JBoss Enterprise Application Platform 5

HTTP コネクタ負荷分散ガイド

JBoss Enterprise Application Platform 向け HTTP 負荷分散

エディッション 5.1.2

Last Updated: 2017-10-26

JBoss Enterprise Application Platform 5 HTTP コネクタ負荷分散ガイド

JBoss Enterprise Application Platform 向け HTTP 負荷分散
エディッション 5.1.2

Jared Morgan
Red Hat, Inc. Engineering Content Services
jmorgan@redhat.com
Lead Writer および Content Architect

Joshua Wulf
Red Hat Engineering Content Services
jwulf@redhat.com

Laura Bailey
Red Hat, Inc. Engineering Content Services
lbailey@redhat.com

Samuel Mendenhall
Red Hat Global Support Services

James Livingston
Red Hat Global Support Services

Jim Tyrell
Red Hat JBoss Solutions Architect

法律上の通知

Copyright © 2011 Red Hat, Inc.

This document is licensed by Red Hat under the Creative Commons Attribution-ShareAlike 3.0 Unported License. If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書では JBoss Enterprise Application Platform HTTP コネクタである mod_jk、mod_cluster、ISAPI、NSAPI のインストールと設定方法について記載しています。また、それらのコネクタを利用したクラスタリングおよび負荷分散についても説明しています。

目次

はじめに

1. ファイル命名規則

読みやすくするために、ファイルパスでは以下の命名規則が使用されています。コンテキストで識別しやすくするために各規則のスタイルは異なります。

JBOSS_EAP_DIST

JBoss Enterprise Application Platform インスタンスのインストールルート。このフォルダには **/jboss-as**、**/seam**、**/resteasy** などの、サーバーを構成する主要なフォルダが含まれます。

JBOSS_EWP_DIST

JBoss Enterprise Web Platform インスタンスのインストールルート。このフォルダには **/jboss-as-web**、**/seam**、**/resteasy.** などの、サーバーを構成する主要なフォルダが含まれます。

JBOSS_EWS_DIST

JBoss Enterprise Web Server インスタンスのインストールルート。このフォルダには **/extras**、**/httpd**、**/tomcat6** などの、サーバーを構成する主要なフォルダが含まれます。

NATIVE

JBoss Native zip のインストールルート。*JBOSS_EAP_DIST* と同じディレクトリレベルに抽出されます。

SJWS

Sun Java Web Server インスタンスのインストールルート。この命名規則のデフォルトのファイル場所は以下のとおりです。

- Solaris 9 x86 または SPARC 64 の場合: **/opt/SUNWwbsrv61**
- Solaris 10 x86 または SPARC 64 の場合: **/opt/SUNWwbsrv70/**

HTTPD_DIST

Apache httpd Server のインストールルート。このフォルダには、**/conf**、**/webapps**、**/bin** などの、サーバーを構成する主要なフォルダが含まれます。JBoss Enterprise Web Server *JBOSS_EWS_DIST* ディレクトリには、*HTTPD_DIST* のルートインストールが含まれます。

PROFILE

テスト用または本番稼働用の設定の一部として使用する JBoss サーバープロファイルの名前。サーバープロファイルは、***JBOSS_EAP_DIST*/jboss-as/server** または ***JBOSS_EWS_DIST*/jboss-as-web/server** に存在します。

パート I. APACHE TOMCAT CONNECTOR (MOD_JK)

第1章 概要

Apache は良く知られた Web サーバーでありプラグインを使用して拡張できます。Apache Tomcat Connector **mod_jk** は Apache httpd Server から Servlet コンテナに要求を転送できるようつくられたプラグインです。 モジュールはスティッキーセッションを維持しながらServelt のセットに対する負荷分散 HTTP 呼び出しに対応します。

HTTP セッションレプリケーションを使用して Web クライアントセッションに関連するステートをクラスタの他のノードにレプリケートします。ひとつのノードが使用不可能になると、クラスタ内の別のノードが無効のノードの負荷と取ります。次の 2 つの機能を実行する必要があります。

- セッションステートのレプリケーション
- HTTP 要求の負荷分散

セッションステートのレプリケーションはアプリケーションレベルで JBoss が処理します (「アプリケーションでのセッションレプリケーションの有効化」 を参照)。

ただし、負荷分散には外部ロードバランサが必要です。負荷分散を管理するコスト効率の良い方法は、Apache httpd と mod_jk を使用してソフトウェアロードバランサを設定することです。

第2章 ダウンロードとインストール

Apache httpd は https://access.redhat.com からダウンロードする JBoss Enterprise Web Serve バイナリにあります。

mod_jk は JBoss Enterprise Application Platform および JBoss Enterprise Web Server 向けのネイティブバイナリのインストールにあります。

JBoss Enterprise Application Platform または JBoss Enterprise Web Server 『Installation Guide』 の手順に従って、適切なプラットフォームとネイティブバイナリをダウンロードし、インストールしてください。

第3章 APACHE と MOD_JK を使用した負荷分散の設定

本章のタスクに従い、Apache と mod_jk コネクタを使用して正しく負荷分散を設定します。

タスク : mod_jk をロードするために Apache を設定する。

このタスクでは、mod_jk をロードするため Apache を設定します。

前提条件

- Apache と mod_jk がインストールされている (Refer to 2章*ダウンロードとインストール*を参照)。

1. ***HTTPD_DIST*/conf/httpd.conf** を開いて、ファイルの最後に 1 行追加します。

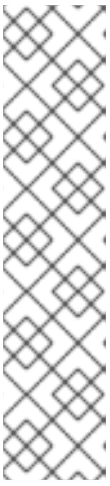
```
# Include mod_jk's specific configuration file
Include conf/mod-jk.conf
```

2. ***HTTPD_DIST*/conf/mod-jk.conf** という新しいファイルを作成します。

3. **mod-jk.conf** ファイルに次の設定を追加します。

重要

LoadModule ディレクティブはインストールしたネイティブバイナリに適用可能な **mod_jk** ライブラリディレクトリの場所を参照する必要があります。



注記

JkMount ディレクティブは Apache が **mod_jk** モジュールにどの URL を転送すべきか指定します。ディレクティブの設定に基づき、**mod_jk** は受信した URL を正しい Servlet コンテナに転送します。

Apache が静的コンテンツ (または PHP コンテンツ) に直接対応し、Java アプリケーションにロードバランサを使用可能なようにするためには、推奨される設定は **/application/*** が **mod_jk** ロードバランサに送られる URL パスですべての要求を指定します。

ロードバランサとして **mod_jk** だけを使用している場合は、ディレクティブの **/*** を指定することですべての URL を **mod_jk** に転送します。

```
# Load mod_jk module
# Specify the filename of the mod_jk lib
LoadModule jk_module modules/mod_jk.so

# Where to find workers.properties
JkWorkersFile conf/workers.properties

# Where to put jk logs
JkLogFile logs/mod_jk.log

# Set the jk log level [debug/error/info]
JkLogLevel info
```

```
# Select the log format
JkLogStampFormat  "[%a %b %d %H:%M:%S %Y]"

# JkOptions indicates to send SSK KEY SIZE
JkOptions +ForwardKeySize +ForwardURICompat -ForwardDirectories

# JkRequestLogFormat
JkRequestLogFormat "%w %V %T"

# Mount your applications
JkMount /application/* loadbalancer

# Add shared memory.
# This directive is present with 1.2.10 and
# later versions of mod_jk, and is needed for
# for load balancing to work properly
JkShmFile logs/jk.shm

# Add jkstatus for managing runtime data
<Location /jkstatus/>
    JkMount status
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
</Location>
```

