
高可用設計とは、z/OSのサブシステムには多く求められる、高可用性を意識したサーバー構成・運用の設計です。

## 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。

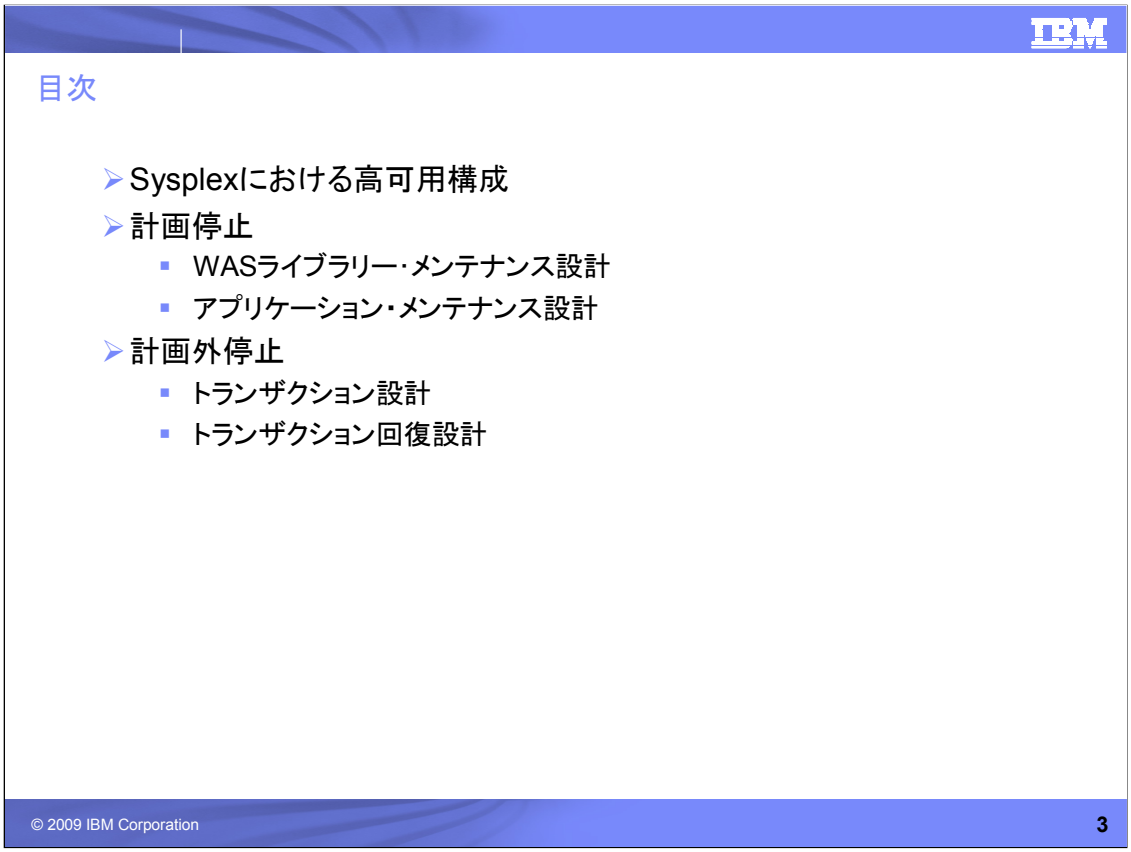
それぞれの項目は、ある特定の状態において正確であることがIBMによって調べられていますが、他のところで同じ、または同様の結果が得られる保証はありません。これらの技術を自身の環境に適用することを試みる使用先は、自己の責任において行う必要があります。

## 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。

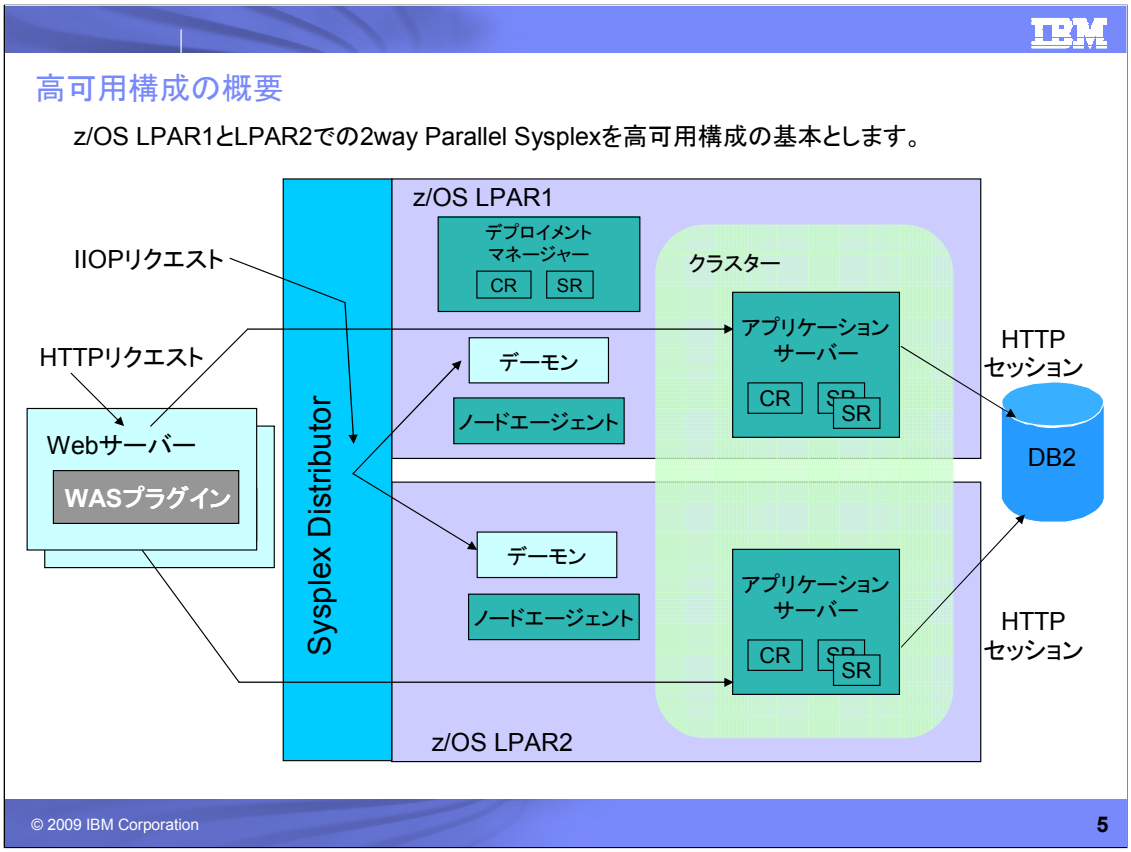



免責事項、登録商標


IBM
目次
Sysplexにおける高可用構成
計画停止
WASライブラリー・メンテナンス設計
アプリケーション・メンテナンス設計
計画外停止
トランザクション設計
トランザクション回復設計
© 2009 IBM Corporation
3

高可用設計　目次


Sysplexにおける高可用構成について記載します。

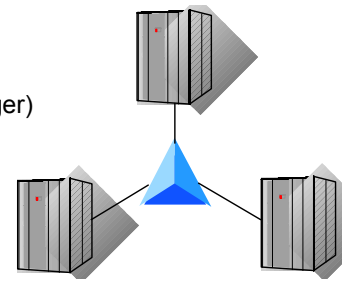



推奨される高可用構成を示します。

## Parallel Sysplex

- IBMメインフレームで提供する最強のクラスター環境

- 可用性に関連するSysplexの主な機能
  - 障害システムの切離し：SFM (Sysplex Failure Manager)
  - 自動再始動：ARM(Automatic Restart Manager)
  - ワークロード管理：WLM(WorkLoad Manager)
  - 資源管理：RRS(Resource Recovery Service)
  - ログ管理：LOGGER(System Logger)



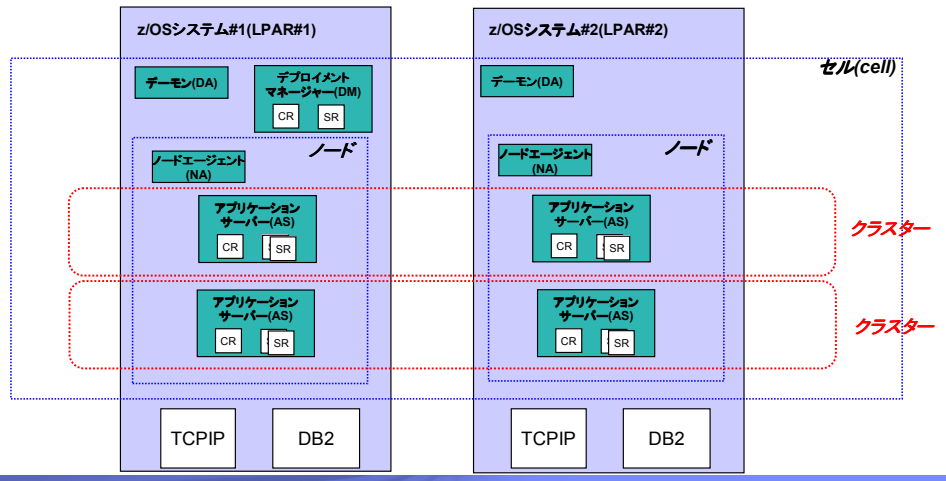
- ただし、単にSysplexであるというだけでは不十分。適切なデザインが必要
  - H/Wクローニング (アクセス機器が同等)
  - S/Wクローニング (S/W構成が同等)
  - 処理の負荷分散 / リクエスト割り振り先の動的変更
  - データ共用 → DB2データ共用
    - 参照： "Parallel Sysplex Availability Checklist"
      http://www.ibm.com/servers/eserver/zseries/library/literature/papers.html




WAS for z/OSの高可用性を検討するためには、z/OSの高可用構成である、Sysplexが必須です。

## WASクラスター構成

- クローンなWASサーバー間のアプリケーションの同一性を保証する仕組み
- Sysplex上でのNetwork Deployment(ND)構成
- WASサーバーの水平クラスター構成が一般的
  - z/OSのアプリケーション・サーバーはCR-SR構成であるため垂直クラスター構成の必要性は低い



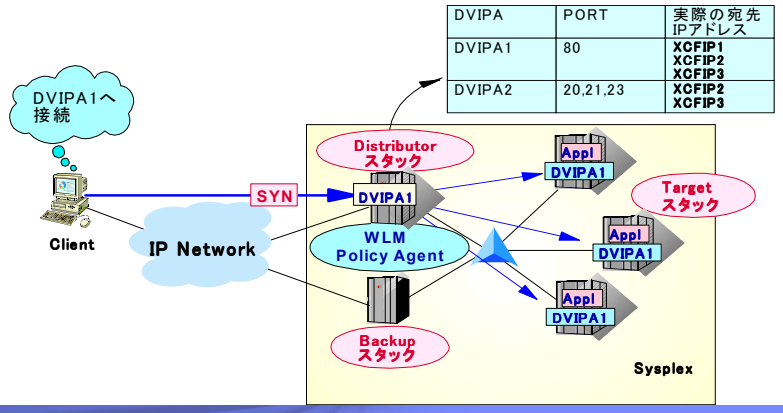




WASのクラスターをSysplex上の各Sysplexメンバーに配置するのが一般的な方法です。

## Sysplex Distributor

- Sysplex TCP/IP Networkingでの負荷分散ソリューション
  - 分散の処理についてはSysplex内部ですべて完結
  - 動的VIPAをクラスターIPアドレスとして使用したシングルイメージ・アクセス.
  - WLMによるシステム負荷情報やポリシー・エージェントからのQoS：サービス品質情報を考慮した負荷分散
  - VIPAテイクオーバー機能を利用し、高可用性にも配慮したソリューション
    - Distributorスタックの機能を即時にBackupスタックで引継ぎ





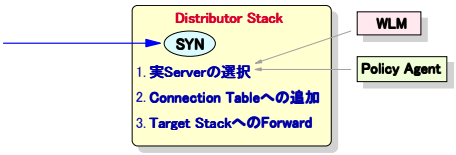


SDはz/OS TCP/IPに設定するだけで使えるようになります。

リアルのIPアドレスとは別にバーチャルのIPアドレスを割り振る必要があります。

## Sysplex Distributor

- DistributorスタックとTargetスタック
  - Distributorスタック　　：　負荷分散処理用のTCP/IPスタック
  - Targetスタック　　：　アプリケーションが稼動するTCP/IPスタック、XCF経由でDistributorスタックに接続
- トリガーとなるのはTCPコネクションの開始要求（SYN）
  - 分散の対象はTCPコネクション単位
- 割り振り処理
  - WLM-based Forwarding
  - WLM/QoS-based Forwarding
  - Random Forwarding




- 割り振り処理フロー
  - ① Distributorスタックはクライアントからの TCPコネクション要求(SYN)を受け取ると、WLMやPolicy Agentと連携し、Sysplex環境において負荷分散を図るためその時点で最適なTargetスタックにその要求を割り振る
  - ② 選択されたスタックにその要求を送り、割り振り結果はコネクション・テーブルに書き込む
  - ③ Targetスタックは、その接続要求を処理し、クライアントに直接返答する
  - ④ 後続パケットはDistributorスタックでコネクション・テーブルに基づきコネクション終了まで選択されたTargetへForward




SDの負荷分散と生死確認の仕組みです。


セッション・アフィニティがある場合は、セッション・オブジェクトがあるサーバーがダウンした場合の回復を検討する必要があります。

方法として、セッション・オブジェクトの配置を二重化するというものが、セッション・パーシスタンスですが、オーバーヘッドがかかります。

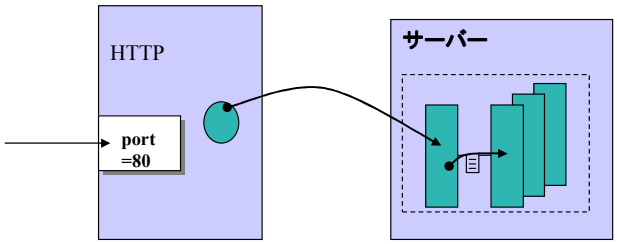
## セッション・アフィニティー

- JSESSIONIDによるセッション・アフィニティーのサポート
  - WebSphere HTTP プラグイン
    - アフィニティーが無ければ、プライマリー・サーバー／ラウンド・ロビン／重み付けによって割り振り先を決定する
    - アフィニティーがある場合は、JSESSIONIDクッキーの中身にあるサーバーに割り振りを行なう
      - プラグインの設定plugin-cfg.xmlでCloneIDに対応するホスト名，ポート番号をも持っている
  - CR

