WebOTX 運用編(チューニング)

WebOTX 運用編

バージョン: 7.1

版数: 第 9 版

リリース: 2011 年 2 月

目次

1. はじめに

本書は WebOTX 実行環境を運用するための運用操作法について概要や具体的な設定項目や設定方法について記載しています。

対象読者

このマニュアルは WebOTX Application Server Web Edition、Standard-J Ediotion、Standard Edition、Enterprise Edition を使って運用環境を構築するシステムエンジニア、日々の運用を行うオペレータを対象としています。

表記について

パス名表記

本書ではパス名の表記については特に OS を限定しない限りセパレータはスラッシュ'/'で統一しています。Windows 環境においては'¥'に置き換えてください。

環境変数表記

インストールディレクトリやドメインルートディレクトリなど環境によって値の異なるものについては環境変数を用いて表します。

${*env*} または $(*env*)で表しています。

例)
${AS_INSTALL} : インストールディレクトリ
${INSTANCE_ROOT}: ドメインルートディレクトリ

製品名表記

以後の説明で「WebOTX」としているものは、「WebOTX Application Server」のことをあらわします。

コマンド操作について

本書中では運用操作に用いるコマンドの詳細についての説明は省略しています。

コマンドの詳細は「運用管理コマンド」、「運用管理コマンドリファレンス」を参照してください。

2. APサーバのチューニング

Standard/Enterprise Editon における AP サーバのチューニングについて説明します。

2.1. アプリケーショングルーピング

WebOTX Standard/Enterprise Edition ではサーバアプリケーションは「アプリケーショングループ」および「プロセスグループ」という単位でグルーピングを行ないます。それぞれどのような観点でグルーピングを行なうべきか説明します。

2.1.1. アプリケーショングループ

アプリケーショングループとは、開始・終了などの運用を共にするアプリケーション群をグルーピングしたものです。例えば、「受発注業務」「在庫数管理業務」といったような業務毎にツリーを分けて管理するといった方法を用います。こうすることで、「受発注業務」は 10 時から 17 時まで動作させ、その後は停止させる。「在庫数管理業務」は 24 時間動作させるなどのアプリケーショングループ単位での独立した運用が行えるようになります。なおアプリケーションに関する設定(例えば、プロセスグループのプロセス数、スレッド数、Java VM オプション等)を変更する場合はアプリケーショングループの再起動が必要となります。

すなわち、業務運用の単位でアプリケーショングループをグルーピングします。

2.1.2. プロセスグループ

プロセスグループとは、ある特定のサービスを提供するプロセス群です。同一のプロセスグループに登録されているコンポーネントは同一のプロセス上でロードされます。 WebOTX はビジネスロジックを記述した 1 つまたは複数のコンポーネントをプロセスグループに登録し、プロセスとして実行します。

プロセスグループの単位で次の項目が共有されます。

- キュー(受信用)

  WebOTX ではプロセスグループ単位にキューを生成します。同一プロセスグループ上のリクエストはすべてこのキューにキューイングされます。

- Java VM(Java 関連のプロセスグループのみ)

  WebOTX ではプロセスグループ単位に Java VM を生成し、配下のコンポーネントをロードします。マルチプロセス構成の場合は同一構成の Java VM が複数生成されます。Java VM に関する設定(ヒープサイズのような VM オプション等)もプロセスグループ単位での設定となります。

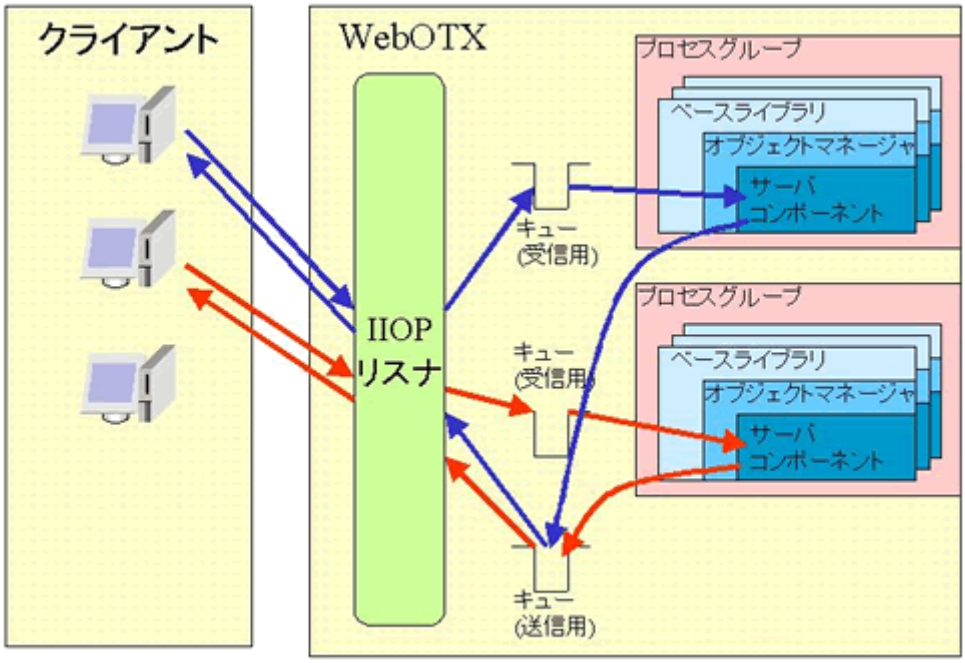
- 実行多重度

  WebOTX ではプロセスグループ単位に実行多重度(プロセス、スレッド)を設定します。そのプロセスグループの特性に応じて実行多重度を設定できます。バックエンドサーバへのコネクション

  AP サーバが利用するバックエンドサーバ(DB や ACOS や TPBASE)へのコネクションはプロセスで共有されます。よってプロセスグループを分けたり、マルチプロセスにしたりする場合はその分コネクションが生成されることになります。また VIS コネクタの場合、全てのプロセスグループで定義している総スレッド数分のコネクションが生成されます。

2.1.3. 処理の流れ

Standard/Enterprise Edition のサーバ AP での処理の流れについて説明します。



上記で説明したように、プロセスグループ単位でキュー(受信用)が生成されます。クライアント(thin クライアントの構成では Web サーバ)からの要求は IIOP リスナを経由して、該当プロセスグループのキューにキューイングされます。そのときにプロセスグループ内でアイドル中(フリーな)のスレッドが存在していた場合、即時にそのスレッドでリクエストは実行されます。アイドル中のスレッドが無い場合は、キューで実行待ちとなります。

プロセスグループでキューは共有しますので、例えば実行時間が非常に長い呼び出しが実行された場合、そのリクエストを処理するために 1 つのスレッドは占有されます。これにより同じプロセスグループ内にある他の呼び出しを実行するスレッドが減少してしまいます。全てのスレッドが長い呼び出しに占有されてしまった場合、そのプロセスグループに対する要求は全てキューイングされることになります。

2.1.4. マルチプロセス・マルチスレッド

プロセスグループを多重実行させる場合、プロセスを多重化させる方法とスレッドを多重化させる方法があります。

各プロセスはシングルスレッドでもマルチスレッドでも動作させることができます。すべてのプロセス上のスレッド群は、単一のキューに対してデータの到着を待ち合わせるので、空きスレッドに効率良くトランザクション要求をディスパッチすることができます。マルチプロセス実行させることにより、アプリケーションの一時的な障害に対してサービスの継続が可能になります。データや環境、タイミングなどの要因でアプリケーション障害が発生しても、そのプロセスだけが異常終了し、他のプロセスは影響を受けません。したがって、システム全体ではサービスを中断させることがありません。

しかしマルチプロセス構成はマルチスレッド構成に比べてより多くのリソースを必要とします。システムで多数のプロセスが起動することによりマシンのリソース不足を招き、サーバ自体が不安定になる恐れがあります。マルチプロセスは予期せぬアプリケーション障害に備えての多重化とし、クライアントからの同時実行に備えての多重化はできるだけマルチスレッドで行う構成が望ましいといえます。ただし、アプリケーションの構成上(スレッド間同期、排他制御)の問題でマルチスレッド動作できないなどの制限がある場合は、マルチプロセス構成で多重度を確保します。

また、システム運用中に発生した想定外の大量のトランザクション要求に対して、動的にアプリケーション多重度(スレッド、プロセス)を増加させることが可能です(注:スレッド数は最初に確保した数以上は増やせません)。これにより、大量のトランザクション要求に対してサーバ側が処理しきれずに要求がキューに滞留しレスポンス悪化することが避けられます。また、負荷が下がったときに安全に実行多重度を

元に戻すこともできます。

運用アシスタント機能により、システムの稼働状況に応じて多重度の増減を推奨（または自動変更）することができます。高負荷により設定された応答時間を満たさないと予想される場合は、多重度を増やすよう通知(または自動変更)します。負荷が下がった状態が続くと多重度を減らすように通知(または自動変更)します。詳しくは「運用編(TP モニタの運用操作) 2.40.3 多重度の最適化支援」を参照してください。

2.1.5. グルーピングのポイント

上記の WebOTX のプロセスグループの特性を考慮してどのようにグルーピングするのが望ましいか説明します。

消費メモリについて

多重度を確保するにはマルチプロセスとマルチスレッドがありますが、特にマルチプロセス構成にする場合メモリ消費量を考慮する必要があります。以下に WebOTX Java AP のメモリ消費量の目安の計算式を示します。

(ヒープサイズ)＋(スレッドスタックサイズ)×スレッド数＋15(MB)
※1 プロセスあたりの計算式です。実際は全てのプロセスの合計値となります
　また、OS によりメモリ消費量は計算式から増減があります。
スレッド数を上げても(スレッドスタックサイズ:既定値 1M)分しか増加しないのに比べ、プロセス数を上げると上記の計算式の値全てのサイズ分増加します。メモリ量を考慮すると、多重度はマルチプロセスよりマルチスレッドで行うべきです。

ライセンスについて

VIS コネクタを使用した場合、TP システムで定義している総スレッド数分コネクションを作成します。VIS コネクタのライセンスはコネクション単位ですので、総スレッド数がライセンスを越えることができません。

例えば
WebOTX VIS コネクタ実行環境 (4)
WebOTX VIS コネクタ実行環境 (+8)
を購入している場合 4+8(=12)ライセンスですので、1 プロセスグループ 1 プロセス 1 スレッド構成にするにしても 12 個しかプロセスグループを作成することができません。1 プロセスグループ 2 プロセス 2 スレッド構成の場合は 3 個(12÷4)のプロセスグループしか作成することができないこととなります。

コネクション数について

バックエンドサーバへのコネクション数についても注意が必要となります。マルチプロセス構成とした場合、プロセスの数分、コネクションを生成することとなります。多くのプロセスを生成すると、バックエンドサーバとのコネクション数の制限を越えてしまうことがあります。

キューについて

上記にも書きましたように、WebOTX はプロセスグループ単位でキューを生成します。ここで作成したコンポーネントを複数のプロセスグループに登録するのと、1 プロセスグループに登録するのではどちらがいいのかの考え方について説明します。

複数のプロセスグループに登録する場合

複数のプロセスグループに登録する場合は、キューやプロセス空間が他のコンポーネントと完全に独立します。あるコンポーネントに問題があり、ストールやレスポンス悪化、アボートが発生した場合でもその影響をほとんど受けません。あるコンポーネントがストールもしくはレスポンス悪化(もともと正常でも時間がかかる呼び出しでも同様)があった場合、プロセスグループ中のスレッドが、全てストールしてしまう可能性があります。アボートの場合は、Java VM 全体でアボートとなり、全てのコンポーネントへの影響が出てきます。このような状態になった場合、そのプロセスグループへの呼び出しは例えその呼び出し自体に問題が無くてもそれ以上処理されなくなりそのプロセスグループ全体がストールしたり、実行プロセスがアボートしてなくなってしまいます。プロセスグループを分けることは、このような現象を未然に予防することができます。

1 プロセスグループに登録する場合

上記で説明したように 1 プロセスグループに登録する場合ストール、レスポンス悪化、アボートに対して注意が必要になります。対策として、マルチスレッドで多重度を十分に確保することおよびマルチプロセ

スでアボートしても、プロセス数が0となることを行っておくことがあげられます。複数のプロセスグループの構成より万全ではありませんがある程度問題は回避できます。

2.2. チューニングのための事前確認

チューニングの方針を立てるためには現在使用中または使用予定の WebOTX のメモリ使用量や性能を正しく把握する必要があります。WebOTX や OS が提供する情報採取方法について説明します。

2.2.1. メモリ使用量の確認

WebOTXが必要とするメモリ使用量はマニュアルの[セットアップガイド]-[1.使用上の条件]-[1.1 必要リソース]を参照してください。WebOTXのエージェントプロセスが使用するJVMヒープサイズの確認については「運用編(モニタリング) 3.3.1 JVMヒープサイズのモニタリング」を参照してください。アプリケーションプロセスのメモリ使用量の確認については「運用編（TPモニタの運用操作）2.37 統計情報の表示」を参照してください。その他のプロセスについてはタスクマネージャ(Windowsの場合)またはtopコマンド(unixの場合)で確認してください。また、OS全体でのメモリ残量はタスクマネージャ(Windowsの場合)またはswapinfoコマンド(unixの場合)で確認できます。HP-UXの場合はHPJmeterなどでヒープ使用量を確認することもできます。