4. **オプション : JKMountFile ディレクティブ**
   JkMount ディレクティブに加え、**JkMountFile** ディレクティブを使用して、マウントポイント設定ファイルを指定することもできます。その設定ファイルには複数の Tomcat 転送 URL マッピングが含まれています。

   a. ***HTTPD_DIST*/conf** に移動します。

   b. **uriworkermap.properties** という名前のファイルを作成します。

   c. 転送する URL と次の構文サンプルをガイドとして使用してワーカー名を指定します。

      サンプルブロックは **mod_jk** を設定し、**/jmx-console** と **/web-console** への要求を Apache に転送します。

      必要な構文の形式は **/url=worker_name** です。

      ```
      # Simple worker configuration file

      # Mount the Servlet context to the ajp13 worker
      /jmx-console=loadbalancer
      /jmx-console/*=loadbalancer
      /web-console=loadbalancer
      /web-console/*=loadbalancer
      ```

   d. ***HTTPD_DIST*/conf/mod-jk.conf** では、次のディレクティブを追加します。

      ```
      # You can use external file for mount points.
      # It will be checked for updates each 60 seconds.
      ```

```
# The format of the file is: /url=worker
# /examples/*=loadbalancer
JkMountFile conf/uriworkermap.properties
```

3.1. MOD_JK のワーカーノードの設定

タスク : mod_jk ワーカーノードを設定する。

このタスクでは、2 つのサーブレットコンテナ間でスティッキーセッションをアクティブにした状態で、重み付きラウンドロビン設定の 2 つの mod_jk ワーカーノードの定義を設定します。

前提条件

- 付録A *参照 : workers.properties* で説明されているとおり **workers.properties** ディレクティブの形式を理解します。
- タスク : mod_jk をロードするために Apache を設定する。

1. ***HTTPD_DIST*/conf/** に移動します。
2. **workers.properties** という名前のファイルを作成します。
3. **workers.properties** に次の情報を追加します。

```
# Define list of workers that will be used
# for mapping requests
worker.list=loadbalancer,status

# Define Node1
# modify the host as your host IP or DNS name.
worker.node1.port=8009
worker.node1.host=node1.mydomain.com
worker.node1.type=ajp13
worker.node1.ping_mode=A
worker.node1.lbfactor=1

# Define Node2
# modify the host as your host IP or DNS name.
worker.node2.port=8009
worker.node2.host=node2.mydomain.com
worker.node2.type=ajp13
worker.node2.ping_mode=A
worker.node2.lbfactor=1

# Load-balancing behavior
worker.loadbalancer.type=lb
worker.loadbalancer.balance_workers=node1,node2
worker.loadbalancer.sticky_session=1

# Status worker for managing load balancer
worker.status.type=status
```

3.2. MOD_JK と動作するよう JBOSS を設定する

タスク : mod_jk を使用して JBoss Enterprise Application Platform が動作するよう設定する。

このタスクでは、クラスタ化したノードで JBoss Enterprise Application Platform インスタンスを正しく準備し、**mod_jk** ロードバランサから転送された要求を受け取ります。

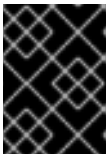
必要な各サーバーインスタンスに対してこのタスクを繰り返し、各ステップでの警告に注意してください。

前提条件

- タスク : mod_jk ワーカーノードを設定する。 を完了する。

1. クラスタ化したサーバーインスタンスの場所に移動します。
2. ***JBOSS_EAP_DIST*/jboss-as/server/*PROFILE*/deploy/jbossweb.sar/server.xml** を開きます。
3. ノード名を指定するには、**server.xml** の **<Engine>** 要素に jvmRoute 属性を追加します。jvmRoute 属性値は ***HTTPD_DIST*/conf/workers.properties** で定義されたノード名です。

```
<!--Preceeding syntax removed for readability -->
<Engine name="jboss.web" defaultHost="localhost" jvmRoute="node1">
<!--Proceeding syntax removed for readability -->
</Engine>
```



重要

クラスタ内で複数のサーバーノードを設定するつもりの場合は、このステップを繰り返すたびに jvmRoute 属性値を一意の名前に変更するようにしてください。

4. **server.xml** で、AJP プロトコル **<connector>** 要素が有効であることを確認してください (非コメント化)。その要素は新しいインストールではデフォルトで非コメント化されています。

```
<Connector protocol="AJP/1.3" port="8009"
address="${jboss.bind.address}"
   redirectPort="8443" />
```

5. これで **mod_jk** ロードバランサを使って Apache httpd Server を正しく設定することができました。これによりクラスタ内のサーブレットコンテナへの呼び出しを分散し、同じサーブレットコンテナ (スティッキーセッション) を使用することができます。

注記

JBoss での mod_jk の使用に関する補足情報は http://www.jboss.org/community/wiki/UsingModjk12WithJBoss の JBoss wiki ページを参照してください。

パート II. JBOSS HTTP CONNECTOR (MOD_CLUSTER)

第4章 概要

JBoss HTTP Connector **mod_cluster** は縮約された設定で、JBoss Enterprise Application Platform にとってインテリジェントな負荷分散のソリューションです。もともとは JBoss mod_cluster コミュニティプロジェクトから開発された技術に基づいています。

JBoss HTTP コネクタは JBoss Enterprise Application Platform と JBoss Enterprise Web Server ワーカーノードに HTTP 要求を負荷分散し、プロキシサーバーとして Apache を使用します。

4.1. 主な特徴

Apache HTTP Server ベース

JBoss HTTP Connector **mod-cluster** はプロキシサーバーとして Apache を使用します。

リアルタイムの負荷分散の計算

JBoss HTTP Connector **mod_cluster** はワーカーノードとプロキシサーバー間のフィードバックネットワークを作成します。**mod_cluster** サービスは各ワーカーノードにデプロイされています。このサービスはプロキシサーバーにリアルタイムの負荷情報をフィードします。 次にプロキシサーバーは各ワーカーノードの現在の負荷に基づきワークを割り振る場所に関してインテリジェントな決定を行います。このリアルタイムの適応した負荷分散によりリソースの最適化が向上します。

ワーカーノードにより報告された情報およびプロキシにより使用された負荷分散ポリシーは両方ともカスタマイズ可能です。

リアルタイムアプリケーションライフサイクルに基づいたルーティング

ワーカーノードにデプロイされた JBoss HTTP Connector **mod_cluster** サービスはアプリケーションライフサイクルイベントをプロキシサーバーにリレーします。これによりサーバーは動的にそのルーティングテーブルを更新できます。アプリケーションがあるノードでデプロイ解除されると、プロキシサーバーはそのノードに対しそのアプリケーションのトラフィックをルーティングしないようになります。

プロキシの自動検出

プロキシサーバーは UDP マルチキャストを介してその存在を知らせるように設定できます。新しいワーカーノードはプロキシサーバーを検出し、それらを自動的に負荷分散クラスタに追加します。これにより必要な設定と管理を大幅に縮小することができます。UDP マルチキャストが使用不可能または希望するものでない場合、ワーカーノードはプロキシの静的リストで設定されます。

複数のプロトコル対応

JBoss HTTP Connector **mod_cluster** は HTTP、HTTPS、または Apache JServ Protocol (AJP) を使用して、プロキシとワーカーノード間の通信を行うことができます。