    ①CRは、入力ストリームをスキャンし、セッション・アフィニティークッキーを探す

    ②もし、見つかるとクッキーからアフィニティートークンを取り出しWLMリージョン・トークンに変換する

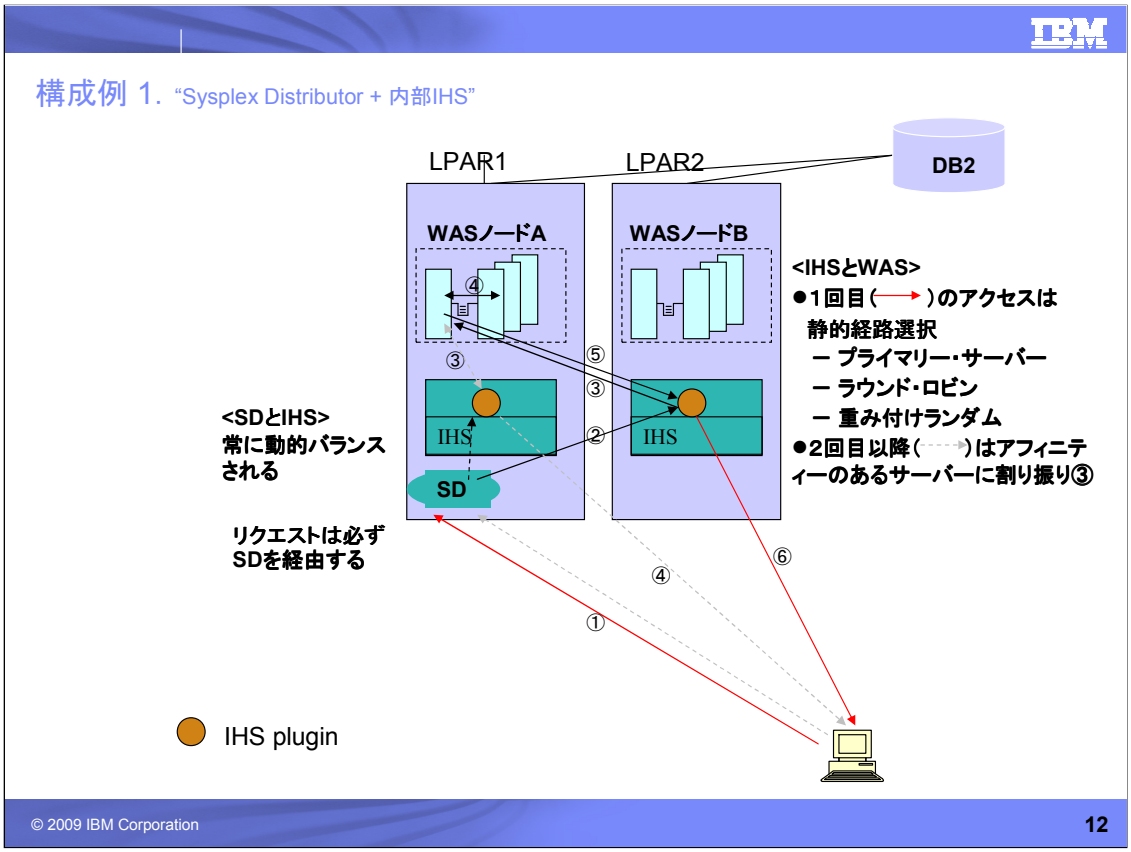
    ③ WLMリージョン・トークンは、リクエストがWLMにキューされた時に正しいSRにルートされるように設定される。



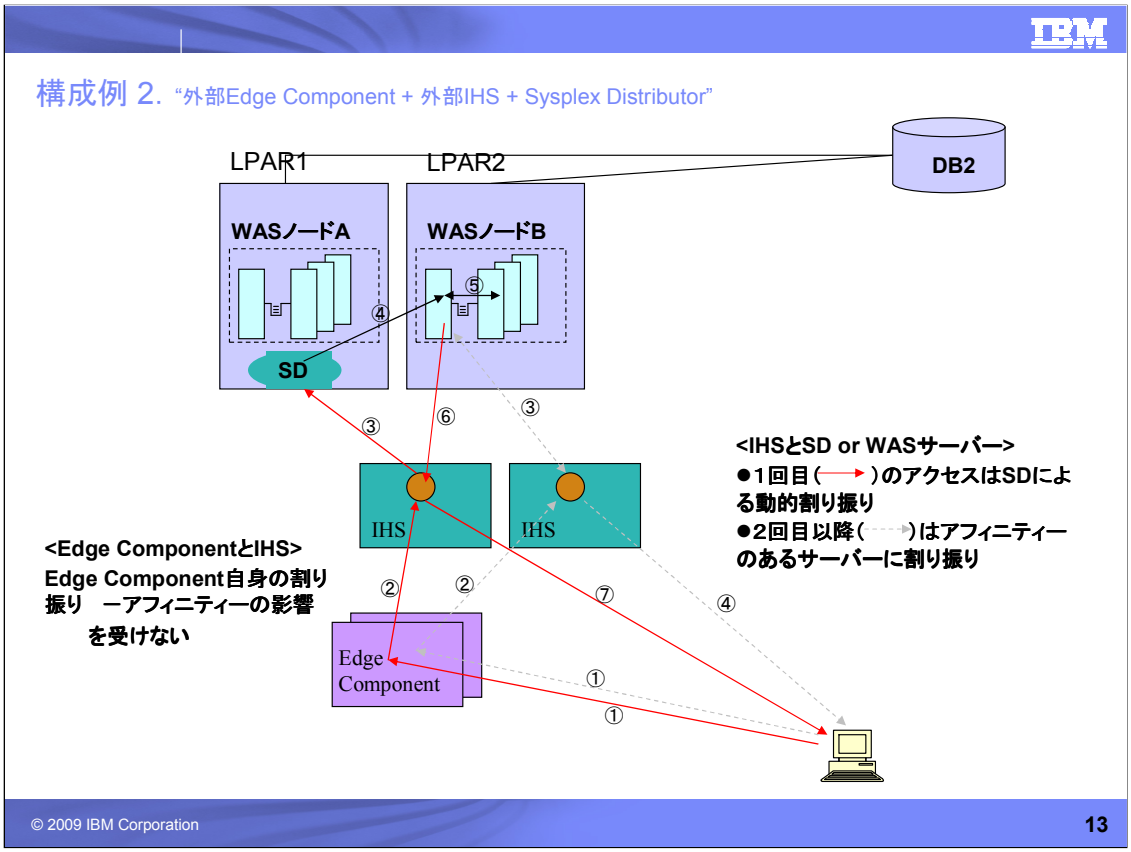




セッション・アフィニティーはIHSプラグインによって認識され、CRから特定のSRに渡されます。




SDのバーチャルIPアドレス(グループアドレス)はブラウザからの直接の接続先となり、その先のIHSを二重化しておくことで、IHSプラグインがセッション・アフィニティに従って正しいWASに割り振ります。



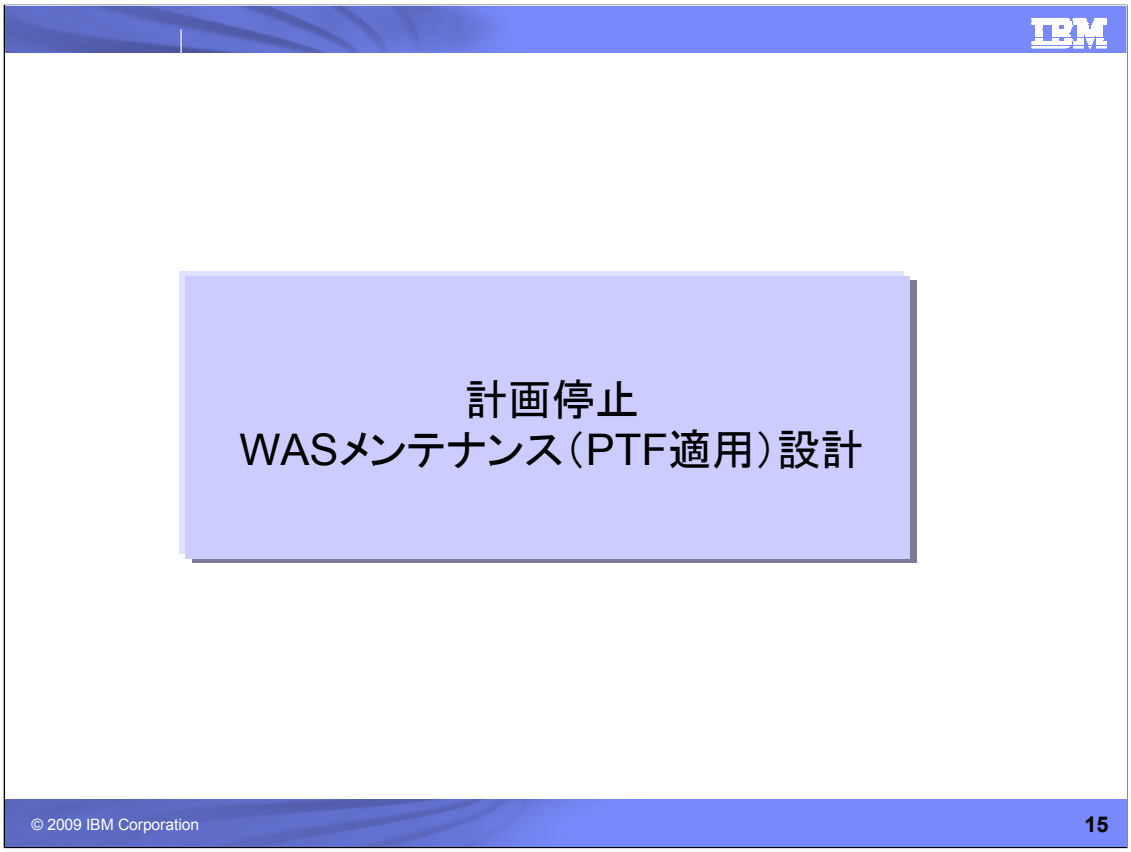

IHSをz/OSの外に置く場合、IHSを二重化するとブラウザからの宛先が二つになってしまうため、Edge Componentをフロントにおきます。

## 参考：デプロイメント・マネージャーの他系起動

- 他系（別ノード）起動は構成時区画が障害である場合の対応
  - 同時に稼動するデプロイメント・マネージャーはひとつ
- 他系起動条件
  - 同時に稼動するデプロイメント・マネージャーはひとつ
    - 他系起動は構成時区画が障害である場合
  - 起動する区画の条件
    - デプロイメント・マネージャーの管轄下のノードが存在すること
    - デーモンが正常に起動していること
  - 構成の条件
    - デプロイメント・マネージャーが持つIPやポートが他系起動に対応していること
      - Sysplex DistributorにおけるDVIPA使用など
    - スタート・プロシージャーが他系からも参照できること
    - デプロイメント・マネージャーの構成HFSが他系からも参照できること




DMgrは二重化することができないため、もしDMgrのあるノードが回復不能（IPLできない場合・ネットワークが接続できない場合など）となった場合に、別のノードで起動する方法が提供されています。

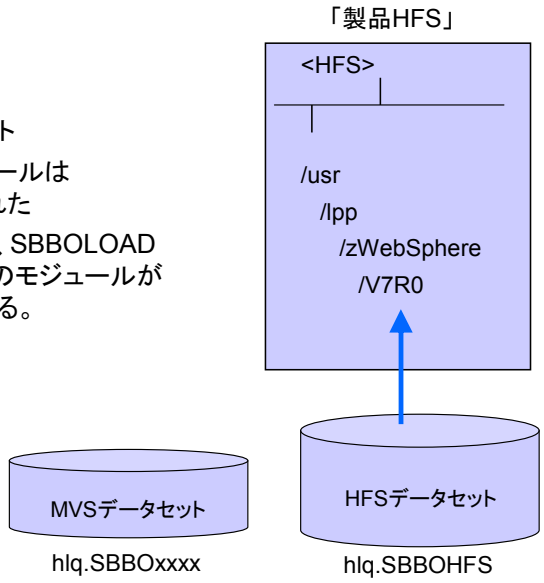



PTFを適用する場合、サーバーの再起動が必要ですので、計画停止をスケジュールします。
この場合にいかにサービス停止を最小化することを検討します。

## WASシステム・ライブラリー

- SMP/Eによるライブラリー管理
- WASライブラリー（製品HFS）の構成
  - HFSまたはzFS
  - サンプルJCLなどのPDSデータセット
    - SBBOLOADなどのロードモジュールは USS上に移動したため、廃止された
    - WASの起動プロシージャーでは、SBBOLOAD のモジュールの代わりにUSS上のモジュールが BPXBATCHの配下でロードされる。

