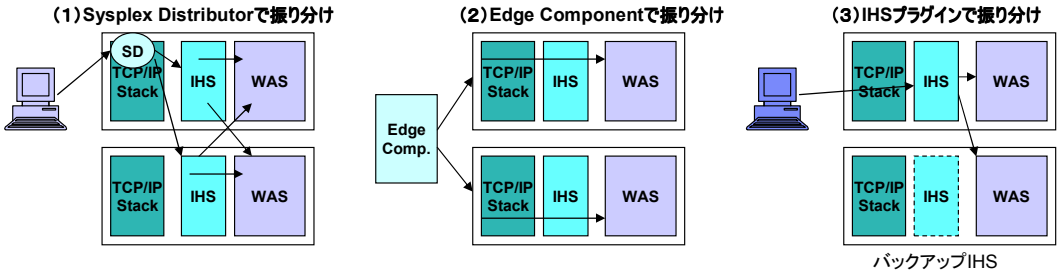
HTTPサーバー・トポロジー 各構成のメリット/デメリット

バリエーション	メリット	デメリット
(1)全てz/OS	コンポーネントが全てz/OS上なので、管理が容易	Sysplex Distributorを使用する場合Edge Componentのようなバックエンドの障害検知機能は無い IHSプラグインを使用する場合、負荷に応じた振り分けはできない
(2)負荷分散をz/OSの外に	Edge Componentは、バックエンドDB2の障害検知が可能 Caching Proxyによる、z/OS側の負荷の軽減	IHSのためのWASのライセンスが必要 管理対象サーバーが増える
(3)IHSをz/OSの外に	IHSをDMZに置くことにより、セキュリティーの強化 z/OSのIHSの負荷を軽減	IHSのためのWASのライセンスが必要 管理対象サーバーが増える
(4)Webコンテナーをz/OSの外に	プレゼンテーション(web)が非常に重く、ビジネスロジック(EJB)の動作に影響を与える場合	WARとEJB-JARとを別にパッケージしなければならず、アプリ管理が面倒 ローカル・インターフェースやパラメーター参照渡しができず、パフォーマンスに影響
(5)DB2をz/OSの外に	z/OS用のDB2のライセンスが無い場合 Oracle等のDBMSを使用している場合	トランザクション管理にXAを使用するため、リカバリー処理には注意が必要 パフォーマンスに十分な注意が必要

前頁のバリエーションにおける、メリデメをまとめました。
すべてを満たす構成はありませんので、取捨選択が必要となります。

リクエスト振り分け機能の配置

HTTPリクエストの振り分け機能としては、Edge Component(Load Balancer)、Sysplex Distributor、IHSプラグインが利用できる

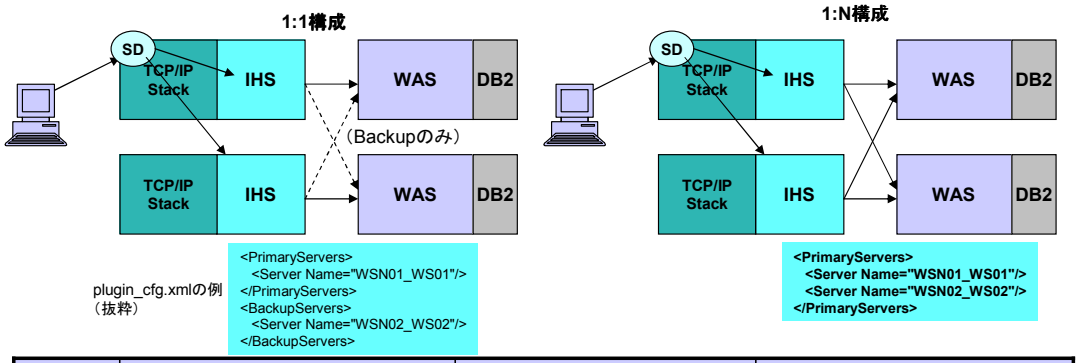


	メリット	デメリット
(1)Sysplex Distributor	z/OSのみで完結していて管理対象が少ない サーバーの負荷に応じた振り分けが可能	DB2障害は検知できないため、WASからエラーが返される場合がある
(2)Edge Component	DB2のダウンなど、より細かなヘルスチェックが可能	Edge Component管理によるコスト増
(3)IHSプラグイン	z/OSのみで完結していて管理対象が少ない	サーバーの負荷に応じた振り分けはできない バックアップIHSはVIPA takeoverの場合のみ

