WebSphere Application Server for z/OS V7.0

パフォーマンス　Hints&Tips

WebSphere software

免責事項

当資料は、2008年9月に発表された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。

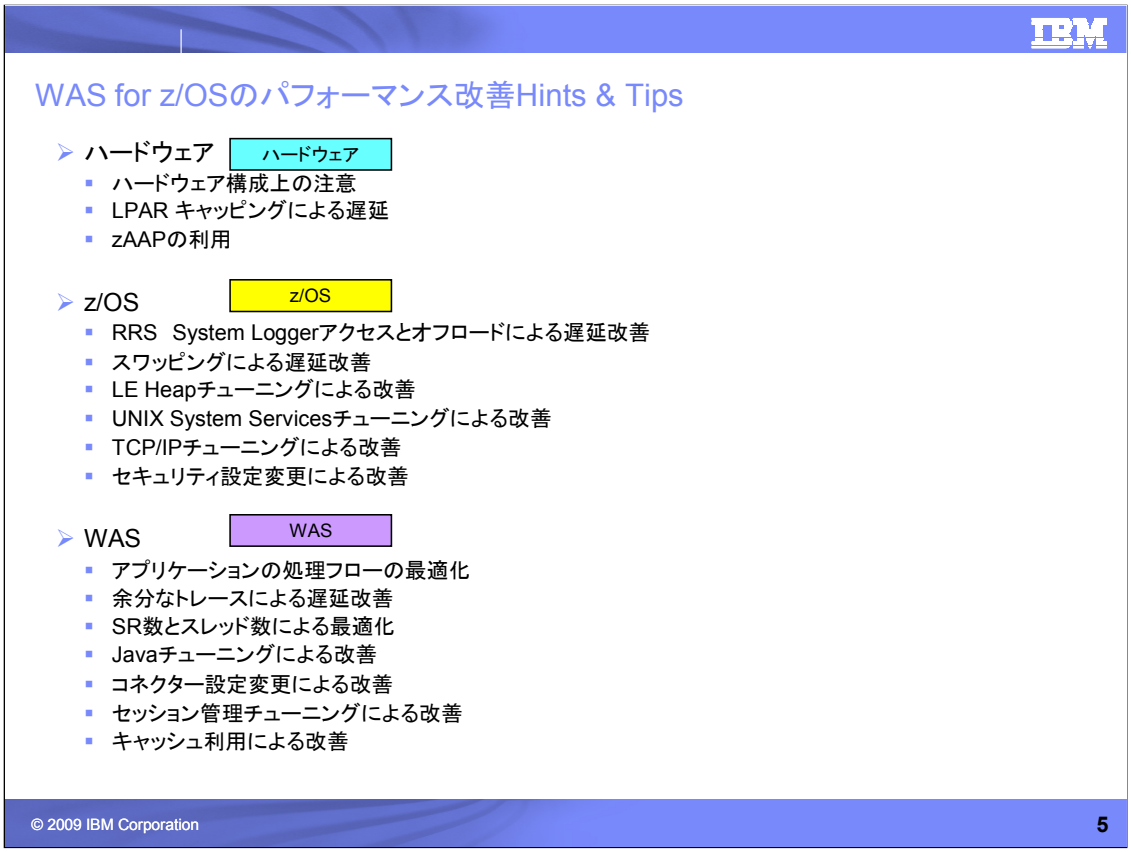
目次

- WAS for z/OS パフォーマンス改善Hints & Tips
- WAS for z/OS パフォーマンス・データ収集

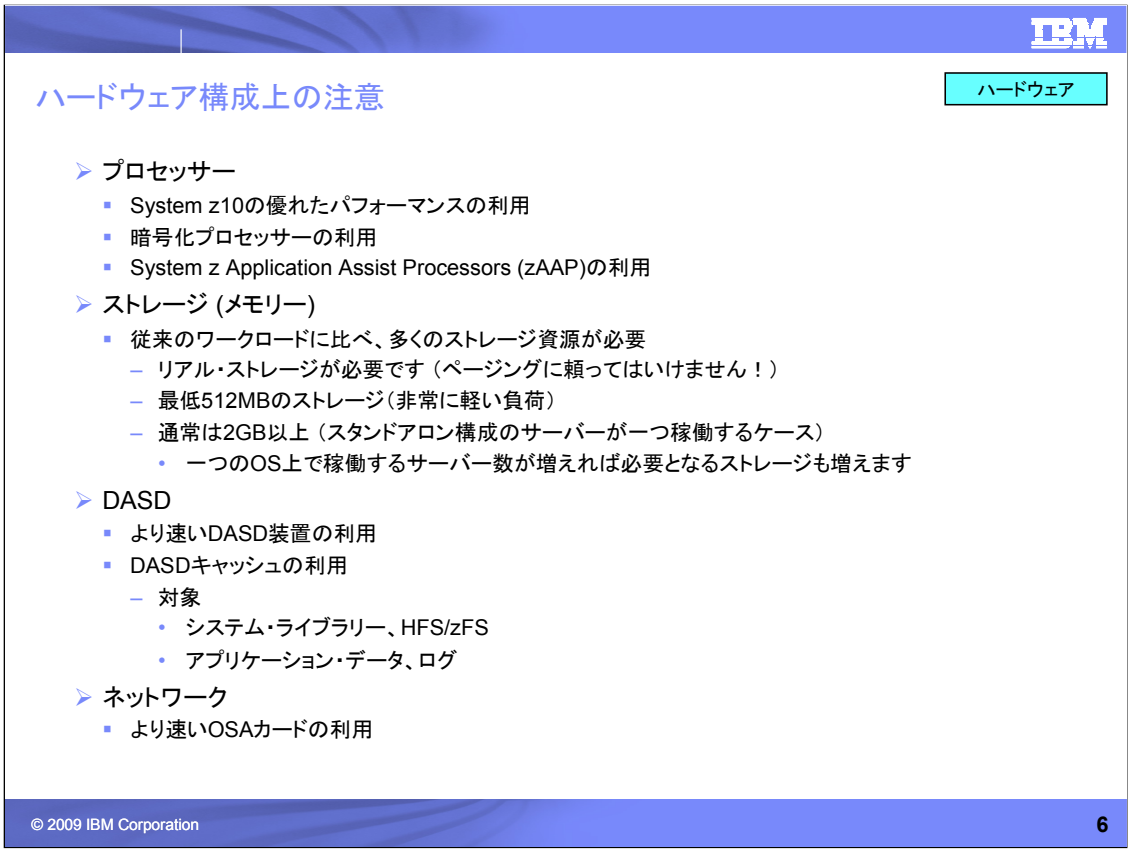
WAS for z/OS

パフォーマンス改善

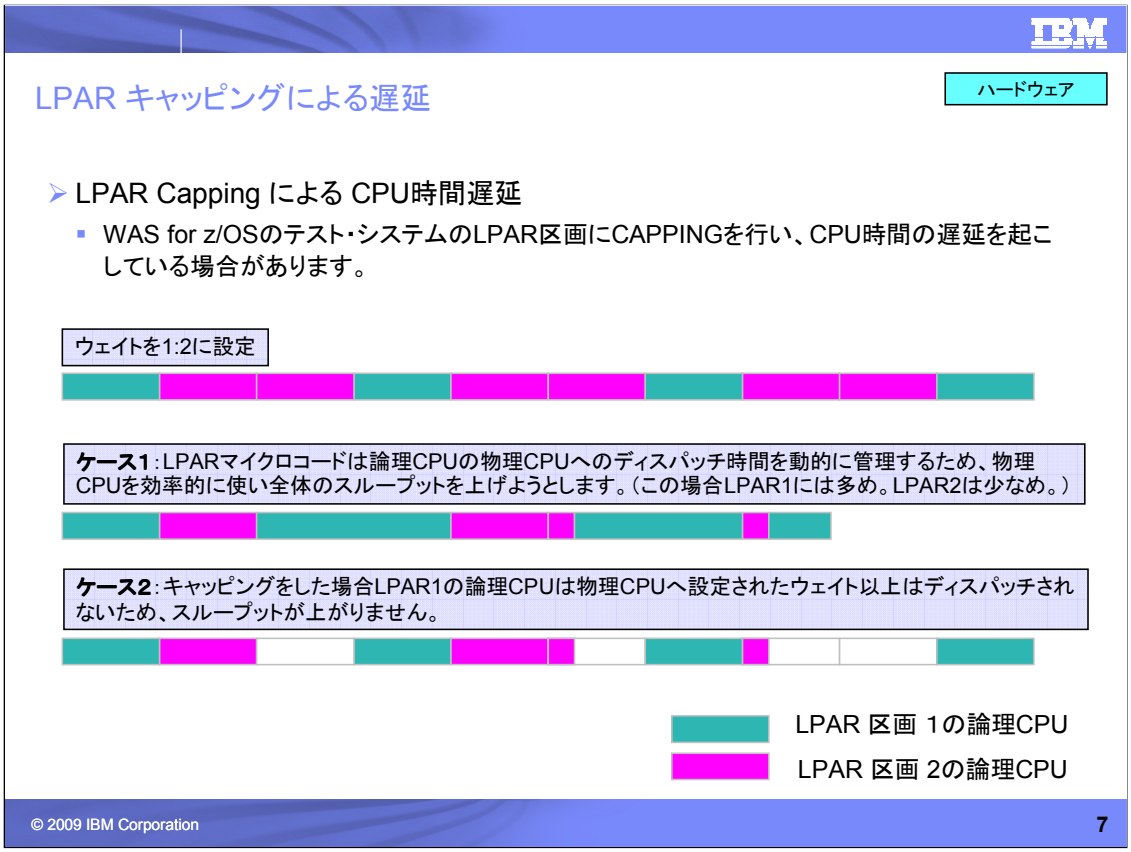
Hints & Tips



WAS for z/OSのパフォーマンス改善のためのHints & Tipsをご紹介します。



ハードウェア構成時には、プロセッサー、ストレージ(メモリー)、DASD、ネットワークで、それぞれ適切な選択が必要です。



ケース1で、LPARは元来ウェイト指定に基づいてCPU資源の適正割り振りを実施しており、設定されたウェイト分のCPU資源量は保証されております。

ケース2の状況が起きているかどうかの判断はRMFのCPUレポートで可能です。RMF CPU レポートにはLPAR BUSYという指標とMVS BUSYという指標がありますが、LPAR BUSYの値 >> MVS BUSYの値 となる状況は、LPAR CAPPINGによりCPU資源の仮想化が上手く機能できていない状況を現します。

System z Application Assist Processors (zAAP)

ハードウェア

- z/OSのJavaワークロードに特化した専用アシスト・プロセッサー
- Javaを使用するワークロードで使用される
  - WAS 5.1 / 6, CICS TS 2.3 / 3, IMS 8 / 9, DB2(Javaを使用したDBアクセス)
- z990、z890、z9、z10 および将来のSystem zで使用可能
  - 通常のCPUと比べ非常に経済的な値段
  - プロセッサー筐体中の通常プロセッサーと同じ数まで購入可能
- 前提
  - z/OS 1.6（or z/OS.e 1.6）以上
  - IBM SDK for z/OS, Java 2 Technology Edition, V1.4* with PTFs UQ88783, UQ90449...
    - JDK 1.4.2 SR3以降がお勧め
- JavaアプリケーションをzAAPで実行しても、既存のIBM S/W製品ライセンス料に影響しない。

バックエンドのデータベース環境とJavaベースのWebアプリケーション一体型の、ハイスピード・高信頼性・高可用性・ハイセキュリティなシステムをより低いTCOで提供する。

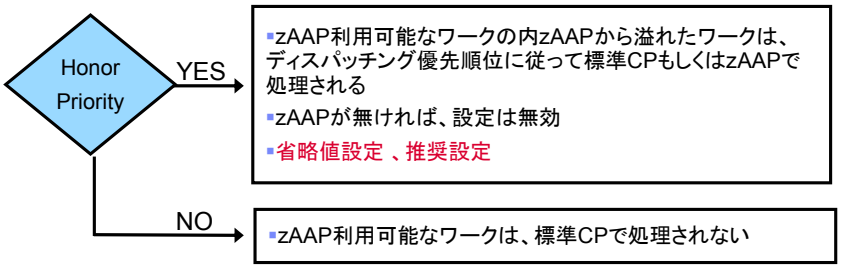
zAAPは、z/OSのJavaワークロードに特化した専用アシスト・プロセッサーです。

zAAP利用のための設定

ハードウェア

z/OS

- **PARMLIB(IEAOPTxx) のオプション (z/OS 1.9以降の場合)**
  - IFAHONORPRIORITY = YES | NO
    - 標準CPで、zAAPから溢れたzAAP利用可能なワークを処理するか否かを設定
  - ZAAPAWMT = n (n=1-499000 microseconds,デフォルト=12000 (12msec))
    - zAAPが他のプロセッサーのヘルプの必要性をチェックする頻度を設定
      - AWM : Alternate Wait Management
  - PROJECTCPU = YES | NO (デフォルト=NO)
    - 標準CPからzAAPやzIIPにオフロードできるワークのプロジェクションを行うか否かを設定
      - zAAPやzIIPが無い場合のみ有効