2.2.2. アプリケーション処理時間の確認

実行時間上限設定を適切に行うために、運用アシスタントの実行時間上限の適正値算出機能を利用してください。詳細は「運用編(TP モニタの運用操作) 2.40.4 実行時間上限の適正値算出」を参照してください。

また、プロセス数やスレッド数を適切に設定するためにはキュー滞留数がひとつの目安になります。統合運用管理ツールでのキュー滞留数の確認方法については「運用編(モニタリング) 3.3.3 キューイング数のモニタリング」を参照してください。さらにquewrtコマンドで確認することもできます。「運用編(TPモニタの運用操作) 2.39.5 キュー滞留数の確認方法」を参照してください。

2.2.3. 多重度の確認

クライアント多重度、スレッド多重度がどこまで使われているのかを確認する方法を説明します。

・ クライアント多重度

使用中の接続クライアント数については「運用編(モニタリング) 3.3.4 接続クライアント数のモニタリング」を参照してください。

・ スレッド多重度

WebOTXは稼動スレッド数などを記録するためにオペレーションジャーナルを採取しています。オペレーションジャーナルを編集することにより、稼動スレッド数を時間毎に調査することができます。「運用編(TPモニタの運用操作) 2.38 オペレーションジャーナルの編集」を参照してください。

2.3. リソースチューニング

WebOTX で消費するメモリについてチューニングする方法について説明します。またディスクサイズを節約する方法について説明します。

2.3.1. ドメインエージェントプロセスのヒープサイズの見直し

ドメインエージェントプロセスのヒープサイズを見直すことにより、ドメインエージェントプロセスの使用メモリをチューニングできます。なおチューニングする値については GC ログを採取するなどして見積もる必要があります。少なくしすぎると OutOfMemoryError が発生します。

設定方法

統合運用管理ツールのドメイン毎に設定

・ [server]-[java-config]-[JVMオプション] -Xmx512m -Xms64m 部分を編集

なお、GC採取時以下のログが出力されていた場合、Javaの通常のヒープとは別のPermanent領域と

いうヒープ領域が足りなくなったことを示しています。

Permanent generation is full...

increase MaxPermSize (current capacity is set to: 67108864 bytes)

ここにはクラスの内部表現が展開されます。アプリケーションによっては非常に多くのクラスをロードするものもあり、Permanent 領域が足りなくなることがあります。このような場合には、Permanent 領域のサイズを変更する必要があります。現在は、Javaのデフォルト値である 64Mに設定されています。

[server]-[java-config]-[JVMオプション]にて、以下を追加してドメインを再起動してください。

-XX:MaxPermSize=<最大サイズ>

例) 最大 256Mに設定

-XX:MaxPermSize=256M

2.3.2. サーバプロセスのヒープサイズ・スタックサイズの見直し

サーバ AP プロセスのヒープサイズ・スタックサイズを見直すことにより、サーバ AP プロセスの使用メモリをチューニングできます。GC ログを採取するなどして見積もる必要があります。少なくしすぎると OutOfMemoryError が発生します。

設定方法

統合運用管理ツールのプロセスグループ毎に設定

- [スレッド制御]-[スレッドスタックサイズ]
- [JavaVMオプション]-[最大ヒープサイズ]及び[初期ヒープサイズ] (Javaのみ)

2.3.3. 起動サービスの見直し

ドメイン作成時の既定値では、全てのサービスが起動するように設定されます。運用上必要のないサービスを起動させない設定にすることによりメモリを削減することができます。

設定方法

統合運用管理ツールのドメイン毎に設定

- [server]-[internal-lifecycle-module]-[(各サービス)]-[設定項目(Configurations)]-[起動の可否]

表示されていないサービスを表示させるためには、統合運用管理ツールの[システム]-[システム設定]-[画面表示]において、管理対象の表示レベルと属性の表示レベルを詳細レベルの情報を表示するように変更してください。

2.3.4. プロセス数・スレッド数の見直し

サーバ AP プロセスのプロセス数およびスレッド数を減らすことによりシステム全体のメモリを削減することが出来ます。それ以上にメモリの節約を必要とする場合はコンポーネントの集約を行うことによりメモリを削減できます。ただし、障害の局所化か難しくなるなどのデメリットもあるので注意が必要です。

設定方法

- 統合運用管理ツールのプロセスグループの[プロセス制御]-[プロセス数]
- 統合運用管理ツールのプロセスグループの[スレッド制御]-[スレッド数]

2.3.5. ディスクサイズの節約

ディスクサイズを節約するためにアプリケーションログのファイルサイズを制限したり、プロセス終了により退避されたアプリケーションログを削除したりすることができます。例えば削除可能なアプリケーションログについては「運用編(ロギング) 2.5.1 サーバアプリケーショントレース採取」に記述している save ディレクトリがあります。但し、ログの削除は障害発生時のプロセスの異常終了の原因究明に支障が出る可

能性があるため注意が必要です。

この save ディレクトリを定期的に削除したい場合はタスクスケジューラ(Windows の場合)や cron(unix の場合)を利用する方法があります。

・ Windows の場合

タスクスケジューラは[コントロールパネル]-[タスク]-[スケジュールされたタスクの追加]で使用できます。save ディレクトリ削除(退避)用バッチを作成してタスクスケジューラに登録してください。

・ UNIX の場合

cron はcrontabファイルの命令に従って実行されます。HP-UX の場合、crontabファイルは実行ユーザ名と同じで、/usr/lib/cron/cron.allow ファイルに該当する名前がある場合に crontab を実行できます。他 OS ではディレクトリやファイル名が異なる場合がありますが、基本的には使い方は同じです。詳細は man コマンドで確認してください。

2.3.6. 配備時のエージェントプロセスメモリ節約

Standard/Enterprise Edition で動作する TP モニタは、オペレーション毎にトランザクション等の実行時情報を管理する機能を備えています。あるモジュールが配備されると、TP モニタでは個々のオペレーションを識別するために番号を付与して、内部管理テーブルと状態情報との関連付けを行います。EJB モジュールの場合には、オペレーションがリモートインタフェースのメソッドに該当します。

メソッド識別情報の割り当て範囲を限定することにより、配備処理中に行われる識別情報の生成数を削減し、エージェントプロセスでのメモリ消費量の削減が可能です。

詳しくは、運用編マニュアル"アプリケーションの配備"中「9.1 EJB のメソッド識別情報の割り当て抑止」を参照してください。

2.4. 性能チューニング

サーバ AP の性能チューニングについて説明します。

2.4.1. 多重度のチューニング

プロセスグループの適正な多重度を算出するための方法について説明します。

理論的に算出

待ち行列理論を用いた計算により理論値を算出する方法について説明します。まずは以下の値について要件を整理します。

- 平均到着間隔:Ta

  単位時間のトランザクション処理件数の逆数です。既存システムもしくはユーザ要件から算出してください。

- 平均サービス時間:Ts

  サーバでのその処理の実行時間です。既存システムもしくはユーザ要件から算出してください。

- レスポンス時間:R

  クライアントからみた応答時間です(サービス時間+待ち時間)。既存システムもしくはユーザ要件から算出してください。

- 多重度:N

  プロセスグループの多重度です。

これらを待ち行列理論で計算して必要な多重度 N を求めます。

N=RTs/(Ta(R-Ts))

例えば

Ta=0.1 (10 件/秒)

Ts=0.2

R=0.5

とすると

N=0.5*0.2/(0.1*(0.5-0.2))=3.3…

となり要求を満たすには 4 多重必要ということになります。

プロトタイプを用いた算出

本番運用を想定したプロトタイプを動作させ多重度を求める方法について説明します。プロトタイプを用いた測定のほうがより実運用に即した値を求めることが出来ます。

単体評価で算出

平均サービス時間:Ts を 1 多重で 1 つのクライアントからアクセスして時間を求めます。その他の値は既存システムもしくはユーザ要件から算出した値を用いて多重度を求めることが出来ます。

負荷評価で確認

実際に平均到着間隔:Ta だけの負荷をかけて多重度を変えながらレスポンス時間を確認します。

運用アシスタントの利用

運用アシスタントにより多重度の適正を診断することができます。多重度が多すぎる/少なすぎる場合は統合運用管理ツールなどを通して通知されます。
また運用アシスタントの多重度自動変更機能を利用することにより、システムの稼働状況に合わせて多重度を自動的に変更させることができます。詳しくは「運用編(TP モニタの運用操作) 2.40.3 多重度の最適化支援」を参照してください。

2.4.2. CPU使用率の確認

応答時間が長い場合はその時の CPU 使用率を確認してください。応答時間が長い場合はその時の CPU 使用率を確認してください。プロセスグループの「動作情報」タブの「プロセス情報」で確認できます。CPU 情報はオペレーションジャーナルに記録され、統計的に解析することが可能です。
CPU 使用率が高い場合はプロセス数やスレッド数を増やしても性能がよくなるとは限らないため、CPU 使用率を減らすなどの対策が必要です。

2.4.3. 重要な処理を優先的に実行したい場合の設定

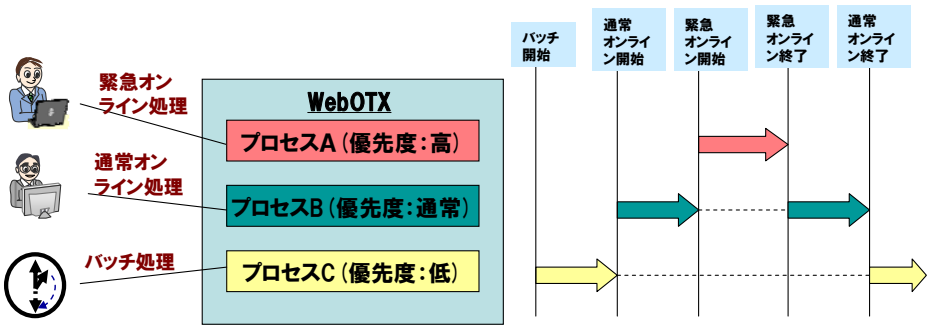
多重度を増やす等の対処をおこなっても、他のプロセスが動作中で CPU 時間がなかなか割り当てられない場合などはレスポンス時間が大きくなってしまいます。プロセスの優先度を指定することにより、高優先度の処理は他のプロセスより優先されるため、CPU 負荷が高い状況でも処理投入後即開始することが可能となります。

例えば、以下のように業務特性に合わせた最適な優先度設定が可能です。(※)

処理	優先度
緊急オンライン処理	高
通常オンライン処理	通常(未設定)
バッチ処理	低

(※)優先度を維持するためには優先度を固定化する必要があります。また Linux OS では優先度を固定することが出来ません。優先度を高く設定しても、実行を続けると優先度が落ちる場合がありま

す。



さらに、呼び出し処理単位の優先度設定（オペレーション優先度）と組合せて、よりきめ細かな設定が可能です。

プロセス優先度の設定はプロセスグループ単位で行います。具体的な設定方法は「運用編(TPモニタの運用操作)」の「2.22 サーバアプリケーション制御のための設定」を参照してください。

オペレーション優先度の設定はオペレーション単位で行います。具体的な設定方法は「運用編(TPモニタの運用操作)」の「2.24 オペレーション制御のための設定」を参照してください。

2.4.4. 性能ネック箇所の特定

性能測定を行なった結果、性能が想定していたより出ない場合、どの部分に問題があるのか特定する方法について説明します。

レスポンス時間の把握

実際にどれくらいの実行時間で運用しているかを確認する方法について説明します。レスポンス時間の把握により問題のあるオペレーションを特定することが出来ます。
オペレーションの性能情報は[統計情報]-[ドメイン名]-[アプリケーション]-[アプリケーション名]-[モジュール名]-[インタフェース名]-[オペレーション名]を参照して下さい。
また、レスポンス時間(キュー待ち時間を含む応答時間)の他に、実行時間（キュー待ち時間を含まない処理時間）と CPU 時間を確認することで性能ネック箇所を切り分けることができます。性能ネック箇所として疑わしいのは以下の箇所となります。

レスポンス時間 >> 実行時間 であれば多重度不足
実行時間 >> CPU 時間 であれば DB やネットワークなどのバックエンド問題
実行時間 ≒ CPU 時間 であれば業務 AP 内でのループ

キュー滞留数の把握

キュー滞留が発生しているかを確認する方法について説明します。キュー滞留が発生している場合、多重度が不足しておりレスポンス悪化になっている可能性が考えられます。

キュー滞留数の確認方法については「運用編(TPモニタの運用操作)」の「2.39 障害解析」を参照ください。

オペレーションジャーナルの活用

オペレーション単位でオペレーションの処理状況(平均時間、最大時間、最小時間、呼び出し回数、平均 CPU 使用時間(ユーザモード/カーネルモード)、最大 CPU 使用時間(ユーザモード/カーネルモード)、最小 CPU 使用時間(ユーザモード/カーネルモード))を確認することができます。これによりより問題のあるオペレーションを特定することが出来ます。

オペレーションジャーナルの確認方法については「運用編(TPモニタの運用操作)」の「2.38 オペレーションジャーナルの編集」を参照ください。

イベントジャーナルの活用

イベントジャーナルを採取してオペレーションの処理の流れを詳細に調べることができます。

イベントジャーナルについては「運用編(TPモニタの運用操作)」の「2.39 障害解析」を参照ください。

プロファイリング(Java APのみ)

Java アプリケーションの場合プロファイリングを行なうことにより、より詳細に問題箇所を特定することが出来ます。

2.5. その他チューニングに関する設定

高負荷になりレスポンスが悪化した場合などに考慮が必要なタイムアウト値や、プロセス起動・停止に必要なタイムアウト値などに関する設定、その他上限設定について説明します。