4.2. コンポーネント

プロキシサーバー

プロキシサーバーでは、JBoss HTTP Connector **mod-cluster** は 4 つの Apache モジュールから構成されています。

Shared Memory Manager: `mod_slotmem.so`

Shared Memory Manager モジュールの **mod_slotmem** はリアルタイムワーカーノード情報を複数の Apache サーバープロセスに対して使用できるようにします。

Cluster Manager Module: `mod_manager.so`

Cluster Manager モジュールの **mod_manager** はノードからワーカーノード登録、ワーカーノード負荷データ、ワーカーノードのアプリケーションライフサイクルイベントなどのメッセージを受信し、認識します。

Proxy Balancer Module: `mod_proxy_cluster.so`

Proxy Balancer Module の **mod_proxy_cluster** はクラスタノードに対し要求のルーティングを処理します。Proxy Balancer はクラスタ内のアプリケーションの場所、各クラスタノードの現在のステート、Session ID (要求が確率したセッションの一部の場合) に基づき、要求を転送する該当するノードを選択します。

Proxy Advertisement Module: `mod_advertise.so`

Proxy Advertisement Module の **mod_advertise.so** は、UDP マルチキャストメッセージを介してプロキシサーバーの存在をブロードキャストします。サーバー告知のメッセージには、負荷分散クラスタに加わりたいノードからの応答に対しプロキシがリッスンする IP アドレスとポート番号が含まれています。

注記

ユーザー設定可能なパラメータなど使用可能なモジュールに関する詳細は 「Apache モジュール」 を参照してください。

ワーカーノードのコンポーネント

Worker node service: `mod-cluster.sar`

JBoss HTTP Connector クライアントサービスの **mod-cluster.sar** は各ワーカーノードにデプロイされています。このサービスはプロキシにワーカーノードのステートのリアルタイム情報を伝え、ライフサイクルイベントの通知を送ります。加えて、ノードが同じネットワークで実行しているすべてのプロキシでそれ自身を検出、登録することができるようにします。

第5章 プロキシサーバーコンポーネントのインストール

本章では JBoss Enterprise Web Server プロキシサーバーに JBoss HTTP Connector **mod-cluster** をインストールする方法を説明します。

5.1. APACHE モジュール

本項では「コンポーネント」で説明した Apache プロキシサーバーモジュールに関する詳細な定義について説明します。これらのモジュールを タスク : プロキシサーバーコンポーネントをインストールする の一部として指定します。

5.1.1. mod_manager.so

Cluster Manager モジュールの **mod_manager** はノードからワーカーノード登録、ワーカーノード負荷データ、ワーカーノードのアプリケーションライフサイクルイベントなどのメッセージを受信し、認識します。

```
LoadModule manager_module modules/mod_manager.so
```

<VirtualHost> 要素で次の関連するディレクティブを定義することもできます。

MemManagerFile

mod_manager が設定の詳細を保存するファイルの場所を定義します。mod_manager はまた、共有メモリおよびロックファイルの生成されたキーに対しこの場所を使用します。**これは絶対パスの名前でなければなりません。** このパスは NFS 共有ではなく、ローカルドライブにあることが推奨されます。デフォルト値は **/logs/** です。

Maxcontext

JBoss mod_cluster が使用するコンテキストの最大数です。デフォルト値は **100** です。

Maxnode

JBoss mod_cluster が使用するワーカーノードの最大数です。デフォルト値は **20** です。

Maxhost

JBoss mod_cluster が使用するホスト (エイリアス) の最大数です。これはロードバランサの最大数でもあります。デフォルト値は **10** です。

Maxsessionid

保存されているアクティブなセッション識別子の最大数です。**5** 分以内にセッションから受信する情報がない場合はセッションはアクティブでないと見なされます。デフォルト値は **0** で、このロジックを無効にします。

ManagerBalancerName

ワーカーノードがロードバランサの名前を与えない場合に使用するロードバランサの名前です。デフォルト値は **mycluster** です。

PersistSlots

on に設定すると、ノード、エイリアス、コンテキストはファイルで永続化されます。デフォルト値は **off** です。

CheckNonce

on に設定すると、セッション識別子はそれらが一意であり以前に発生したことがないことを確認します。デフォルト値は **on** です。

警告

このディレクティブを **off** に設定することで、ご使用のサーバーをリプレーアタックに対し脆弱性を有するようにできます。

SetHandler

ハンドラを定義し、クラスタ内のワーカーノードに関する情報を表示します。これは **Location** 要素で定義されます。

```
<Location $LOCATION>
  SetHandler mod_cluster-manager
  Order deny,allow
  Deny from all
  Allow from 127.0.0.1
</Location>
```

ブラウザで **Location** 要素に定義された *$LOCATION* にアクセスすると、次のようなものが表示されます (この場合、*$LOCATION* は **mod_cluster-handler** としても定義されます)。

Transferred はワーカーノードに送られた POST データに該当します。**Connected** は、このステータスページが要求されたときに処理された要求数に該当します。**Sessions** はアクティブなセッション数に該当します。このフィールドは **Maxsessionid** が **0** の場合は存在しません。

5.1.2. mod_proxy_cluster.so

Proxy Balancer Module の **mod_proxy_cluster** はクラスタノードに対し要求のルーティングを処理します。Proxy Balancer はクラスタ内のアプリケーションの場所、各クラスタノードの現在のステート、Session ID (要求が確率したセッションの一部の場合) に基づき、要求を転送する該当するノードを選択します。

```
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
```

<VirtualHost> 要素で次の関連するディレクティブを定義し、負荷分散の動作を変更することができます。

mod_proxy_cluster directives

CreateBalancers

Apache HTTP Server の仮想ホストでのロードバランサの作成方法について定義します。次の値は **CreateBalancers** で有効です。

0

Apache HTTP Server で定義されたすべての仮想ホストでロードバランサを作成します。**ProxyPass** ディレクティブでロードバランサを設定するようにしてください。

1

バランサを作成しません。この値を使用する場合は、**ProxyPass** または **ProxyPassMatch** でもロードバランサの名前を定義しなければなりません。

2

メインサーバーのみ作成します。これは **CreateBalancers** のデフォルト値です。

UseAlias

定義された **Alias** が **ServerName** に対応することを確認するか定義します。次の値は**UseAlias** で有効です。

0

ワーカーノードからの Alias 情報を無視します。これは **UseAlias** のデフォルト値です。

1

定義されたエイリアスがワーカーノードのサーバー名に対応していることを検証します。

LBstatusRecalTime

ワーカーノードのステータスを計算するプロキシ間での秒単位の間隔を定義します。デフォルトの間隔は 5 秒です。

ProxyPassMatch; ProxyPass

ProxyPass はリモートサーバーをローカルサーバーの名前空間にマップします。ローカルサーバーがアドレス **http://local.com/** を持っている場合、次の **ProxyPass** ディレクティブは **http://local.com/requested/file1** のローカル要求を **http://worker.local.com/file1** のプロキシ要求に変換します。

```
ProxyPass /requested/ http://worker.local.com/
```

ProxyPassMatch は Regular Expressions を使用して、プロキシされた URL が適用するべきローカルパスに適合します。

いずれのディレクティブの場合でも、**!** はある特定のパスがローカルであり、そのパスの要求がリモートサーバーにルーティングされるべきではないことを示しています。例えば、次のディレクティブは **.gif** ファイルがローカルに提供されることを指定します。

```
ProxyPassMatch ^(/.*\.gif)$ !
```