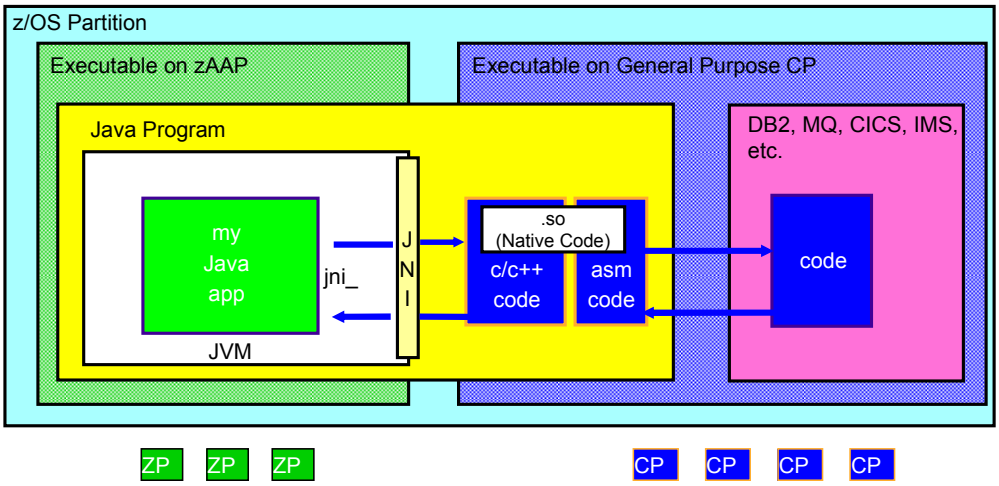
IFA： Integrated Facility for Application = zAAP

zAAP利用のための設定は、SYS1.PARMLIB(IEAOPTxx)に設定する必要があります。

zAAPが利用可能となるタイミング

ハードウェア

- WAS内で稼働するJavaコードはzAAPを使用して稼働
- JNIでラップされたネイティブ・コードはzAAPではなく標準CPで処理
- Java コードはJITされてもzAAPを使用することが可能



WAS内で稼働するJavaコードはzAAPを使用して稼働することが可能です。JNIでラップされたネイティブ・コードはzAAPではなく汎用CPUで処理されます。

Java コードはJITされてもzAAPを使用することが可能です。

RRSとSystem Logger

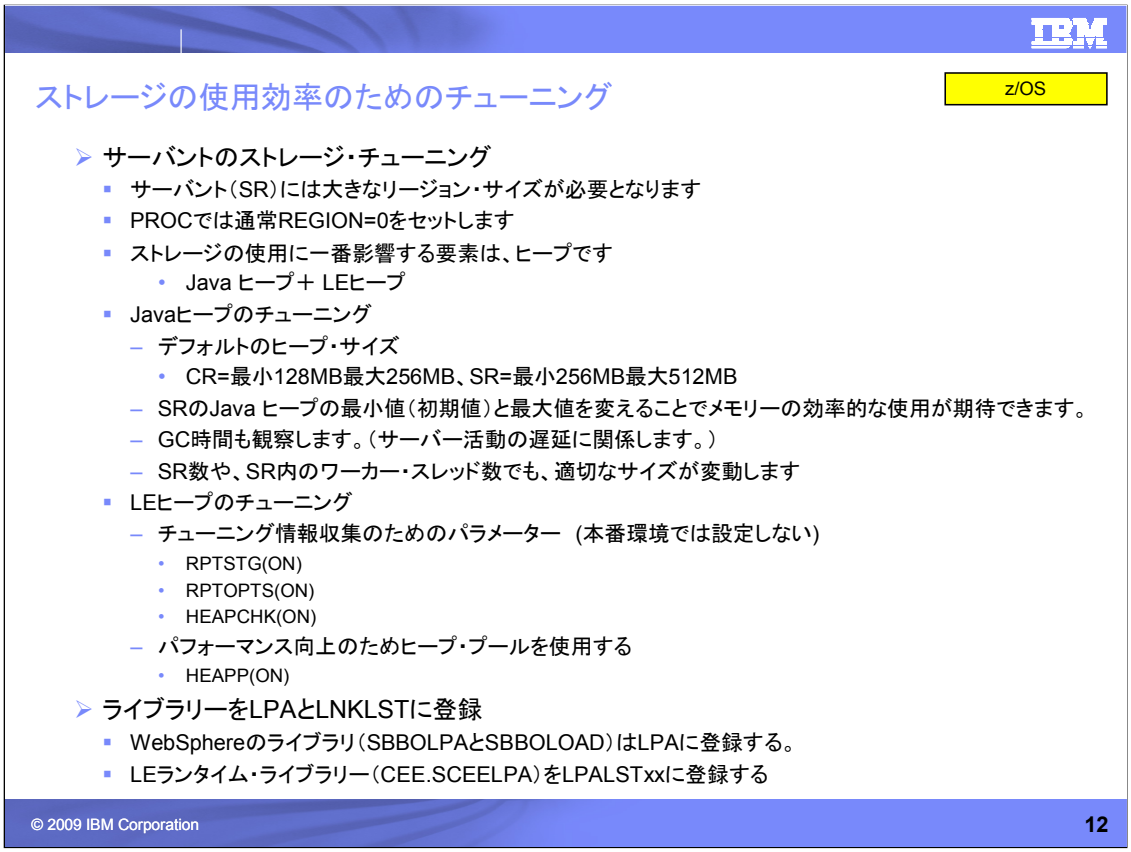
z/OS

- CF (Coupling Facility)がある場合は、CF Loggerを使用するのが基本
  - DASD Loggerを使用する場合は、できるだけ高速なDASDを使用し、他のデータセットとは混在させない
- ログストリームのサイズは十分大きく
  - ログストリーム・サイズが小さいと余分なオフロードが発生します
  - 大きめのサイズ設定がお勧めとなります
- Archive Logは必ずしも必要ではありません
  - システムが安定稼働をしてきたら、削除することによりスループットの向上が期待できます
  - z/OS z.10の機能拡張で、RRS Archive Logを動的にDisableにできるようになりました

```
SETRRS ARCHIVELOGGING,DISABLE
ATR174I RRS ARCHIVE LOGGING HAS BEEN DISABLED.
IXC579I NORMAL DEALLOCATION FOR STRUCTURE LOGR_RRSARCH IN
        COUPLING FACILITY  002097.IBM.02.00000004054F
                           PARTITION: 08    CPCID: 00
HAS BEEN COMPLETED.
PHYSICAL STRUCTURE VERSION: C451EBD0 6C31C38B
INFO116: 13572800 01 2800 0000000A
TRACE THREAD: 00003513.
```


RRS（Resource Recovery Service）はWAS for z/OSで稼働するJ2EEアプリケーションのトランザクション・ステータスを管理します。また、RRSはグローバル・トランザクションにおける2フェーズ・コミット処理のコーディネータ役でもあります。

RRSが管理するトランザクション・ログは物理的にはSystem Loggerにより管理されるログストリームに書き込まれます。



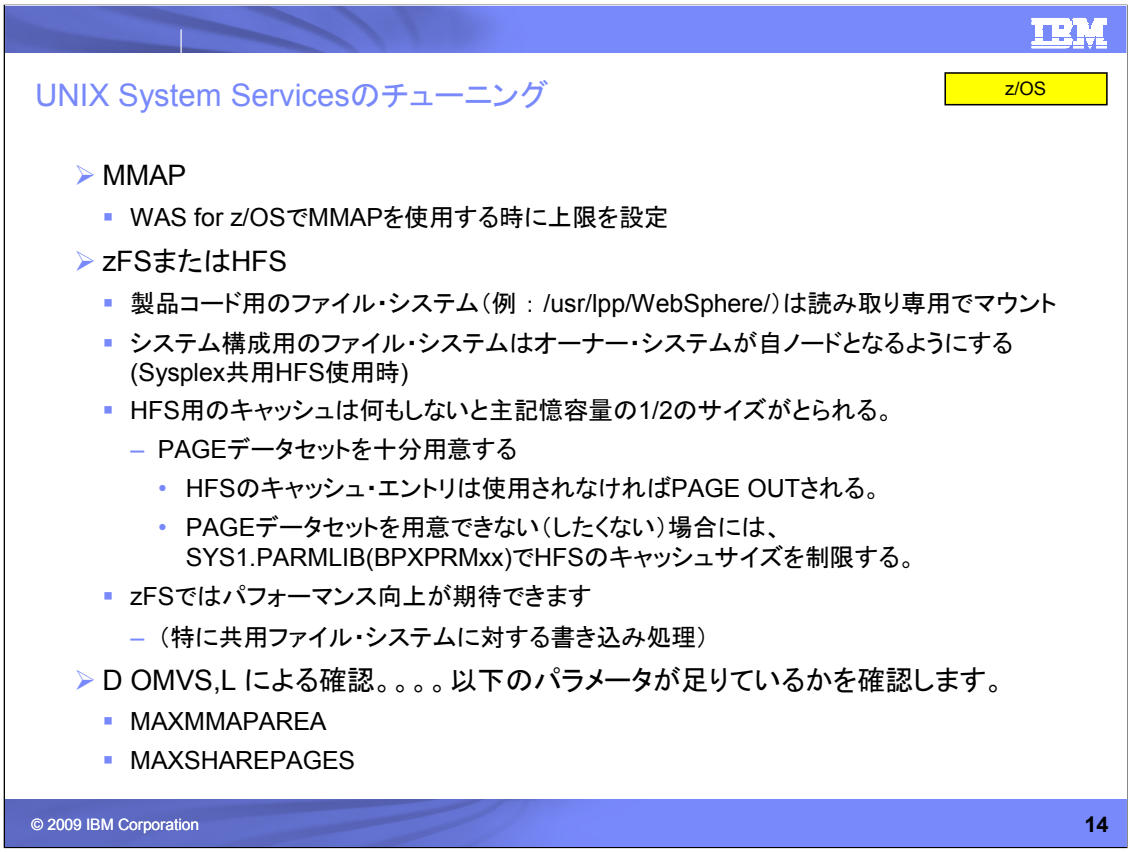
サーバントのストレージ・チューニングでは、Javaヒープのチューニングと LEヒープのチューニングが大切です。

ページング／スワッピングに関する方針

z/OS

- 良好なパフォーマンス維持の観点からは、WAS for z/OSのシステムはページングの無い（少ない）環境で運用するのが望ましい
- WAS（Java）はガベージ・コレクション（GC）により、JVMヒープというメモリー領域を掃除し、アプリケーションが使用できる領域を確保しながら稼動を続ける。という仕組みで動いているためGCとページングがぶつかる状況は好ましくありません
- 一番良いのはページング0（ゼロ）に近い状態で稼動させることであるが、少なくとも秒当たり二桁のページングが発生している状況では、資源の増強を含めキャパシティ計画を見直すべきです。
- WASはスワッピングの対象となるべきではない。
  - BBOSR、BBOCTL、BBODAEMN のWAS関連のモジュールはSYS1.PARMLIB(SCHEDxx)においてPPT登録をし、NON-SWAPPABLEにして下さい。

WAS for z/OSのシステムはページングの無い（少ない）環境で運用してください。



MMAPの設定

システム・デフォルトはBPXPRMxxのMAXMMAPAREAで指定、プロセス単位にはRACFユーザーID毎のOMVSセグメントでMMAPAREAMAXを指定します。