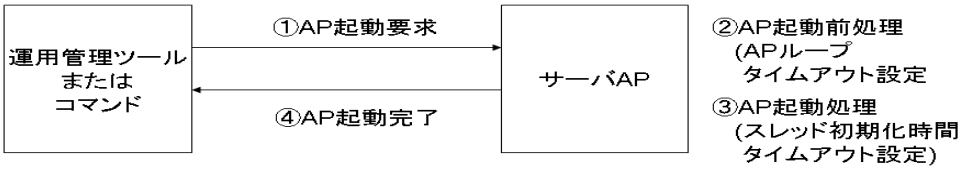


図1 AP起動の流れ(2.5.1, 2.5.2参照)

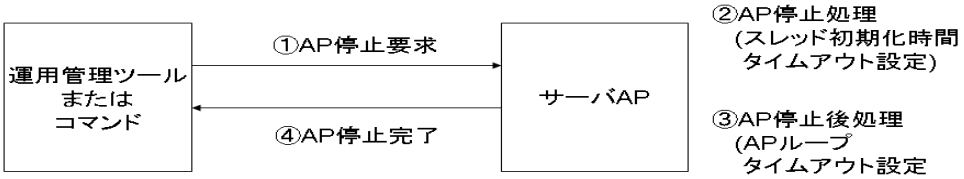
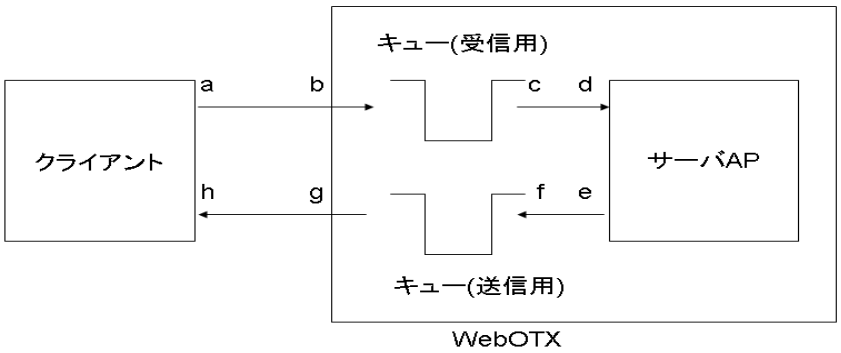


図2 AP停止の流れ(2.5.1, 2.5.2参照)



区間a-h ： 呼び出しタイムアウト(クライアント側) ※2.5.3参照

区間b-g ： 呼び出しタイムアウト(サーバ側) ※2.5.3参照

区間d-e ： 実行時間タイムアウト ※2.5.4参照

図3 オペレーション呼び出しの流れ

2.5.1. 起動・停止タイムアウト(メッセージ表示)

プロセスの起動操作、停止操作の完了を待ち合わせる時間を設定します。

タイムアウトした場合はエラーメッセージが表示されます。但し、プロセスの起動処理自体は実行され続けます。

設定項目	説明	既定値	設定値範囲	補足事項
1. システム起動タイムアウト	システムの起動処理に関するタイムアウトの設定をします。	120 秒	1 以上の整数で秒単位	内部プロセスの電文応答待ち時間である 70 秒よりも長い値を指定してください
2. システム停止タイムアウト	システムの停止処理に関するタイムアウトの設定をします。	120 秒	1 以上の整数で秒単位	内部プロセスの電文応答待ち時間である 70 秒よりも長い値を指定してください
3. アプリケーショングループ起動タイムアウト	アプリケーショングループの起動処理に関するタイムアウトの設定をします。	120 秒	1 以上の整数で秒単位	内部プロセスの電文応答待ち時間である 70 秒よりも長い値を指定してください
4. アプリケーショングループ停止タイムアウト	アプリケーショングループの停止処理に関するタイムアウトの設定をします。	120 秒	1 以上の整数で秒単位	内部プロセスの電文応答待ち時間である 70 秒よりも長い値を指定してください
5. プロセスグループ起動タイムアウト	プロセスグループの起動処理に関するタイムアウトの設定をします。	120 秒	1 以上の整数で秒単位	内部プロセスの電文応答待ち時間である 70 秒よりも長い値を指定してください
6. プロセスグループ停止タイムアウト	プロセスグループの停止処理に関するタイムアウトの設定をします。	120 秒	1 以上の整数で秒単位	内部プロセスの電文応答待ち時間である 70 秒よりも長い値を指定してください

設定方法

1. otxadmin> set tpsystem.startTimeOut=120
2. otxadmin> set tpsystem.stopTimeOut=120
3. otxadmin> set tpsystem.applicationGroups.アプリケーショングループ名.startTimeOut=120
4. otxadmin> set tpsystem.applicationGroups.アプリケーショングループ名.stopTimeOut=120
5. otxadmin> set tpsystem.applicationGroups.アプリケーショングループ名.processGroups.プロセスグループ名.startTimeOut=120
6. otxadmin> set tpsystem.applicationGroups.アプリケーショングループ名.processGroups.プロセスグループ名.stopTimeOut=120

2.5.2. 起動・停止タイムアウト(プロセス終了)

アプリケーションプロセスの起動または停止に長い時間がかかる場合はタイムアウトを検出して異常終了することがあります。この場合、処理時間が妥当か検証し、妥当でないなら処理を見直し、妥当ならタイマ値を見直す必要があります。

設定項目	説明	既定値	設定値範囲
1. スレッド初期化時間	スレッド初期化にかかる時間のタイムアウト値を設定します。	600 秒	1-2147483647 で秒単位
2. AP ループ	スレッド初期化前及びスレッド終了後にかかる時間のタイムアウト値を設定します。	600 秒	1 以上の整数で秒単位

スレッド初期化時間のタイムアウト値は以下の場所で時間のかかる処理を行っている場合に考慮が必要です。

・ コンストラクタ・デストラクタ(アパートメントの場合)

・ 常駐オブジェクトのコンストラクタ・デストラクタ

設定を反映させるためにはアプリケーショングループを再起動してください。

また、AP ループのタイムアウト値は以下の場所で時間のかかる処理を行っている場合に考慮が必要です。

・ コンストラクタ・デストラクタ(ステートレスフリーの場合)

・ コンポーネント初期化インタフェース

・ 実装情報の定義関数

・ プロセス起動および終了時のコールバック関数

・ WO_ObjectListener (JavaAP でフリースレッドモデルの場合のみ)

・ BindingHooks (ステートレスフリーで名前サーバへの登録方法が一時的の場合のみ)

設定を反映させるためには TP システムを再起動してください。

設定方法

1. スレッド初期化時間

統合運用管理ツールのプロセスグループの[スレッド制御]-[スレッド初期化時間]

2. AP ループ

ファイル(unix の場合) : /opt/WebOTX/domains/<domain 名>/config/tpsystem/tpbase.cnf

修正前 : APLOOP 600

修正後 : APLOOP xxx

(注: xxx にプロセス起動または停止に必要な時間以上の値を秒単位で入れてください)

2.5.3. 呼び出しタイムアウト

・ クライアント側で設定する場合

クライアント側でオペレーション実行要求を出してからその応答をもらうまでのタイムアウト値を設定します。このタイムアウト値を設定することで、クライアントが無応答となることを抑止します。このタイムアウト値はサーバでの実行時間のほかに通信に要する時間およびキュー待ち時間を考慮して設定します。

設定項目	説明	既定値	設定値範囲
1. Java クライアントからのオペレーション呼び出しタイムアウト時間	EJB、JNDI など RMI-IIOP 通信を使用する呼び出しのタイムアウト時間です。	30 秒	0 以上 ※0 を指定した場合は無制限
2. Java クライアントでの無通信監視タイムアウト時間	クライアント側で一定時間送受信要求のないコネクションをクローズするまでのタイムアウト時間です。	無制限	0 以上 ※0 を指定した場合は無制限
3. オペレーション呼び出しのタイムアウト時間	CORBA 通信を使用する呼び出しのタイムアウト時間です。	30 秒	0 以上 ※0 を指定した場合は無制限
4. コネクションラウンドロビンでのオペレーション呼び出しのタイムアウト時間	多重化オブジェクトによるコネクションラウンドロビン機能を利用する場合に有効になります。 個々のサーバへのオペレーション呼び出しのタイムアウト時間	オペレーション呼び出しのタイム	0 以上 ※0 を指定した場合は無制限

	です。	アウト時間の値	

設定方法

1. Java クライアントからのオペレーション呼び出しタイムアウト時間

   ・ Java コマンドラインで指定する場合

   以下の形式でシステムプロパティを指定

   -Djp.co.nec.orb.RequestTimeout=<タイムアウト時間(秒)>

   ・ J2EE サーバ(Web コンテナ、JNDI サーバ)に指定する場合

   統合運用管理ツールの server.java-config の「JVM オプション」に以下の形式でシステムプロパティを追加

   -Djp.co.nec.orb.RequestTimeout=<タイムアウト時間(秒)>

   ・ プロセスグループ(EJB コンテナ)に指定する場合：

   統合運用管理ツールの tpsystem.applicationGroups.<アプリケーショングループ名>.processGroups.<プロセスグループ名>の「Java システムプロパティ」で行の追加を実行し、1 列に「jp.co.nec.orb.RequestTimeout」、2 列に秒数を指定

2. Java クライアントでの無通信監視タイムアウト時間

   ・ Java コマンドラインで指定する場合

   以下の形式でシステムプロパティを指定

   -Djp.co.nec.orb.ClientAutoTimeout=<タイムアウト時間(秒)>

   ・ J2EE サーバ(Web コンテナ、JNDI サーバ)に指定する場合

   統合運用管理ツールの server.java-config の「JVM オプション」に以下の形式でシステムプロパティを追加

   -Djp.co.nec.orb.ClientAutoTimeout=<タイムアウト時間(秒)>

   ・ プロセスグループ(EJB コンテナ)に指定する場合

   統合運用管理ツールの tpsystem.applicationGroups.<アプリケーショングループ名>.processGroups.<プロセスグループ名>の「Java システムプロパティ」で行の追加を実行し、1 列に「jp.co.nec.orb.ClientAutoTimeout」、2 列に秒数を指定

3. オペレーション呼び出しのタイムアウト時間

   ・ 統合運用管理ツールで指定する場合

   server.objectbrokerconfig の「共通」タブの「リクエスト呼び出しのタイムアウト時間」に秒数を指定

   ・ コマンドで指定する場合

   otxadmin> set server.objectbrokerconfig.RequestTimeout=<設定値>

   【注意】

   Java クライアントの場合、上記の「Java クライアントからのオペレーション呼び出しタイムアウト時間」が設定されていれば、そちらが本設定よりも優先されます

4. コネクションラウンドロビンでのオペレーション呼び出しのタイムアウト時間

   ・ 統合運用管理ツールで指定する場合

   server.objectbrokerconfig.の「Cpp」タブの「多重化使用時の最大待ち時間」に秒数を指定

   ・ コマンドで指定する場合

   otxadmin> set server.objectbrokerconfig.ConnectionRoundRobinTimeout=<設定値>

・ サーバ側で設定する場合(AP 応答監視タイマ)

オペレーション実行要求をサーバが受け付けてからその応答を返すまでのタイムアウト値を設定します(既定値:上限なし)。このタイムアウト値を設定することで、クライアントが無応答となることを抑止します。このタイムアウト値はサーバでの実行時間のほかに通信に要する時間およびキュー待ち時間を考慮して設定します。各オペレーションの実行時間(予測値)の最大値よりは大きな値を設定する必要があります。設定を反映させるためにはアプリケーショングループの再起動が必要です。

なお、AP 応答監視タイマを超過した場合、クライアントにエラーは返りますがサーバ AP は処理を継続したままなのでサーバ AP の無応答状態が改善するわけではありません。サーバ AP の無応答状態を改善させるためには[2.5.4 実行時間タイムアウト]を設定してください。

設定方法

統合運用管理ツールのプロセスグループの[上限設定]-[AP 応答監視タイマ]

2.5.4. 実行時間タイムアウト

サーバ側でオペレーション実行を行なってから実行完了までのタイムアウト値を設定します(既定値:上限なし)。この値はサーバでの実行時間のみ考慮して設定します(キュー待ち時間は含みません)。

実行時間上限設定を適切に行うために、運用アシスタントの実行時間上限の適正値算出機能を利用してください。詳細は「運用編(TP モニタの運用操作) 2.40.4 実行時間上限の適正値算出」を参照してください。

なお、実行時間タイムアウト値を超過した場合、AP は終了します。AP を再起動させるためには再起動回数(統合運用管理ツールの TP システムの[上限設定]-[プロセス障害時の再起動回数])を 2 以上にする必要があります。推奨はデフォルトの 5 です。プロセス障害時の再起動回数が 1 で AP が終了した場合、プロセスは終了したままで再起動しませんので注意してください。

設定方法

統合運用管理ツールのオペレーションの[オペレーションの自動活性]-[実行時間上限]

2.5.5. クライアント無通信監視間隔

サーバとクライアントの間の無通信状態を監視します(既定値:上限なし)。本タイマ値以上無通信状態(具体的には要求も応答も流れない状態)が続いた場合はコネクションを切断します。設定を反映させるためには TP システムの再起動が必要です。

設定方法

統合運用管理ツールの[IIOPListener]-[クライアント制御]-[クライアント無通信監視を行う]及び[クライアント無通信監視間隔]