5.1.3. mod_advertise.so

Proxy Advertisement Module の **mod_advertise.so** は、UDP マルチキャストメッセージを介してプロキシサーバーの存在をブロードキャストします。サーバー告知のメッセージには、負荷分散クラスタに加わりたいノードからの応答に対しプロキシがリッスンする IP アドレスとポート番号が含まれています。

このモジュールは **VirtualHost** 要素の **mod_manager** に沿って定義されます。次のコードスニペットの識別子は **advertise_module** です。

```
LoadModule advertise_module modules/mod_advertise.so
```

mod_advertise は次のディレクティブを取ります。

ServerAdvertise

告知しているメカニズムの使用方法を定義します。

On に設定した場合、告知しているメカニズムを使用して、ワーカーノードにこのプロキシにステータス情報を送るよう指示します。次の構文 **ServerAdvertise On http://hostname:port/** でホスト名とポートを指定することもできます。これが必要なのは、名前ベースの仮想ホストを使用する場合か、仮想ホストが定義されていない場合だけです。

デフォルト値は **Off** です。**off** に設定すると、プロキシはその場所を告知しません。

AdvertiseGroup

告知するマルチキャストのアドレスを定義します。構文は **AdvertiseGroup *address:port*** であり、*address* は **AdvertiseGroupAddress** に該当し、*port* はワーカーノードの **AdvertisePort** に該当するはずです。

ワーカーノードが JBoss Enterprise Application Platform ベースで、**-u** スイッチが起動時に使用される場合は、デフォルト **AdvertiseGroupAddress** は **-u** スイッチにより渡された値です。

デフォルト値は **224.0.1.105:23364** です。*port* が指定されていないと、指定されるデフォルトのポートは **23364** です。

AdvertiseFrequency

IP アドレスとポートを告知するマルチキャストメッセージ間の秒単位の間隔です。デフォルト値は **10** です。

AdvertiseSecurityKey

JBoss Web の JBoss HTTP Connector mod_cluster を特定するために使用する文字列を定義します。デフォルトでは、このディレクティブは設定されておらず何も情報は送られません。

AdvertiseManagerUrl

プロキシサーバーに情報を送るためにワーカーノードが使用する URL を定義します。デフォルトでは、このディレクティブは設定されておらず何も情報は送られません。

AdvertiseBindAddress

マルチキャストメッセージを送るアドレスとポートを定義します。構文は **`AdvertiseBindAddress address:port`** です。これにより、複数の IP アドレスを使ってマシンにアドレスを指定できます。デフォルト値は **`0.0.0.0:23364`** です。

5.2. プロキシサーバーコンポーネントのインストール

タスク : プロキシサーバーコンポーネントをインストールする

このタスクに従い、JBoss Enterprise Web Server に JBoss HTTP Connector をインストールします。

JBoss HTTP Connector はプロキシサーバーとして JBoss Enterprise Web Server を使用した実稼働のみに対応しています。JBoss Enterprise Web Server のダウンロードとインストールについては、JBoss Enterprise Web Server 『Installation Guide』 を参照してください。

Native コンポーネントは Operating System およびプロセッサアーキテクチャに固有のものです。ご使用のサーバー Operating System およびプロセッサアーキテクチャに対して適切な Native Components パッケージをダウンロードするには、JBoss Enterprise Application Platform 『Installation Guide』 を参照してください。

前提条件

- JBoss Enterprise Web Server v1.0.1 またはそれ以降のバージョンがインストールされている。
- JBoss Enterprise Application Platform 5 Native コンポーネントがダウンロードされている。

1. **Native Components ダウンロードから Apache モジュールを抽出します。**
   4 つのモジュール **`mod_advertise.so`**、**`mod_manager.so`**、**`mod_proxy_cluster.so`**、**`mod_slotmem.so`** を使用しているプロセッサアーキテクチャに対して **`native/lib/httpd/modules`** または **`native/lib64/httpd/modules`** いずれかの適切な Native Components パッケージディレクトリから抽出します。

2. **JBoss Enterprise Web Server に Apache モジュールをコピーします。**
   JBoss HTTP Connector モジュールを JBoss Enterprise Web Server の ***`JBOSS_EWS_DIST`*`/httpd/modules`** ディレクトリにコピーします。

3. **mod_proxy_balancer モジュールを無効にします。**
   JBoss Enterprise Web Server Apache 設定ファイルである ***`JBOSS_EWS_DIST`*`/httpd/conf/httpd.conf`** を編集し、# を冒頭に追加することで次の行をコメントアウトします。

   ```
   LoadModule proxy_balancer_module modules/mod_proxy_balancer.so
   ```

   このモジュールは JBoss HTTP Connector と互換性はありません。

4. **JBoss HTTP Connector モジュールにロードするようサーバーを設定します。**

   a. ***JBOSS_EWS_DIST*/httpd/conf.d/JBoss_HTTP.conf** を作成します。

   b. 次の行を ***JBOSS_EWS_DIST*/httpd/conf.d/JBoss_HTTP.conf** に追加します。

```
LoadModule slotmem_module JBOSS_EWS_DIST/modules/mod_slotmem.so
LoadModule manager_module JBOSS_EWS_DIST/modules/mod_manager.so
LoadModule proxy_cluster_module
JBOSS_EWS_DIST/modules/mod_proxy_cluster.so
LoadModule advertise_module
JBOSS_EWS_DIST/modules/mod_advertise.so
```

5. **JBoss Enterprise Web Server Apache サービスを再開します。**
   詳細な方法は JBoss Enterprise Web Server ドキュメントを参照してください。

第6章 基本的なプロキシサーバーの設定

本章の指示に従い、 JBoss HTTP connector mod_cluster を使用するため JBoss Enterprise Web Server を設定します。

6.1. 基本的なプロキシ設定の概要

プロキシサーバーを設定するには、必須とオプションのステップそれぞれ 1 つずつあります。

1. ワーカーノード接続要求とワーカーノードフィードバックを受信するよう Proxy Server リスナを設定します。
2. オプション : サーバーの告知を無効にします。

サーバー告知

プロキシサーバーは UDP マルチキャストを使用してそれ自身を告知できます。UDP マルチキャストがプロキシサーバーとワーカーノード間のネットワーク上で使用可能な場合、サーバーの告知によってプロキシサーバーに別途設定する必要なく、ワーカーノードに最低限の設定だけでワーカーノードを追加できます。

UDP マルチキャストが使用不可能または希望するものでない場合、 「静的プロキシの設定」 で説明されているとおりプロキシサーバーの静的リストでワーカーノードを設定します。どちらの場合でもワーカーノードのリストでプロキシサーバーを設定する必要はありません。

6.2. HTTP CONNECTOR を使用した負荷分散プロキシの設定

本項では JBoss HTTP Connector を使用する負荷分散プロキシの設定方法について説明します。

タスク : プロキシサーバーリスナを設定する

このタスクに従い、JBoss HTTP Connector を使用して JBoss Enterprise Web Server Apache サービスが負荷分散プロキシとして動作するよう設定します。

前提条件

- JBoss Enterprise Web Server をインストールしている。詳細は JBoss Enterprise Web Server 『Installation Guide』 を参照してください。
- JBoss HTTP Connector モジュールをインストールしている。詳細は 5章 *プロキシサーバーコンポーネントのインストール* を参照してください。