HFSキャッシュの制御。。。SYS1.PARMLIB(BPXPRMxx)の例

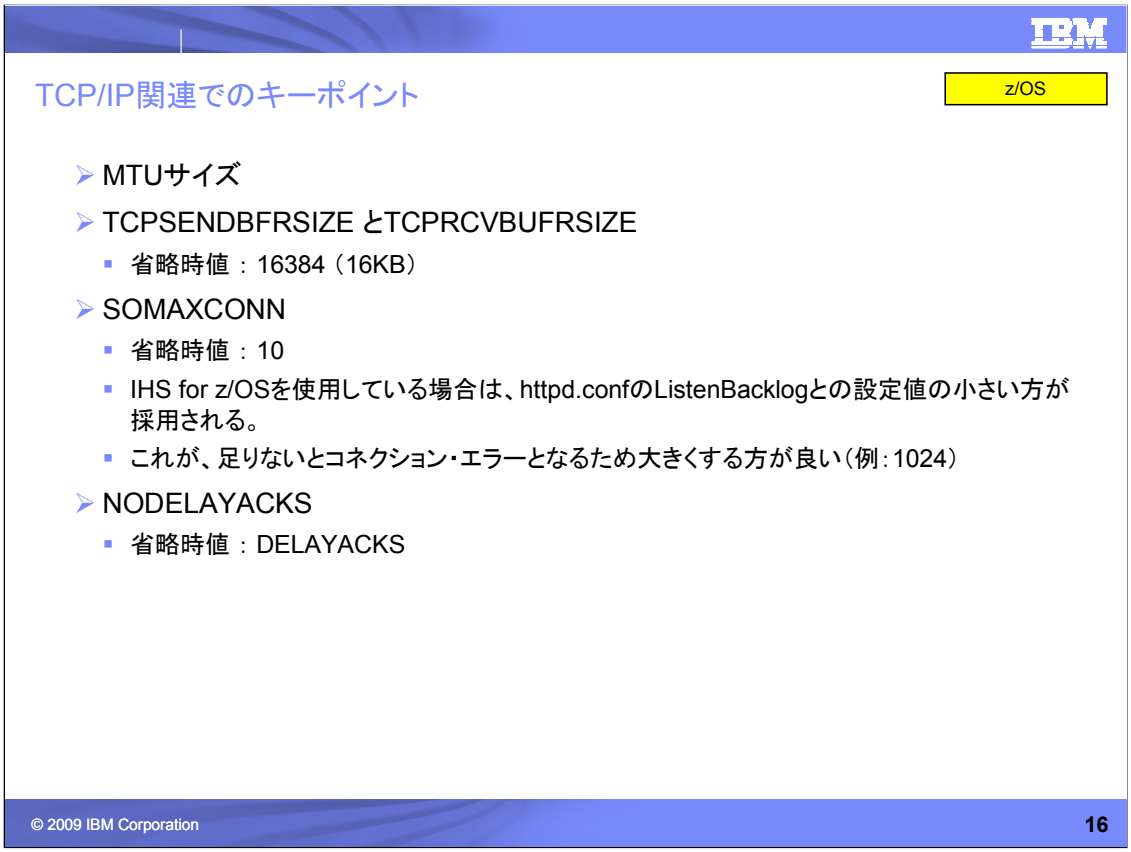
```
FILESYSTYPE TYPE(HFS)              /* Filesystem type HFS       */
         ENTRYPOINT(GFUAINIT)
         PARM('SYNCDEFAULT(60) FIXED(0) VIRTUAL(100)')   ← HFSキャッシュは100MBが上限となります。
```

UNIX System Servicesのチューニング

z/OS

- ソケット数の確認
  - BPXPRMxxの設定
    - MAXFILEPROCを十分大きな数にして下さい。
      - OMVSカーネルのストレージ使用に影響があります。
      - 全てのUSSユーザー・プロセスに対して有効となります。(RACFのOMVSセグメントで個々のユーザー毎の設定も可能)
    - AF_INETファイル・システムに関するMAXSOCKETSの値を十分大きくする。
      - 少なくともMAXFILEPROCと同じにする(OMVSカーネルのストレージ使用に影響はない。)

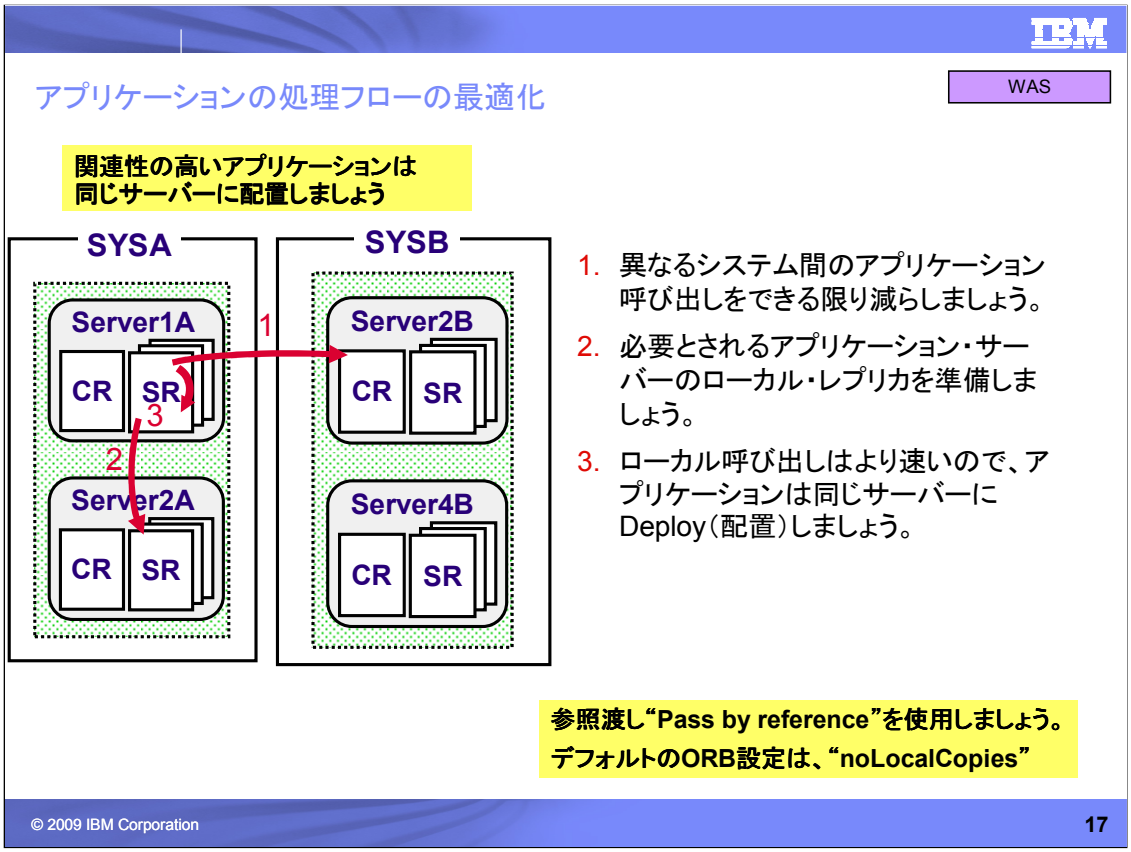
USSのチューニングでは、MAXFILEPROCを十分大きな数にして下さい。



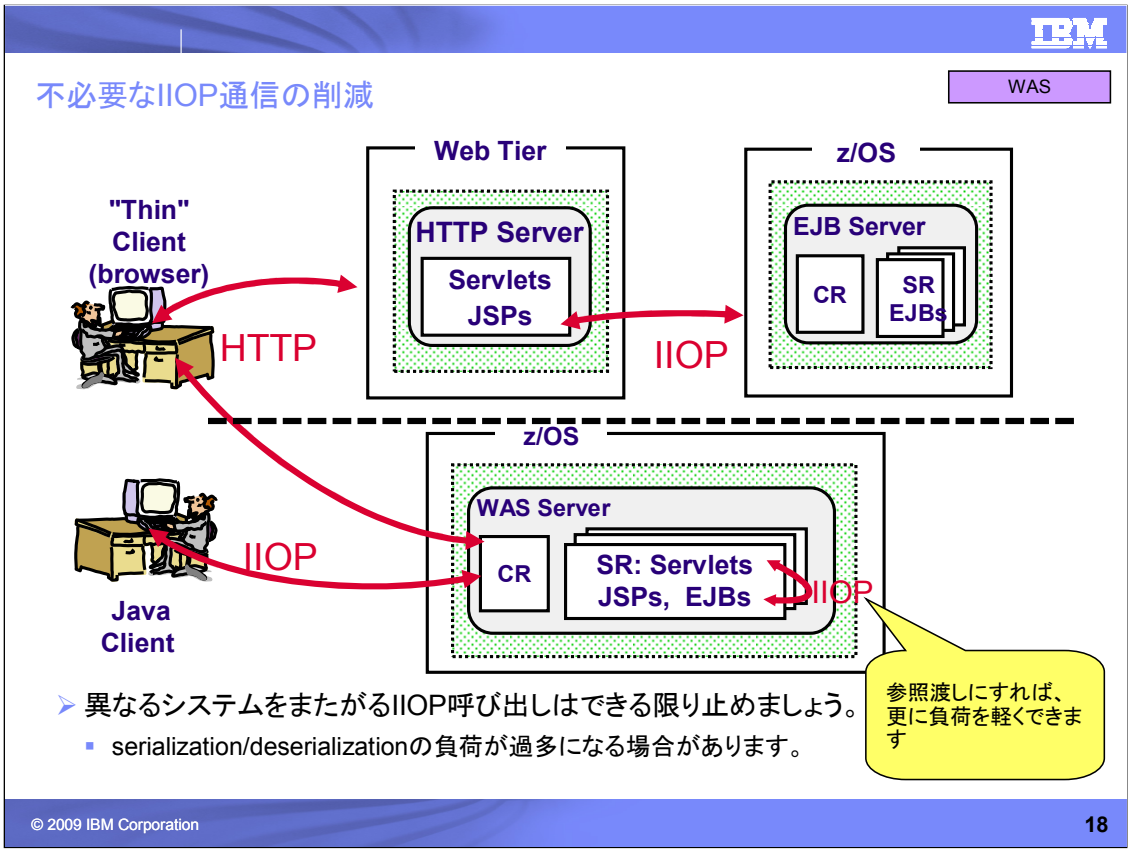
CPUが十分に使われている訳でもないのに、システムのスループットが上がらない場合にこちらであげたTCP/IPのパラメータをチェックすると良いことがあるかもしれません。

SOMAXCONN

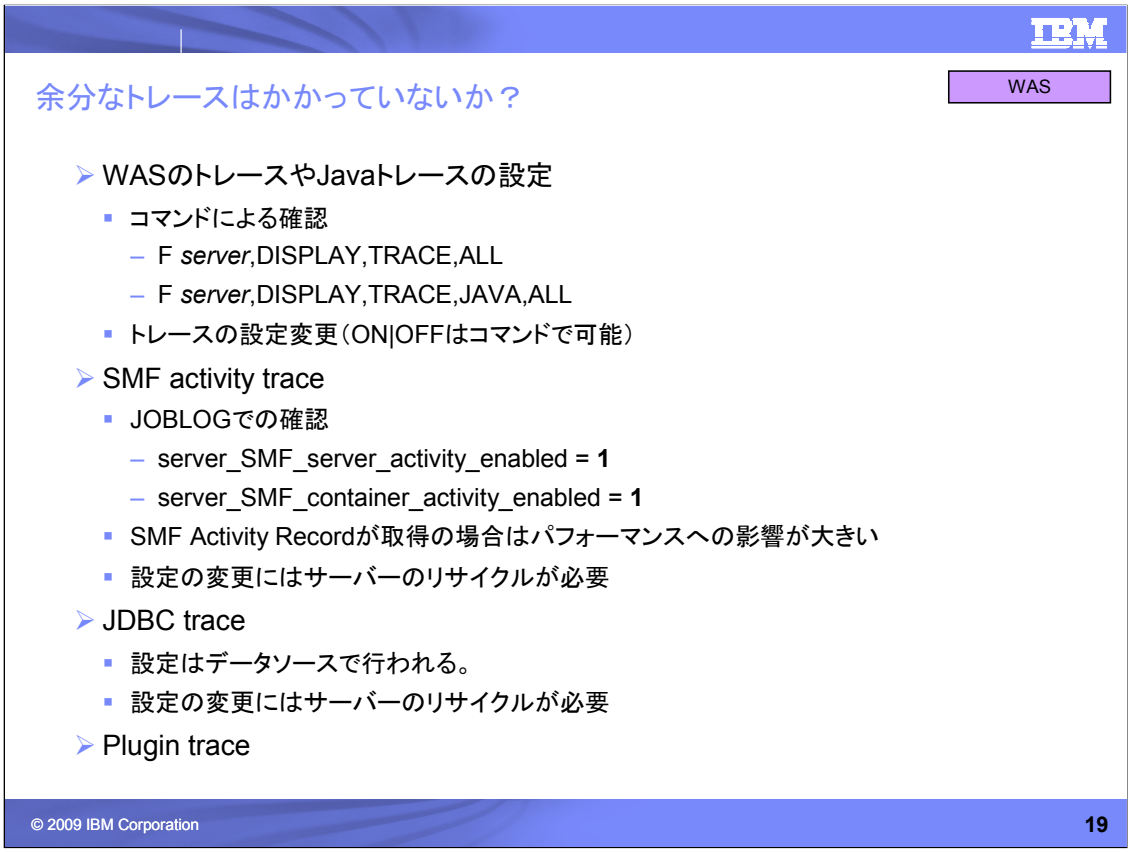
Listenしているソケットに対するコネクション・リクエストの最大数を指定します。



システムを跨ったり、サーバーを跨ると、オーバヘッドが大きくなります。連携するアプリケーションは同じサーバーで実行される環境を用意しましょう。



EJBにおいても、異なるシステムをまたがるIIOP呼び出しはできる限り止めましょう。

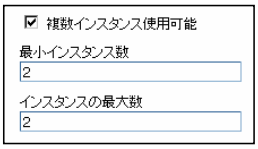


パフォーマンス問題が発生した際には、余計なトレースがかかっていないか？を、ご確認ください。

SR数とスレッド数による最適化

WAS

- SR数とSR内のワーカー・スレッド数による同時稼動数の最適化
  - バランスが大事
    - SRが多いとメモリーを多く消費します
    - 一つのJVMの中に多くのスレッドを作ると、コンテンションや過多なGCの原因となります
- SR数の変更
  - 管理コンソールにて変更
    - サーバー> サーバー・タイプ>WebSphere Application Server> server_name >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、最小数、最大数を設定する