※以下ではHFS/zFSを総称して、HFSと表記します。



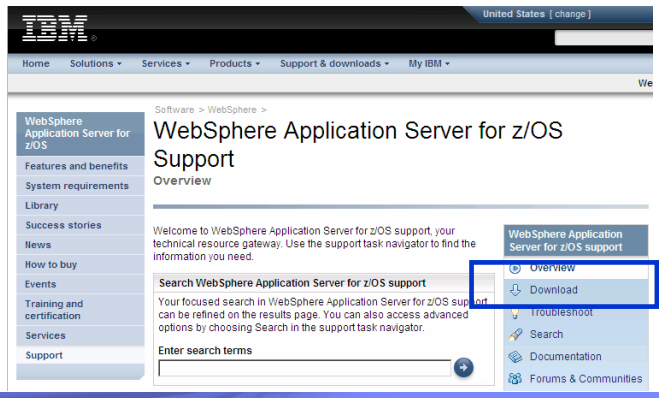




WAS for z/OSは、他のz/OS製品と同様にSMP/Eによるライブラリー管理となります。
V7では、MVSデータセットは、ほとんどなくなりました。

## WASシステム・ライブラリー

- メンテナンス単位は、個別のAPARで当てるのではなく、サービスレベル単位（W7.0.0.xなど）。サービスレベルは複数PTFの集合
- 最新レベルとPTF一覧の確認
  - http://www.ibm.com/software/webservers/appserv/zos_os390/support/
  - Download > Fixes by Version > APAR/PTF Table for WebSphere Application Server for z/OS V7.0 と辿る



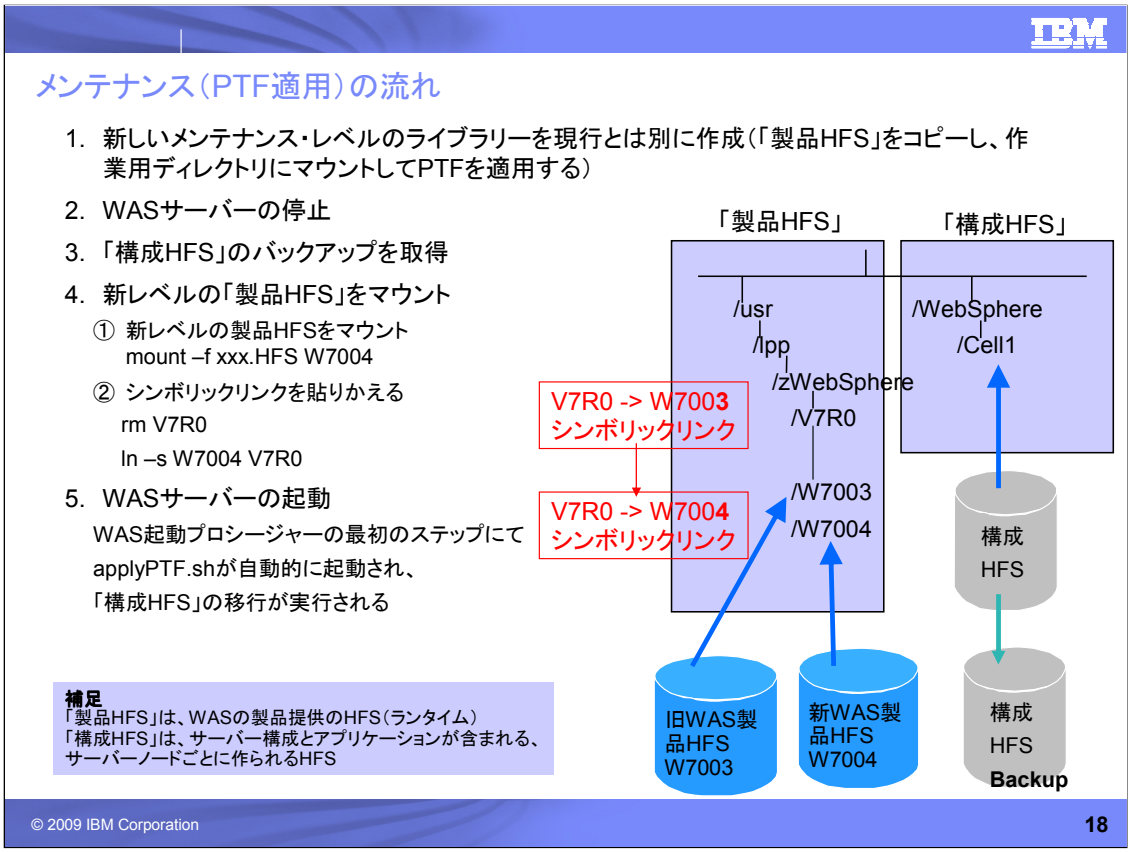




メンテナンス単位は、サービスレベル単位の複数PTFの集合であり、PTFを選択して適用することはできません。

それぞれの最新のレベルの確認は、それぞれのサイトで行ってください。サービスレベルとAPAR番号とPTF番号を関連付いて記述されています。

ひとまとまりのサービスレベルになっているPTFはまとめて適用しなければなりません。あるサービスレベルを抜かして、次のサービスレベルにレベルアップすることはできません。




WASの構成やアプリケーションはWASの構成ファイル(Config.HFS)に保存されます。

WASの構成時に設定した、Sysplex名、システム名、HFSライブラリーのパス名などは、WASの構成を行ってから変更することはできません。

WASの構成を保持したまま、WASのサービスレベルを上げることができます。

新しいサービスレベルを参照するようにしたサーバーを起動すると、WAS起動プロシージャーの最初のステップにてapplyPTF.shが起動されWAS構成HFSの移行が実行されます。

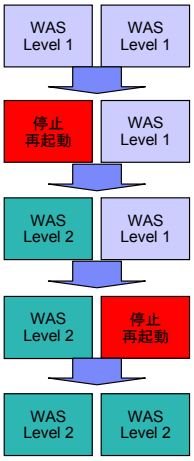
サービスレベルのフォールバックが必要になる場合や、applyPTF.shが正常に機能しない場合を想定して、WAS構成ファイルの構成HFSは、必ずバックアップを取得してからメンテナンス・レベルアップを実施してください。

## ローリング・メンテナンスとは

- ローリング・メンテナンスとは
  - クラスター構成のWASを、「可能な限り中断なく、あるPTFレベルから次のレベルへのアップグレードを可能にする」方法
  - ND構成のそれぞれのサーバーをシステム毎にひとつずつアップグレード
  - 単一システムの単一セル構成では実現不可

- ローリング・メンテナンス実現のために
  - 異なるメンテナンス・レベルの共存をサポート
    - 条件：デプロイメント・マネージャーのレベルを最初にアップグレード
  - メンテナンス・レベルが判りやすいデータセットのネーミング・ルール
    - データセット名にレベル名を持つ
      - WAS70.*W7003*.SBBOHFS のように、レベル名を挟む
  - システム毎に製品のコードを置き換えることが可能なHFS設計
    - 方法1. バージョン固有HFSでのバージョン切替
    - 方法2. システム固有HFS配下のHFSリマウント切替
    - 方法3. HFSのリマウント切替(非共用HFS環境の場合)

**詳細は次のページ以降にて。**







ローリング・アップグレードとは、クラスター構成のWASを、「可能な限り中断なく、あるサービス・レベルから次のレベルへのアップグレードを可能にする」方法です。ND構成のそれぞれのサーバーをシステム毎にひとつずつアップグレードすることにより、サービスを停止することなく、WASをアップグレードすることができます。

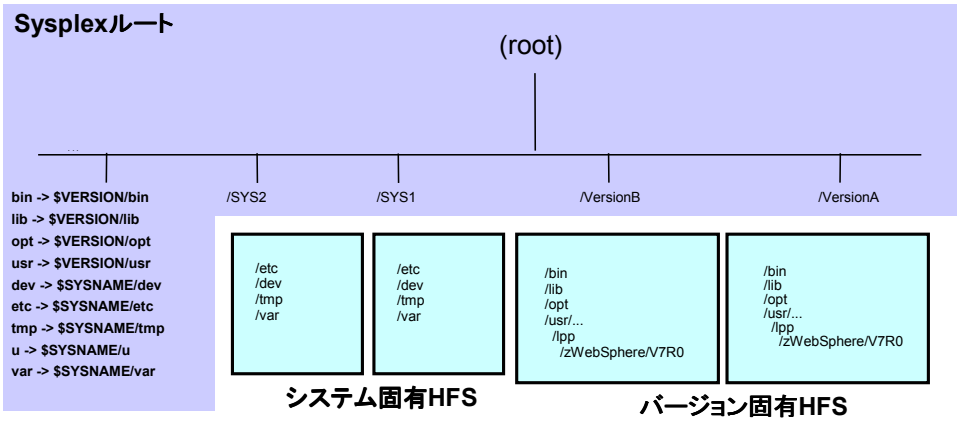
単一システム構成では、WASのアップグレードのためには、サービスを停止しなければなりません。

ローリング・アップグレード実現のためには、WASは異なるメンテナンスレベルの共存をサポートします。ユーザーは、システム毎に製品のコードを置き換えることが可能なように、設計しておく必要があります。

WAS for z/OSの各コード・レベルは、原則としてこれよりも古いコード・レベルを許容するように設計されています。WAS for z/OSの異なるレベルは、アップグレード・プロセス中のND構成内で互換的に共存できます。アップグレードの手順においては、デプロイメント・マネージャーのレベルを最初にアップグレードする必要があります。

## HFS設計の前に

- 共用HFSとは？
  - Sysplex内のメンバー間でのHFSのRead/Write共用を可能にするための機能
  - SYSBPX XCFシグナリング・グループに属するシステムが対象
  - BPXPRMxxでSysplex(YES)を指定
- 共用HFSで使用するHFS







HFS設計の前に、共用HFSについて記載します。共用HFSとは、Sysplex内のメンバー間でのHFSのRead/Write共用を可能にするための機能です。

共用HFSで使用するHFS

Sysplexルート

ディレクトリーとシンボリック・リンクのみが存在・HFSデータセットとしてSysplex内に1つだけ存在

全システムからRead/Writeマウント

バージョン固有HFS

USSおよび関連プロダクトのExecutableが存在 (従来のルートHFSに相当、/bin, /usr, /lib, /opt, ...)

HFSデータセットとしてSysplex内に存在 (複数のバージョンHFSの混在可能)　全システムからRead-onlyマウント（推奨）

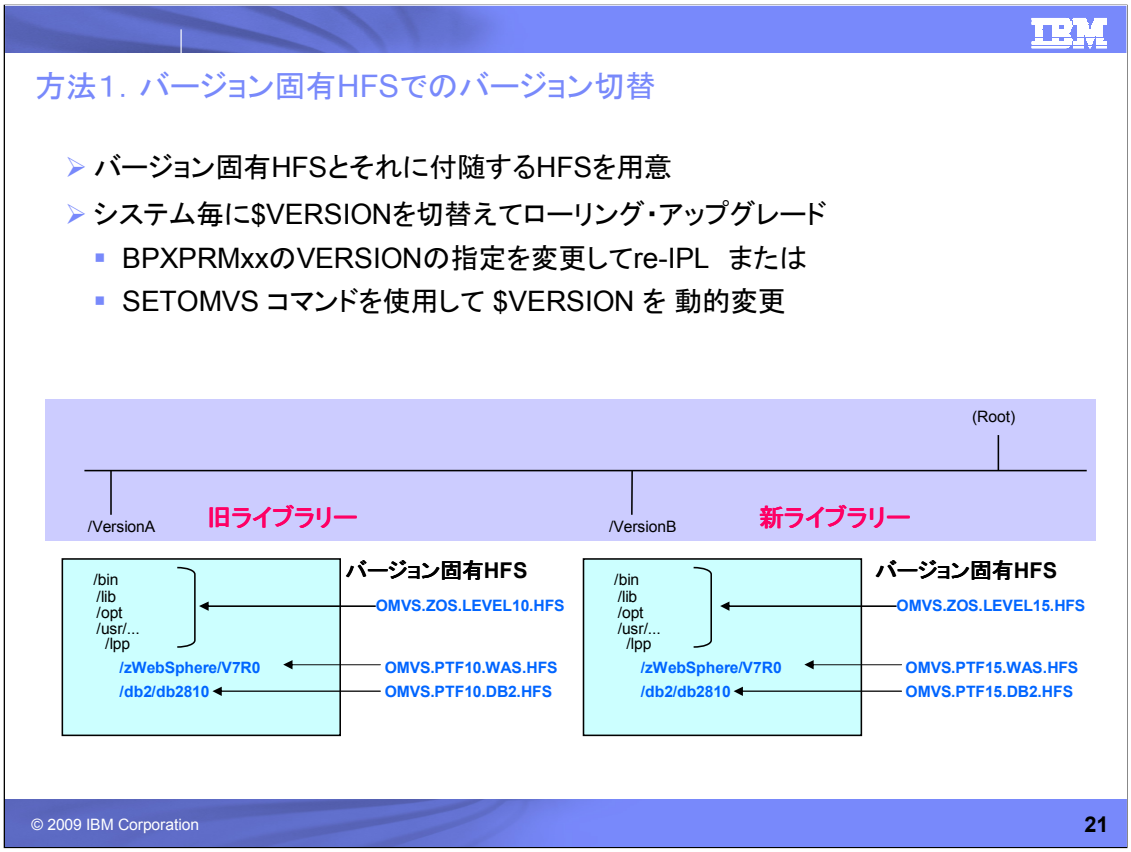
バージョンHFSのマウント・ポイントはBPXPRMxxのVERSIONの指定により、IPL時に動的に作成される

システム固有HFS

各システム固有のディレクトリー/ファイルが存在 (/dev, /tmp, /var, /etc.)