1. **プロキシサーバーに対しリッスンディレクティブを作成します。**
   次を追加して、設定ファイル ***JBOSS_EWS_DIST*/httpd/conf.d/JBoss_HTTP.conf** を編集します。

   ```
   Listen IP_ADDRESS:PORT_NUMBER
   ```

   IP_ADDRESS はワーカーノードと通信するためのサーバーネットワークインターフェースの IP アドレスです。 *PORT_NUMBER* はインターフェースがリッスンするポートです。

注記

ポート *PORT_NUMBER* は受信 TCP 接続に対するサーバーファイアウォールでオープンでなくてはなりません。

例6.1 Listen ディレクティブのサンプル

```
Listen 10.33.144.3:6666
```

2. **仮想ホストを作成します。**
次の <VirtualHost> ブロックを ***JBOSS_EWS_DIST*/httpd/conf.d/JBoss_HTTP.conf** ファイルに追加します。

```
<VirtualHost IP_ADDRESS:PORT_NUMBER>

   <Directory>
      Order deny,allow
      Deny from all
      Allow from 10.33.144.
   </Directory>

   KeepAliveTimeout 60
   MaxKeepAliveRequests 0

   ManagerBalancerName mycluster
   AdvertiseFrequency 5

</VirtualHost>
```

IP_ADDRESS と *PORT_NUMBER* は Listen ディレクティブからの値です。

3. **オプション : サーバーの告知を無効にします。**
AdvertiseFrequency ディレクティブの存在により、UDP マルチキャストを介してサーバーが定期的にサーバー告知メッセージを送るようになります。ここでは 5 秒に設定されています。

これらサーバー告知のメッセージには VirtualHost 定義で指定した *IP_ADDRESS* と *PORT_NUMBER* が含まれています。サーバー告知に応答するよう設定されたワーカーノードは、この情報を使用してプロキシサーバーを使って登録します。

サーバー告知を無効にするには、次のディレクティブを **VirtualHost** 定義に加えます。

```
ServerAdvertise Off
```

サーバー告知を無効にする、または UDP マルチキャストがプロキシサーバーとワーカーノード間でのネットワークで使用できない場合は、プロキシサーバーの静的リストでワーカーノードを設定する必要があります。

4. **JBoss Enterprise Web Server Apache サービスを再開します。**
詳細な方法は JBoss Enterprise Web Server ドキュメントを参照してください。

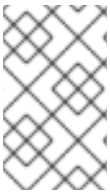
第7章 基本設定でのノードのインストール

本章ではワーカーノードに JBoss HTTP Connector をインストール方法、直ちにノードが動作を開始するための基本設定の実装方法について説明します。

7.1. ワーカーノードの要件

サポートされるワーカーノードタイプ

- JBoss Enterprise Platform 5 JBoss Web コンポーネント
- JBoss Enterprise Web Server Tomcat サービス



注記

JBoss Enterprise Platform ワーカーノードはすべての JBoss HTTP Connector が持つ機能に対応しています。JBoss Enterprise Web Server Tomcat ワーカーノードは JBoss HTTP Connector が持つ機能性のサブセットに対応します。

JBoss HTTP Connector Enterprise Web Server ノードの制約

- 非クラスタ化モードのみである。
- 負荷分散係数を計算する場合、1 度に 1 つの負荷メトリックだけしか使えない。

7.2. ワーカーノードのインストールと設定

本項には多数のタスクがあります。適切なタスクに従い、JBoss HTTP Connector または JBoss Enterprise Web Server でワーカーノードのインストールおよび設定を行ってください。

タスク : JBoss Enterprise Application Platform ワーカーノードのインストールと設定を行う

この手順に従い、JBoss Enterprise Application Platform ノードに JBoss HTTP Connector をインストールし、非クラスタ化運用に対し設定します。

前提条件

- 対応している JBoss Enterprise Application Platform をインストールしている。

1. **ワーカーノードサービスをデプロイします。**
   ***JBOSS_EAP_DIST*/mod_cluster** ディレクトリから **mod-cluster.sar** を **jboss-as/server/*PROFILE*/deploy** にコピーします。

2. **JBoss Web にリスナを追加します。**
   次の **Listener** 要素を ***JBOSS_EAP_DIST*/jboss-as/server/*PROFILE*/deploy/jbossweb.sar/server.xml** の他の Listener の下に追加します。

```
<Listener
className="org.jboss.web.tomcat.service.deployers.MicrocontainerInte
grationLifecycleListener" delegateBeanName="ModClusterService"/>
```

3. **サービスの依存性を設定します。**
   次の **depends** 要素を ***JBOSS_EAP_DIST*/jboss-as/server/*PROFILE*/deploy/jbossweb.sar/META-INF/jboss-beans.xml** の他の **depends** 要素の下に追加します。

```
<depends>ModClusterService</depends>
```

4. **ワーカーに一意のアイデンティティを加えます。**
   以下のとおり **jvmRoute** 属性と値を **Engine** 要素に追加し、***JBOSS_EAP_DIST*/jboss-as/server/*PROFILE*/deploy/jbossweb.sar/server.xml** を編集します。

```
<Engine name="jboss.web" defaultHost="localhost"
jvmRoute="worker01">
```

   各ノードに一意の jvmRoute 値を使用します。

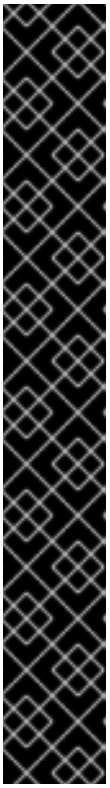
5. **オプション : マルチキャスト Proxy Server 告知を受け取るためにファイアウォールを設定します。**
   JBoss HTTP Connector を使用するプロキシサービスは、UDP マルチキャストを介してそれ自身を告知できます。ワーカーノードが動的にプロキシサービスを検出できるようにするには、ワーカーノードのファイアウォールで UDP 接続のポート **23364** を開きます。

   Linux ユーザーの場合

```
/sbin/iptables -A INPUT -m state --state NEW -m udp -p udp --dport
23364 -j ACCEPT
-m comment --comment "receive mod_cluster proxy server
advertisements"
/sbin/iptables save
```

   Automatic Proxy Discovery (プロキシの自動検出 を参照) を使用していない場合、プロキシの静的リストでワーカーノードを設定します。方法は 「静的プロキシの設定」 を参照してください。この場合、次の警告メッセージは安心して無視してください。

```
[warning] mod_advertise: ServerAdvertise Address or Port not
defined, Advertise disabled!!!
```



重要

ノードが Red Hat Enterpise Linux を稼働している別のマシンにある場合は、互いを自動的には認識しません。JBoss Clustering は jGroups が提供する UDP (User Datagram Protocol) マルチキャスティングに依存しています。

Red Hat Enterprise Linux に同梱している SELinux 設定はこうしたパケットをデフォルトでブロックします。パケットを許可するには、(ルートで) iptables ルールを修正します。次のコマンドは 192.168.1.x に一致する IP アドレスに適用します。

```
/sbin/iptables -I RH-Firewall-1-INPUT 5 -p udp -D
224.0.1.0/24 -j ACCEPT
/etc/init.d/iptables save
/sbin/iptables -I RH-Firewall-1-INPUT 5 -p udp -d
224.0.0.0/4 -j ACCEPT
  /sbin/iptables -I RH-Firewall-1-INPUT 9 -p udp -s
192.168.1.0/24 -j ACCEPT
  /sbin/iptables -I RH-Firewall-1-INPUT 10 -p tcp -s
192.168.1.0/24 -j ACCEPT
  /etc/init.d/iptables save
```