- ワーカー・スレッド数の変更
  - 管理コンソールにて変更
    - サーバー> サーバー・タイプ>WebSphere Application Server> server_name > コンテナー・サービス >ORB サービス>z/OS の追加設定 (z/OS additional settings)
- 詳細は「WAS for z/OSのカスタマイズ基礎編」をご参照ください

スループットが上がらない場合、アプリケーションの同時稼動数が不足している場合があります。SR数とSR内のワーカースレッド数を調整して、適切な値を設定してください。

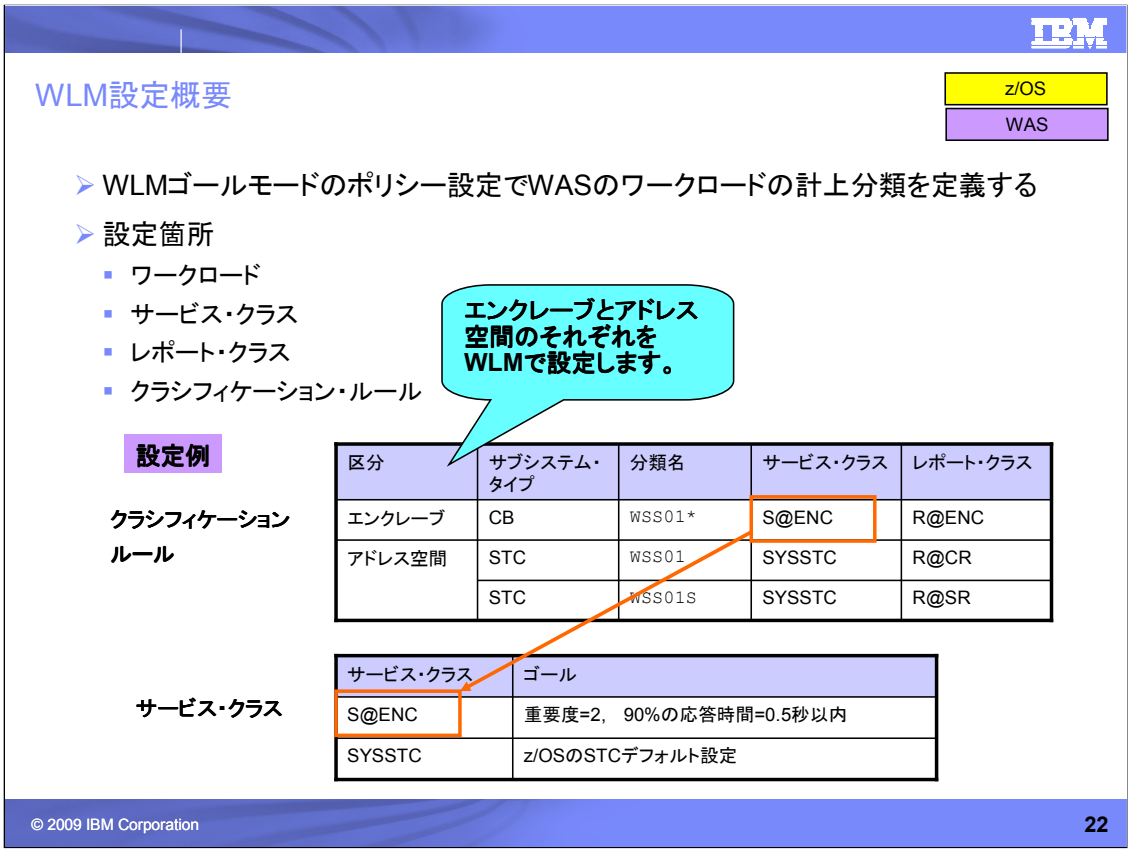
WLMでのワークロードのクラス割り振り

z/OS
WAS

- STC (Started Tasks)
  - CR, SR, CRAのSTCとしてのサービスクラス割り当てのため、STCクラシフィケーション・ルールに登録
- OMVS
  - applyPTF.shのサービス・クラス指定のため、CRのジョブ名をOMVSクラシフィケーション・ルールに登録
- **CB**
  - **アプリケーション (トランザクション)** のサービス・クラス指定
    - **HTTP**
    - **MDB**
    - **IIOP**
- リソース・マネージャー
  - DB2
  - CICS
  - IMS
  - MQ
  - Network QoS

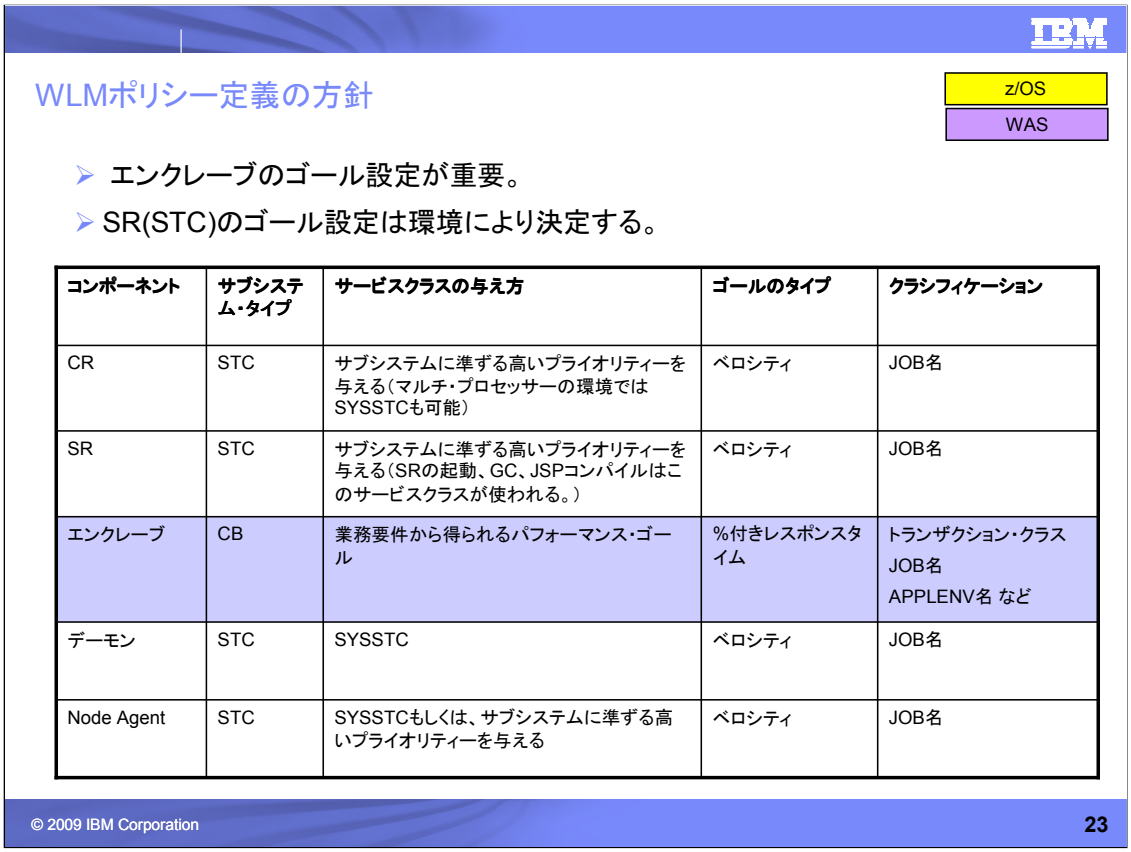
```
WLM Subsystem Type Selection List for Rules

                                                        Default
Action  Type      Description                           Service
  __    CB        CB Class'n w/WLM Trans. CLASSes       CBCLASS
  __    CICS      Use Modify to enter YOUR rules
  __    DB2       Use Modify to enter YOUR rules
  __    DDF       Use Modify to enter YOUR rules        DB_DDF
  __    IMS       Use Modify to enter YOUR rules
  __    IWEB      IWEB rules                            IWEBFAST
  __    JES       Batch Classification Rule             BAT_MED
  __    OMVS      E_Biz Classification Rule             EBIZ_DEF
  __    STC       Started Task Classification Rule      OPS_DEF
  __    TSO       TSO Classification Rule               TSO_DEF
```


WLMでのワークロードクラス割り振りでは、STC, OMVS, CB についてそれぞれ設定してください。



WLMゴールモードのポリシー設定でWASのワークロードの計上分類を定義してください。

表はWLM設定例です。



WASに関するWLMポリシー設定に関しては、ユーザー・アプリを処理する部分をエンクレーブとして処理します。

トランザクション・タイプのワークロードであれば%付きレスポンス・タイムで設定しますが、比較的1トランザクションの処理量が多いような場合（ショート・バッチ）には、ベロシティ・ゴールやマルチ・ピリオドのゴールも考えられます。

エンクレーブ以外のサービスクラスはサブシステム・タイプSTCですが、マルチ・プロセッサー環境であれば、できる限り高いサービスクラスを割り当てることが望ましいです（SR以外は基本的にSYSSTC）が、SR(STC)に関しては以下の考慮があります。

SRのサービスクラス高（SYSSTCなど）にする場合

マルチ・プロセッサーで資源に余裕のある環境でお勧め。

SRの起動、GC、JSPコンパイル等が高優先順位で処理されるため、RMF モニターIIIで、ほかのワークロードを遅延させないかの監視は必要です。

SRのサービスクラス低

単一CPUなど、比較的資源に余裕の無い環境でとる対策。
SRの起動に遅延が発生することが考えられ、余り長いとWLMタイム・アウトの調整の検討をして下さい。

Application Servers > *server* > **ORB Service > Advanced Settings**（デフォルト300秒）

他に、GCの遅延によるアプリ応答時間の影響も考慮してください。

WLMポリシー設定例

z/OS
WAS

コンポーネント（JOB名）	ワークロード	サブシステム・タイプ	サービスクラス名	重要度	ゴール (タイプ)	クラシフィケーション
CR	WAS_WKL	STC	STC_HI	1	70% （ベロシティ）	JOB名
SR	WAS_WKL	STC	STC_MED	2	40% （ベロシティ）	JOB名
エンクレーブ	WAS_APPL_WKL	CB	WAS_HI	2	80% 1.0秒 （%付きレスポンスタイム）	トランザクション・クラス
エンクレーブ	WAS_APPL_WKL	CB	WAS_MED	3	70% 2.5秒 （%付きレスポンスタイム）	トランザクション・クラス
DMN	WAS_WKL	STC	SYSSTC	N/A	N/A	JOB名
Node Agent	WAS_WKL	STC	SYSSTC	N/A	N/A	JOB名

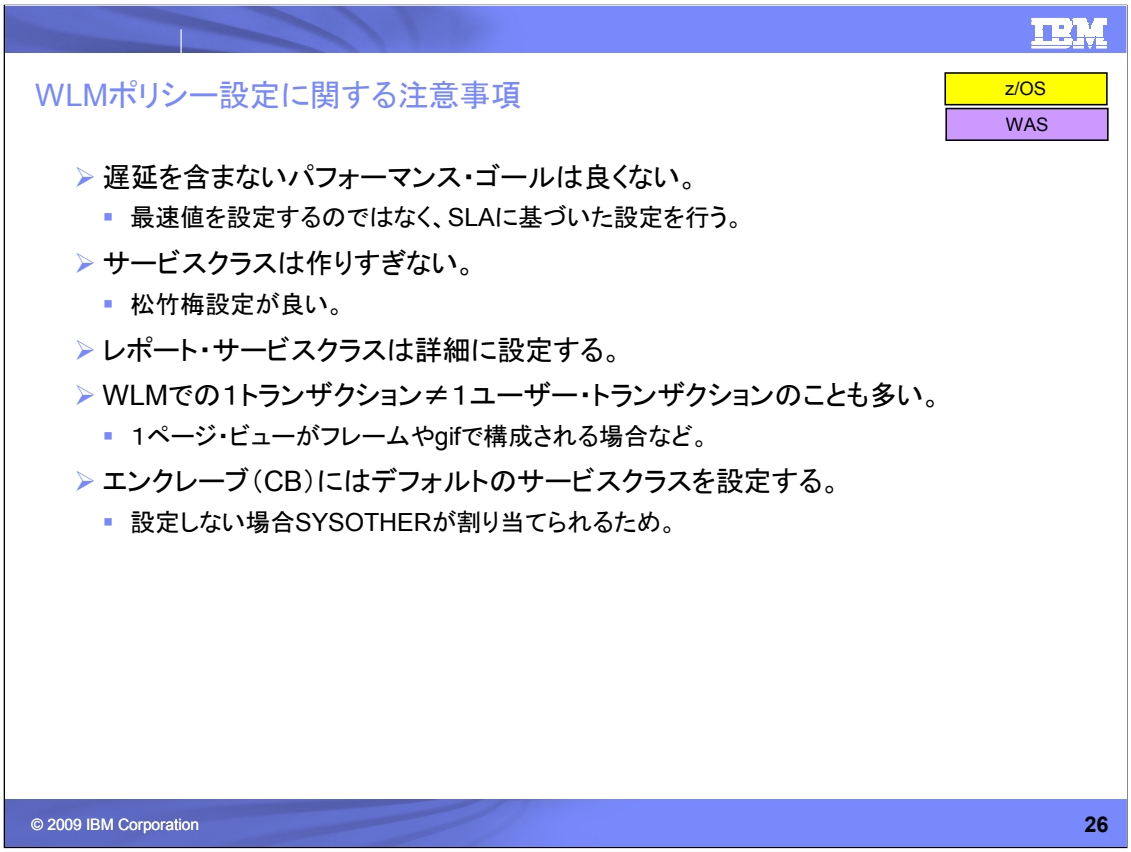
WASのWLMポリシー設定例です。
前ページの定義方針と合わせて参考にして下さい。