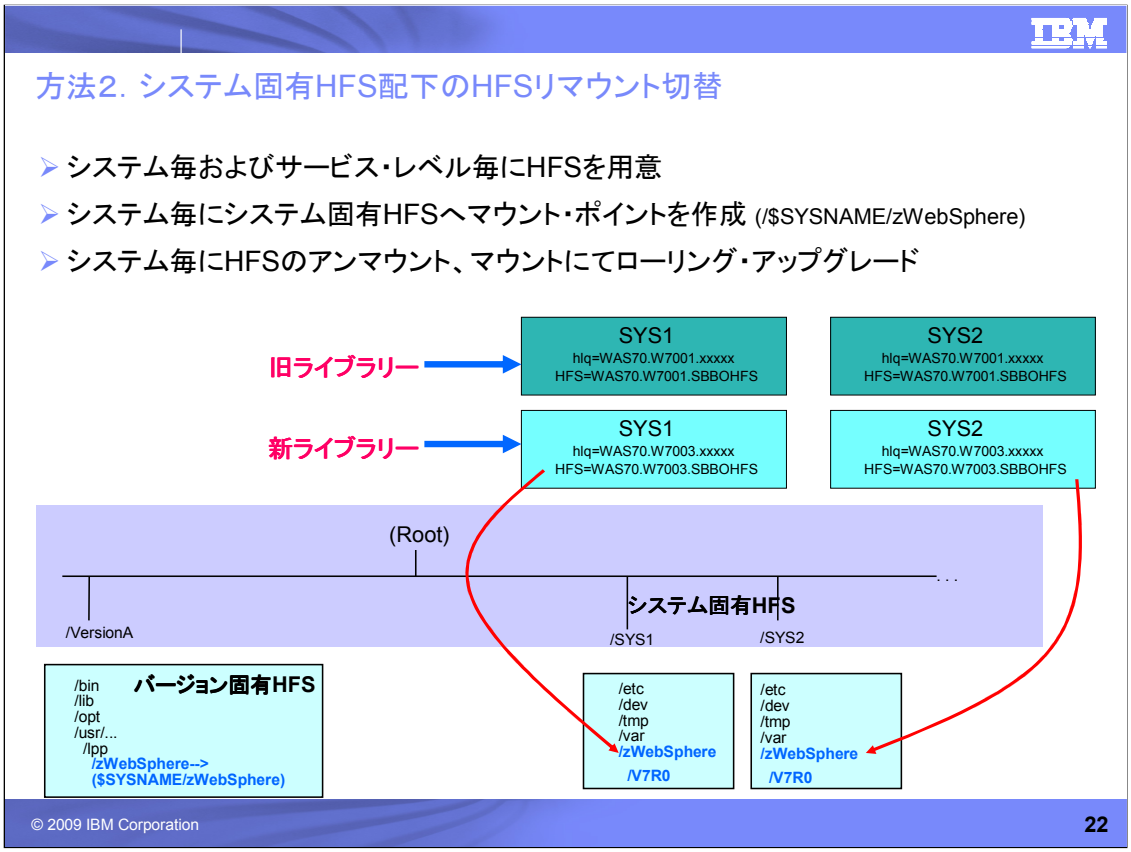
HFSデータセットとしてSysplexのシステム毎に存在

自システムからRead/Writeマウント (他システムからは/$SYSNAME/...でアクセス可能)



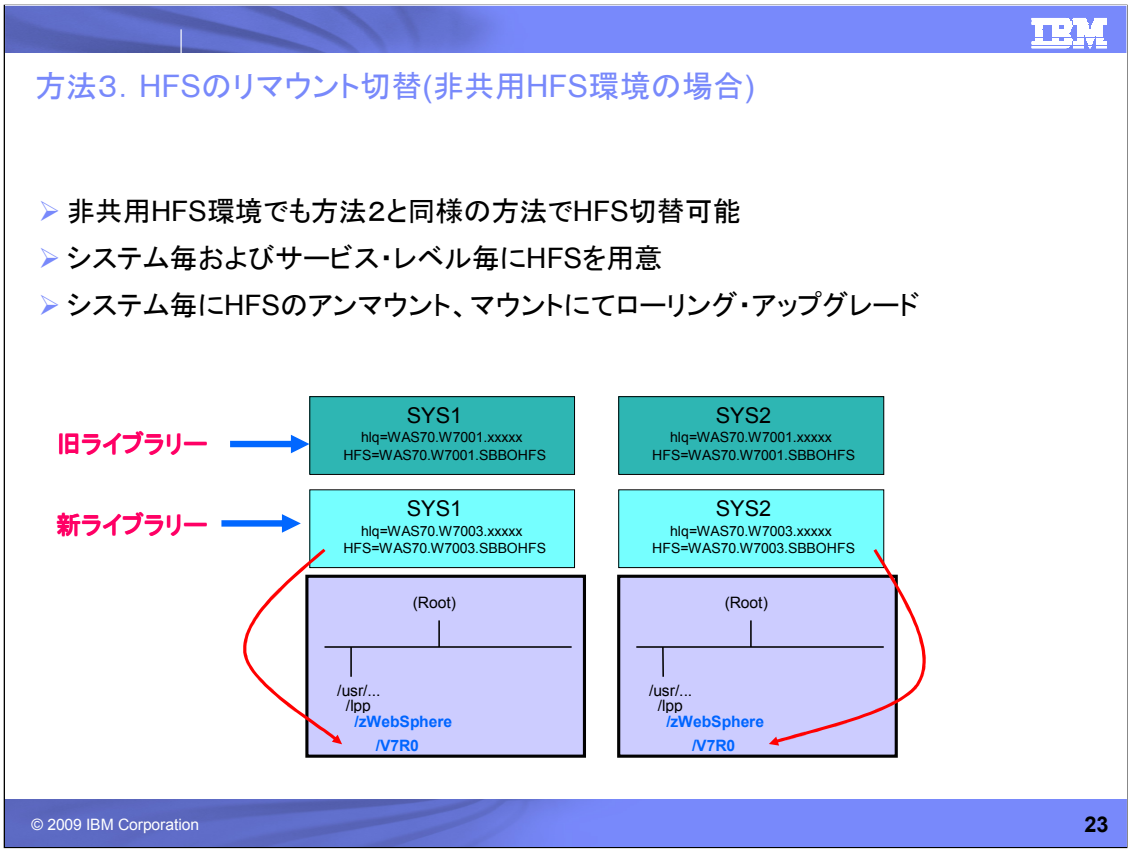

方法1. バージョン固有HFSでのバージョン切替

OMVSを共用HFSで構成した場合に設定するVERSIONを利用してHFSを切替える例です。VERSIONの値はシステムごとに異なる値を持つことが可能であるため、システム毎に順々にメンテナンス・レベルのアップグレードが可能です。




方法2. システム固有HFS配下HFSのリマウント切替

OMVSを共用HFSで構成した場合に、システム固有HFS配下HFSをマウント、アンマウントすることにより切替える例です。システム固有はシステム毎に持つため、システム毎に順々にメンテナンス・レベルのアップグレードが可能です。




方法3. HFSのリマウント切替(非共用HFS環境の場合)

非共用HFS環境でも方法2と同様の方法でHFS切替可能です。共用していないため、システム毎に順々にメンテナンス・レベルのアップグレードが可能です。

## ローリングアップグレードのためのHFS設計比較

| | バージョン固有HFSでのバージョン切替 | システム固有HFS配下のHFSリマウント切替 | HFSリマウント切替<br>(非共用HFS環境の場合) |
|---|---|---|---|
| HFSの準備 | バージョン固有HFSとそれに付随するHFS。<br>システム毎に複製は不要。 | WASおよび関連製品のHFS。<br>システム毎に複製が必要。 | WASおよび関連製品のHFS。<br>システム毎に複製が必要。 |
| 切替方法 | $VERSION切替 | /$SYSNAME配下のHFSリマウント切替 | /usr/lpp/zWebSphere配下のHFSリマウント切替 |
| 選択基準 | 共用HFSが必要な場合。<br>z/OS OMVSのHFSごとの切替のため、DASD容量が多く必要。<br>WASレベルアップ時には他製品との一斉レベルアップを計画する場合。 | 共用HFSが必要な場合。<br>WASおよび関連製品のHFSの複製のため、DASD容量は必要最小限。 | 共用HFSが不要な場合。<br>WASおよび関連製品のHFSの複製のため、DASD容量は必要最小限。 |




ローリングアップグレードのためのHFS設計比較です。

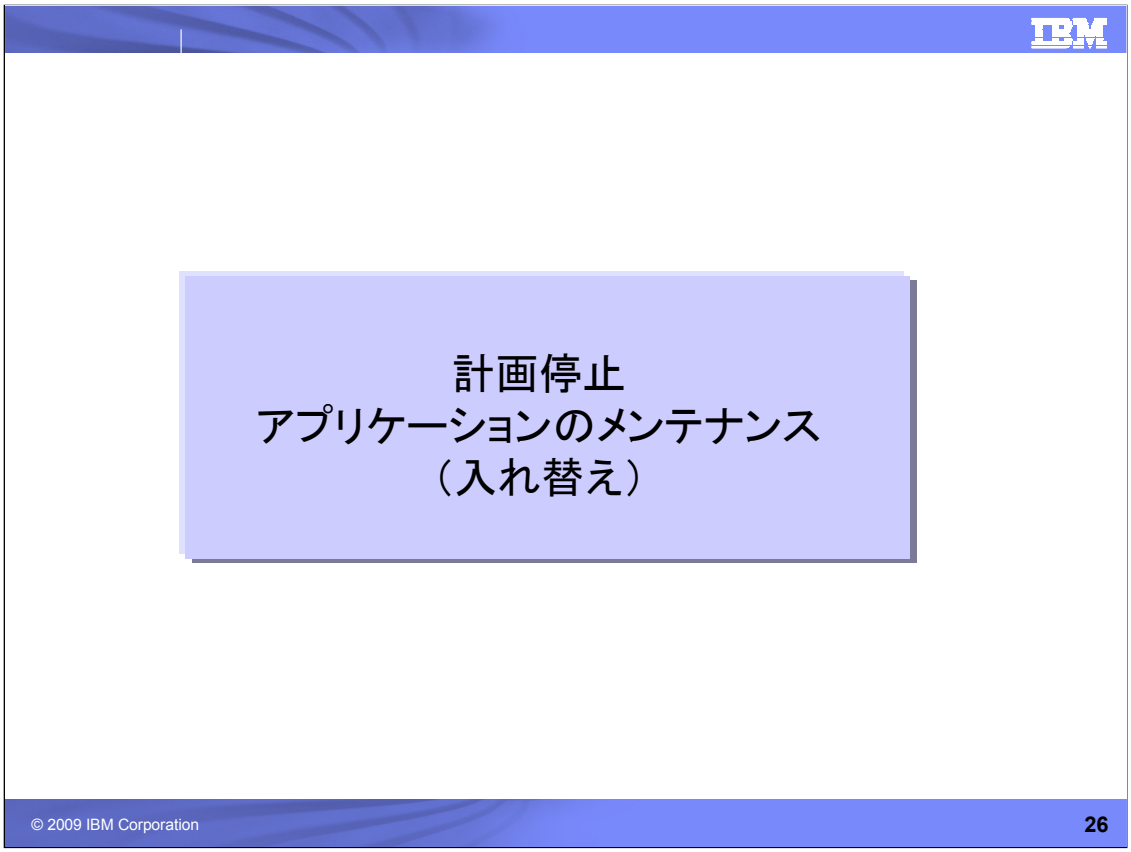
## サービス停止メンテナンスとローリング・メンテナンス

| | サービス一時停止によるメンテナンス | ローリング・メンテナンス |
|---|---|---|
| 製品レベルの保証 | 製品レベルに依存せずに実施可能 | すべてのレベルでローリング・アップグレードが保証されているわけではない |
| サービス停止時間 | あり<br>（テスト環境でテスト済みのライブラリーのコピーによる停止時間の最小化は可能） | 原則としてなし |
| 運用上の負荷 | システム全体の切り替えで、比較的シンプルな手順 | 割り振りの順次停止・再開、順次切り替えの手順が複雑 |
| テストでの負荷 | メンテナンス後ライブラリーによる一通りのアプリケーション稼動テストの実施。<br>本番環境での変更手順の確立。 | 左に加え、以下が主な追加項目となる<br>①新旧システム共存環境でのアプリケーション稼動テストの実施。<br>②順次システム切り替えの運用手順の確立 |
| 特記事項 | 他のサブシステム（OS, MQ, DB2など）の保守と同期を取った保守が可能。 | |




サービス停止によるメンテナンスとローリング・アップグレードの比較です。

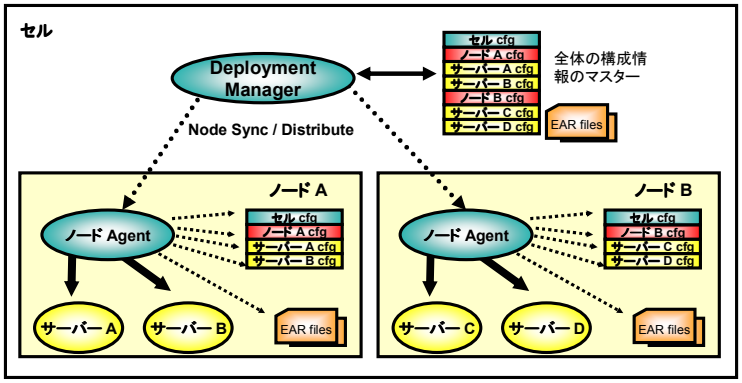
ローリング・アップグレードを行う場合は、異なるサービスレベルでのサーバー共存のテスト実施、手順の確立の負荷、および本番環境でのオペレーション・ミス発生のリスクが高くなります。




アプリケーションの修正や追加などの変更の場合に、計画停止が必要となるケースがあります。

## アプリケーション・メンテナンス関連機能 (1/3)

- ファイル同期サービス
  - デプロイメント・マネージャーで更新されたアプリケーション/構成情報を配下のノードにコピーする機能
  - 自動の場合は **System Administration > Node Agents >** *node_agent_name* **> File Synchronization Service** で設定
    - 始動同期、自動同期(分単位の同期間隔)を指定できる
  - 管理コンソールやシェル・スクリプトでの手動同期も可能
  - 注意: デプロイメント・マネージャーでファイルが誤った状態に変更されたときにはセル内の全てのアプリケーションサーバーに影響がでます







「ファイル同期サービス」は、デプロイメント・マネージャーで更新されたアプリケーション/構成情報を配下のノードにコピーする機能です。

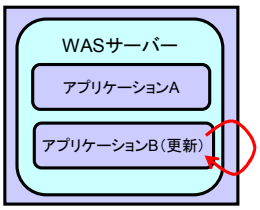
自動の場合は **System Administration > Node Agents >** *node_agent_name* **> File Synchronization Service** で設定します。ここで、始動同期、自動同期(分単位の同期間隔)を指定できます。