2.5.6. キューイング数上限

サーバアプリケーションすべてのスレッドがクライアントからのリクエスト処理を行なっている場合、新たな要求はキューイングされ実行待ちとなります。既定値ではメモリの許す限りキューイングされてしまい、クライアントから見るとキュー数が増えれば増えるほど無応答時間が長くなります。ある程度以上のキューイングを抑制することによりクライアントのレスポンスを保証することが出来ます。

(平均待ち時間)=(平均サービス時間)*{(キュー数)+1}

設定方法

統合運用管理ツールで TP システム毎、アプリケーショングループ毎、プロセスグループ毎(後者ほど設定優先度高)に設定できます。

・ TPシステムの[上限設定]-[キューの最大数]

・ アプリケーショングループの[キューの最大数]-[キューの最大数]

・ プロセスグループの[キューの最大数]-[キューの最大数]及び[アプリケーショングループと同様の最大数とする]

2.5.7. 接続クライアント数上限

richクライアントの場合、想定以上の接続を受けるとサーバ側資源(主にメモリとファイルディスクリプタ)を余分に使います。接続数に上限を設けることにより設定以上のクライアントから要求を受け付けなくすることが出来ます。ただし接続クライアント数をぎりぎりに設定してしまうとゴーストセッションが残っている場合、接続できなくなる可能性がありますので、ゴーストセッションの対策をとった上である程度の余裕を持たせてください。ゴーストセッションの対策については運用編「トラブルシューティング(障害解析) クライアント接続数オーバへの対応」を参照してください。

設定方法

統合運用管理ツールの[TP システム]-[IIOP リスナ]-[上限設定]-[利用可能な同時接続クライアント数]

2.5.8. 1 セッションあたりの同時実行数

thin クライアント構成の場合、Web サーバから 1 つのセッションを通して多重にリクエスト要求が発行されます。クライアント数やトランザクション処理件数に応じた 1 セッションあたりの同時実行数を設ける必要があります。

Web サーバが複数台ある場合、多重度の設定は個々の Web サーバに対してそれぞれ適用されます。全ての Web サーバの中で、もっともリクエストの多重度が大きい値を設定してください。

設定方法

統合運用管理ツールの[TP システム]-[IIOP リスナ]-[クライアント制御]-[1 プロセス当たりの多重度]

2.5.9. 電文長の長いメッセージの送受信

サーバ側の送受信の上限は設定の有無にかかわらず 9,999,998 バイト(10M)となります。

また、クライアントが VB または C++の場合、電文の受信最大サイズは 8,388,608 バイト(8M)となります。クライアント側での受信電文長を 10M まで増やしたい場合はクライアントマシンのレジストリ変更が必要です。

設定方法

レジストリの名前:HKEY_LOCAL_MACHINE¥SOFTWARE¥NEC¥ObjectSpinner¥1¥MaxMessageSize

属性:文字列属性(REG_SZ)

値　:0～4294967295(単位:バイト)

意味:受信可能とする最大サイズを指定します。

補足:送信には関係ありません。

2.5.10. 接続要求最大保留数

クライアントから瞬時に大量の接続要求が来た場合、WebOTX 側での接続処理が間に合わずクライアントへエラーが返ってしまうことがあります。このような場合には本パラメータの値を大きくすることにより、接続要求をサーバ側で保留することができます。

設定方法

統合運用管理ツールの[TP システム]-[IIOP リスナ]-[クライアント制御]-[接続要求最大保留数]

2.5.11. メモリプールサイズ

クライアントからのリクエストはメモリプールとして事前に確保した共有メモリ上に受信されます。また、サーバからの応答もメモリプールに展開された後に送信を行います。これらの領域は、クライアントへの送信が完了した時点で解放されます。
このような管理方式にすることで、外因によらず迅速で確実なバッファ取得ができることになりますが、一方で、このサイズを適当な値に設定する必要があります。
バッファ使用区間はリクエストの受信から送信までですので、サーバ側でのオペレーションの同時実行数が上がるほどバッファが必要になります(キューイングされているものも含む)。また、リクエストや応答のメッセージサイズが大きいほどバッファが必要になります。

なお、本用途に使用されるバッファ領域はメモリプールサイズ×0.9 となります。

必要なメモリプールサイズの最大値は次の通り計算できます。

(メモリプールサイズの最大値) = (メモリブロック数) * (ブロックサイズ)
(メモリブロック数) = {もっとも大きい電文のサイズ/(ブロックサイズ) + 1} *
(WebOTX で同時に処理するオペレーション数)
(ブロックサイズ) = 4704 bytes

設定方法

統合運用管理ツールの[TP システム]-[システムパラメータ/動的情報]-[メモリプールサイズ]

3.Webサーバ(Apache)のチューニング

全 Editon における Web サーバ(Apache) のチューニングについて説明します。

3.1. リクエスト処理数

Web サーバ(Apache)は、親プロセスと複数の子プロセスで構成されます。ブラウザから受信したリクエストは、子プロセスに割り当てられて、リクエスト処理を実行します。

Apache のバージョンやプラットフォームにより、子プロセスでのリクエストの処理方法が異なります。

UNIX の場合かつ Apache1.3 の場合、リクエストを受ける度に子プロセスが起動し、1 つの子プロセスで 1 つのリクエストを処理します。子プロセスの数は受け付けたリクエスト数により増減します。

UNIX の場合かつ Apache2.0 の場合、マルチスレッド動作をする子プロセスが複数個起動し、子プロセス内の 1 スレッドで 1 つのリクエストを処理します。子プロセスの数は受け付けたリクエスト数により増減します。

Windows の場合、マルチスレッド動作をする子プロセスが1個起動し、子プロセス内の 1 スレッドで 1 つのリクエストを処理します。子プロセスの数は受け付けているリクエスト数に関係なく常に1固定です。

3.1.1. 最大同時リクエスト処理数

Web サーバが同時に処理できるリクエストの数を変更するには、最大同時リクエスト数の値を変更します。${INSTANCE_ROOT}/config/WebServer/httpd.conf を変更します。

UNIX

```
MaxClients 150
```

Windows

```
ThreadsPerChild 250
```

※本設定値に大きな値を設定すると、その分メモリ等のシステム資源が必要になりますので、値を変更する際には十分注意してください。

3.1.2. その他の設定項目

Web サーバのリクエスト処理数に関係するその他の設定項目は以下のとおりです。システム要件に合わせて各値を調整します。${INSTANCE_ROOT}/config/WebServer/httpd.conf を変更します。

UNIX: Apache1.3

指示子	説明	既定値
MaxClines	最大同時リクエスト処理数。同時に起動する最大子プロセス数。4096 まで指定可能。	150
StartServers	Web サーバ起動初期化時の子プロセス数。	5
MinSpareServers	アイドル状態での子プロセスの最小数。	5
MaxSpareServers	アイドル状態での子プロセスの最大数。	10

MaxRequestsPerChild	子プロセスが処理するリクエストの最大総数。本指定数のリクエストを受信すると子プロセスは終了します。0 を指定すると子プロセスは動作し続けます。	0

各指示子の詳細については、以下を参照してください。

http://httpd.apache.org/docs/1.3/server-wide.html#process

UNIX: Apache2.0

指示子	説明	既定値
MaxClients	最大同時リクエスト処理数。子プロセスで動作するスレッドの総数。本値を変更する場合には、ThreadsPerChild/ServerLimit/ThreadLimit の各値を調整します。	150
ThreadsPerChild	子プロセスで生成されるスレッド数。 ThreadLimit 以下の値を設定します。	25
StartServers	起動初期化時のプロセス数。	2
MinSpareThreads	アイドル状態でのスレッド総数の最小数。	25
MaxSpareThreads	アイドル状態でのスレッド総数の最大数。	75
MaxRequestsPerChild	子プロセスが処理するリクエストの最大総数。本指定数のリクエストを受信すると子プロセスは終了します。0 を指定すると子プロセスは動作し続けます。	0
ServerLimit	子プロセスの上限値。20000 まで指定可能。	16
ThreadLimit	子プロセスで動作するスレッド数の上限値。 15000 まで指定可能。	64

各指示子の詳細については、以下を参照してください。

http://httpd.apache.org/docs/2.0/mod/worker.html

Windows

指示子	説明	既定値
ThreadsPerChild	最大同時リクエスト処理数。子プロセスで動作するスレッドの総数。4096 まで指定可能。	50 (1.3) 250 (2.0)
MaxRequestsPerChild	子プロセスが処理するリクエストの最大総数。本指定数のリクエストを受信すると子プロセスは終了します。0 を指定すると子プロセスは動作し続けます。	0

各指示子の詳細については、以下を参照してください。

http://httpd.apache.org/docs/2.0/mod/mpm_winnt.html

4. Webコンテナのチューニング

全 Editon における Web アプリケーション実行時の Web コンテナのチューニングについて説明します。

4.1. プロセッサ数

Web コンテナには、Web ブラウザからのリクエストを処理するプロセッサが存在し、リクエストが来るとスレッドが割り当てられ、プロセッサがリクエスト処理を実行します。

同時に複数のリクエストが来た場合それだけプロセッサを消費しますが、その数を調整することで、同時に処理できるリクエストの数を制限したり、拡大したりすることができます。

4.1.1. 最大プロセッサ数

同時に処理できるリクエストの数を拡大するには、最大プロセッサ数の値を変更します。以下をチューニングの指針としてください。

【シングルプロセスモード】

[Web サーバの最大同時接続数] ≦ [Web サーバプラグインの最大同時接続数] ≦ [Web コンテナの最大プロセッサ数] + 5

※[Web コンテナの最大プロセッサ数] + 5 としているのは別途制御用スレッドが存在するためです。

Web コンテナの最大プロセッサ数を変更するには、otxadmin を使用して次のコマンドを実行します。

```
otxadmin> set   --user <ユーザ名> --password <パスワード>
                --host <ホスト名> --port <管理ポート>
                server.http-service.http-listener.<リスナ ID>.max-processors=<最大数>
```

get コマンドを利用すれば、現在の値を確認できます。

```
otxadmin> get server.http-service.http-listener.<リスナ ID>.*
```

なお、標準では以下のリスナ(リスナ ID)が定義されています。

- 外部 Web サーバと連携している場合

  ajp-listener-1

- 内蔵 Web サーバを利用している場合

  http-listener-1, http-listener-2

【マルチプロセスモード】

[Web サーバの最大同時接続数] ≦ [Web サーバプラグインの最大同時接続数] ≦ [Web コンテナの最大処理スレッド数 * プロセス数 + プロセスグループのキュー数]

Web コンテナの最大処理スレッド数を変更するには、運用管理コンソールの以下の設定値を変更しま

す。

[TP システム] – [アプリケーショングループ] – <アプリケーショングループ名> – [プロセスグループ] –<プロセスグループ名> の順にクリックしていき、[属性]タブ – [スレッド制御] – [スレッド数] の値を変更します。

Web コンテナのプロセス数を変更するには、運用管理コンソールの以下の設定値を変更します。

[TP システム] – [アプリケーショングループ] – <アプリケーショングループ名> – [プロセスグループ] –<プロセスグループ名> の順にクリックしていき、[属性]タブ – [プロセス制御] – [プロセス数] の値を変更します。

4.1.2. 最小プロセッサ数

最大プロセッサ数に応じて、最小プロセッサ数の値を変更します。この設定はシングルプロセスモード時のみ適用されます。

Web コンテナのスレッド数は同時リクエスト数に応じて最小プロセッサ数から最大プロセッサ数に向けて増加していきますが、 スレッド数の増加が追いつかない場合、接続バックログにキューイングされます。この接続バックログがあふれると、コネクションが切断され、エラーとなります。 よって、以下をチューニングの指針としてください。

（最大プロセッサ数 – 最小プロセッサ数） > 接続バックログ数

otxadmin を使用して次のコマンドを実行します。

```
otxadmin> set   --user <ユーザ名> --password <パスワード>
                --host <ホスト名> --port <管理ポート>
                server.http-service.http-listener.<リスナ ID>.max-processors=<最小数>
```

get コマンドを利用すれば、現在の値を確認できます。

```
otxadmin> get server.http-service.http-listener.<リスナ ID>.*
```

なお、標準では以下のリスナ（リスナ ID）が定義されています。

- 外部 Web サーバと連携している場合

  ajp-listener-1

- 内蔵 Web サーバを利用している場合

  http-listener-1, http-listener-2

4.2. プラグインのリクエスト処理数

ドメインを WebOTX Web Server などの外部の Web サーバと連携した場合、Web サーバとドメインを接続するプラグインというモジュールが存在します。

Web サーバやドメインでリクエストを処理するプロセッサ（スレッド）が調整可能であるように、プラグインでもリクエスト処理数を調整することができます。

4.2.1. 最大同時リクエスト処理数

同時に処理できるリクエストの数を拡大するには、cachesize の値を変更します。

cachesize は、ドメインディレクトリの config/WebCont 配下の workers.properties（シングルプロセスモード時） ior_workers.properties（マルチプロセスモード時） に定義します。このファイルをエディタで開いて編集してください。

具体的には、次のようになります。"worker.ajp13"（シングルプロセスモード時） “worker.otxiiop”（マルチプロセスモード時） については通常固定と考えてください（プラグインの負荷分散機能を利用した場合は可変となります）。デフォルト値は 150 になっています。

worker.ajp13.cachesize=150 （シングルプロセスモード時）

worker.otxiiop.cachesize=150 （マルチプロセスモード時）

値を反映するには Web サーバを再起動する必要があります。

また、マルチプロセスモード時は追加で以下の設定が必要になります。

運用管理コンソールより、[TP システム] – [IIOP リスナ] – [クライアント制御] – [1 プロセスあたりの多重度] に cache_size と同じ値を設定します。