WLM: エンクレーブへのサービス・クラス割り振り

z/OS
WAS

- サブシステム・タイプ CBを使用します。以下の分類が使用可能です。
  - サーバー名 (**CN**)
  - サーバーインスタンス名 (**SI**)
  - トランザクションに割り振られたユーザーID (**UI**) – 余り使われない
  - **トランザクション・クラス** (**TC**)
    - TC はXMLファイルで設定します。(詳細は次ページ)
- %(パーセンテージ)付きレスポンスタイム・ゴールがお勧め
  - レスポンスタイム・ゴールはベロシティ・ゴールに比べ環境変化の影響を受け難く、ビジネス・ゴールを反映させ易い。
    - (例) 80%のトランザクションが0.5秒以内に完了
  - マルチ・ピリオドのゴールは設定しない。
  - デフォルトはSYSOTHER
- その他の考慮点
  - リクエストが既にエンクレーブ・トークンを持っている場合にはそのエンクレーブへのサービス・クラスが使用されます。

エンクレーブのWLM設定の説明です。



遅延を含まないパフォーマンス・ゴールというのは、例えば、そのトランザクションが遅延なしで0.2秒で応答できる時に、0.2秒のゴールを設定することを指します。WLMは全体最適を目指すメカニズムですので、こういったゴール設定では、最適化されたワークロード調整は望めません。

SLAやシステム・デザイン時の目標として、例えば3秒以内のレスポンス・タイムとしたのであれば、たとえ、最速0.2秒のトランザクションでも、N/Wのサービス時間を0.5秒引いて2.5秒というように設定してください。

サービスクラスの作りすぎは管理を複雑にし、WLMの処理負荷を増す原因となります。アクティブなサービスクラスの総数が30程度となるように計画して下さい。

SYSOTHERはシステムで予約されているサービスクラスです。ディスクリショナリー・ゴールとなるため、資源が配分されにくくなることがありえます。

エンクレーブ（CB）では、マルチ・ピリオドのサービスクラスを設定することも可能です。バッチ的な処理や、長短混合のトランザクション環境では検討してください。

より詳細な割り振り. . . WLMトランザクション・クラス

z/OS
WAS

- レポート・クラス、サービス・クラスが重要
  - トランザクション・クラスは、WLMレポート・クラス、サービス・クラスと関連付けることが可能
  - 実行優先度の観点
    - トランザクション・クラスごとに別サービス・クラスを割当てると、クラスごとにSRが起動しエンクレーブの優先度が設定される
      - サーバー・インスタンスを複数起動できるようにしておく
  - レポートの観点
    - レポート・クラスにより、細かいレベルのRMFワークロード・レポートを取得できる

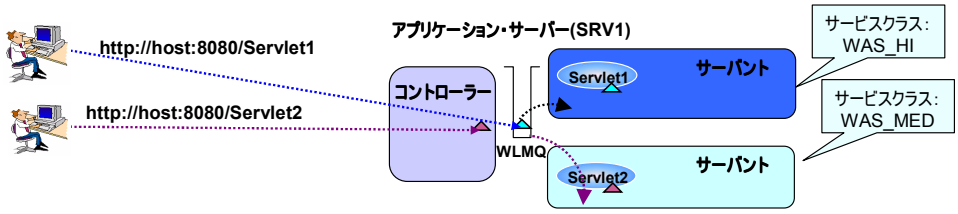
- 参考
  - サーバー・インスタンスの複数起動設定
    - 1. 管理コンソール：サーバー > アプリケーション・サーバー名 > サーバー・インフラストラクチャー > Javaおよびプロセス管理 > サーバー・インスタンス にて “複数インスタンス使用可能” にチェックし、インスタンスの最大数を2以上にする
    - 2. WAS for z/OS V7からサーバント数は動的に変更できるようになり、サーバント数の変更においてサーバーの再起動が不要になった
      - Modify WLM_MIN_MAX=min,maxコマンド
      - 注： wlm_dynapplenv_single_server=0 (“複数インスタンス使用可能”にチェック) で稼動している必要がある
      - コマンド使用例： F WAS7A,WLM_MIN_MAX=2,2

WLMトランザクション・クラスの活用例を2つ示します。
参考として、SRを複数起動するための設定を提示します。

より詳細な割り振り. . . WLMトランザクション・クラス

z/OS
WAS

- 実行優先度の観点
  - トランザクション・クラスごとに別サービス・クラスを割当てると、クラスごとにSRが起動しエンクレーブの優先度が設定される
  - 以下の設定例で、Servlet1とServlet2は別サーバントで実行される
  - サービス・クラスは、
    - Servlet1が実行されるサーバントはWAS_HIサービス・クラス
    - Servlet2が実行されるサーバントはWAS_MEDサービス・クラス　となる



wlm_classification_file (WAS側)

host:port	URI名	トランザクション・クラス名
host:8080	/Servlet1/*	TCLAS1
host:8080	/Setvlet2/*	TCLAS2

ワークロード定義(WLM側)

LEVEL	TYPE	NAME	Service Class	Report Class
1	CN	SRV1	WAS_MED	R@ENC
2	TC	TCLAS1	WAS_HI	R@ENCT1
2	TC	TCLAS2	WAS_MED	R@ENCT2

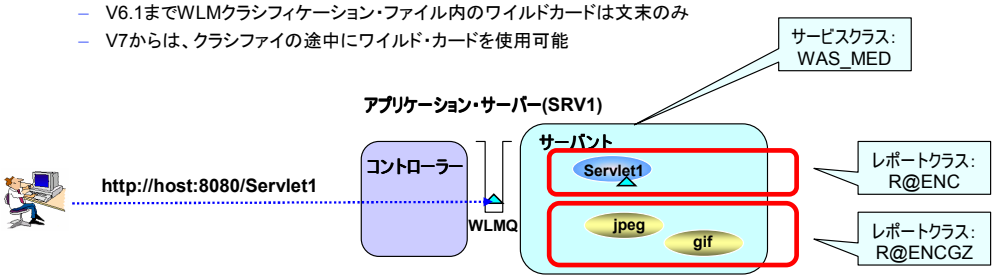
WLMトランザクション・クラスの活用例を1つめ、「実行優先度の観点」での利用です。

より詳細な割り振り. . . WLMトランザクション・クラス

z/OS
WAS

- レポートの観点
  - レポート・クラスにより、細かいレベルのRMFワークロード・レポートを取得できる
    - 同じサーバー上の処理を、アプリケーション毎に集計できる
    - Servlet処理と、画像処理のワークロードを別々のレポート・クラスに設定できる

➢ 参考
- WLMクラシフィケーション・ファイルの記載方法がV7で改善された
  - V6.1までWLMクラシフィケーション・ファイル内のワイルドカードは文末のみ
  - V7からは、クラシファイの途中にワイルド・カードを使用可能



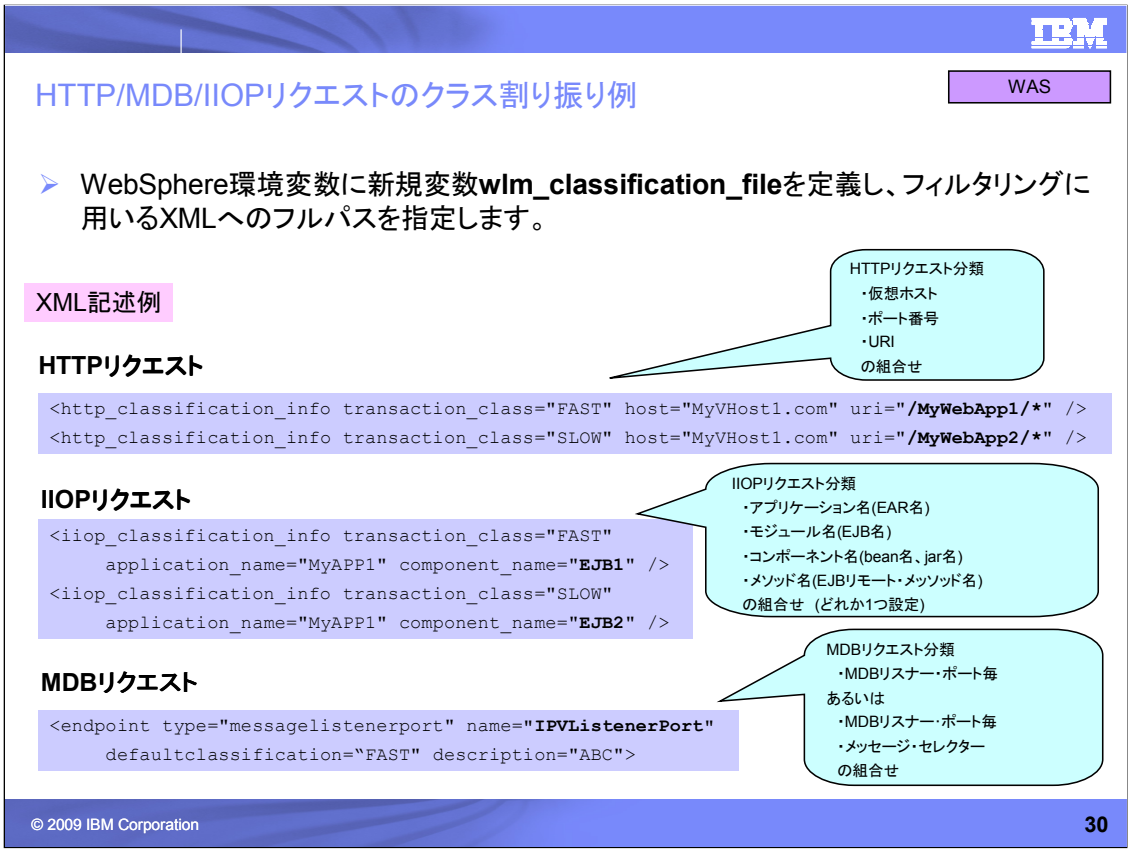
wlm_classification_file (WAS側)

host:port	URI名	トランザクション・クラス名
host:8080	/Servlet1/*.jpeg	TCLAS1
host:8080	/Setvlet1/*.gif	TCLAS2

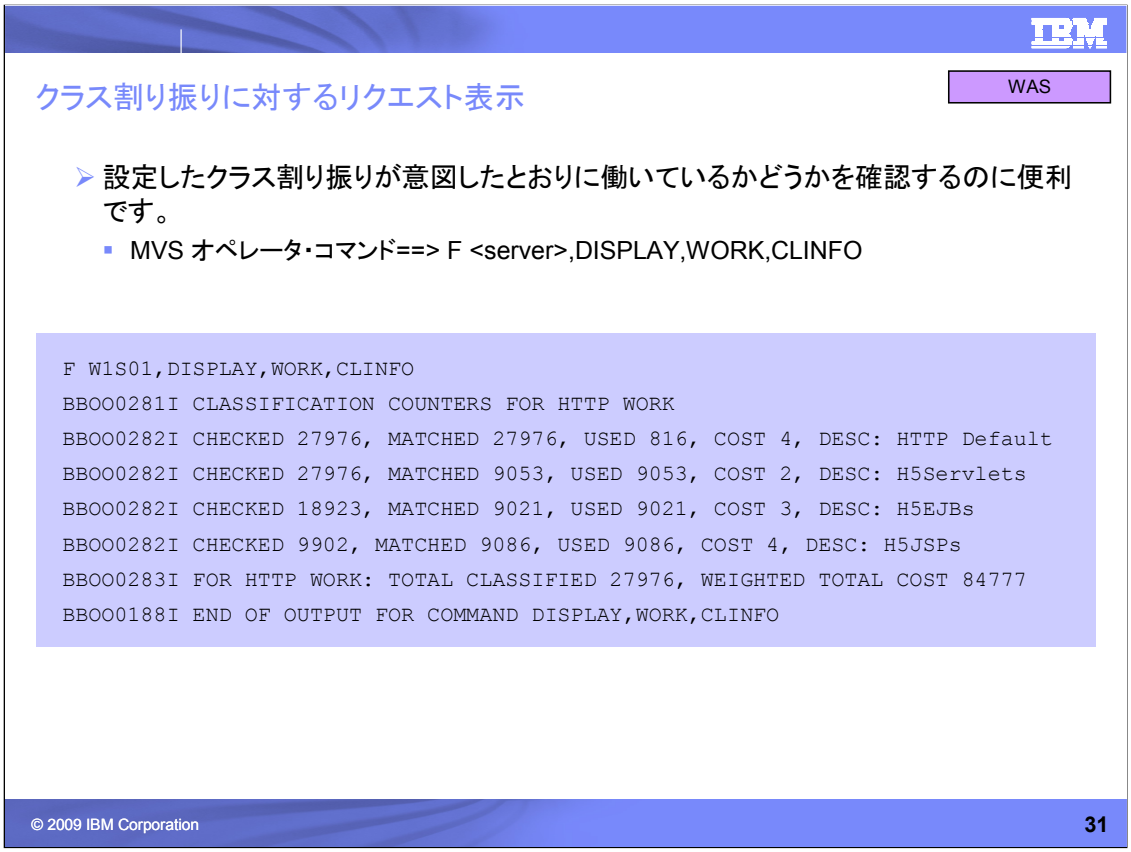
ワークロード定義(WLM側)