クラスター間のワークロード振り分けを行う方法にもいくつかの選択肢があります。

（1）すべてz/OS

SDはセッション・アフィニティを制御しないため、セッション・アフィニティを持つWebアプリケーションでは、IHSとWASの配置を1:1にすることはできない。1:Nにする必要がある。



plugin_cfg.xmlの例（抜粋）

```
<PrimaryServers>
  <Server Name="WSN01_WS01"/>
</PrimaryServers>
<BackupServers>
  <Server Name="WSN02_WS02"/>
</BackupServers>
```

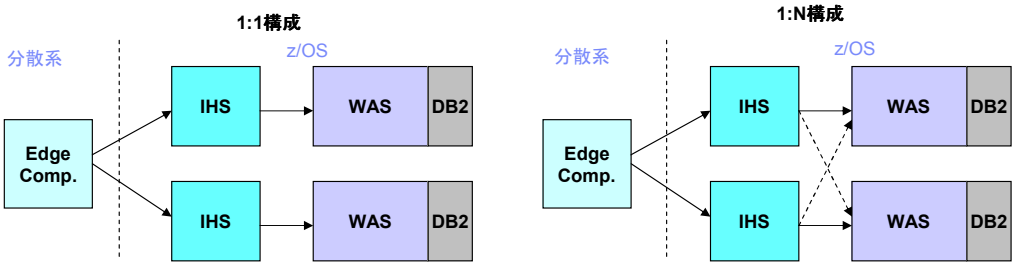
```
<PrimaryServers>
  <Server Name="WSN01_WS01"/>
  <Server Name="WSN02_WS02"/>
</PrimaryServers>
```

	特徴	メリット	デメリット
1:1構成	IHSからWASへの接続はローカルのみ。ローカルのWASがダウンしている場合は他系のクラスターメンバーに接続する	IHS-WASがローカル接続主体となり、パフォーマンス向上が期待できる	SDはアフィニティに従った（JSESSIONIDを意識した）割り振りはしないため、アフィニティのあるWebアプリでは使用不可。
1:N構成	IHSからWASへの接続はローカル・リモートの両方。	SDで複数IHSへの割り振り、IHSプラグインで複数WASへの割り振りができ、資源を有効に使いきることができる。	IHS-WASの接続は、常にクロスに跨る。

1:1構成でPrimary/Backup構成にして、WASクラスター間でセッション共有（Memory to MemoryやSession DB）を使う構成が取れなくはないが、IHS-WASの接続オーバーヘッドに対してセッション共有のコストの方が大きく、このような構成はBad Practiceといえます。

（2）Edge Componentで振り分け

Edge Component（Load Balancer）は分散系のサーバーに配置。Load Balancerはセッション・アフィニティをコントロールできるので、IHSとWASの配置を1:1にすることも1:Nにすることもできる。

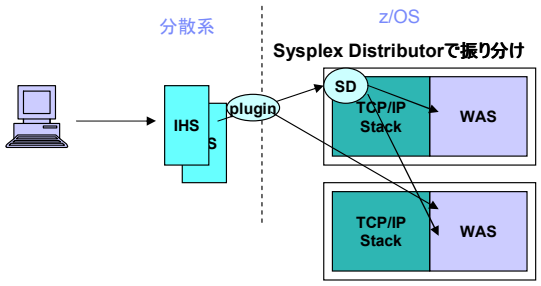


	特徴	メリット	デメリット
1:1構成	Edg Componentが振り分けた通りにWASにリクエストが届く ヘルスチェック・サーブレットによりEdgeがDB2までの経路上での障害を検知可能	Edgeがヘルスチェック・サーブレットによる障害を検知すると、そちらにリクエストを振らないようにできる	WASサーバーと同数のIHSが必要
1:N構成	Edgeが振り分けても、IHSのプラグインがセッション・アフィニティにしたがって振り分ける	WASよりも少数のIHSで運用可能 WAS障害時は、プラグインがそちらに振らないようにできる	DB2障害は検知できないため、WASからエラーが返される場合がある