タスク : JBoss Enterprise Web Server Worker Node のインストールと設定を行う

この手順に従い、JBoss Enterprise Web Server ノードに JBoss HTTP Connector をインストールし、非クラスタされた動作に対し設定します。

前提条件

- 対応している JBoss Enterprise Web Server をインストールしている。
- 付録B *参照 : Java プロパティ* で説明されている Proxy Configuration パラメータを理解している。

1. **ワーカーノードサービスをデプロイします。**
   ***JBOSS_EAP_DIST*/mod_cluster/JBossWeb-Tomcat/lib** ディレクトリのすべてのライブラリファイルをコピーします。これらのファイルを ***JBOSS_EWS_DIST*/tomcat6/lib/** に移動します。

2. **Tomcat に Listener を追加します。**
   次の **Listener** 要素を ***JBOSS_EWS_DIST*/*tomcat6*/conf/server.xml** の他の Listener 要素の下に追加します。

   ```
   <Listener className="org.jboss.modcluster.ModClusterListener"
   advertise="true" stickySession="true" stickySessionForce="false"
   stickySessionRemove="true"/>
   ```

3. **このワーカーに一意のアイデンティティを加えます。**
   以下のとおり **jvmRoute** 属性と値を **<Engine>** 要素に追加し、***JBOSS_EWS_DIST*/*tomcat6*/conf/server.xml** を編集します。

   ```
   <Engine name="Catalina" defaultHost="localhost"
   jvmRoute="worker01">
   ```

4. **オプション : Proxy Server の告知を受け取るためにファイアウォールを設定します。**
JBoss HTTP Connector を使用するプロキシサービスは、UDP マルチキャストを介してそれ自身を告知できます。こうしたマルチキャストメッセージを受け取るには、ワーカーノードのファイアウォールで UDP 接続のポート 23364 を開きます。

   Linux ユーザーの場合

   ```
   /sbin/iptables -A INPUT -m state --state NEW -m udp -p udp --dport
    23364 -j ACCEPT
   -m comment -comment "receive mod_cluster proxy server
   advertisements"
   ```

   Automatic Proxy Discovery (プロキシの自動検出 を参照) を使用していない場合、プロキシの静的リストでワーカーノードを設定します。方法は 「静的プロキシの設定」 を参照してください。この場合、次の警告メッセージは安心して無視してください。

   ```
   [warning] mod_advertise: ServerAdvertise Address or Port not
   defined, Advertise disabled!!!
   ```

第8章 サーバーの追加設定

本章ではサーバーの追加設定の実装について説明します。

8.1. APACHE サーバーディレクティブ

サーバー全体に影響するよう次の Apache サーバーディレクティブを Apache サーバー設定に追加します。

8.1.1. CreateBalancers

CreateBalancers ディレクティブは HTTP バランサが VirtualHosts でどのように作成されるか決定します。

CreateBalancers の値

0

すべての VirtualHosts でバランサを作成します。

1

バランサは作成しません。

2

メインサーバーにのみバランサを作成します。

第9章 詳細設定

本章では JBoss HTTP Connector の高度な機能の設定について説明します。

9.1. 静的プロキシの設定

サーバー告知によりワーカーノードはプロキシサーバーで動的にそれらを検出し、登録することができます。UDP ブロードキャストが使用不可能、またはサーバー告知が無効の場合は、ワーカーノードはプロキシサーバーアドレスおよびポートの静的リストで設定する必要があります。

タスク : 静的プロキシリストで Application Platform ワーカーノードを設定する

このタスクに従い、JBoss Enterprise Application Platform ワーカーノードがプロキシサーバーの静的リストと動作するよう設定します。

前提条件

- JBoss Enterprise Application Platform ワーカーノードが設定されている。方法は *7章 基本設定でのノードのインストール* を参照してください。

1. **動的プロキシの検出を無効にします。**
   ファイル ***JBOSS_EAP_DIST*/jboss-as/server/*PROFILE*/mod-cluster.sar/META-INF/mod-cluster-jboss-beans.xml** を編集し、**advertise** プロパティを false に設定します。

   ```
   <property name="advertise">false</property>
   ```

2. 次の静的プロキシのオプションからひとつを選択し、実装します。

   - **オプション 1 : 静的プロキシサーバーのリストを作成する。**
     ***JBOSS_EAP_DIST*/jboss-as/server/*PROFILE*/mod-cluster.sar/META-INF/mod-cluster-jboss-beans.xml** ファイルを編集し、**proxyList** プロパティの *IP_ADDRESS:PORT* の形式でコンマで区切られたプロキシのリストを追加します。

     例9.1 静的プロキシリストのサンプル

     ```
     <property
     name="proxyList">10.33.144.3:6666,10.33.144.1:6666</property>
     ```

   - **オプション 2 : パラメータとして静的プロキシリストでワーカーノードを開始する。**

     a. ***JBOSS_EAP_DIST*/server/*PROFILE*/mod-cluster.sar/META-INF/mod-cluster-jboss-beans.xml** を編集します。

     b. 次の行を追加します。

     ```
     <property name="domain">${jboss.modcluster.domain:}</property>
     ```

     c. ノードを開始するとき、**jboss.modcluster.proxyList** パラメータとしての *IP_ADDRESS:PORT* の形式でプロキシのコンマで区切られたリストを追加します。

     例9.2 静的プロキシリストのパラメータのサンプル

```
-Djboss.modcluster.domain=10.33.144.3:6666,10.33.144.1:6666
```

タスク : 静的プロキシリストで Web Server ワーカーノードを設定する。

この手順に従い、プロキシサーバーの静的リストと動作するよう JBoss Enterprise Web Server ワーカーノードを設定します。

前提条件

- JBoss Enterprise Web Server ワーカーノードが設定されている。方法は *7章 基本設定でのノードのインストール* を参照してください。
- 付録B *参照 : Java プロパティ* で説明されている Proxy Configuration パラメータを理解している。

1. **動的プロキシの検出を無効にします。**
   ***JBOSS_EWS_DIST*/*tomcat6*/conf/server.xml** ファイルを編集し、ModClusterListener の **advertise** プロパティを false に設定します。

2. **mod_cluster リスナを定義します。**
   <Listener> 要素を **server.xml** に追加します。

```
<Listener className="org.jboss.modcluster.ModClusterListener"
advertise="false" stickySession="true" stickySessionForce="false"
stickySessionRemove="true"/>
```

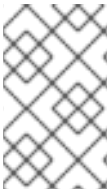
3. **静的プロキシサーバーのリストを作成します。**
   ModClusterListener <Listener> 要素の **proxyList** プロパティとして *IP_ADDRESS:PORT* の形式でプロキシのコンマで区切られたリストを追加します。

例9.3 静的プロキシリストのサンプル

```
<Listener className="org.jboss.modcluster.ModClusterListener"
advertise="false" stickySession="true" stickySessionForce="false"
stickySessionRemove="true"
proxyList="10.33.144.3:6666,10.33.144.1:6666"/>
```