LEVEL	TYPE	NAME	Service Class	**Report Class**
1	CN	SRV1	WAS_MED	**R@ENC**
2	TC	TCLAS1	WAS_MED	**R@ENCGZ**
2	TC	TCLAS2	WAS_MED	**R@ENCGZ**

WLMトランザクション・クラスの活用例を2つめ、「レポートの観点」での利用です。



wlm_classification_fileに指定するXMLファイルの記述例です。



設定したクラス割り振りが意図したとおりに働いているかどうかを確認する方法です。

Javaチューニング

WAS

- JavaのレベルはSRのジョブログの中でレポートされます
  - SDKのレベルは最新の状態を保つようにしましょう
  - ジョブログの中でJITの有効｜無効もレポートされます
- JITが有効になっていることの確認
  - サーバー各々でDisable JITが選択されていないことを確認します
- アプリケーション・リロード間隔
  - class reloadingを止める
    - “Reload Enabled” のチェックを外す。
    - ReloadがEnabledになっているアプリケーションでは、applications -> enterprise applications -> <application name> で “Reload Interval” をゼロにセット。
    - Reloadを止めることができない場合は“Reload Interval” の値を大きくする。

Javaチューニング項目です。

Javaチューニング

WAS

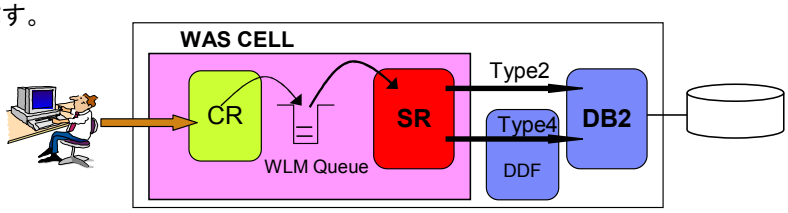
- GCから判断するheapサイズ・チューニング
  - verboseGC出力確認 (SYSOUTへ出力)
  - 確認ポイント
    - GC後のフリーメモリーサイズ
      - フリーメモリーの推移を見ることで、メモリーリークが起きていないことを確認する
    - GC後のフリーメモリーサイズとトータルメモリーサイズ
      - Java heapサイズを決めるための基礎数値となる
    - GC間隔と、GCにかかった時間
      - Java heapサイズが適切な値であることを確認する
- GCコレクター・ポリシーの変更
  - 必要に応じて、GCコレクター・ポリシーを変更を検討
    - -Xgcpolicy:optthrougput (default)
    - -Xgcpolicy:optavgpause
    - -Xgcpolicy:gencon
    - -Xgcpolicy:subpool
- Compressed references や Large Pages の利用の検討

Javaチューニング項目(続き)です。

コネクター・パフォーマンス

WAS

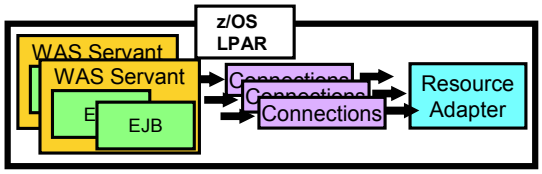
- 一般論
  - リモートではなくローカル・コネクションを使用しましょう。
    - N/W遅延が無くなります。
    - 必要なCPU資源が少なく済みます。
  - コネクターとリソース・アダプター間のキューイングの仕組み
    - Pooled connection
- DB2: JDBC Type2 vs. Type4
  - DB2 for z/OSとWAS for z/OSを同じOSイメージ上に配置する構成がベストで、且つこの場合Universal JDBC DriverでのType2接続がお勧め。
  - 一般的にSQLJ(静的SQL)はJDBC(動的SQL)に比べ処理効率が良い。
    - JDBCの場合は動的ステートメント・キャッシュを利用するとパフォーマンスは大幅に改善します。



DBと接続する際の注意点です。

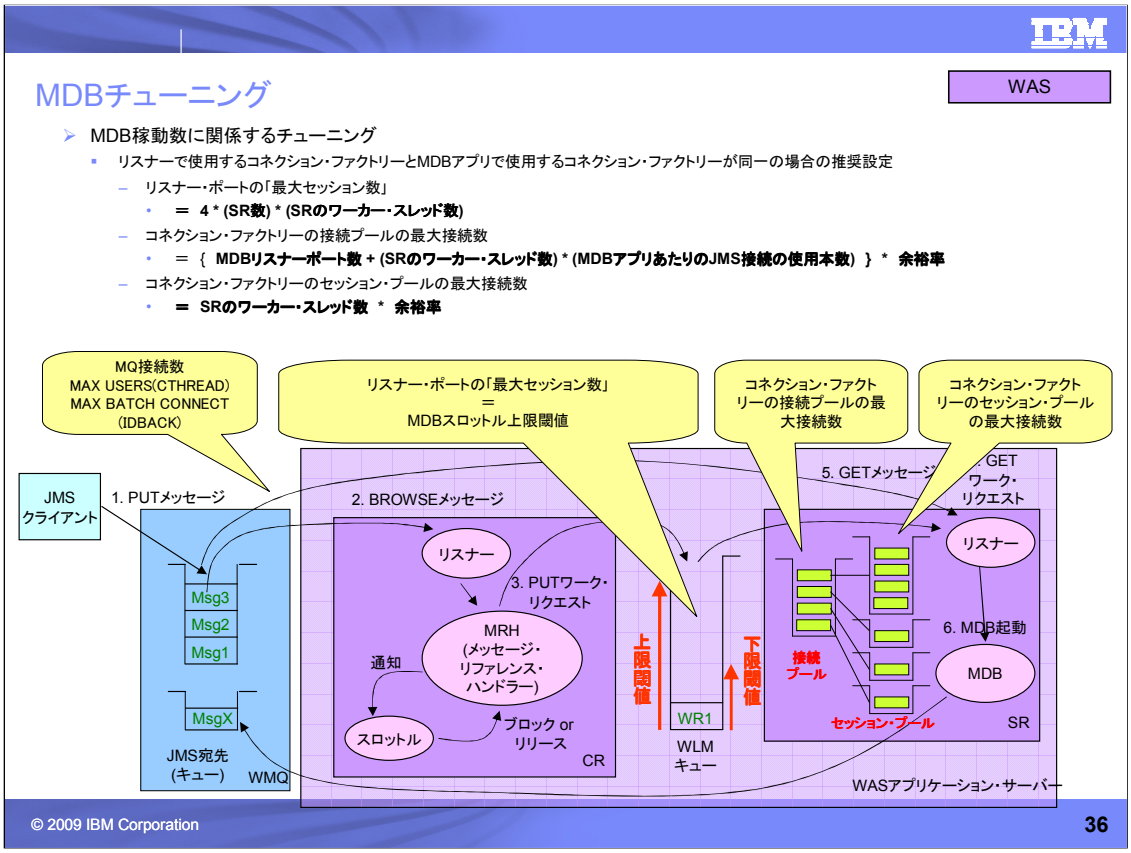
コネクター・パフォーマンス

- CICS：CICS Transaction GatewayによるEXCIの利用
  - パイプとスレッド数の監視と制御
- IMS：ローカル接続 vs. MSC vs. リモートIMSコネクト
- MQ：バインディング・モード vs. クライアント・モード



詳細は“"WebSphere for z/OS Connectivity Architectural Choices" RedBook SG24-6365 ”

CICS、IMS、MQと接続において、パフォーマンスに関する確認項目です。
詳細は、提示のRedbookをご参照ください。



MDB稼動数に関係するチューニングにポイントについての図です。

MDB チューニング

WAS

- MDB稼動数に関係するチューニングについて説明します。
- リスナーで使用するコネクション・ファクトリーとMDBアプリで使用するコネクション・ファクトリーが同一の場合の推奨設定は以下になります。
  - 設定値:リスナー・ポートの「最大セッション数」
    - 推奨: **4 * (SR数) * (SRのワーカー・スレッド数)**
    - 最小設定: 2 * (SR数) * (SRのワーカー・スレッド数)
    - 理由:
      - 例えば、SR数=2、SRワーカー・スレッド数=18 とすると、 最低限の式で計算すると 最大セッション数 = 2 * 2 * 18 =72 となります。スロットルメカニズムの関係で、最大セッション数の半分より小さい整数値が、low threshold となります。最大セッション数=72の時は、low threshold=36となります。一旦、MDBスロットルの上限閾値に達すると、合計で36スレッドありながら、WLMキューのワーク・リクエスト数が35以下にならなければ、新たなリクエストは入らないことになります。遊ぶスレッドを作らないためには、low threshold > SRワーカー・スレッドs の設定が必要です。Information Centerには、2*(WT+N)という式が提示されています。WT=全ワーカー・スレッド、N=バックログ、を意味しています。バックログの最大は、全ワーカー・スレッドなので、2*(WT+N)=4*WTと置き換えられます。従って、上記の推奨式になります。
  - 設定値:コネクション・ファクトリーの接続プールの最大接続数
    - 推奨: { **MDBリスナーポート数 + (SRのワーカー・スレッド数) * (MDBアプリあたりのJMS接続の使用本数)** } * **余裕率**
    - 理由:
      - MDBリスナー・ポートの数と同じMQ接続数が必要で、加えて、MDBアプリケーションあたりのJMS接続の使用本数が必要です。プール戻されるタイムラグを加味して、1-2割の余裕を持たせてください。
  - 設定値:コネクション・ファクトリーのセッション・プールの最大接続数
    - 推奨:**SRのワーカー・スレッド数 * 余裕率**
    - 理由:
      - ひとつのSRのワーカー・スレッド数と同じ数に設定するのが推奨です。MDBのトリガー・メッセージをGETするには、ひとつのコネクションに複数のセッションが作成されます。そのため、ひとつのSRのワーカー・スレッド数と同じ数のセッションが必要です。プール戻されるタイムラグを加味して、1-2割の余裕を持たせてください。