Edge ComponentはZ/OSには配置できません。

（3）IHSを分散系サーバーに外出し

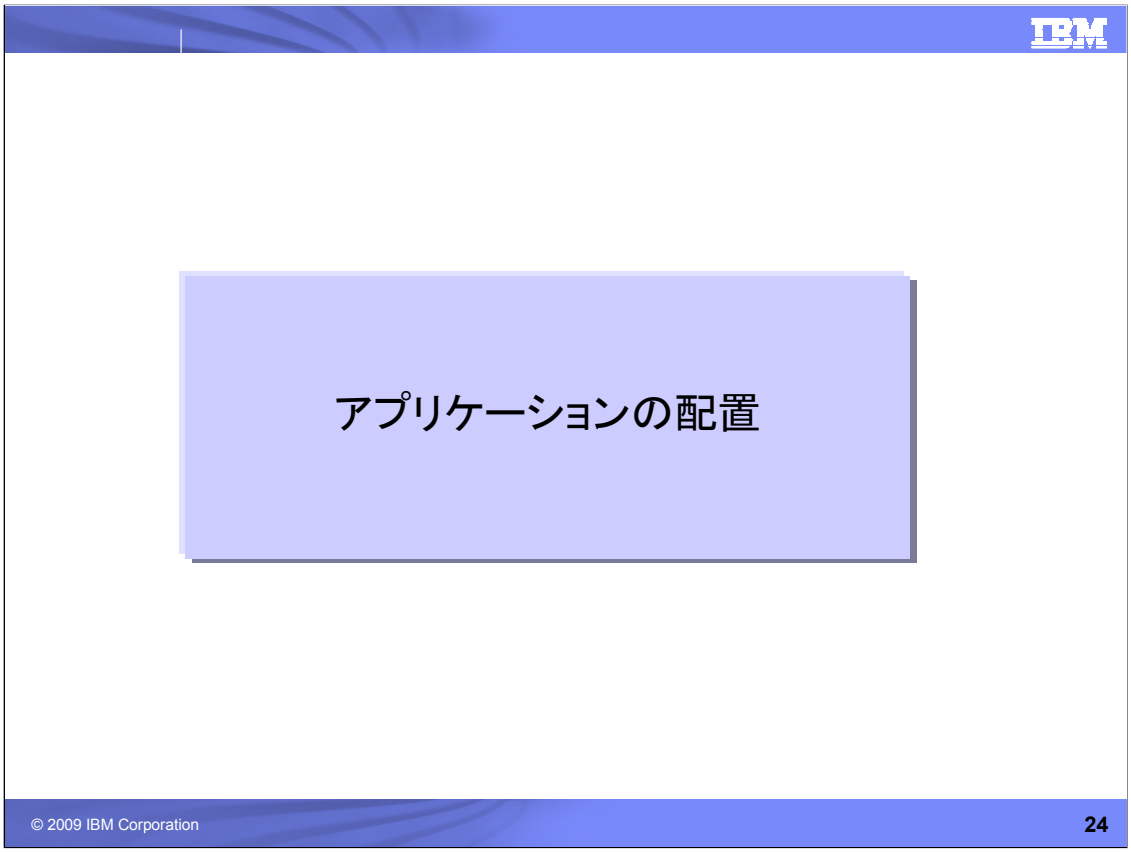
IHSを分散系サーバー上に配置し、SDでWASへのリクエストを振り分けることも可能。セッションアフィニティがある場合は、SDを経由せず直接特定のホストに接続される。



	メリット	デメリット
IHSをz/OSの外に置き、SDで振り分け	IHSのワークロードをz/OSから追い出し可能 IHSをDMZに配置することが可能 サーバーの負荷に応じた振り分けが可能	管理対象が、z/OSと分散系サーバーの2つとなる

SDはTCP/IPレベルの振り分けを行うだけですので、HTTP電文に含まれるセッション・アフィニティをハンドリングすることはできません。アフィニティがない場合は、SDに振り分けを任せることも可能です。