5. JMSのチューニング

JMS のチューニングについて説明します。ここでは、主に、メッセージの滞留に関するチューニングポイントについて説明しています。

以降の説明で、「設定方法」には、設定対象となる統合運用管理ツール上の項目名や、MOの属性名を記述しています。設定方法の詳細については、マニュアル「運用編(JMSの運用操作)」-「JMSの運用操作」を参照してください。

5.1. JMSサーバの設定

JMS サーバの設定について説明します。

5.1.1. JVMヒープサイズの設定

JMS サーバにおける JVM ヒープの最大サイズは、デフォルトで 192M バイトに設定されています。通常、大量のメッセージ送受信で JMS サーバに負荷がある場合は、JVM ヒープサイズを大きくする必要があります。

JMS サーバの JVM ヒープの消費は、主に JMS サーバ内に滞留するメッセージの数とサイズに比例します。システムの運用上想定している数のメッセージが滞留した場合に、どの程度のメモリが消費されるかを確認して十分なヒープサイズを設定することが必要になります。また、逆にシステムのリソースに合わせて、後述する設定を行うことにより滞留自体を抑止することも有効です。

確認方法

JVM ヒープ内のその時点での総メモリ量と、使用可能な空きメモリ量は、JMS サーバに関する統計情報から確認できます。

統合運用管理ツールの場合:

総メモリ量 [WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[統計情報]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]-[MemoryTotal]
空きメモリ量 [WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[統計情報]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]-[MemoryFree]

運用管理コマンドの場合:

総メモリ量 otxadmin> get --monitor=true server.jms-service.MemoryTotal-Count
空きメモリ量 otxadmin> get --monitor=true server.jms-service.MemoryFree-Count

統計情報取得に必要な設定などの詳細については、マニュアル「運用編(JMS の運用操作)」-「JMS の運用操作」-「JMS サービスの統計情報取得」を参照してください。

設定方法

必要な値を設定し、JMSサーバを再起動してください。

統合運用管理ツールで設定する場合:

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]-[起動属性]-[JavaVM 引数]

運用管理コマンドで設定する場合:

otxadmin> set server.jms-service.vmargs=<JavaVM 引数>

例)

統合運用管理ツールでの指定 : -Xms128 -Xmx384
運用管理コマンド : otxadmin> set server.jms-service.vmargs="-Xms128 -Xmx384"

5.1.2. JMS起動タイムアウトの設定

JDBC ストア利用時に、送信先にメッセージが大量に蓄積されていると、JMS サーバの起動に時間がかかり、JMS サーバの起動に失敗する場合があります。

この場合、ドメイン起動における JMS サーバ起動完了待ち時間を長くする必要があります。

JMS サーバ起動完了待ち時間のデフォルト値は、60(秒)です。

確認方法

JMS サーバの起動に要する時間は、環境によって異なりますが、参考となる値は ${INSTANCE_ROOT}/logs/wojms/wojmsserver.log から調べることができます。

ログレベルがデフォルトの INFO である場合、

```
[2005-07-26 10:06:59,159]
================================================================
WebOTX JMS
NEC Corporation.
================================================================
```

が起動開始を示し、

```
[2005-07-26 10:07:00,291] [B1039]: Broker "wojmsbroker@xxx:9700" ready.
```

が起動完了を示しますので、それぞれの先頭で出力されているタイムスタンプから、起動に必要となる時間を算出してください。

設定方法

設定した値は、次回のドメイン起動から有効になります。

統合運用管理ツールで設定する場合:

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]-[起動属性]-[起動完了待ち時間]

運用管理コマンドで設定する場合:

otxadmin> set server.jms-service.init-timeout-in-seconds=<起動完了待ち時間>

5.2. 送信先の設定

物理的な送信先の設定について説明します。

5.2.1. プロデューサ最大数の設定

JMS サーバに必要以上にメッセージを滞留させないためには、メッセージを生成するプロデューサ数を制限することも有効な方法です。

プロデューサ最大数のデフォルト値は -1(無制限)です。

確認方法

送信先の現在のプロデューサ数を確認するには、次の JMS のコマンドを利用します。

wojmscmd query dst -u admin -p adminadmin -t <送信先タイプ> -n <送信先名>

コマンドの詳細については、「運用管理コマンドリファレンスマニュアル」-「JMS」-「wojmscmd query dst」を参照してください。

設定方法

統合運用管理ツールで設定する場合:

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]-[送信先名]-[<送信先名>]-[拡張]-[プロデューサの最大数]

運用管理コマンドで設定する場合:

otxadmin> set server.jms-service.jms-physical-destitination.<送信先名>.maxNumProducers=<プロデューサの最大数>

5.2.2. アクティブコンシューマ数の設定

複数のキューコンシューマが送信先のメッセージを処理する効率は、ここで説明するアクティブコンシューマ数と、JMS サーバが一度の処理でコンシューマに配信するメッセージ数(「メッセージ滞留抑止の設定」参照)によって左右されます。

効率よくメッセージを処理するためには、十分な数のアクティブコンシューマがメッセージの配信に遅れずに対応する必要があります。メッセージがキューに蓄積している場合、メッセージを処理するアクティブコンシューマ数が不十分であることが考えられます。

確認方法

送信先に対する現在のアクティブコンシューマ数は、次の操作で確認できます。

統合運用管理ツールの場合:

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]- [送信先名]-[<送信先名>]-情報取得

運用管理コマンドの場合:

otxadmin> get-jmsdest-info <送信先名>

上記の操作で表示される、「Current Number of Active Consumers」の値が現在のアクティブコンシューマ数です。

操作の詳細については、マニュアル「運用編(JMS の運用操作)」-「JMS の運用操作」-「送信先の操作」を参照してください。

設定方法

統合運用管理ツールで設定する場合:

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]-[送信先名]-[<送信先名>]-[拡張]-[アクティブコンシューマの最大数]

運用管理コマンドで設定する場合:

otxadmin> set server.jms-service.jms-physical-destitination.<送信先名>.maxNumActiveConsumers=<アクティブコンシューマの最大数>

アクティブコンシューマ数のデフォルト値は、-1(無制限)です。

5.2.3. コンシューマへのフロー制限数の設定

JMS サーバが一度の処理でコンシューマに配信するメッセージ数が大きすぎる場合、メッセージがコンシューマ上で滞留する可能性があります。

たとえば、コンシューマへのフロー制限数が大きすぎると、ある一つのコンシューマがキュー内のすべてのメッセージを受信し、そのほかのコンシューマは何も受信していないことがあります。コンシューマの処理が高速であれば、これは問題になりませんが、コンシューマでの処理に時間がかかるような場合、メッセージを各コンシューマに均等に分散させるために、コンシューマへのフロー制限数を小さくする必要があります。

コンシューマでの滞留もメモリ消費につながるため、クライアントの JVM ヒープサイズと合わせてフロー制限数を調整する必要があります。

設定方法

統合運用管理ツールで設定する場合:

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]-[送信先名]-[<送信先名>]-[拡張]-[コンシューマへのフロー制限数]

運用管理コマンドで設定する場合:

otxadmin> set server.jms-service.jms-physical-destitination.<送信先名>.consumerFlowLimitNum=<コンシューマへのフロー制限数>

ただし、各コンシューマが使用するコネクションファクトリで指定された制限値の方が小さい場合は、その値のほうが優先されます。

コンシューマへのフロー制限数のデフォルト値は 1000 です。

5.2.4. メッセージ滞留抑止の設定

送信先のメッセージ滞留を抑止するために、送信先単位、または、JMS サーバ全体でメッセージ数とメッセージの合計サイズを設定できます。送信先のメッセージ数、または、メッセージのバイト数が設定された制限に達した場合の動作は、次のものから選択することができます。

REMOVE_OLDEST ： もっとも古いメッセージを削除する
REMOVE_LOW_PRIORITY ： メッセージの有効期間に従いもっとも優先度の低いメッセージを削除する
FLOW_CONTROL ： プロデューサとの間でフロー制御が行われ、プロデューサがブロックされる
REJECT_NEWEST ： 新しいメッセージを拒否する。永続メッセージの場合はメッセージ拒否の例外をスローする

メッセージの最大数、メッセージの最大合計サイズは、送信先、JMS サーバとも、-1(無制限)です。
送信先の制限到達時の振る舞いのデフォルト値は REJECT_NEWEST です。

確認方法

送信先に設定されている制限到達時の振る舞いは、以下で確認できます。

統合運用管理ツールの場合：
[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]-[送信先名]-[<送信先名>]-[拡張]-[制限到達時の振る舞い]

運用管理コマンドの場合：
otxadmin> get server.jms-service.jms-physical-destitination.<送信先名>.limitBehavior

送信先、または、JMS サーバ全体の現在のメッセージ数とメッセージのバイト数を監視するには、統合運用管理ツールから参照できる統計情報を利用します。

送信先の場合は、[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[統計情報]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]- [送信先名]-[<送信先名>]、JMS サーバ全体の場合は、[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[統計情報]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]の「統計情報採取開始」を実行します。

送信先
メッセージ数 ：MessagesNow
メッセージのバイト数 ： MessageBytesNow

JMS サーバ全体
メッセージ数 ： MessagesNow
メッセージのバイト数 ： MessageBytesNow

【注意】
JMSサーバ全体のデータには、運用管理操作(統計情報の採取も含む)での情報も含まれており、JMSメッセージだけのメッセージ数やバイト数というわけではありません。

統計情報取得に必要な設定などの詳細については、マニュアル「運用編(JMS の運用操作)」-「JMS の運用操作」-「JMS サービスの統計情報取得」を参照してください。

設定方法

送信先ごとのメッセージ滞留抑止に関する値は、以下で設定します。

統合運用管理ツールで設定する場合：

メッセージの最大数
[WebOTX管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMSサービス]-[送信先名]-[<送信先名>]-[拡張]-[メッセージの最大数]

メッセージの最大サイズ
[WebOTX管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMSサービス]-[送信先名]-[<送信先名>]-[拡張]-[メッセージの最大合計サイズ]

制限到達時の振る舞い
[WebOTX管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMSサービス]-[送信先名]-[<送信先名>]-[拡張]-[制限到達時の振る舞い]

運用管理コマンドで設定する場合：

メッセージの最大数
otxadmin> set server.jms-service.jms-physical-destitination.<送信先名>.maxNumMsgs=<メッセージの最大数>

メッセージの最大サイズ
otxadmin> set server.jms-service.jms-physical-destitination.<送信先名>.maxTotalMsgBytes=<メッセージの最大合計サイズ>

制限到達時の振る舞い
otxadmin> set server.jms-service.jms-physical-destitination.<送信先名>.limitBehavior=<制限到達時の振る舞い>

JMS サーバ全体のメッセージ滞留抑止に関する値は、以下で設定します。

統合運用管理ツールで設定する場合:

メッセージの最大数
[WebOTX管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMSサービス]-[JMSサーバ情報]-[JMSサーバ内メッセージ最大数]

メッセージの最大サイズ
[WebOTX管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMSサービス]-[JMSサーバ情報]-[JMSサーバ内メッセージ最大合計サイズ]

運用管理コマンドで設定する場合:

メッセージの最大数
otxadmin> set server.jms-service.systemMaxCount=<JMSサーバ内メッセージ最大数>

メッセージの最大サイズ
otxadmin> set server.jms-service.systemMaxSize=<JMSサーバ内メッセージ最大合計サイズ>

5.3. コネクションの設定

コネクション関連の設定について説明します。

5.3.1. サーバクライアント間の相互監視の設定

WebOTX JMS は、サーバとクライアント間の相互通信に TCP/IP プロトコルを利用していますが、データ受信を主体としている場合、ネットワークの異常を検知できないような事象(簡単な例だと、相手側のケーブルが抜けたような場合)が発生する場合があり、次のような問題を引き起こす可能性があります。

- JMS サーバが、クライアントの異常を検知できずに、クライアントに関するリソースがパージできない。
- JMS クライアントが、サーバの異常を検知できずに、メッセージ配信を待ちつづけてしまい、JMS サーバのクラスタ切替に対応できない。

このような問題を防ぐために、クライアントアプリケーションのライブラリレベルで、JMS サーバとの接続を定期的にチェックする機能を提供しています。

この機能はデフォルトで動作しますが、監視間隔は任意の値に変更することができます。

設定方法(JMSサーバ)

JMS サーバに対する設定は、次のプロパティが対象になります。

wojms.ping.enabled

wojms.ping.interval

起動引数として設定する方法と、config.properties ファイルに設定する方法とがあります。設定方法や設定値の詳細については、「運用編(コンフィグレーション)」-「JMS の設定」をご覧ください。

起動引数

統合運用管理ツールの場合:
[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サーバ]-[起動属性]-[起動引数]で以下の設定を行ってください。

-Dwojms.ping.enabled=true -Dwojms.ping.interval=120

運用管理コマンドの場合:

set コマンドで以下の設定を行ってください。

```
otxadmin> set server.jms-service.start-args=" -Dwojms.ping.enabled=true
    -Dwojms.ping.interval=120"
```

config.properties

ファイルに以下の設定を行ってください。

```
wojms.ping.enabled=true

wojms.ping.interval=120
```

設定方法（JMSクライアント）

コネクションファクトリリソース

統合運用管理ツールの場合：

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[JMS リソース]-[コネクションファクトリリソース]-[<コネクションファクトリ名>]-[接続]-[接続監視の間隔]