管理コンソールやシェル・スクリプトでの手動同期も可能です。

## アプリケーション・メンテナンス関連機能(2/3)

- ホット・デプロイメント（常時配置）
  - アプリケーションサーバーを停止させずにアプリケーションを配置する機能
    - ✓ 基本はデプロイメント・マネージャー配下のアプリケーション/構成を更新
  - ファイル同期サービスと併用することが可能
    - ✓ 配置時の同期オプションを指定することにより、デプロイメント・マネージャー配下のアプリケーション/構成の更新を各ノードのアプリケーション/構成にコピーし、アプリケーションの停止・再起動を行うことも可能

- 動的再ロード
  - 既存のコンポーネントを変更する機能
  - アプリケーション(EAR)単位の設定 （再ロード使用可否、再ロード間隔（秒））
  - 再ロード間隔毎に更新の有無を確認し、更新されている場合に再ロード
  - 変更を有効にするためにサーバーを再始動する必要はありません
  - 動的再ロードには、以下が含まれます
    - ✓ アプリケーションのコンポーネントのインプリメンテーションに対する変更
    - ✓ アプリケーションの設定に対する変更
  - クラスローダーポリシーの設定が“単一”である場合は動的再ロードはされません





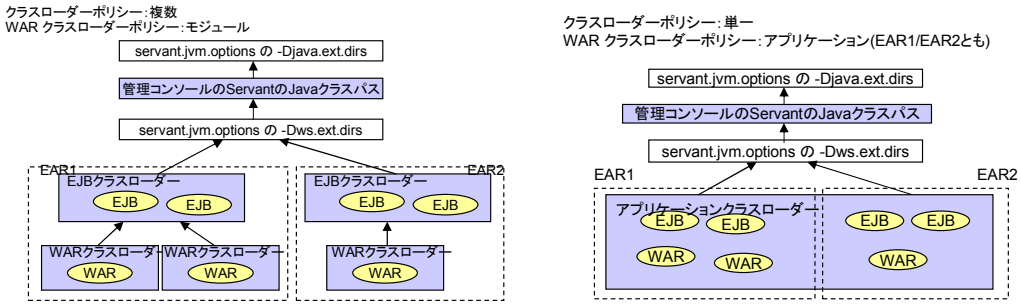


「ホット・デプロイメント（常時配置）」は、アプリケーションサーバーを停止させずにアプリケーションを配置する機能です。基本はデプロイメント・マネージャー配下のアプリケーション/構成を更新します。ファイル同期サービスと併用することが可能です。配置時の同期オプションを指定することにより、デプロイメント・マネージャー配下のアプリケーション/構成の更新を各ノードのアプリケーション/構成にコピーし、アプリケーションの停止・再起動を行うことも可能です。

「動的再ロード」は、既存のコンポーネントを変更する機能です。アプリケーション(EAR)単位の設定（再ロード使用可否、再ロード間隔（秒））が可能です。再ロード間隔毎に更新の有無を確認し、更新されている場合に再ロードします。変更を有効にするためにサーバーを再始動する必要はありません。クラスローダーポリシーの設定が“単一”である場合は動的再ロードはされません。

## アプリケーション・メンテナンス関連機能(3/3)

- クラスローダーポリシー
  - サーバーポリシーの設定
    - **複数**に設定すると、EJB-JARのクラスローダーはEAR単位となる。
    - **単一**に設定すると、そのサーバーのすべてのEARに含まれるEJB-JARに対して共通のクラスローダーが使われる。
  - エンタープライズ・アプリケーションのWAR クラスローダーポリシーの設定
    - **モジュール**にすると、WARクラスローダーは個々のWAR単位になり、それらのWARクラスローダーの親としてEJBクラスローダーが位置する。
    - **アプリケーション**にすると、そのEARに含まれるすべてのEJB-JARとWARに対して共通のクラスローダーが使われる。







アプリケーションのメンテナンスでは、「クラスローダーポリシー」を理解することが重要です。「クラスローダーポリシー」は2つ設定します。

1. サーバーポリシーの設定

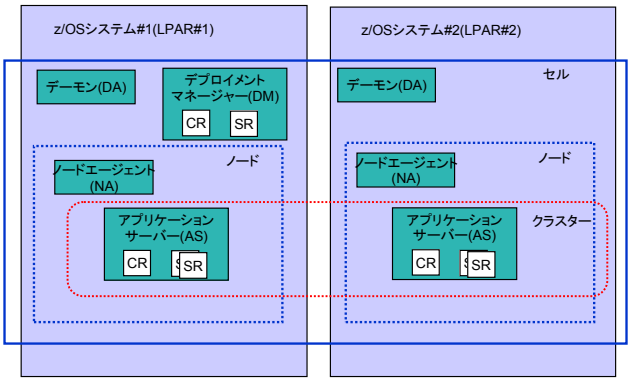
複数に設定すると、EJB-JARのクラスローダーはEAR単位となります。単一に設定すると、そのサーバーのすべてのEARに含まれるEJB-JARに対して共通のクラスローダーが使われます。

2. エンタープライズ・アプリケーションのWAR クラスローダーポリシーの設定

モジュールにすると、WARクラスローダーは個々のWAR単位になり、それらのWARクラスローダーの親としてEJBクラスローダーが位置します。アプリケーションにすると、そのEARに含まれるすべてのEJB-JARとWARに対して共通のクラスローダーが使われます。

## アプリケーション・メンテナンス – 前提

- 当資料で可用性の高いアプリケーション・メンテナンスを検討するにあたり、前提は以下に定めます
  - ND (Network Deployment)構成
  - セルはSysplexでひとつ。ノードはシステム毎にひとつ。サーバーは水平クラスタですべてのノードに配置
- 上記前提により
  - アプリケーションは、クラスタ単位に配置します
  - 管理コンソール上で管理されるアプリケーションの状態（始動、停止）はクラスタ単位にひとつです
  - クラスタにデプロイしたアプリケーションを、あるサーバーのものだけ手動で停止することはできません







当資料で可用性の高いアプリケーション・メンテナンスを検討するにあたり、当資料では前提は以下に定めます。アプリケーション・メンテナンスの典型的な例を判りやすいご理解いただくためです。

・ND構成

・セルはSysplexでひとつ。ノードはシステム毎にひとつ。サーバーは水平クラスタですべてのノードに配置

IBM

## サービス停止を最小限にするアプリケーション・メンテナンス

| | 方法1:<br>縮退稼動切替 | 方法2:<br>2セル構成切替 |
|---|---|---|
| 切替方法 | 複数システムにまたがるクラスタ構成で、ひとつひとつ順にアプリケーションを入替 | z/OSの外でリクエストの割振り先を切替 |
| 考慮 | Webアプリケーションにおいては、セッション・アフィニティーに考慮あり | |
| 稼動状況 | 縮退稼動 | 通常構成での稼動 |
| サービスの停止 | 旧アプリケーションと新アプリケーションの混在が<br>・可能であれば停止時間なし<br>・不可であれば停止時間あり(振分停止と振分再開の間の時間) | |
| 必要なシステム資源 | 通常のSysplex構成(N+1) | 各システムでの追加資源(CPU, メモリー, DASD 等)が必要 |
| メリット | 追加資源が不要。 | 旧アプリケーションから新アプリケーションへのスムーズな移行。<br>旧アプリケーションへの戻しが容易。 |




サービス停止を最小限にするアプリケーション・メンテナンスは、大きく2つの方法があります。

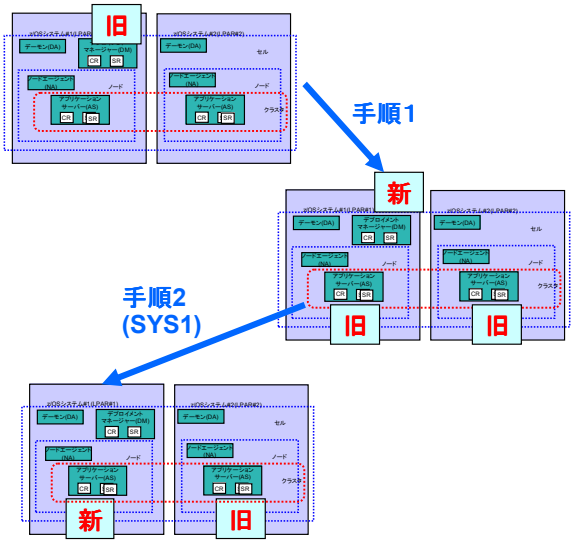
方法1は、N+1のSysplexにおいて、ND構成のクラスターにおいてひとつひとつアプリケーションを入替える方法です。
方法2は、方法1でのセル構成を2つ構成し、z/OSの外でリクエストの割振り先を切替える方法です。

Webアプリケーションで、セッション・オブジェクトを使用している場合には、リクエストの割振り制御の際、セッション・アフィニティーの考慮が必要です。

## 方法1： 縮退稼動切替

- 手順

1. アプリケーションの更新デプロイ
   - 保管時「ノードと変更を同期化」にチェックしない。
2. 各ノードにて
   - ① リクエスト振り分け停止
   - ② サーバーの停止
   - ③ ノードの同期
   - ④ サーバーの起動
   - ⑤ リクエスト振り分け再開







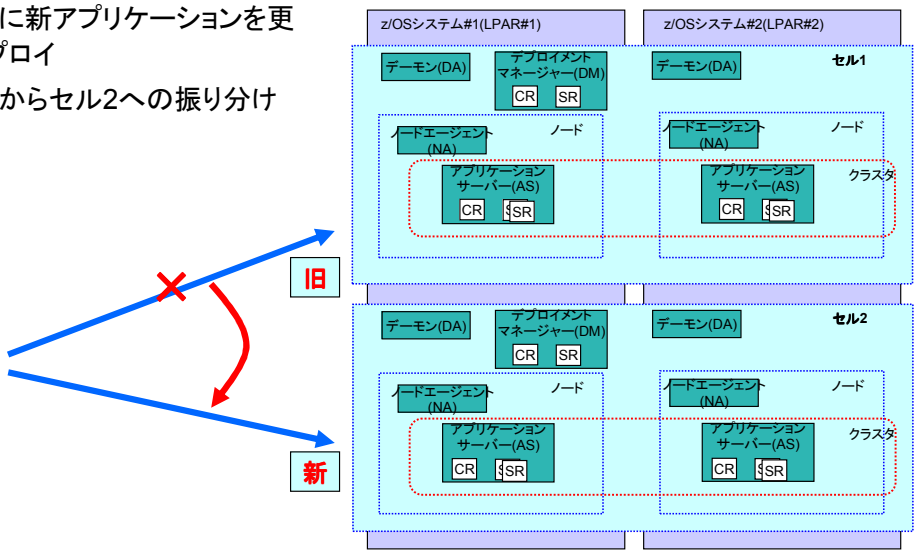
方法1の縮退稼動切替においては、旧アプリケーションと新アプリケーションの混在が可能であれば、サービスの停止なしにアプリケーションの切替えが可能です。混在ができない場合は、振分け停止と振分け再開との間でサービス停止が発生します。

手順の流れとしては、アプリケーションの更新デプロイの保存時に「ノードと変更を同期化」のチェックをしないことにより、アプリケーションの更新をデプロイメント・マネージャーの構成範囲に留めておきます。その後、各ノードにて、リクエストの振分け停止作業後、サーバーの停止、ノードの同期、サーバーの起動、リクエストの振分け再開を行います。各ノードの構成に更新後のアプリケーションが反映されるのは、ノードの同期のタイミングです。

## 方法2： 2セル構成切替

- 手順

1. セル2に新アプリケーションを更新デプロイ
2. セル1からセル2への振り分け切替

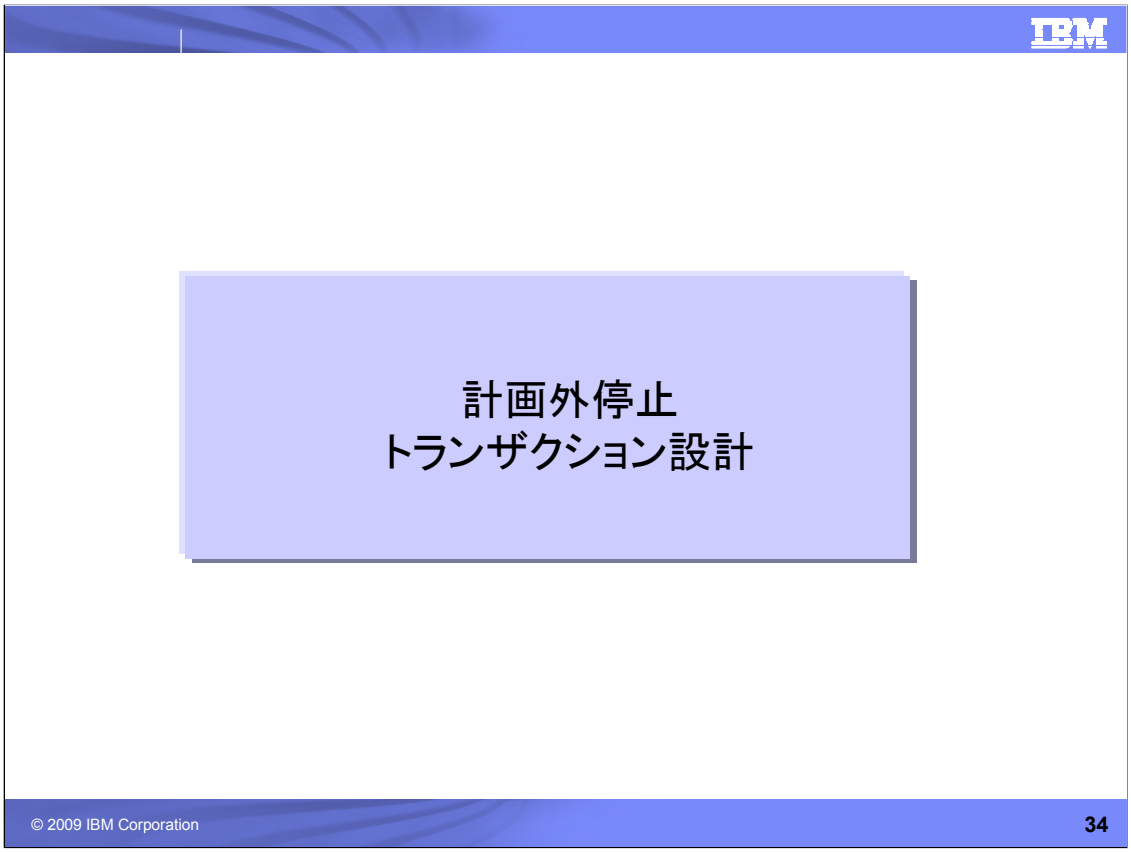






方法2の2セル構成切替においては、旧アプリケーションと新アプリケーションの混在が可否に関係なく、サービスの停止なしのアプリケーションの切替えが可能です。WAS構成は、通常稼動しているものと同じ構成をもうひとつ持つことになりますので、システム資源も追加の資源（CPU、メモリー、DASD　等）が必要になります。