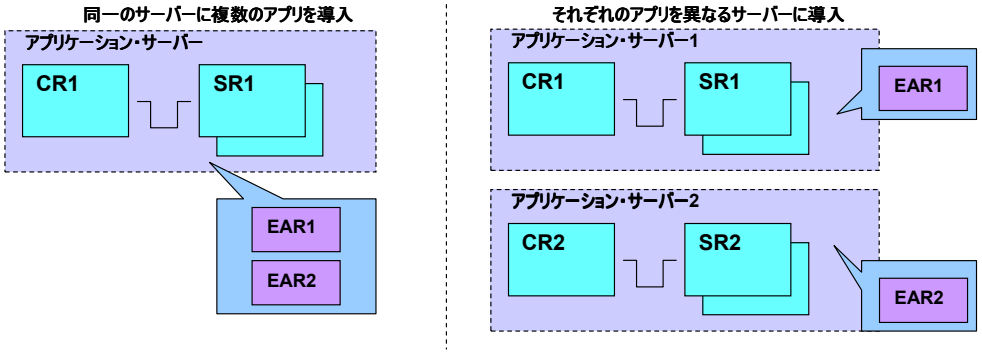
IHS pluginはそれが可能です。



アプリケーションをどのようにサーバーにデプロイするか、も検討が必要です。

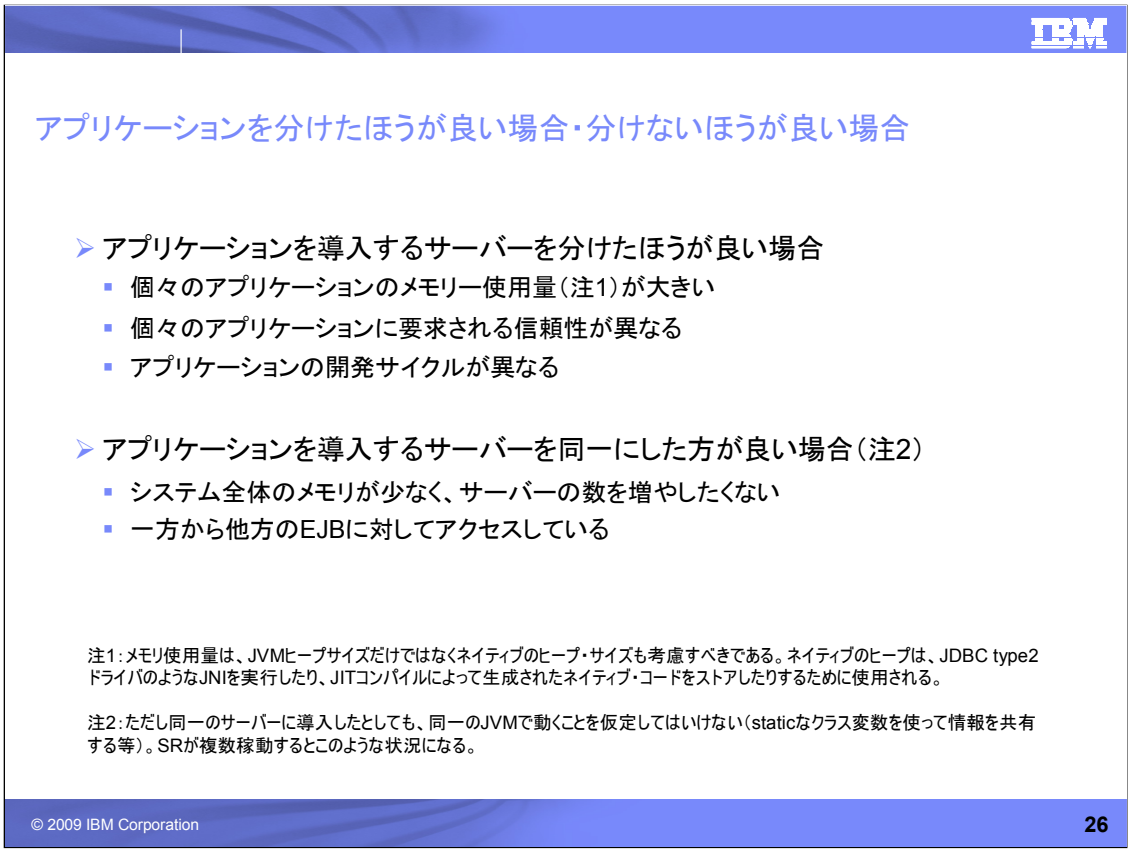
アプリケーションのインストール

複数のアプリケーション（EAR）を分ける必要がある場合



- 以下のような場合、アプリケーションをデプロイするサーバーを分ける必要がある。
  - JVM単位で設定するパラメーターに対する要件が異なる
    - トランザクション・タイムアウト、クラスローダー設定等
  - バージョン等の違いにより実装の異なるライブラリーをそれぞれ使用している
  - それぞれが、異なるDB2サブシステムにアクセスする必要がある(Type2の場合)

アプリケーションが数多くあり、それらの性質が異なる場合や運用上別々に管理したい場合などは、デプロイするサーバーを別々に用意するのが良いでしょう。



サーバーにデプロイできるアプリケーションの量は、物理的には上限はありませんが、運用上妥当な範囲はあります。