運用管理コマンドの場合：

```
otxadmin> set server.resources.jms-resource.jms-connection-factory.<コネクションファクトリ名
>.wojmsPingInterval=<接続監視の間隔>
```

5.3.2. JMSサーバからの受諾通知タイムアウト時間の設定

プロデューサからのメッセージ送信に対して、JMS サーバが受諾通知を返すようになっているときに、JMS サーバへの負荷が高い場合や、ネットワークの通信速度が遅いような場合、プロデューサが受諾通知を受け取るまでに時間がかかることがあります。このような場合、デフォルトでは、JMS サーバから受諾通知が届くまで、プロデューサの呼び出しスレッドはブロックするようになっています。

この、受諾通知を待ち合わせる時間は、コネクションファクトリの wojmsAckTimeout(通知タイムアウト)で決めることができます。デフォルトでは、タイムアウトせず、必ず待ちあわせる設定になっています。

受諾通知を待ち続けるのではなく、決められた時間以内に届かないときにはエラーを発生させたい場合、この値を設定してください。

デフォルトでは、JMS サーバは、プロデューサから受け付けた永続メッセージに対してのみ受諾通知を返すようになっています。

確認方法

コネクションファクトリの現在の通知タイムアウト値は、以下で確認できます。

コネクションファクトリリソース

統合運用管理ツールの場合：

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[JMS リソース]-[コネクションファクトリリソース]-[<コネクションファクトリ名>]-[フロー制御]-[通知タイムアウト]

運用管理コマンドの場合：

```
otxadmin> get server.resources.jms-resource.jms-connection-factory.<コネクションファクトリ名
>.wojmsAckTimeout
```

設定方法

コネクションファクトリリソース

統合運用管理ツールの場合：

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[JMS リソース]-[コネクションファクトリリソース]-[<コネクションファクトリ名>]-[フロー制御]-[通知タイムアウト]

運用管理コマンドの場合：

```
otxadmin> set server.resources.jms-resource.jms-connection-factory.<コネクションファクトリ名
>.wojmsAckTimeout=<通知タイムアウト>
```

5.4. JMSクライアントのチューニング情報採取

JMS クライアントについてのチューニング情報採取について説明します。ここで採取した情報から、クライアント側にメッセージが滞留することによるメモリ負荷の状況を把握することができます。

まず、JMSサーバ側で提供する情報によって、どのJMSクライアントに問題がありそうかを把握し（「5.4.1JMSサーバ側での情報採取」）、その後、問題のありそうなJMSクライアント側で情報を採取する（「5.4.2JMSクライアント側での情報採取」）という手順になります。

5.4.1. JMSサーバ側での情報採取

JMS サーバ側ではコネクション単位にメモリ情報を採取し、コネクション一覧で採取した情報を表示します。

設定方法

統合運用管理ツールで設定する場合：

以下の属性をチェックします。
[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]-[JMS サーバ情報]-[JMS クライアントメモリ情報採取]

運用管理コマンドで設定する場合：

```
otxadmin> set server.jms-service.enableClientMetrics=true
```

採取情報

コネクション単位に次の情報を採取します。

項目	説明
Max Memory	JavaVM の最大メモリ量(バイト数)。
Current Memory	JavaVM の総メモリ量(バイト数)
Peak Memory	JavaVM のメモリ最大使用量(バイト数)

確認方法

採取された情報は、以下の操作で確認できます。

統合運用管理ツールで確認する場合：

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[JMS サービス]-<コネクション一覧の取得>

運用管理コマンドで確認する場合：

```
otxadmin> list-jmsdest-connections
```

確認の結果、「Peak Memory」が、「Max Memory」に近いものがあれば、さらにそのクライアントに対して次項で説明しているログを採取し、コネクションファクトリのフロー制御に関する設定値や、メモリサイズを調整します。

5.4.2. JMSクライアント側での情報採取

JMS クライアント実行時に Java システムプロパティを設定することにより、メモリ状況や、フロー制御に関する項目をログ出力することが可能です。

コネクションファクトリの wojmsConsumerFlowLinit(滞留可能なメッセージ数の制限値)や、wojmsConsumerFlowThreshold(配信を再開するポイント)、JMS クライアントのメモリサイズの決定には、ここで採取できる情報も参考になります。

設定方法

JMS クライアントの Java システムプロパティとして、以下の値を指定します。

プロパティ	設定値
wojms.client.metrics	true
wojms.client.metrics.interval	例） 10000 省略可能。デフォルト値は、5000（ミリ秒）。
wojms.client.metrics.log	例） C:¥client.log 省略可能。ログファイル名を指定しない場合は、標準出力に表示。

プロパティの詳細については、「運用編（コンフィグレーション）」-「JMS の設定」-「JMS 設定項目一覧」-「JMS のプロパティ／属性一覧」-「JMS クライアントのプロパティ」をご覧ください。

採取情報

ログに出力する情報は次のとおりです。

- メモリ状況に関する項目

項目	説明
TotalMemory	JavaVM の総メモリ量(バイト数)
FreeMemory	JavaVM の空きメモリ量(バイト数)
PeakMemory	JavaVM の最大使用メモリ量(バイト数)

- コンシューマレベルのフロー制御に関する項目

項目	説明
Count of unprocessing	コンシューマで未消費のメッセージ数
Peak count of unprocessing	コンシューマ起動後の未消費メッセージ数の最大値
Pause count	停止した回数
Resume count	配信再開要求(RESUME_FLOW)を行った回数
Last resume count	最後の配信再開要求(RESUME_FLOW)で、JMS サーバに要求したメッセージ数

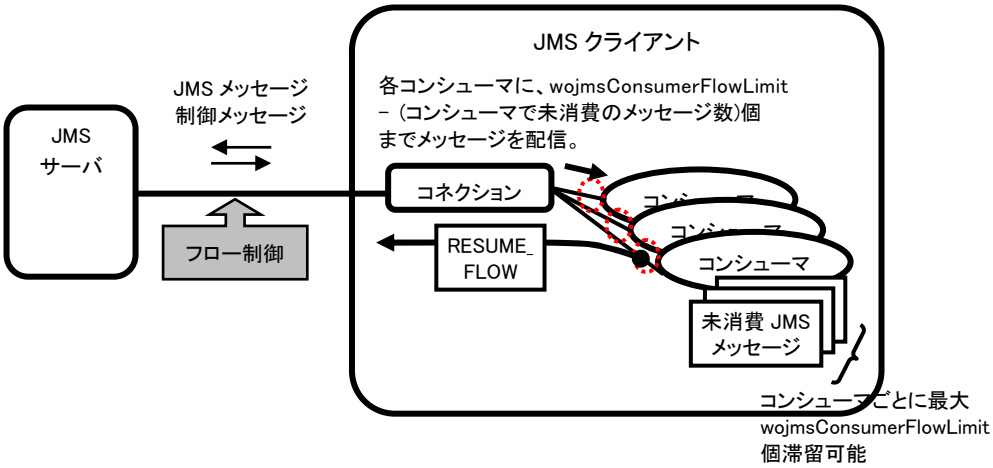
- コネクションレベルのフロー制御に関する項目

項目	説明
Count of unprocessing	コネクションで未消費のメッセージ数
Peak count of unprocessing	クライアントが起動後の未消費メッセージ数の最大値
Pause count	停止した回数
Resume count	配信再開要求(RESUME_FLOW)を行った回数

コンシューマレベルのフロー制御について

コンシューマレベルでは、コンシューマ単位に滞留可能なメッセージ数の制限値(wojmsConsumerFlowLimit)と、配信を再開するポイント(wojmsConsumerFlowThreshold)をコネクションファクトリに設定してフロー制御を行っています。

【備考】
wojmsConsumerFlowLimit のデフォルト値は1000、wojmsConsumerFlowThreshold のデフォルト値は、50(%)です。



クライアントへ送信された JMS メッセージ数が、wojmsConsumerFlowLimit に達すると、JMS サーバはメッセージの配信を停止し、未消費メッセージ数が wojmsConsumerFlowThreshold(%)を下回ると配信を再開します。配信再開後、JMS サーバから送信されるメッセージ数は、「wojmsConsumerFlowLimit －（コンシューマで未消費のメッセージ数)」となります。

1 つのコンシューマでは、最大、「Peak count of unprocessing」×（1 メッセージのサイズ)のメモリが滞留メッセージによって消費されることになります。「Peak count of unprocessing」の値を参考に、コネクションファクトリの wojmsConsumerFlowLimit の値や、コンシューマ数、メモリサイズを調整します。

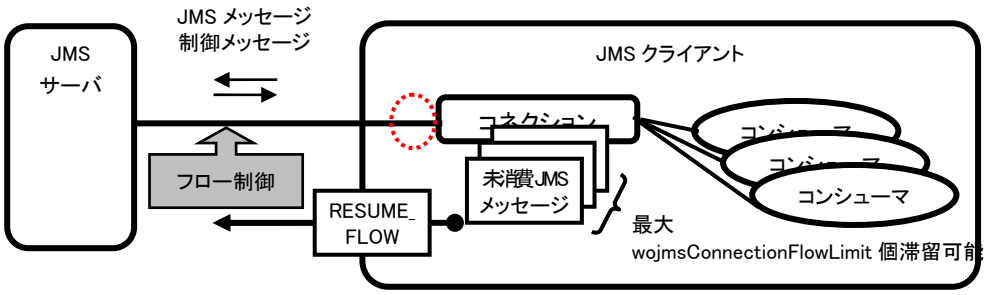
また、「Pause count」は、wojmsConsumerFlowLimit の値に達して停止した回数を示し、「Resume count」は、wojmsConsumerFlowThreshold を下回って、配信を再開する要求を行った回数になります。受信したメッセージ数と比較して、「Pause count」や、「Resume count」の値が大きい場合は、wojmsConsumerFlowLimit が小さすぎるか、wojmsConsumerFlowThreshold が大きすぎて、スループットを低下させている可能性があるため、これらの値を調整する目安とします。

【備考】
wojmsConnectionFlow
Limit のデフォルト値は
1000 です。

コネクションレベルのフロー制御について

コネクションレベルでは、同一コネクション上のすべてのコンシューマで滞留可能なメッセージ数の制限値(wojmsConnectionFlowLimit)をコネクションファクトリに設定してフロー制御を行います。ただし、このフロー制御は、wojmsConnectionFlowLimitEnabled が true のとき行われ、デフォルトでは制御を行わない(false)設定になっています。

コンシューマ数が決まっていないような場合は、コネクションレベルでの制御が有効です。



コネクション上のすべてのコンシューマの未消費メッセージが wojmsConnectionFlowLimit を超えると、合計数がコネクションの制限を下回るまで、コネクション経由のメッセージ配信を停止します。

コネクションレベルで制御を行っている場合は、1 つのコネクションについて、最大、「Peak count of unprocessing」×（1 メッセージのサイズ)のメモリが滞留メッセージによって消費されることになります。「Peak count of unprocessing」の値を参考に、メモリサイズの調整を行ってください。

6. Transactionサービスのチューニング

Transaction サービスのチューニングについて説明します。

以降の説明で、「設定方法」には、設定対象となる統合運用管理ツール上の項目名や、MOの属性名を記述しています。設定方法の詳細については、マニュアル「運用編(コンフィグレーション)」-「11.Transactionサービスに関する設定」を参照してください。

6.1. トランザクションタイムアウト時間について

ここではトランザクションタイムアウト時間と ObjectBroker のリクエスト呼び出しのタイムアウト時間の関係について、トランザクション実行中にサーバアプリケーションを呼び出さない場合とトランザクション実行中にサーバアプリケーションを呼び出す場合で分けて説明します。

なお、データベース等のリソースのタイムアウト設定はトランザクションタイムアウト時間を元に Transaction サービスで適宜変更しているので調整の必要はありません。

「トランザクションタイムアウト時間」

トランザクションを開始してから一定時間が経過してもコミットやロールバックなどの完了処理が発行されない場合に、自動的にトランザクションをロールバックするまでの時間です。Transaction サービスの機能であり、統合運用管理ツールなどで設定することができます。

「ObjectBroker のリクエスト呼び出しのタイムアウト時間」

クライアントアプリケーションからサーバアプリケーションに対して処理要求を行う際に、ObjectBroker が基盤となって橋渡しを行います。クライアントアプリケーションでは、この処理要求が一定時間経過しても戻ってこない場合に要求が失敗したとみなしてエラーとする機能を提供しています。この一定時間のことを「ObjectBroker のリクエストタイムアウト時間」と呼び、統合運用管理ツールなどで設定することができます。
詳細は次のマニュアルを参照してください。
「運用編(コンフィグレーション)」-「12.ObjectBroker.に関する設定」