手順の流れとしては、切替え先となるセル2へ新アプリケーションの更新デプロイし、準備ができた段階で、z/OSの外でのリクエストの振分け切り替えを行います。旧アプリケーション構成がそのまま残っているので、新アプリケーションに何等かの不具合があった場合の戻し作業もスムーズに行えます。




計画外停止（意図せぬサーバー・ダウン）について、考えられるパターンと全面ダウンを最小化する方法について検討します。

## WAS for z/OSにおけるトランザクション設計の原則

- 1フェーズ・コミット処理可能な場合は1フェーズにする。無意味に2フェーズにしない。
  - リカバリー不要な資源をトランザクション処理に含めない
  - ひとつのアプリケーションからのDB2接続では同じデータソースを使用する
  - ひとつのアプリケーションからのMQ接続では同じキュー接続ファクトリーを使用する
- 2フェーズ・コミット処理の場合は、同期点処理時間を短く設計する
  - 資源はz/OS内で完結させる
  - リモート資源をトランザクション処理に含めない
- ひとつのトランザクション・タイプに統一する
  - RRSトランザクションとXAトランザクションの混在を避ける
  - どちらでも選択可能であれば、RRSトランザクションに統一する
    - DB2接続：Type4(XA)よりType2(RRS)を使用する
    - MQ接続：トランスポート・タイプでCLIENT(XA)よりBINDINGS(RRS)を使用する




WAS for z/OSにおけるトランザクション設計の原則は、大きく3つあります。いずれも、障害発生時の影響範囲を最小限にするための原則です。

## 2フェーズ・コミットの問題と対応

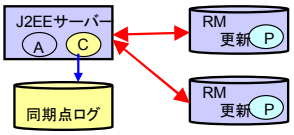
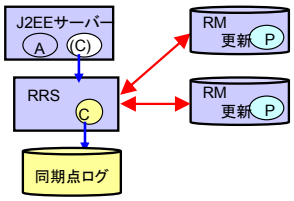
- 2フェーズ・コミットの問題は、障害時の排他制御の経過時間とデータ整合性とのトレード・オフです

| 問題 | 対応 |
|---|---|
| 同期点処理の経過時間が長い<br>障害でインダウトになる時間が比較的長い | 同期点処理フローはできる限りネットワークを経由させない |
| リソース・マネージャーを跨るデッドロックの恐れがある<br>各リソース・マネージャーのロック管理機能では対応できない<br>コーディネーターはデッドロックに対応できない | 開発標準やフレームワークで資源へのアクセス順をコントロール<br>発生してしまったらタイムアウト検知しロールバック |
| トランザクションの結果はコーディネーターだけが知っている<br>リソース・マネージャーだけの判断で資源の更新をコミットまたはロールバックすると、他の資源との整合性を損なう恐れがある | ヒューリスティックな決断（コーディネーター抜きでの回復）のルールと手順を用意しておく<br>アプリケーションに障害回復後の整合性チェックの機能を持たせる |
| 同期点処理中のコーディネーターの障害により、リソースの可用性が落ちる<br>コーディネーターが回復するまで同期点処理は完結しない<br>その間、資源の排他制御は解除できない<br>可用性はコンポーネントの可用性の掛け算 | コーディネーターの可用性をできる限り高くする<br>コーディネーターとリソース・マネージャーをできる限り同じコンポーネントに配置する<br>（その究極がローカル・トランザクション） |




2フェーズ・コミットの問題と対応を列記してあります。2フェーズ・コミットの問題は、障害時の排他制御の経過時間とデータ整合性とのトレード・オフです。

## XAトランザクションとRRSトランザクション

| | XAトランザクション | RRSトランザクション |
|---|---|---|
| コーディネーター | J2EEサーバー<br>  <br><br>A: アプリケーション<br>C: コーディネーター<br>RM: リソース・マネージャー<br>P: パーティシパント | RRS<br>  <br><br>(c): RRSへコーディネーターを委譲 |
| 同期点ログ | ファイルシステム上のファイル　または<br>システム・ロガーのログストリーム(WASログ) | システム・ロガーのログストリーム(RRSログ) |
| 可用性 | J2EEサーバーの可用性(単一）と<br>同期点ログの可用性　の掛け算 | RRSの可用性(Sysplexで複数)と<br>同期点ログの可用性　の掛け算 |
| リソース | ・DB2 JDBC/SQLJ Type 4<br>・MQ JMS トランスポート・タイプ：CLIENT | ・DB2 JDBC/SQLJ Type 2<br>・MQ JMS トランスポート・タイプ：BINDINGS |




XAトランザクションとRRSトランザクションとの違いの表です。

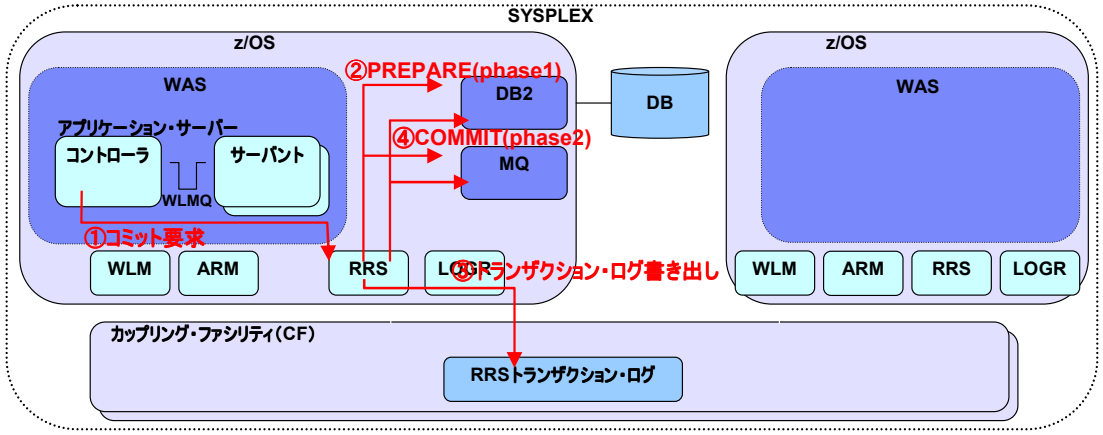
XAトランザクションでは、J2EEサーバーがコーディネーターになります。
RRSトランザクションでは、J2EEサーバーはRRSへコーディネーターを委譲しますので、RRSがコーディネーターになります。

XAトランザクションにおける可用性は、J2EEサーバーの可用性と同期点ログの可用性の掛け算になります。障害発生時にログから資源のリカバリー処理を行うのは、J2EEサーバーです。J2EEサーバーが再起動するまで、更新された資源のin-doubtに関連するロックは開放されないままになります。プロセッサーやOS障害の際に、J2EEサーバーが再起動されるまで時間がかかる場合も同様にin-doubtに関連するロックの開放までに長い時間を要することになります。

RRSトランザクションにおける可用性は、RRSの可用性と同期点ログの可用性の掛け算になります。RRSは、それぞれのOS上で稼動していますので、Sysplexでは複数存在します。プロセッサーやOS障害の際にも、障害が発生していない正常稼動のOS上のRRSによりログが読まれ、in-doubt解消は速やかに実行されます。

表の最後の行にXAリソースとなるもの、RRSリソースとなるものを記載しました。

## RRSによる同期点コントロール

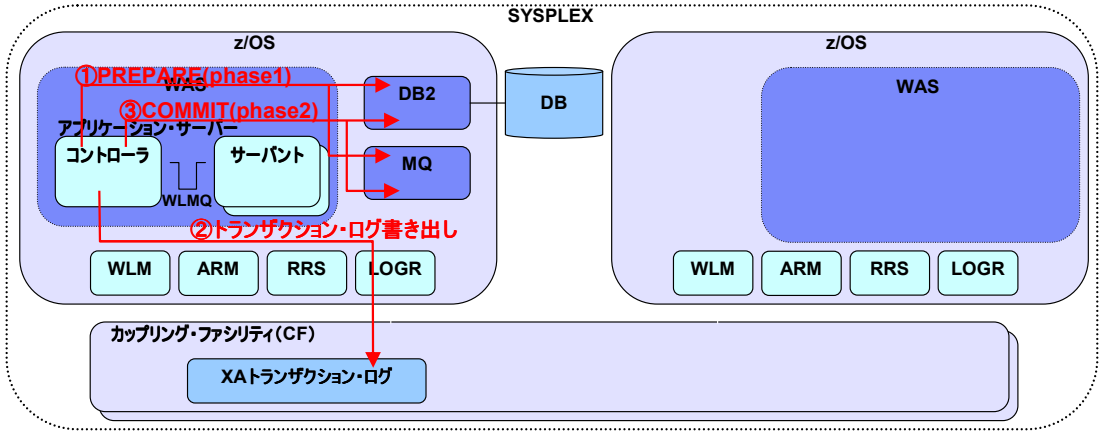



- RRSはz/OS上で稼動するミドルウェアやDBマネジャーにまたがる同期点処理を制御
  1. WASはコミット処理をRRSにリクエストする
  2. RRSは各リソース・マネジャー(DB2,MQ)にPREPAREを要求する(2フェーズ・コミットのPhase1)
  3. RRSはトランザクション・ログをLOGR経由でCFに書く(このログは障害回復で使用される)
  4. RRSは各リソース・マネジャー(DB2,MQ)にCOMMITを要求する(2フェーズ・コミットのPhase2)




WASからRRSにコミット要求が出ると、2フェーズコミット処理はRRSとDB2/MQの間のみで行われますので、万一のインダウト発生時でもWASの再起動が不要。というのがポイントです。