9.2. クラスター化したノードの動作

JBoss HTTP Connector は非クラスタモードまたはクラスタモードで動作します。



注記

JBoss Enterprise Application Platform ノードのみ JBoss HTTP Connector を使ったクラスタ化した動作に対応します。JBoss Enterprise Web Server ノードは非クラスタ化した動作のみに対応します。

JBoss HTTP Connector の非クラスタ化した動作

非クラスタモードでは、各ノードは直接プロキシと通信します。

JBoss HTTP Connector クラスタ化した動作

クラスタモードでは、複数のワーカーノードが JBoss HA (High Availability) クラスタドメインを形成します。単一のワーカーノードがクラスタドメインの他のノードに代わってプロキシと通信します。

第10章 負荷分散のデモ

JBoss HTTP Connector には負荷分散のデモが含まれており、サーバー側のシナリオが負荷分散プロキシサーバーによって実行されたクライアント要求のルーティングに与える影響がどう違うか示しています。 必須の設定は **jboss-eap-5.1/mod_cluster/demo** ディレクトリにあります。

アプリケーションは 2 つの主要なコンポーネントで成り立っています。

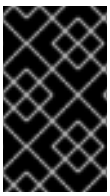
/server/load-demo.war

JBoss Enterprise Application Platform または JBoss Enterprise Web Server でデプロイされる WAR ファイルです。この WAR には多くのサーブレットが含まれています。

/client/lib/mod-cluster-demo.jar

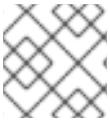
ユーザーがスレッドのプールを起動でき、ロードバランサを介して **load-demo.war** のプライマリサーブレットに要求を送信する Web アプリケーション。このアプリケーションは要求を処理するサーバーに関する情報を表示します。また、**load-demo.war** の負荷生成サーブレットにも個別の要求を送信でき、ユーザーは特定の負荷条件がどのように要求の負荷分散に影響を与えるかを確認できます。

デモは、さまざまなワーカーサイドシナリオがどのようにプロキシサーバーのルーティング決定に影響を与えるかを示すために使用できます。



重要

JBoss Enterprise Web Server でデモを実行する場合、利用可能なメトリクスは、System および JVM メトリクスのみとなっています。Load-demo app は Tomcat 6 で他のメトリクスとやり取りを行うように設計されているわけではありません。



注記

デモは、クラスタ構成が処理できる最大負荷を示しません。

10.1. デモの設定

以下の手順は、デモの設定方法と起動方法について簡単に示しています。これらの手順については、詳しく説明されます。デモが実行されたら、「デモとの対話」を参照してください。

タスク: デモの起動

このタスクを完了すると、デモの基本要件が設定されます。

前提条件

- ワーカーノードをインストールおよび設定します。「ワーカーノードのインストールと設定」を参照してください。
- プロキシサーバーをインストールおよび設定します。「静的プロキシの設定」を参照してください。

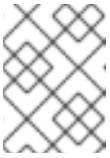
1. **プロキシサーバーの起動**
   ***HTTPD_DIST*/sbin** に移動し、プロキシサーバーを起動します。

```
[sbin]$ apachectl start
```

2. **ワーカーノードの起動**
   ターミナルで、以下のコマンドを実行します。

   - JBoss Enterprise Web Server の場合:

     ```
     [home]$ ./JBOSS_EWS_DIST/tomcat6/bin/startup.sh
     ```

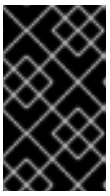
     注記

     JBoss Enterprise Web Server 上で実行Sul場合は、jvm メトリクスのみが Load-demo アプリケーションで利用可能となっています。

   - JBoss Enterprise Application Platform の場合:

     ```
     [home]$ ./JBOSS_EAP_DIST/bin/run.sh
     ```

3. **JBoss Enterprise Web Server では、Catalina Service Name を指定します。**

   重要

   この手順は JBoss Enterprise Web Server で Tomcat 6 を利用する場合のみ該当します。JBoss Enterprise Application サーバーでデモアプリケーションを実行する場合はこの手順を飛ばしてください。

   ***$JBOSS_EWS_DIST*/mod_cluster/src/demo/resources/web.xml** にて、**<web-app>** 配下に、サービスとして Catalina を指定する **<context-param>** ディレクティブを追加します。

   ```
   <context-param>
      <param-name>service-name</param-name>
      <param-value>Catalina</param-value>
   </context-param>
   ```

4. **ワーカーノードへのデモ Web アーカイブのデプロイ**
   ***JBOSS-EAP_DIST*/mod_cluster/demo/server** の **load-demo.war** を以下のディレクトリにコピーします。

   - JBoss Enterprise Web Server の場合:

     ***JBOSS_EWS_DIST*/*tomcat6*/webapps**

   - JBoss Enterprise Application Platform の場合:

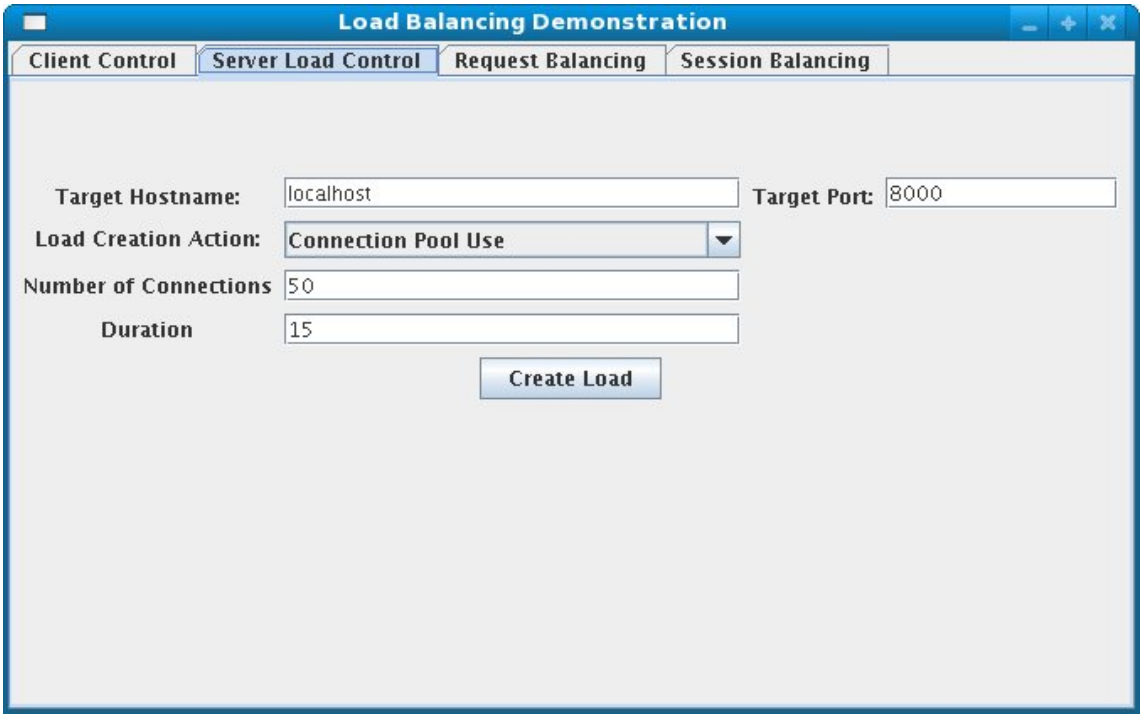
     ***JBOSS_EAP_DIST*/jboss-as/server/*PROFILE*/deploy**