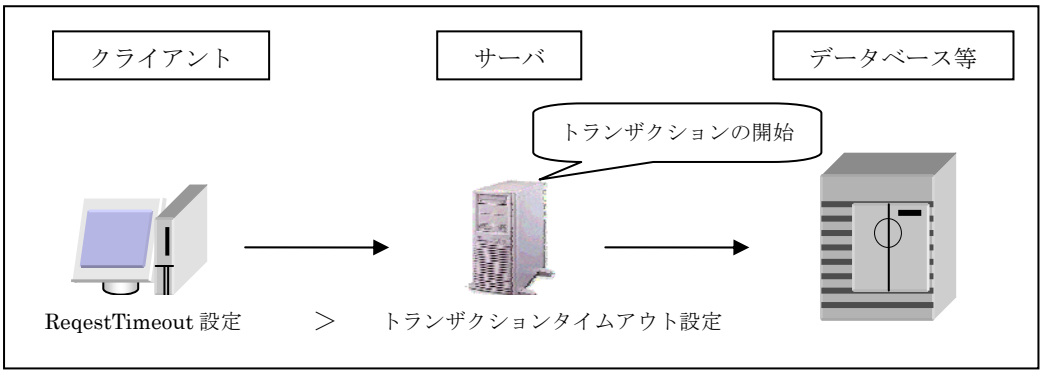
・ トランザクション実行中にサーバアプリケーションを呼び出さない場合

トランザクション実行中にサーバアプリケーションを呼び出さない場合とは、サーバアプリケーションでトランザクションを開始し、他のプロセス上のサーバアプリケーションを呼び出すことなくトランザクションが完結する場合のことです。自プロセス上のサーバアプリケーションの呼び出しもこの場合に該当します。

この場合は、クライアントで設定するサーバアプリケーションを呼び出す際の通信のタイムアウト時間より短い値をトランザクションタイムアウト時間に設定します。

これはクライアントからサーバへの呼び出しの中でトランザクションを実行するので、トランザクションのタイムアウトが発生した時点で通信のタイムアウトにより通信が切断されているとクライアントにトランザクションの成否を通知する手段が無くなるためです。

具体的にはクライアントで設定される ObjectBroker のリクエスト呼び出しのタイムアウト時間
(RequestTimeout: 既定値 30 秒)より短い値をサーバのトランザクションタイムアウト時間に設定します。



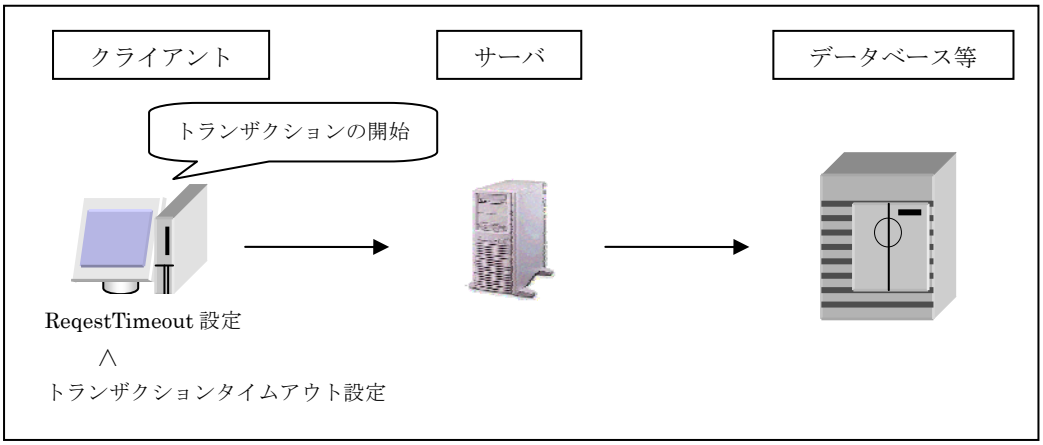
・　トランザクション実行中にサーバアプリケーションを呼び出す場合

トランザクション実行中にサーバアプリケーションを呼び出す場合とは、クライアントアプリケーションでトランザクションを開始した後、サーバアプリケーションを呼び出し実行する場合、もしくはサーバアプリケーションでトランザクションを開始した後、他のプロセス上のサーバアプリケーションを呼び出し実行する場合のことです。

この場合は、クライアントからサーバアプリケーションを呼び出す際の通信のタイムアウト時間より長い値をトランザクションタイムアウト時間に設定します。

これはトランザクションの開始と終了の間でサーバの呼び出しが行われるため、いくら通信のタイムアウト時間を長くしてもトランザクションタイムアウト時間が経過した時点でトランザクションはロールバックしてしまうためです。

具体的にはクライアントで設定される ObjectBroker のリクエスト呼び出しのタイムアウト時間(RequestTimeout: 既定値 30 秒)より長い値をクライアントまたはサーバ(トランザクションを開始する方)のトランザクションタイムアウト時間に設定します。



設定方法

サーバアプリケーションでトランザクションを開始し、統合運用管理ツールで設定する場合：

[WebOTX[<ホスト名>]]-[<ドメイン名>]-[server]-[transactionservice]-[TM 設定]-[トランザクションタイムアウト時間]

サーバアプリケーションでトランザクションを開始し、運用管理コマンドで設定する場合：

otxadmin> set server.transactionservice.tx-timeout=600(秒)

クライアントアプリケーションでトランザクションを開始する場合：

Java クライアントでは以下の形式でシステムプロパティを指定します。

-DTxTimeout=<トランザクションタイムアウト時間(秒)>

6.2. トランザクション高速化設定

DBコネクション事前生成機能無効化

EJB を利用した場合、トランザクションの開始(begin())の前に取得した DB コネクションをそのトランザクションに参加させることができます。これは DB コネクションの管理を行なっているために可能となっています。ただしこの機能を利用することが無い場合、DB コネクションの管理を行なう必要がなく、トランザクションの高速化させることが可能です。

具体的には以下のようなケースで高速化させることが可能です。

- トランザクションの開始の前に取得した DB コネクションをそのトランザクションに参加させることはなく、DB コネクションの取得、及び、解放はトランザクションの開始と終了の間で行なわれている。

```
// DB コネクション事前生成機能利用(高速化不可)
UserTransaction utx = …;
DataSource ds = …;

Connection con = ds.getConnection();
utx.begin();
// コネクション(con)を利用した DataBase 更新をここで行なう
utx.commit();
con.close();
```

```
// DB コネクション事前生成機能未使用(高速化可能)
UserTransaction utx = …;
DataSource ds = …;

utx.begin();
Connection con = ds.getConnection();
// コネクション(con)を利用した DataBase 更新をここで行なう
con.close();
utx.commit();
```

- 他の Bean を呼び出す前には DB コネクションを解放し、DB コネクションの引継ぎを利用しない。

```
// DB コネクションの引継ぎを利用(高速化不可)
DataSource ds = …;

Connection con = ds.getConnection();
// コネクション(con)を利用した DataBase 更新をここで行なう
// 別 Bean の呼び出し
// コネクション(con)を利用した DataBase 更新をここで行なう
con.close();
```

```
// DB コネクションの引継ぎを利用しない(高速化可能)
DataSource ds = …;

Connection con = ds.getConnection();
```

```
// コネクション(con)を利用した DataBase 更新をここで行なう
con.close();
// 別 Bean の呼び出し
Connection con = ds.getConnection();
// コネクション(con)を利用した DataBase 更新をここで行なう
con.close();
```

トランザクションの高速化は、${INSTANCE_ROOT}/config/TS/jta.conf に以下を追加したあとドメイン再起動することにより行なうことができます。

wojta.enlistPrecreatedConnection=false

7.Webサービスのチューニング

Web サービスのチューニングについて説明します。

7.1. SOAP通信高速化設定

JAX-PRC を利用した Web サービスでは、高速な XML パーサを利用することができます。高速な XML パーサを有効にするには Web サービスを実行するドメインの起動オプションに設定を追加します。この設定により、SOAP メッセージの xml 階層が深いパターン等で高速化の効果が期待できますが、最終的には必ずご自身の作成したアプリケーションを用いてこの設定の有無を変えてパフォーマンスを測定し、安定して高いパフォーマンスが得られる設定を採用してください。なお、この設定を行うと UTF-8 の SOAP メッセージしか扱うことができないなどの制限事項 がありますので、注意してください。制限事項の詳細については、マニュアル「注意制限事項」 - 「4.1. 相互接続性に関する制限事項」を参照してください。

MEMO

SOAP 通信高速化オプションを設定しない場合、UTF-8、UTF-16 を扱うことができます。

[Web Edition/Standard-J Edition/Standard Edition 以上かつシングルプロセスモードの場合]

SOAP 通信高速化設定は、運用管理コマンド(otxadmin)を使い、次のように設定します。

otxadmin> login --user admin --password adminadmin --port 6212 （これはログインの例です）

otxadmin> create-jvm-options -Dwoparser=true

※ 設定を有効にするにはドメインの再起動が必要です。

この設定を無効化するには、運用管理コマンド(otxadmin)を使い、次のように設定します。

otxadmin> login --user admin --password adminadmin --port 6212 （これはログインの例です）

otxadmin> delete-jvm-options -Dwoparser=true

※ 設定を有効にするにはドメインの再起動が必要です。

[Standard Edition 以上かつマルチプロセスモードの場合]

SOAP 通信高速化設定は、運用管理コマンド(otxadmin)を使い、次のように設定します。

otxadmin> login --user admin --password adminadmin --port 6212 （これはログインの例です）

otxadmin> stop-apg [アプリケーショングループ名]

otxadmin> add-pg-javasystem-property --apgroup [アプリケーショングループ名] --javasystemprop woparser --value true [プロセスグループ名]

otxadmin> start-apg [アプリケーショングループ名]

この設定を無効化するには、運用管理コマンド(otxadmin)を使い、次のように設定します。

otxadmin> login --user admin --password adminadmin --port 6212 （これはログインの例です）

otxadmin> stop-apg [アプリケーショングループ名]

otxadmin> delete-pg-javasystem-property --apgroup [アプリケーショングループ名] --javasystemprop woparser [プロセスグループ名]

otxadmin> start-apg [アプリケーショングループ名]

8. JDBCデータソースのチューニング

JDBC データソースのチューニングについて説明します。ここでは、主に、コネクションプール数に関するチューニングポイントについて説明しています。

以降の説明で、「設定方法」には、設定対象となる統合運用管理ツール上の項目名や、MOの属性名を記述しています。設定方法の詳細については、マニュアル「運用編(コンフィグレーション)」 - 「9. JDBC データソースに関する設定」を参照してください。

8.1. コネクションプール数の設定

コネクションプール数の設定について説明します。

8.1.1. 最小プールサイズの設定

アプリケーションの性能に大きな影響があるため、最小プールサイズを適切にチューニングすることが重要です。

最小プールサイズ[minPoolSize]にはデフォルトで 4 が設定されています。最小プールサイズに、アプリケーションの動作スレッド数と同じ値を設定すると、最も高い性能を得ることができますが、反面、多くのコネクションやメモリなどの資源が必要となります。このため、一般的には、データベースサーバ側の接続数の制限に合わせてコネクションプール数のチューニングを行います。

最小プールサイズには、システム内で使用される全コネクション数がデータベースサーバ側の接続数の制限を越えない範囲で、アプリケーションの動作スレッド数か、それより小さい値を設定してください。コネクションプールは、アプリケーションプロセス毎に、JDBC データソースの定義数分存在します。それらのコネクションプールの最小プール数を合計し、運用管理操作で必要となるコネクション数などを考慮した上で、最小プールサイズ、または、データベースサーバ側の接続数の制限値を調整してください。

最小プールサイズを超えて払い出されたコネクションは、アプリケーションが使い終わった後で破棄されます。その際、アプリケーションが使い終わった直後にコネクションが破棄されると、負荷が高い場合に、コネクションの接続および切断を繰り返すことで性能が極端に低下することが懸念されます。そのための対策として、コネクション解放の待ち合わせ時間[shrinkDelayTime]のデフォルト値が 15 秒に設定されています。それでもなお、度々コネクションの接続および切断が繰り返される状況になる場合は、コネクション解放の待ち合わせ時間を長くすることで性能を改善できます。

確認方法

アプリケーションの動作スレッド数は、最小プールサイズに設定する最大値です。その値と、負荷をかけた際の次の統計情報(使用中コネクションの最大数)の値を照らし合わせて、最終的に最小プールサイズに設定すべき値を決定してください。

統合運用管理ツールで確認する場合:

エージェントで動作する場合

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[統計情報]-[<ドメイン名>]-[アプリケーションサーバ]-[リソース]-[jdbc-datasource.JDBC データソース名]-[プール中のコネクション数]-[NumConnUsed]-[使用中コネクション数の最大値]

Standard/Enterprise Editionのプロセスグループで動作する場合

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[統計情報]-[<ドメイン名>]-[TP システム]-[アプリケーショングループ]-[アプリケーショングループ名]-[プロセスグループ]-[プロセスグループ名]-[process]-[プロセス ID]-[リソース]-[jdbc-datasource.JDBC データソース名]-[プール中のコネクション数]-[NumConnUsed]-[使用中コネクション数の最大値]

運用管理コマンドで確認する場合:

エージェントで動作する場合

```
otxadmin> get --monitor *.*.jdbc-datasource*.NumConnUsed-HighWaterMark
```

Standard/Enterprise Editionのプロセスグループで動作する場合
otxadmin> get --monitor tpsystem.*.*.*.*.*.*.jdbc-datasource*.NumConnUsed-HighWaterMark

統計情報は、JDBC データソースの統計情報の採取レベルを ON に設定した上で、JDBC データソースから最初にコネクション取得を行った際に表示されます。採取レベルの変更方法は次の通りです。

統合運用管理ツールで設定する場合:

次の属性の値を変更し、更新ボタンを押してください。
[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[アプリケーションサーバ]-[モニタリングサービス]-[モジュールモニタリングレベル]-[jdbc datasource モニタリングレベル]