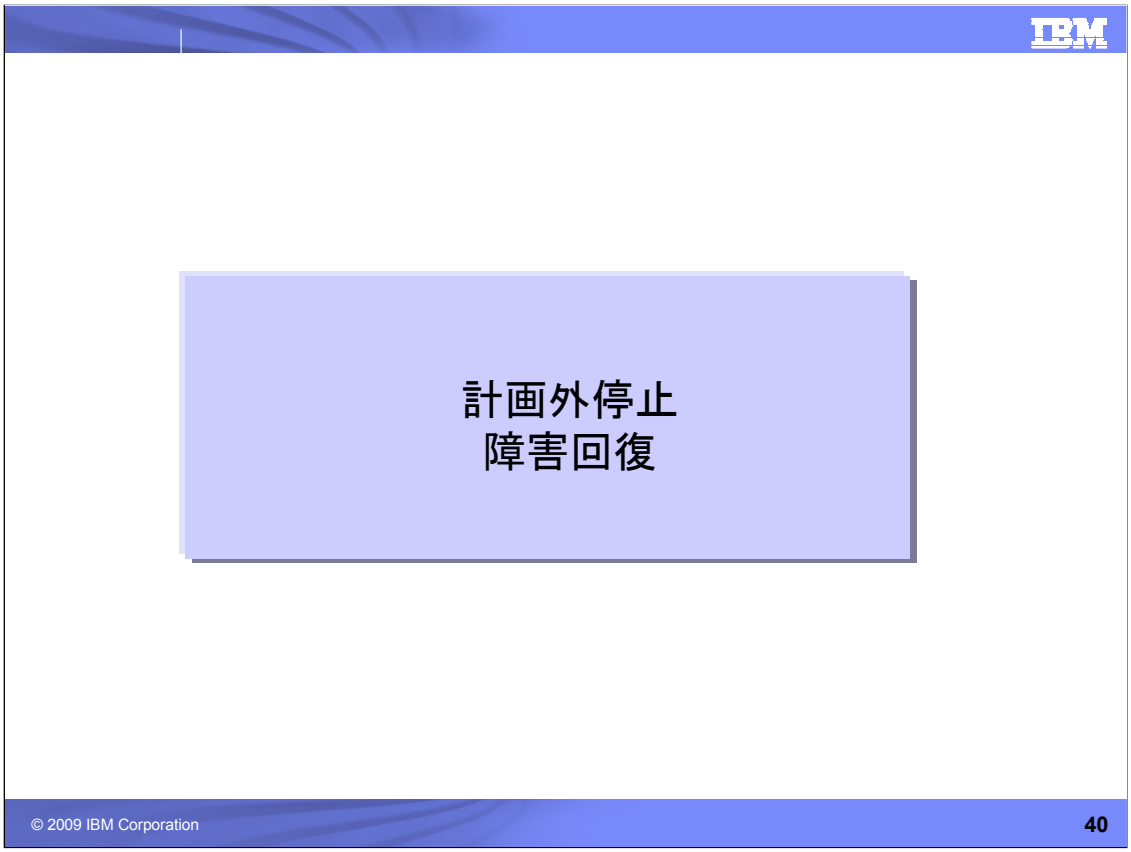
## XA選択時の同期点コントロール



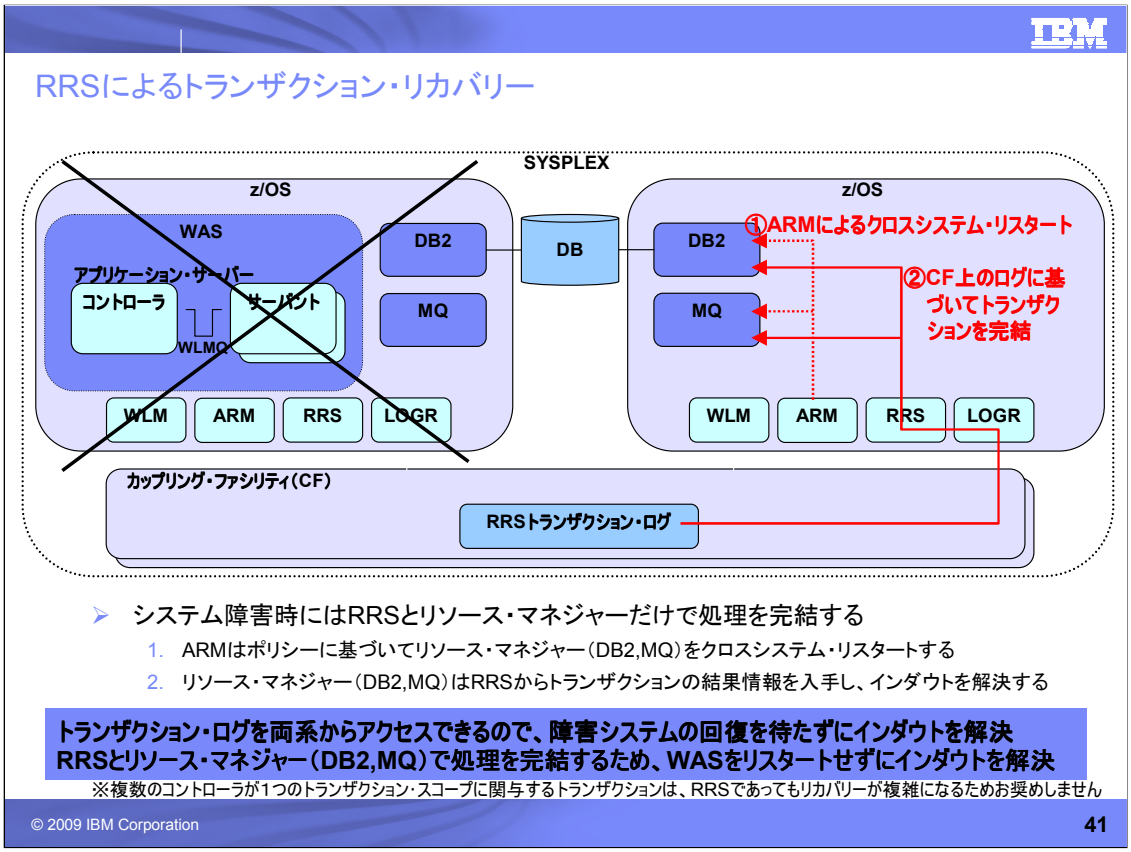

- XAを選択した場合はWAS自身が同期点処理をコントロールする
  1. WASは各リソース・マネジャー(DB2,MQ)にPREPAREを要求する(2フェーズ・コミットのPhase1)
  2. WASはトランザクション・ログをLOGR経由でCFに書く(このログは障害回復で使用される)
  3. WASは各リソース・マネジャー(DB2,MQ)にCOMMITを要求する(2フェーズ・コミットのPhase2)




XAトランザクションはWASがコーディネーターとなりますので、万一のインダウト発生の場合のトランザクション回復にWASの再起動が必要となります。(HA Managerがある場合は、再起動がいらなくなります。)

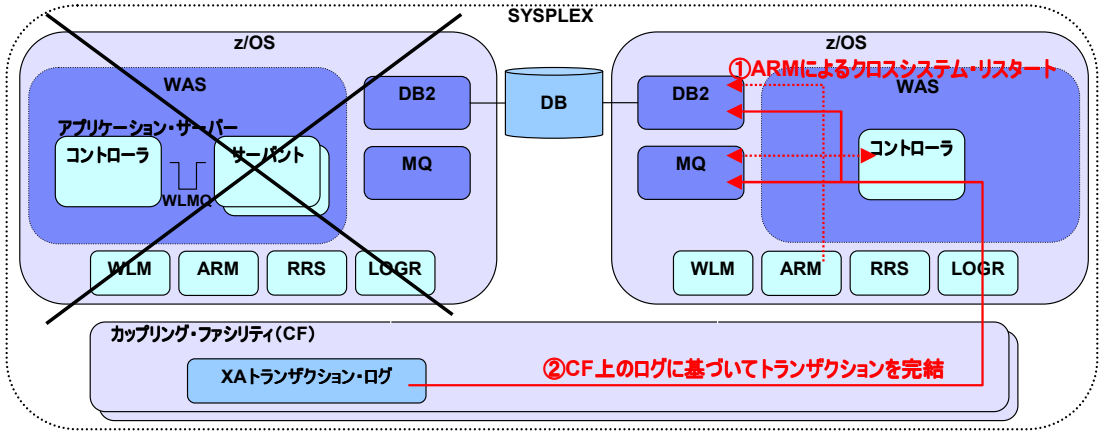



障害が発生してしまった場合の回復方法についてまとめます。




RRSのトランザクション・ログをCF上のストラクチャーに置くことで、Sysplexのすべてのメンバーがアクセスすることができます。

## XA選択時のトランザクション・リカバリー




- システム障害時にはWASもリスタートして処理を完結する
  1. ARMはポリシーに基づいてリソース・マネジャー(DB2,MQ)とWASをクロスシステム・リスタートする
  2. リソース・マネジャー(DB2,MQ)はWASからトランザクションの結果情報を入手し、インダウトを解決する

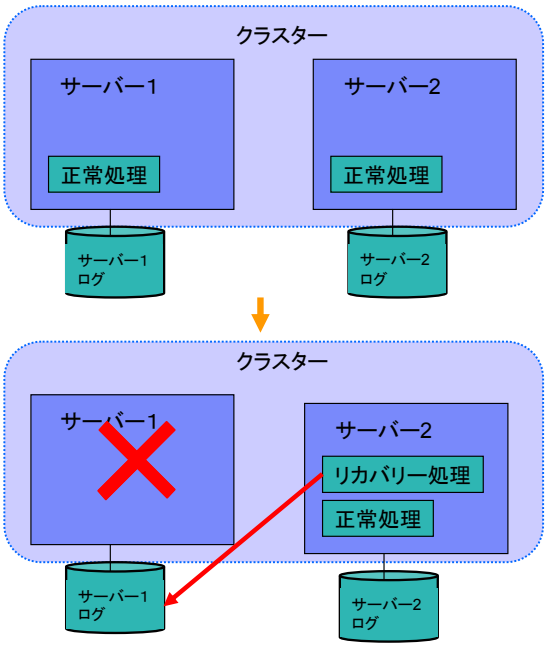
**トランザクション・ログを両系からアクセスできるので、障害システムの回復を待たずにインダウトを解決**
**WAS再始動やWAS－リソース・マネジャー間の通信再確立が必要のため、RRSに比べ回復時間大**




XAの場合は、HA Managerを使わない場合、障害が起こっていないSysplexメンバーでDB2/MQのリソースマネージャのほかに、コーディネーターのWASも再起動することが必要です。

## HA ManagerによるXAトランザクションのリカバリー

- High Availability Manager
- サーバー障害の検知
- ピア・リカバリー
  - 稼動中のクラスター・メンバーが障害サーバーのフェールオーバーを実施できる







HA ManagerはV6からのWASの機能で、クロスシステム・リスタートをせずにXAトランザクションを回復することができるようになりました。

## WAS for z/OS V7におけるトランザクション・リカバリー

- Sysplex内の1ノードがシステム障害を起こした場合のトランザクション・リカバリー
  - システム障害でシステムが再起動されるまでにリカバリー可能な方法

| | HA機能を使用しない | HA機能を使用 |
|---|---|---|
| XAトランザクション | （システム再起動まで）不可 | 可 |
| RRSトランザクション | 可 | 不要 |




HA Managerの機能とRRSトランザクションの回復機能は重複していませんので、RRSトランザクションのみの場合はHA Manager機能をオフにすることもでき、それによってWASの消費リソースを節約することができます。

## 障害コンポーネントと影響範囲

| No. | 障害の現象 | 関連して発生する障害 | 障害の切離し | 障害中の新規業務 | 回復(例) |
|---|---|---|---|---|---|
| 1 | AppServer<br>SR ABEND<br>(1回) | なし | SR自身がWLMキューよりリクエストを拾い処理を行うため切離しの必要はない。 | 他のSRにより処理される。<br>他のSRが無い場合にはリクエストはSRの再起動まで待たされる。<br>他のAppServerへのリクエストは正常に処理される。 | WLMによる自動再始動。 |
| 2 | AppServer<br>SR 連続ABEND | ABENDが単位時間に一定回数を超えるとWLMのAPPLENVがSTOP状態になりAppServer SRは起動されなくなる。 | WLMが単位時間にあたりのSR起動回数を制限し必要以上にSRの起動を繰り返さない。<br>該当AppServerのCRは稼動したままとなる。 | 該当AppServer CRへのリクエストはエラーとなる。SDでQoSを加味した割振りにしている場合は、該当AppServerが割振り対象から外される。 | ABEND原因を除去し、<br>RESUMEコマンドにて回復。 |
| 3 | AppServer CR<br>ABEND | 該当AppServer配下のSRがABENDする。 | SDが該当AppServerのポートが閉じられたことを検知。<br>SDは該当AppServerへの割振りを停止する。 | 障害切離しが行われているので、新規リクエストは正常系システムへ割振られ正常に処理される。 | ARMによる自動再始動。<br>SDによる割振りはWAS AppServerのポートがオープンされた際に自動で再開される。 |
| 4 | Node Agent<br>ABEND | 管理コンソールにおいて、該当Nodeの状況把握、操作ができなくなる。 | アプリケーション処理に直接関連しないため、アプリケーション・フローからの切離し処理はない。 | アプリケーション処理に直接関連しないため、Node Agentが上がっていないNodeにおいてもリクエストは正常に処理される。 | ARMによる自動再始動。<br>Node Agent起動後、オペレータ操作なしに管理コンソールにおける該当Nodeの状況把握、操作が可能となる。 |




WASは多くのコンポーネントで成り立っていますので、それらのどこで障害が発生したかによって回復方法が異なります。

このページ以降で列挙します。

## 障害コンポーネントと影響範囲

| No. | 障害の現象 | 関連して発生する障害 | 障害の切離し | 障害中の新規業務 | 回復(例) |
|---|---|---|---|---|---|
| 5 | Deployment Manager SR ABEND（1回） | 管理コンソール自体が使用不可となる。<br>回復するまで管理コンソールは使用不可。 | アプリケーション処理に直接関連しないため、アプリケーション・フローからの切離し処理はない。 | アプリケーション処理に直接関連しないため、Deployment Managerが上がっていなくてもリクエストは正常に処理される。 | WLMによる自動再始動。 |
| 6 | Deployment Manager CR ABEND | Deployment ManagerのSRがABENDする。<br>管理コンソール自体が使用不可となる。<br>回復するまで管理コンソールは使用不可。 | アプリケーション処理に直接関連しないため、アプリケーション・フローからの切離し処理はない。 | アプリケーション処理に直接関連しないため、Deployment Managerが上がっていなくてもリクエストは正常に処理される。 | ARMによる自動再始動。 |
| 7 | DB2 ABEND<br>(Universal Type2 driver) | AppServerからDB2へのスレッドが切断される。該当コネクションにアクセスするとStaleConnectionExceptionになる。<br>DB2再起動後は再接続可能。 | 自動での障害の切離し処理は行われない。<br>DB2が再起動されるまでのリクエストはエラーとなる | 該当DB2と同じシステムで稼動するAppServerへの新規リクエストは、DB2が再起動されるまでの間はエラーとなる。正常系リクエストは正常に処理される。ただし、Retained Lockがある場合ロックされた資源へのアクセスは不可。 | DB2はARMにより自動再始動。<br>DB2再起動後は新たなDB2アクセスのアプリケーションが実行されるタイミングでDB2と再接続される。 |
| 8 | DB2 ABEND<br>(Universal Type4 driver) | AppServerからDB2へのTCP/IPコネクションが切断される。該当コネクションにアクセスするとStaleConnectionExceptionになる。<br>DB2が再起動されるとTCP/IPコネクションは再確立される。 | 自動での障害の切離し処理は行われない。<br>DB2が再起動されるまでのリクエストにはエラーとなる。 | 該当DB2と同じシステムで稼動するAppServerへの新規リクエストは、DB2が再起動されるまでの間はエラーとなる。正常系リクエストは正常に処理される。ただし、Retained Lockがある場合ロックされた資源へのアクセスは不可。 | DB2はARMにより自動再始動。<br>DB2再起動後は新たなDB2アクセスのアプリケーションが実行されるタイミングでDB2とのTCP/IPコネクションが再確立される。 |