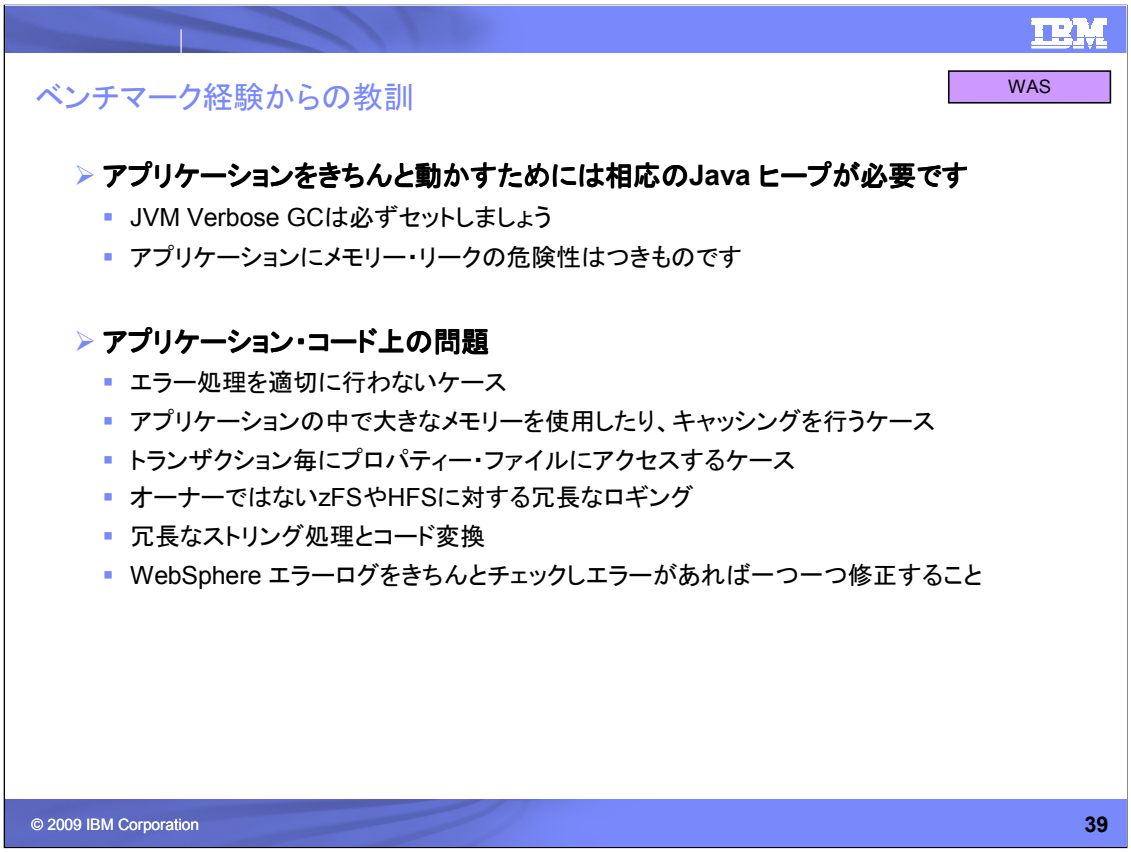
MDB稼動数に関係するチューニングについて説明します。

セッション管理のチューニング・トピック

WAS

- セッション終了時にはjavax.servlet.http.HttpSession.invalidate() で、HTTPセッションを開放する。
- 各サーブレット・JSPの外でのHTTPセッション・オブジェクトの保存・再利用をできるだけしない。
- HTTPセッションに新しいオブジェクトを置く場合はjava.io.Serializableインターフェースをインプリメントする。
- HTTPセッション・オブジェクトに大きなオブジェクトを置かない（ルール作りが必要）
- キャッシュ・ヒットを高めるために、セッション・アフィニティを活用する。
- メモリー間&DBセッション・レプリケーションを計画する場合は、チューニング項目をきちんとチェックする。
- J2Cコネクションには、connectionFactory キャッシングを利用する。

セッション管理のチューニング・トピックです。



ベンチマーク経験からの教訓です。

WAS for z/OS

パフォーマンス・データ収集

パフォーマンス・データ収集手段

	RMF (*1)	TPV (*2)	SMF(*3) Type120
概要	z/OS上で稼動するサブシステムすべてに共通の資源モニター	分散系WASと共通のツールで、WASから吐き出されるPMIデータを利用する	WAS用のSMFレコードからレポート作成
データの粒度	**資源（CPU、Diskなど）の実使用量**・レスポンスタイムなど（アドレススペース単位など）	WAS上で稼動するアプリケーション単位での**レスポンスタイム**	WAS上で稼動するアプリケーション単位の**細かい粒度でのCPU使用時間**
データ取得のオーバーヘッド	Monitor I,II ：低 Monitor III ：低～中	低～中 デフォルトでPMI有効。	Interval：低～中 Activity：高 Request: 低
使用目的	統計情報・リアルタイム	リアルタイム	統計情報
使い方	手軽。ただしz/OSの使用経験による	手軽だが、管理コンソールでの操作が必須	2段階のデータ加工が必須

※1 Resource Measurement Facility. RMFはSMF70-79のレコードを使用している。
※2 Tivoli Performance Viewer
※3 System Management Facilities

WAS for z/OSにおける主なパフォーマンスのデータ収集手段は、RMF, TPV, SMFです。

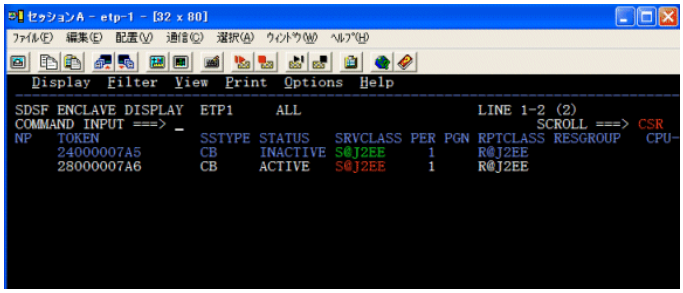
RMFモニターⅠを使う前に・・・WLMを設定する

- RMF使用の如何に拘らず、WLM（Workload Manager）の設定は必須です。
- WASのエンクレーブ用のサブシステムタイプCB（*)が用意されています。

(*) CBはWASの原型である、Component Brokerに由来

  - エンクレーブとは？
    - サーバー（CRやSR）のアドレス空間から独立した優先度がアサインされる
    - 複数のアドレス空間にまたがるトランザクションをエンクレーブという単位で制御し優先度をひとまとめに設定する
      - 例：WAS上のサーブレットがJDBC Type2でDB2に要求を出す場合、DB2のCPU時間がWASのエンクレーブにincludeされる

- RMF Monitor Iで、ワークロードを別々に集計するために、WLMレポート・クラスの設定をしてください。



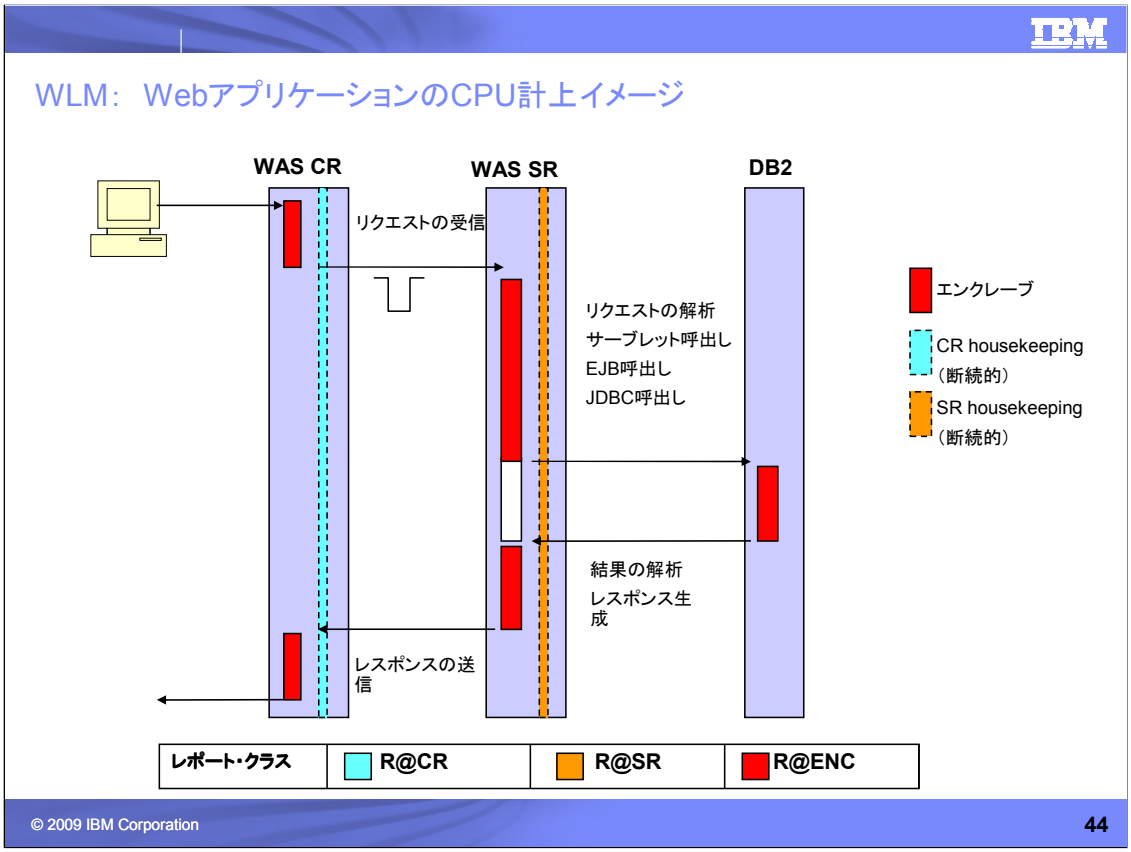
エンクレーブはSDSFのENCメニューで確認することができます。

RMFで細かくデータ収集するためには、適切なWLM設定が必要です。

WLM: CPU時間の計上箇所

- エンクレーブ
  - J2EEアプリケーションのリクエストはエンクレーブで処理される
    - JDBCやJNIコールも含む
      - DB2 Type 2 driver - DB2 CPU 時間はCBエンクレーブに計上される
      - DB2 Type 4 driver - DB2 CPU 時間はDDFエンクレーブに計上される
- STC（アドレス空間）
  - WAS CR (control region)
    - 通信の終端点：IIOP/HTTP/MDBリクエストを受け取る
    - IIOPリクエストに関するセキュリティ・チェック
    - リクエストのWLMクラス割り振りとWLMキューへのキューイング
  - WAS SR (servant region)
    - WLMキューからのサービス・クラスにあったリクエストの選択
    - ガベージ・コレクション
    - （アプリケーションによるスレッド作成）

エンクレーブに計上されるCPUと、STCに計上されるCPUの説明です。



エンクレーブに計上されるCPUと、STCに計上されるCPUのイメージ図です。

RMFデータ取得 - RMFモニター I

- バッチ処理(ジョブでレポートを生成)型モニタリング
- RMFデータの抽出・フォーマット

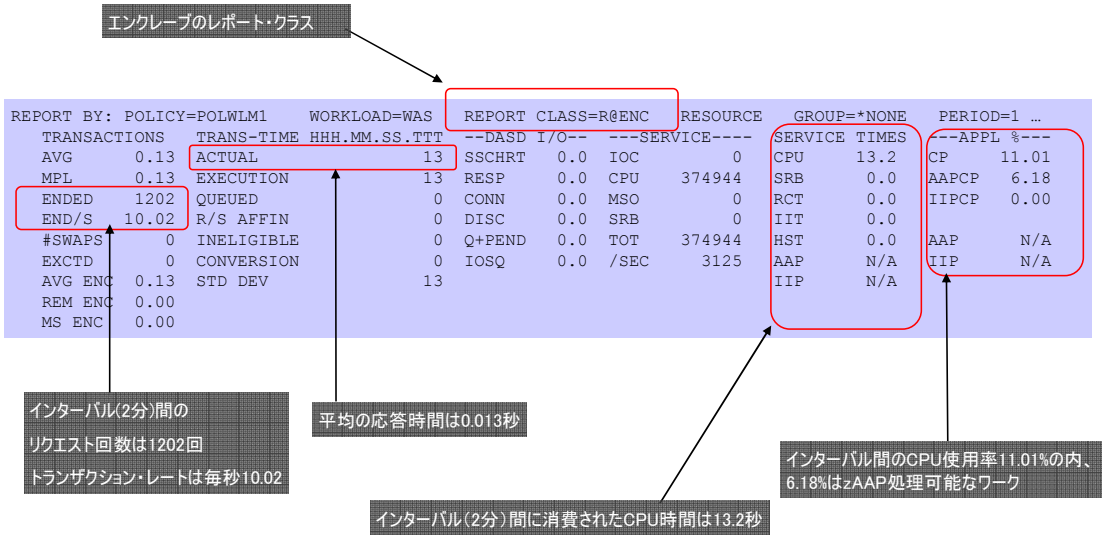
```
//RMFPOST  JOB MSGCLASS=X,PERFORM=3,NOTIFY=&SYSUID
//ST01     EXEC PGM=ERBRMFPP,REGION=0M
//MFPMSGDS DD   SYSOUT=*
//MFPINPUT  DD DISP=SHR,DSN=MK.SMF.DUMP
//SYSOUT   DD   SYSOUT=*
//SYSIN    DD   *
  DATE(05192006,05192006)      /* (MMDDYYYY,MMDDYYYY) DAY WINDOW    */
  RTOD(1900,1905)
  DINTV(0005)                  /* (HHMM) REPORTING INTERVAL         */
  REPORTS(CPU)
  SYSRPTS(
    WLMGL(POLICY,
     RCLASS(R@ENC,R@CR,R@SR),
     SCPER(S@ENC)
     ))
  SYSOUT(X)
/*
```