運用管理コマンドで設定する場合:
otxadmin> set server.monitoring-service.module-monitoring-levels.jdbc-datasource=ON

設定方法

統合運用管理ツールで設定する場合:

次の属性の値を変更し、更新ボタンを押してください。
[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクションプール]-[最小プールサイズ]

[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクションプール]-[コネクション解放までの待ち時間]

運用管理コマンドで設定する場合:
otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.minPoolSize=10
otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.shrinkDelayTime=60

8.1.2. 初期プールサイズの設定

初期プールサイズ[initialPoolSize]に 0 が設定されている場合、アプリケーションがコネクションを取得する際に、必要に応じて接続されます。このため、業務の運用開始直後は、コネクションの接続を行う分、業務の性能が悪くなります。

EJB アプリケーションや Web アプリケーションでは、初期プールサイズで指定された数のコネクションが、アプリケーションプロセス起動時に予め接続されていますので、業務の運用開始直後の性能劣化を防ぐことができます。

初期プールサイズには、最小プールサイズで指定された数以下の値を指定してください。

設定方法

統合運用管理ツールで設定する場合:

次の属性の値を変更し、更新ボタンを押してください。
[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクションプール]-[初期プールサイズ]

運用管理コマンドで設定する場合:
otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.initialPoolSize=4

8.1.3. 最大プールサイズの設定

特定のアプリケーションプロセスで、極端に多くのコネクションを取得してしまうと、他のアプリケーションプロセスでデータベースに接続できなくなってしまう恐れがあります。このような状況を防止するための対策としては、最大プールサイズの設定が有効です。最大プールサイズ[maxPoolSize]には、最小プールサイズよりも大きく、データベースサーバ側の最大接続数よりも小さい値を設定してください。

設定方法

統合運用管理ツールで設定する場合:

次の属性の値を変更し、更新ボタンを押してください。
[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクションプール]-[最大プールサイズ]

運用管理コマンドで設定する場合：
otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.maxPoolSize=10

8.2. その他の設定

その他の設定について説明します。

8.2.1. ステートメントの最大プール数の設定

JDBC ドライバでステートメントプール機能をサポートしている場合、その機能を利用することで、ステートメントの生成コスト分の性能を改善することができます。

JDBC データソースがサポートする JDBC ドライバの中では、Oracle と SequeLink の JDBC ドライバで、ステートメントプール機能をサポートしています。

ステートメントプール機能では、文字列化された SQL 命令をキーにしてステートメントの検索が行われます。このため、文字列化された SQL 命令に、可変の値が含まれない java.sql.preparedStatement を利用し、検索効率を高めることが重要です。

設定方法

統合運用管理ツールで設定する場合：

次の属性の値を変更し、反映ボタンを押してください。
[WebOTX 管理ドメイン[<ホスト名>]]-[<ドメイン名>]-[リソース]-[jdbc データソース]-[JDBC データソース名]-[コネクションプール]-[ステートメントの最大プール数]

運用管理コマンドで設定する場合：
otxadmin> set server.resources.jdbc-datasource.<JDBC データソース名>.maxStatements=20

9.EJBコンテナのチューニング

Standard-J Edition および Standard/Enterprise Edition における EJB コンテナのチューニング方法に関して説明します。

9.1. EJB

EJB は、ステートフルセッション Bean、ステートレスセッション Bean、エンティティ Bean、メッセージドリブン Bean の 4 種類あり、用途に応じて使い分けられます。

9.1.1. ステートレスセッションBeanのチューニング

ステートレスセッション Bean は名前の通り状態（ステート）を持ちません。そのため事前にインスタンスを生成し、プールしておくことが可能です。インスタンス生成は CPU リソースを消費し、時間がかかりますので、運用中のインスタンス生成を避けるために事前生成とプーリングが行われます。

また複数クライアントから同時に呼び出しが行われることを想定して、複数のインスタンスをプールしておくことも可能です。プールするインスタンス数を増やすと、プール数に比例してメモリを消費します。

ステートレスセッション Bean のチューニングはインスタンスプールのプール数の調整によって行われます。これは Edition によって設定項目が変わります。

Standard/Enterprise Editionの場合

リモートインタフェースを使用した場合、プールするインスタンス数はプロセスグループのスレッド数と同じ値になります。「2.4.1 多重度のチューニング」を参考にスレッド数を設定してください。

配備されたアプリケーション単位でプールするインスタンス数を調整することも可能です。以下のどちらかの方法により 0～スレッド数の間で設定してください。

- 統合運用管理ツール
  [(ドメイン名)]-[アプリケーション]-[WebOTXJ2EE アプリケーション]-[(モジュール名)]-[(ファイル名)]-[(アプリケーション名)]-[(リモートインタフェース)]の「一般」タブの「事前生成オブジェクト数」
- 運用管理コマンド
  otxadmin> set applications.j2ee-applications.(モジュール名).(ファイル名).(アプリケーション名).(リモートインタフェース).priorGenerationCount=(設定したい数)

ローカルインタフェースを使用した場合、プールするインスタンス数は EJB コンテナの設定項目である「通常プールサイズ」で設定されます。既定値では「通常プールサイズ」は 0 になっています。インスタンス事前生成機能を有効にするには、「通常プールサイズ」を 1 以上にしてください。
設定方法は、次の「Standard-J Edition の場合」を参照してください。

Standard-J Editionの場合

プールするインスタンス数は EJB コンテナの設定項目である「通常プールサイズ」で設定されます。既定値では「通常プールサイズ」は 0 になっています。インスタンス事前生成機能を有効にするには、「通常プールサイズ」を 1 以上にしてください。

- 統合運用管理ツール
  [(ドメイン名)]-[アプリケーションサーバ]-[EJB コンテナ]の「通常プールサイズ」
- 運用管理コマンド
  otxadmin> set server.ejb-container.steady-pool-size=(設定したい数)

9.1.2. ステートフルセッションBeanのチューニング

ステートフルセッション Bean は状態を持ちますので事前生成は行えません。クライアントからステートフルセッション Bean が利用される時、必ずインスタンス生成／破棄が行われます。状態が必要ないアプリケーションの場合、ステートレスセッション Bean で設計し、事前生成・プーリング機能を有効にした方が

実行性能は良くなります。

ステートフルセッション Bean ではプールの代わりにキャッシュが使われます。メモリ上に作られ、破棄されていないインスタンスが長時間アクセスされない場合、状態がファイルにキャッシュされます。頻繁にキャッシングが行われる場合、アプリケーションの問題（クライアントから bean の remove()メソッドが呼ばれていない）もしくはインスタンスのタイムアウト設定時間が短すぎることが考えられます。インスタンスのタイムアウト設定時間は次のように設定します。

- 統合運用管理ツール
  [(ドメイン名)]-[アプリケーションサーバ]-[EJB コンテナ]の「アイドルタイムアウト（キャッシュ）」
- 運用管理コマンド
  otxadmin> set server.ejb-container.pool-idle-timeout-in-seconds=(設定したい数)

9.1.3. エンティティBeanのチューニング

エンティティ Bean はプールとキャッシュ両方を利用します。プーリングについては「9.1.1.ステートレスセッション Bean のチューニング」、キャッシングについては「9.1.2.ステートフルセッション Bean のチューニング」を参照してください。

9.1.4. メッセージドリブンBeanのチューニング

ステートレスセッション Bean と同様に事前生成・プーリングが行われます。「9.1.1.ステートレスセッション Bean のチューニング」を参照してください。ただし Standard-J Edition での通常プールサイズの設定は EJB コンテナではなく、独自の MDB コンテナの設定に拠ります。

- 統合運用管理ツール
  [(ドメイン)]-[アプリケーションサーバ]-[MDB コンテナ]
- 運用管理コマンド
  otxadmin> set server.mdb-container.steady-pool-size=(設定したい数)

10. 通信に関するチューニング

ブラウザから Web サーバを経由し AP サーバにアクセスする場合の通信に関するチューニングについて説明します。

10.1. TCP/IPに関する設定について

全ての通信の基本である OS の TCP/IP に関する設定について説明します。

概要

TCP/IP に関する最も重要な設定は、**送信タイムアウト時間**と **KeepAlive** の設定です。例えば、通信相手のサーバマシンの電源が落ちて、TCP/IP の切断処理が行われないままとなった場合、クライアントでは、**送信タイムアウト時間**と **KeepAlive** のどちらかの設定によって障害が検出されるまで、通信できない状態になります。

大抵の場合、**送信タイムアウト時間**の設定によって、長くて 10 分程度で障害を検出することができます。ただ、稀に受信待ちとなった場合には、**KeepAlive** の OS のデフォルト設定で約 2 時間障害を検出できません。

KeepAlive の設定は、サーバ側で無効なコネクションを破棄するために利用することがより一般的です。

システムの障害復旧時間の要件に応じて、これらの設定のチューニングを行ってください。

OSの設定方法

設定内容や設定方法は、次に示す通り、OS 毎に異なります。詳細については、各 OS のリファレンスをご覧ください。

HP-UXの場合）

/etc/rc.config.d/nddconf に次のように設定します。

送信タイムアウト時間（データ送信時）:

```
TRANSPORT_NAME[num]=tcp
NDD_NAME[num]=tcp_ip_abort_interval
NDD_VALUE[num]=600000
```

送信タイムアウト時間（接続要求時）:

```
TRANSPORT_NAME[num]=tcp
NDD_NAME[num]=tcp_ip_abort_cinterval
NDD_VALUE[num]=75000
```

KeepAlive:

次のデフォルトの設定では、tcp_ip_abort_interval の値を加えた2時間10分で障害を検出します。

```
TRANSPORT_NAME[num]=tcp
NDD_NAME[num]=tcp_keepalive_interval
NDD_VALUE[num]=7200000
```

Linuxの場合）

/etc/sysctl.conf に次のように設定します。

送信タイムアウト時間（データ送信時）:

次のデフォルトの設定では、13～30 分で障害を検出します。

net.ipv4.tcp_retries = 15

送信タイムアウト時間（接続要求時）:

net.ipv4.tcp_syn_retries = 5

KeepAlive:

次のデフォルトの設定では、7200 秒 + 75 x 9 秒で、約2時間11分で障害を検出します。

net.ipv4.tcp_keepalive_intvl = 75

net.ipv4.tcp_keepalive_proves = 9

net.ipv4.tcp_keepalive_time = 7200

Solarisの場合）

/etc/system に次のように設定します。

送信タイムアウト時間（データ送信時）:

set tcp:tcp_ip_abort_interval = 480000

送信タイムアウト時間（接続要求時）:

set tcp:tcp_ip_abort_cinterval = 180000

KeepAlive:

次のデフォルトの設定では、tcp_ip_abort_interval を加えた2時間8分で障害を検出します。

set tcp:tcp_ip_keepalive_interval = 7200000

Windowsの場合）

レジストリ HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥services¥Tcpip¥Parameters キーの値を次のように設定します。

MEMO

Windows の初期値は OS により様々です。OS のリファレンスやホームページで確認してください。

送信タイムアウト時間（データ送信時）:

次の設定では、アダプタ毎のレジストリ値 TcpInitialRTT の値が 3000 ミリ秒である場合、

3 + (3 x 2) + (6 x 2) + (12 x 2) + (24 x 2)秒で、約93秒で障害を検出します（リトライする度に、リトライの間隔が直前の間隔の2倍になります）。

TcpMaxDataRetransmissions	REG_DWORD	5

送信タイムアウト時間（接続要求時）:

次の設定では、アダプタ毎のレジストリ値 TcpInitialRTT の値が 3000 ミリ秒である場合、

3 + (3 x 2) + (6 x 2)秒で、約21秒で障害を検出します（リトライする度に、リトライの間隔が直前の間隔の2倍になります）。

TcpMaxConnectRetransmissions	REG_DWORD	3

KeepAlive:

次のデフォルトの設定では、7200 秒 + 5x1000 ミリ秒で、約2時間で障害を検出します。

KeepAliveTime	REG_DWORD	7200000
KeepAliveInterval	REG_DWORD	1000
TcpMaxDataRetransmissions	REG_DWORD	5

次のレジストリ値 TcpInitialRTT は、設定場所やデフォルト値が OS 毎に異なります。

デフォルト値については OS 毎に確認して頂く必要がありますが、TcpMaxConnectRetransmissions や TcpMaxDataRetransmissions といった再送回数のチューニングだけを行うようにすれば、設定場所まで意識する必要はありません。

TcpInitialRTT	REG_DWORD	3000

KeepAliveを有効にするための設定

WebOTX では、デフォルトの設定で KeepAlive の動作が有効になります。このため、設定変更を行う必要はありません。ここでは、WebOTX で利用することが多い Oracle クライアントでの設定方法について説明します。詳細については、Oracle のリファレンスを参照してください。

Oracle thinドライバでKeepAliveを有効にするための設定

JDBC データソースのデータソース名[dataSourceName]に設定する接続文字列として、次の形式（例）で設定してください。

"jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=xxx.xxx.xxx.xxx)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=ORCL))(ENABLE=BROKEN))"

Oracle OCIドライバおよびC++アプリケーションでKeepAliveを有効にするための設定

Oracle Net Service の tnsnames.ora ファイルに、次の形式（例）で設定してください。

ORCL = (DESCRIPTION= (ADDRESS=(PROTOCOL=TCP)(HOST=xxx.xxx.xxx.xxx)(PORT=1521))
(CONNECT_DATA= (SERVICE_NAME=ORCL))(ENABLE=BROKEN))