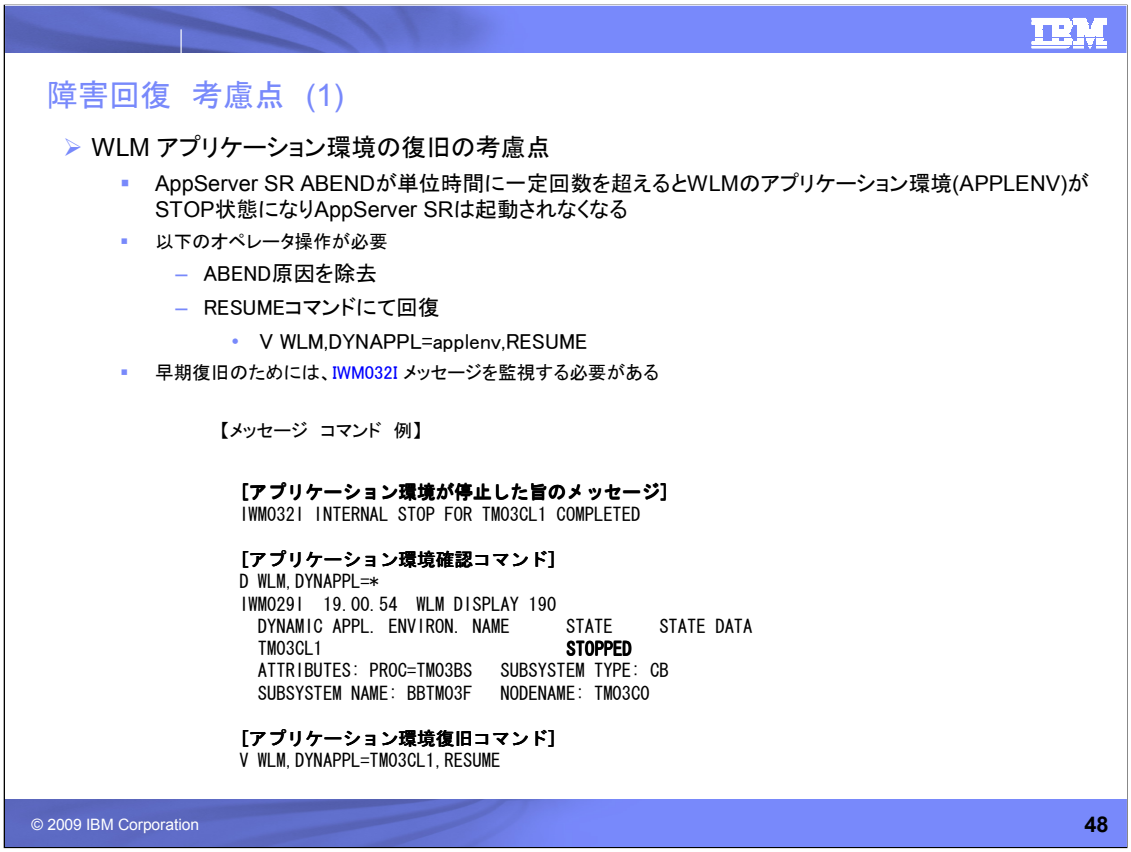

前頁の続き

## 障害コンポーネントと影響範囲

| No. | 障害の現象 | 関連して発生する障害 | 障害の切離し | 障害中の新規業務 | 回復(例) |
|---|---|---|---|---|---|
| 9 | TCP/IP ABEND | WASのTCP/Iコネクションがすべて切断される。<br>TCP/IP回復後、すべてのWASリージョンの停止/再始動が必要。 | SDの前提設定である動的VIPAによりIPが他のシステムにテイクオーバーされる。Distributorスタック障害では、バックアップにDistributor機能が引き継がれる。 | 該当システムへの割振り要求はIPを引き継いだ他システムで処理される。アフィニティーのあるリクエストの場合、セッション・パーシスタンスの工夫が無ければエラーとなる。正常系システムへのリクエストは正常に処理される。 | ARMによる自動再始動。<br>TCP/IP回復後、すべてのWASリージョンの停止/再始動が必要。 |
| 10 | LOGGER ABEND | RRSによるロギングが休止する。<br>障害系のWASの同期点処理が休止する。 | 切離し処理はない。<br>LOGGER障害が長引くようであれば、システムごと割振り対象から切離す必要がある。 | 該当システムへのリクエストのうち、RRSのロギングを必要とする処理はすべて休止する。正常系システムへのリクエストは正常に処理される。 | 手動または自動化によるLOGGER再起動。 |
| 11 | RRS ABEND | 該当RRSが稼動していたシステムのすべてのWASリージョンがABENDする。 | AppServer CRもABENDするため、SDが該当AppServerのポートが閉じられたことを検知。<br>SDは該当AppServerへの割振りを停止する。 | 障害切離しが行われているので、新規リクエストは正常系システムへ割振られ正常に処理される。 | ARMによる自動再始動。 |
| 12 | System Down | 停止したシステム上のすべてのリージョンが停止。 | SFMによりSysplexからシステムが切離される。<br>ネットワークもSDの前提設定である動的VIPAによりIPが他のシステムにテイクオーバーされる。 | 障害切離しが行われているので、新規リクエストは正常系システムへ割振られ正常に処理される。ただし、Retained Lockがある場合ロックされた資源へのアクセスは不可。 | Retained Lock解消のため、DB2を他系で再始動する。ARMによる自動再起動が可能。ロック解消後は新規リクエストを受け付けずに停止。システムを再起動して回復。 |




前頁の続き

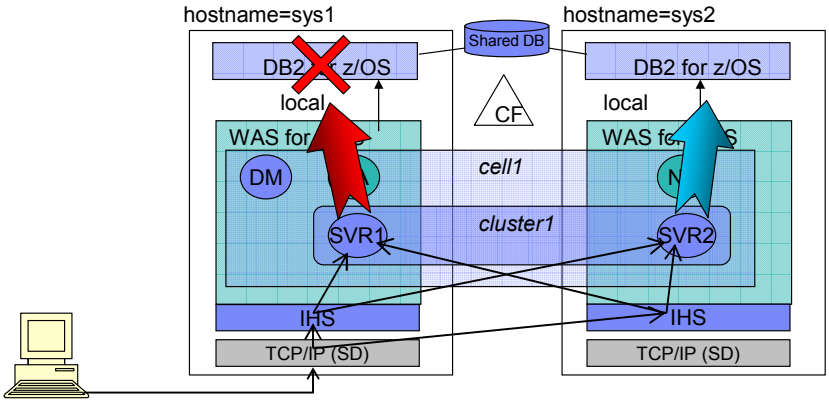



WASのSRは、WLM applenvという機能でABEND時に自動的に再起動されるのですが、起動しきらないうちに再びABENDを繰り返すとapplenvが停止します。この場合は原因究明の上、手作業でのresumeが必要です。

## 障害回復　考慮点　(2)

- DB2 障害の考慮点（Type2 JDBC接続）
  - DB2メンバーの片系ダウンでも正常系へのリクエストは処理可能 (Retained-Lockを除く)
  - 障害系へのリクエスト割振りを停止し，正常系へ割振るための自動化処理が必要
    - DB2再起動までの時間とWAS-DB2の再接続を考慮
    - そのノードでのWASクラスターメンバーのリスナーを停止し、回復時にリスナーを開く

```
F server,PAUSELISTNERS
F server,RESUMELISTENERS
```







WASのバックエンドにDB2接続がある場合、DB2の障害が発生するとWASのアプリケーションとして成立しなくなります。そこで、その場合には前面のWASの処理受付（リスナー）のみを停止させ、DB2回復時に再開する方法がよいでしょう。

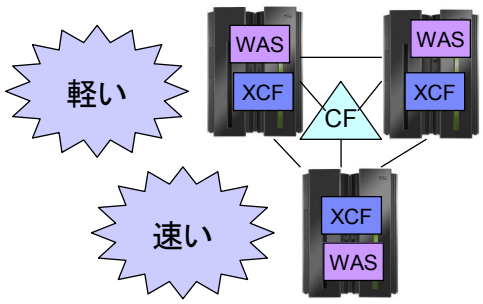
WASの停止・起動のオーバーヘッドがなく、MVSコマンドで簡単に停止・再開ができます。

## WAS for z/OS V7新機能：XCFによるHAマネージャー障害検知

- XCFによるHAマネージャー障害検知
  - コア・グループ・メンバー障害検知のプロトコルにXCFが使用可能
    - コア・グループの全てのサーバーがWAS for z/OS V7.0であること
    - コア・グループの全てのサーバーが、ひとつのSysplex上で稼動していること
    - XCFによるHAマネージャー障害検知の設定がされていること（デフォルトではない）

    →サーバーは起動時にXCFグループのメンバーとしてJOINします
- XCFを使用しない障害検知は、TCP/IPでのハートビート

- メリット
  - 従来のハートビートの仕組みに比べ
    - CPU使用率の軽減
    - 障害検知時間の短縮




※ XCF：Cross System Coupling Facility
※ XCFグループ：Sysplexの中の複数OS間のシグナリング・サービスを使用して連携を取る機能グループ




WAS V7で障害検知の機能が拡張されました。

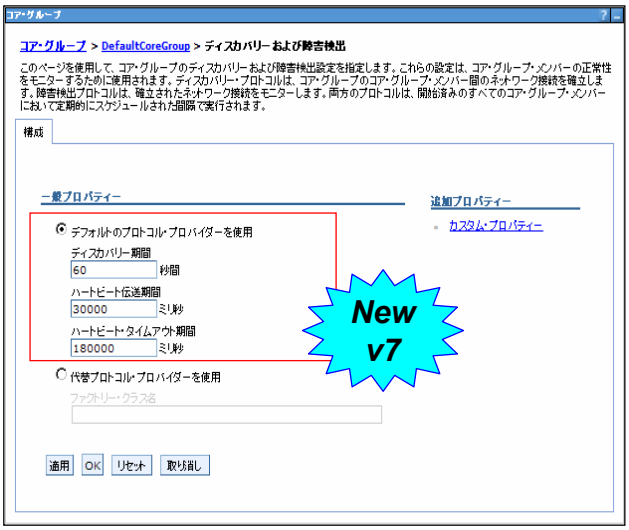
従来のHAマネージャーでの障害検知は、コア・グループ・メンバーがActiveであることを確認するために、メンバー間でTCP/IPプロトコルによるハートビートが行われていました。

WAS for z/OS V7で新しく登場した“XCFによるHAマネージャー障害検知”では、WASがXCFグループのメンバーとして参加することで、障害検知にXCFを利用することができるようになりました。XCFグループとは、Sysplexの中の複数OS間のシグナリング・サービスを使用して連携を取る機能グループです。

“XCFによるHAマネージャー障害検知”では、従来のハートビートの仕組みに比べ、CPU使用率を軽減でき、障害検知時間を短縮することができます。

## WAS for z/OS V7新機能：XCFによるHAマネージャー障害検知

- HAマネージャー障害検知のデフォルト設定
  - 管理コンソール：コア・グループ設定 > DefaultCoreGroup > ディスカバリーおよび障害検出 > ディスカバリーおよび障害検出
  - デフォルトのプロトコル・プロバイダーを使用では、以下の設定が可能
    - ディスカバリー期間(秒)
      - デフォルト180秒
    - ハートビート伝送期間(ミリ秒)
      - デフォルト30000ミリ秒
    - ハートビート・タイムアウト期間(ミリ秒)
      - デフォルト180000ミリ秒
  - WAS V6.xのカスタムプロパティは非推奨
    WAS V6.xとWASV7の混合セル環境でのみ使用
    - IBM_CS_UNICAST_DISCOVERY_INTERVAL_SECS
    - IBM_CS_FD_PERIOD_SECS
    - IBM_CS_FD_CONSECUTIVE_MISSED







WAS V7でも、HAマネージャー障害検知のデフォルト設定は、TCP/IPプロトコルによるハートビートです。

管理コンソールの、コア・グループ設定 > DefaultCoreGroup > ディスカバリーおよび障害検出 > ディスカバリーおよび障害検出 で設定します。

“デフォルトのプロトコル・プロバイダーを使用”では、ディスカバリー期間(秒)、ハートビート伝送期間(ミリ秒)、ハートビート・タイムアウト期間(ミリ秒)の設定が可能です。

WASV6.xの下記カスタムプロパティは非推奨になりました。WASV6.xとWASV7の混合セル環境でのみご使用下さい。

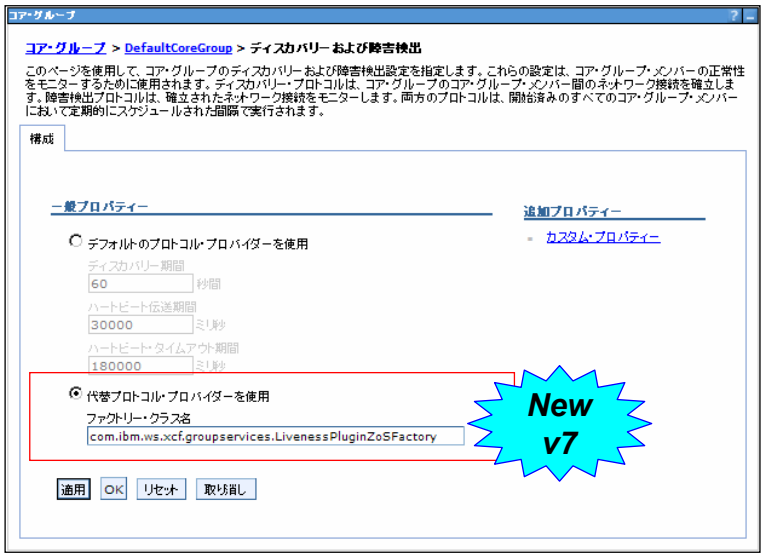
・IBM_CS_UNICAST_DISCOVERY_INTERVAL_SECS

・IBM_CS_FD_PERIOD_SECS

・IBM_CS_FD_CONSECUTIVE_MISSED

## WAS for z/OS V7新機能: XCFによるHAマネージャー障害検知

- XCFによるHAマネージャー障害検知の設定
  - 管理コンソール:コア・グループ設定 > DefaultCoreGroup > ディスカバリーおよび障害検出 > ディスカバリーおよび障害検出
    - 「代替プロトコル・プロバイダーを使用」 を選択
    - ファクトリー・クラス名に、「com.ibm.ws.xcf.groupservices.LivenessPluginZoSFactory」を指定








XCFによるHAマネージャー障害検知の設定は、前ページと同じページで設定します。

管理コンソールの、コア・グループ設定 > DefaultCoreGroup > ディスカバリーおよび障害検出 > ディスカバリーおよび障害検出

「 XCFによるHAマネージャー障害検知」へ変更するには、

・「代替プロトコル・プロバイダーを使用」のラジオ・ボタンをチェックし

・ブランクになっているファクトリー・クラス名にXCFを使用するファクトリー・クラス名を設定します。

ファクトリー・クラス名 :

**com.ibm.ws.xcf.groupservices.LivenessPluginZoSFactory.**