WLM設定概要で設定したレポート・クラスについて出力する

RMFモニター I のPOST処理(レポート出力)のJCL例です。

RMF モニター I ワークロード・アクティビティ・レポート

- エンクレーブのレポート・クラス - Report Class = R@ENC の例



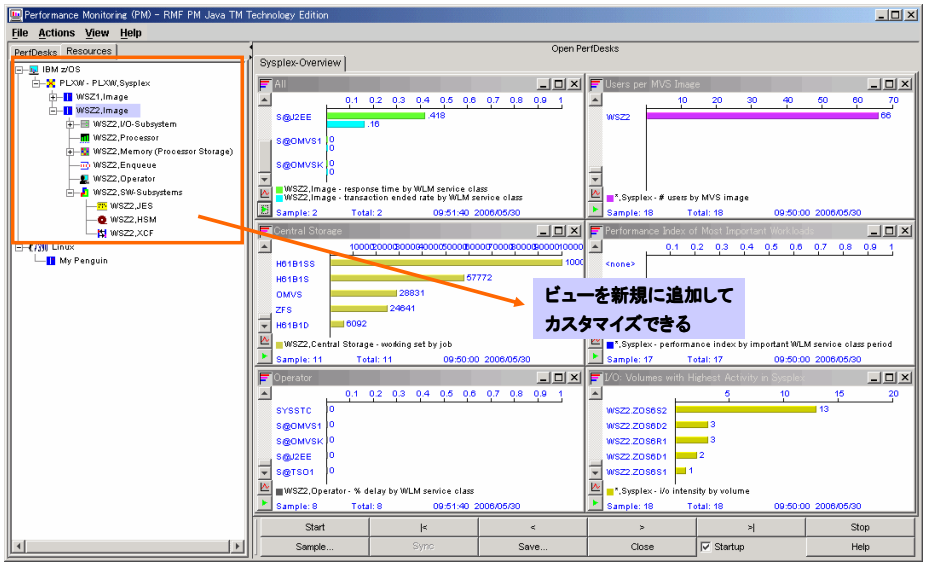
RMFモニター I のレポート出力例です。

RMFモニターⅢ

- 対話型（モニタリングは数分間隔）モニタリング
- Data gathererが必須
  - 起動コマンド　F RMF, START III,MEMBER(xx)
    - SYS1.PARMLIB(ERBRMFxx)のサフィックスを入れる。
- 3270画面上にテキストベースの表示
- RMF Performance Management (RMF PM)ツールを使用すればPCからGUIでモニターⅢを見ることができる。
  - http://www.ibm.com/servers/eserver/zseries/zos/rmf/rmfhtmls/pmweb/pmweb.html
  - 準備として、DDS (Distributed Data Server)のセットアップが必要。

RMFモニターⅢの説明です。

RMFモニターIII - RMF PM



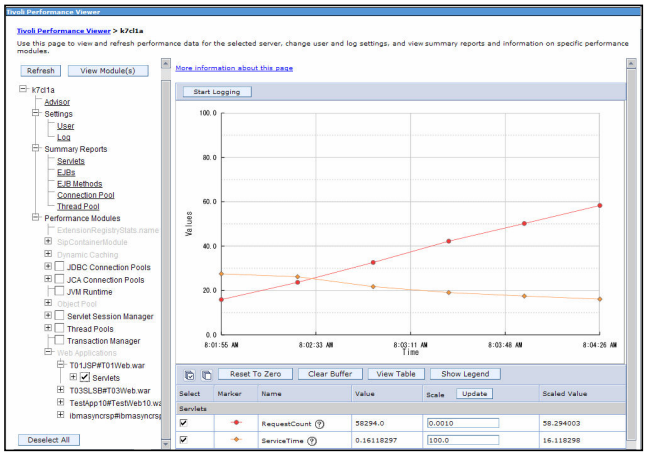
RMF PMを利用すると、ひとつの画面で、複数のz/OSを同時にモニターできます。

RMF PM Java Technology Edition

http://www-03.ibm.com/servers/eserver/zseries/zos/rmf/product/rmfhtmls/pmweb/pmweb.html

Tivoli Performance Viewer (TPV)

- PMI (Performance Monitoring Infrastructure) データを表示する
- V6以降はデフォルトでPMIオン。TPVはISC（管理コンソール）に統合
- TPVにはできないこと
  - WAS外部のリソース状況監視
    - CPU・DASD I/Oなど
  - 表示項目を増やしすぎるとTPV自体の負荷が高くなることに注意
- RMFにできないこと
  - WAS内部のリソース状況監視
    - JDBCコネクションプールなど



WAS for z/OS 7.0 TPVコンソール

Tivoli Performance Viewer (TPV)についての説明です。

SMFレコード120取得

- SMFレコード120とは
  - WAS for z/OSが出力するパフォーマンス情報
- V6までのSMFレコード120(タイプ1,3,5,6,7,8)は
  - SMF Activity Recordが取得の場合はパフォーマンスへの影響が大きい
  - 設定の変更にはサーバーの停止・再始動が必要
- V7で取得可能になったSMFレコード120タイプ9は
  - パフォーマンス・データ取得のオーバー・ヘッドが軽減
  - データの取得、未取得をサーバーの再起動なしに、動的に設定可能

内容	スコープ	レコード・タイプ	主な出力データ
サーバー	サーバー毎	タイプ1(Activity) タイプ3(Interval)	・起動されているサーバー・リージョンの数 ・接続クライアント・セッション数 ・セッション毎のデータ転送量
EJBコンテナ	アプリケーション毎	タイプ5(Activity) タイプ6(Interval)	・メソッド毎のコール数 ・平均、最大応答時間
Webコンテナー	Webアプリケーション(warファイル)毎	タイプ7(Activity) タイプ8(Interval)	・HTTPセッション数 ・サーブレット毎のコール数 ・サーブレット毎の平均、最大応答時間
リクエスト	**リクエスト毎**	**タイプ9**	**・タイプ1,5,7と同様のデータ**

New v7

※SMF: System Management Facility 。。。z/OS上のサブシステムが出力する情報をファイルに書き出すz/OSサービス

SMFレコード120についての説明です。

SMFレコード120-9取得

- SMFレコード120タイプ9の取得設定
  - **静的設定:**

    server_SMF_request_activity_enabled　　0 | 1 (0=OFF(デフォルト), 1=ON)

    server_SMF_request_activity_CPU_detail　　0 | 1

    server_SMF_request_activity_timestamps　　0 | 1

    server_SMF_request_activity_security　　0 | 1

  - **動的設定: (MVS Modify (F)コマンド)**

    F <server>,SMF,REQUEST,[ON | OFF]

    F <server>,SMF,REQUEST,CPU,[ON | OFF]

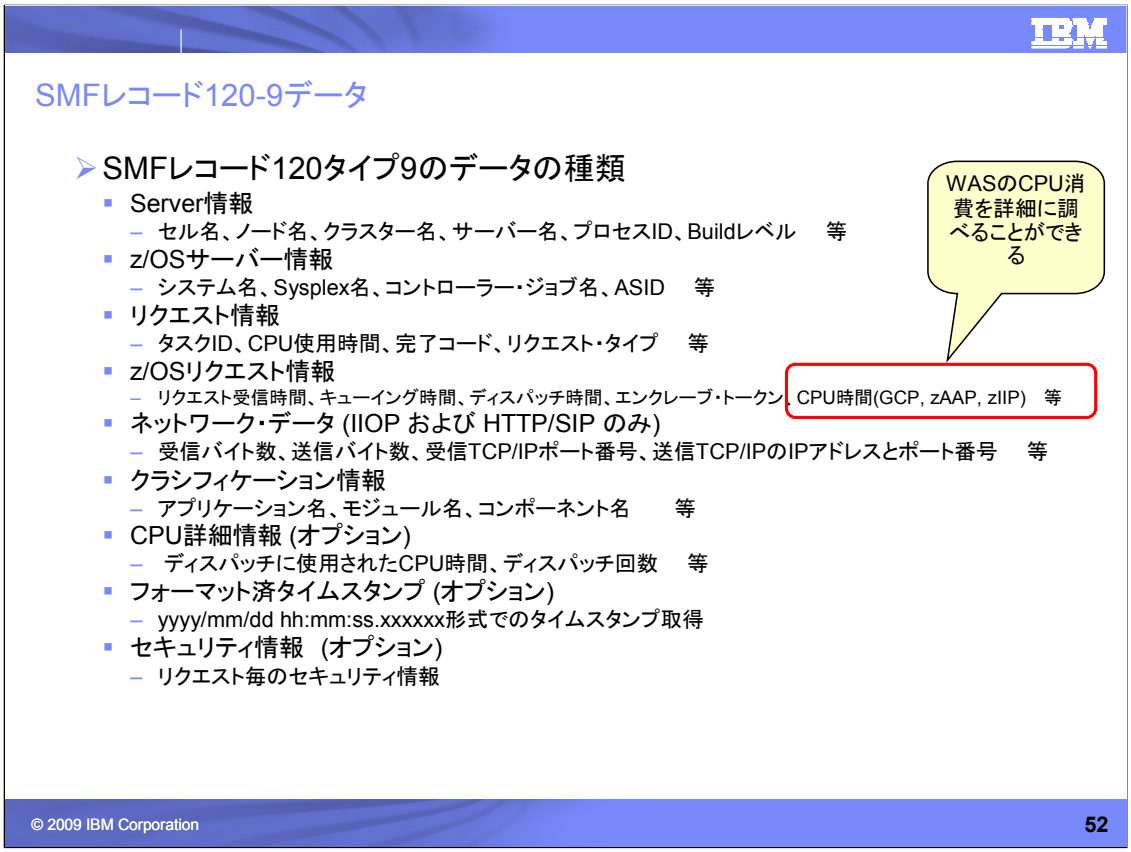
    F <server>,SMF,REQUEST,TIMESTAMPS,[ON | OFF]

    F <server>,SMF,REQUEST,SECURITY,[ON | OFF]

  - **設定確認コマンド:**

    F <server>,DISPLAY,SMF

SMFレコード120タイプ9取得の設定です。



SMFレコード120タイプ9データの種類です。

SMF タイプ120のフォーマット

- SMFレコードはそのままでは見ることができない。
- SMFをいったんデータセットにダンプし、SMF Browserでテキスト形式に変換する
  - SMFのダンプを行うJCL

```
//SMFDUMP  JOB MSGCLASS=H,CLASS=A,MSGLEVEL=(1,1)
//SMFDUMP  EXEC PGM=IFASMPDP
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//DUMPIN   DD DSN=SYS1.WSZ1.MANX,DISP=SHR
//DUMPOUT  DD DSN=MK.TEST.SMF120,
//     DISP=(NEW,CATLG,DELETE),UNIT=3390,VOR=SER=WORK11,
//     LRECL=32760,SPACE=(CYL,(10,10),RLSE)
//SYSIN    DD *
       INDD (DUMPIN,OPTIONS(DUMP))
       OUTDD (DUMPOUT,TYPE(120))
/*
```

  - SMF Browserの入手
    - ダウンロード・サイト
      - https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=zosos390
  - SMF Browserの実行

詳細レポート

サマリー・レポート

```
java -cp bbomsmfv.jar com.ibm.ws390.sm.smfview.Interpreter "MK.TEST.SMF120" "./MK.TEST_sum.txt" > MK.TEST_det.txt
```


SMFレコード120のフォーマット方法の説明です。
なお、SMFブラウザーの入手にはIBM IDが必要です。

SMF Browser 出力例

- SMF 120-9 サマリー・レポート例

```
SMF 120 Performance Summary V700              Date:  Wed Oct 8 09:10:41 EDT 2008   SysID: SYSA, Page 1

 - Record subtypes: 1:Svr_Act. 3:Svr_Int. 5:EJB_Act. 6:EJB_Int. 7:Web_Act. 8:Web_Int. 9:Request
   - subtype 9 Sections: CPU:CPU, N:Network, Cl: Classification, S:Security, T:Timestamps, U:UserData

SMF -Record Time     Server   Bean/WebAppName        Bytes  Bytes # of El.Time   Enclave_CPU_Time(uSec)
Numbr -Type hh:mm:ss Instance  Method/Servlet        toSvr  fmSvr Call  (msec)    GCP      zAAP      zIIP
1---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0--
    2 120.9  9:10:41 WAS7A1   MyWAR.name/MyServlet  1234   3456   12    7232     490        96        12
    3 120.9  9:10:47 WAS7A1   MyEJB.name/doMethod   1234   3456   12    7232     490        96         7
1---+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8----+----9----+----0--

REQUEST Recs: Avg Bytes, TranCount & Times =       9    3159   177       66     30596         0

===SMF=120=V700===== End of Report ====== ====== End of Report =SMF=V700=JMH= Oct 8,2008
Run Time:       0 seconds     =      746 milliseconds
```

メソッド名　応答時間　CPU使用

SMFレコード120タイプ9データの出力例です。

アプリケーションのCPU時間取得

- WAS for z/OSではアプリケーションの任意の箇所でCPU時間を取得できます
- **SMFJActivity.obtainTotalCpuTimeUsed()メソッド**
  - AppServer/plugins/com.ibm.ws.runtime.jar に含まれています
- 使用例

```
import com.ibm.ws390.sm.smf.SmfJActivity;
...
                    long startTime;
                    long stopTime;
                    long cpuTime;
                    startTime = SmfJActivity.obtainTotalCpuTimeUsed();

      < main Java code or method calls here >

                    stopTime = SmfJActivity.obtainTotalCpuTimeUsed();
                    cpuTime = stopTime - startTime;
                    System.out.println("CPU Time: " + cpuTime + " microseconds");
...
```


WAS for z/OSでは、アプリケーションの任意の箇所でCPU時間を取得できます。