5. **デモの起動**
   ***JBOSS_EAP_DIST*/mod_cluster/demo/client/** に移動し、デモを起動します。

   ```
   [client]$ ./run-demo.sh
   ```

結果

デモが起動し、[Load Balancing Demonstration] ウィンドウが開きます。タスク: [Client Control] タブフィールドの設定 に進みます。



10.2. デモクライアントの設定

デモの Client Control パラメータを設定し、デモをとおしてクライアントが期待どおりに稼働するようにする必要があります。

タスク: [Client Control] タブフィールドの設定

このタスクを完了し、[Load Balancing Demonstration] の **[Client Control]** タブで指定する必要がある値について学習します。

前提条件

このタスクを続行する前に タスク: デモの起動 を完了します。

1. **[Client Control]** タブをクリックします。
2. 以下のフィールド定義に基づいて、**[Client Control]** タブのすべてのフィールドに対して値を指定します。

Proxy Hostname

負荷分散プロキシサーバーのホスト名またはプロキシサーバーが要求をリッスンしている IP アドレス。このフィールドのデフォルト値は **localhost** であるか、または **mod_cluster.proxy.host** システムプロパティにより決定されます (設定されている場合)。

デモを使用するたびにこの値を再設定することを回避するために、**run-demo.sh** の **-Dmod_cluster.proxy.host=localhost** 値を編集します。

Proxy Port

負荷分散サーバーが要求をリッスンするポート。デフォルト値は **8000** であるか、または **mod_cluster.proxy.port** プロパティによって決定されます (設定されている場合)。

デモを使用するたびにこの値を再設定することを回避するために、**run-demo.sh** の **-Dmod_cluster.proxy.port=8000** 値を編集します。

Context Path

要求を指定する要求 URL の部分は **load-demo.war** です。

Session Life

クライアントスレッドがセッションを無効にし、破棄する前にセッションを使用する時間 (秒数)。これは小さい値である必要があります。小さい値でないと、セッションスティッキネスによりサーバー負荷の変化がプロキシサーバーのルーティング決定に影響を与えなくなります。スティッキーセッションが有効な場合 (強く推奨) は、新しいセッションを作成すると、プロキシが負荷を分散します。

Invalidate

チェックされた場合、スレッドがセッションの使用を停止したときにセッションがクライアントスレッドにより無効になります。チェックされた場合、セッションは破棄され、セッションタイムアウトが終了するまでワーカーノードに存在します。

Session Timeout

ワーカーノードの期限が切れ、セションが削除されるまでセションが未使用のままになる時間 (秒数)。

Invalidate を選択解除し、セッションライフよりも大きい値を設定すると、大量の未使用のセッションがサーバーに蓄積されます。

Num Threads

起動するクライアントスレッドの数。各スレッドは **[Stop]** ボタンが押されるか、要求が HTTP 200 以外の応答を受けとるまで要求を繰り返し行います。

Sleep Time

要求間にクライアントスレッドがスリープ状態にする (ミリ秒数）。

Startup Time

アプリケーションがクライアントスレッドの起動を整理する時間 (秒数)。セッションの起動時間を整理すると、すべてのセッションが開始され、ほとんど同時に終了する非現実的な状況が回避されます。起動時間を整理すると、プロキシが新しいセッションを連続的に確認し、セッションをルーティングする方法を決定します。

3. 値を指定したら、タスク: デモとの対話 に進みます。

10.3. デモとの対話

用語

アクティブセッション

セッションは、クライアントスレッドがセッションに関連付けられた要求を送信する場合にアク

ティブと見なされます。クライアントスレッドはセッションの使用を停止すると、セッションを無効にする要求を送信するか、該当するセッションクッキーを含めないようにすることによりセッション破棄します。

セッションが破棄されると、セッションは **Session Balancing** チャートで反映されなくなりますが、セッションタイムアウト値に基づいて削除されるまでワーカーノードに存在し続けます。

クライアント合計数

[Start] ボタンがクリックされてから作成されたクライアントスレッドの数。

ライブクライアント数

現在実行されているクライアントスレッド数。

失敗したクライアント数

HTTP 200 応答以外の結果となる要求などを異常終了したクライアントスレッドの数。

この項では、デモの設定方法とデモの使用の開始方法について説明します。

タスク: デモとの対話

このタスクを完了し、デモで負荷分散を実験します。

タスクの前提条件

- タスク: デモの起動 を完了します。
- タスク: [Client Control] タブフィールドの設定 を完了します。

1. **[Request Balancing]** タブをクリックし、各ワーカーノードに送信される要求の数を確認します。
2. **[Session Balancing]** タブをクリックし、各ワーカーノードによりホストされるアクティブセッションの数を確認します。
3. いくつかのワーカーノードを停止するか、**`load-demo.war`** をデプロイ解除し、要求およびセッション分散に与える影響を確認します。
4. いくつかのワーカーノードを再起動するか、**`load-demo.war`** を再デプロイし、要求およびセッション分散に与える影響を確認します。
5. 1 つまたは複数のワーカーノードに人工的な負荷を追加して実験を行い、負荷およびセッション分散に対する影響を確認します (詳細については、「人工的な負荷の生成」 を参照してください)。

10.3.1. 人工的な負荷の生成

負荷分散デモを使用してさまざまな種類の負荷を生成するようワーカーノードに指示し、その負荷が要求およびセッション分散にどのように影響を与えるかを追跡できます。負荷生成は **[Server Load Control]** タブにより制御されます。

ターゲットホスト名、ターゲットポート

負荷を生成するサーバーのホスト名とポート番号。これらの値を設定するには 2 つの方法があります。

1. プロキシサーバーのホスト名とポートを使用します。プロキシは負荷をワーカーノードにルーティングします。ただし、クライアントはこれらの要求のセッションクッキーを保持しないため、生成された負荷は同じワーカーに必ずしもルーティングされません。

2. ワーカーノードが HttpConnector と AJP コネクタを実行している場合、ワーカーの HttpConnector がリッスンしている IP アドレスとポートを指定できます (デフォルト値は **8080**)。

負荷作成アクション

ワーカーノードが生成する負荷の種類を指定します。

利用可能なアクション数

アクティブセッション

ターゲットサーバーでセション作成を開始することによりサーバー負荷を生成します。

データソース使用

設定された時間に対して **java:DefaultDS** データソースから接続を取得することによりサーバー負荷を生成します。

接続プール使用

設定された時間に対する webserver 接続プールのスレッドをブロックすることによりサーバー負荷を生成します。

ヒープメモリプール使用

設定された時間に対して空きヒープメモリの 50% を占有することによりサーバー負荷を生成します。

CPU 使用

スレッドでタイトループを開始することによりサーバー CPU 負荷を生成します。

サーバー受信トラフィック

設定された時間の間、1 秒ごとにサーバーに大きいバイトアレイを転送することによりサーバートラフィック受信負荷を生成します。

サーバー送信トラフィック

1 秒ごとに要求 (サーバーが大きいバイトアレイで応答する) を行うことによりサーバートラフィック送信負荷を生成します。

要求数

複数の要求を行うことによりサーバー負荷を生成し、ターゲットサーバーの要求数を増やします。

パラメータ

指定された負荷作成サーブレットに渡すゼロ以上のパラメータ (スクリーンショットに示された接続数や時間など)。表示されるパラメータ、パラメータ名、およびその意味は選択された負荷作成アクションによって異なります。各パラメータのラベルには、使用方法を説明するヒントが含まれます。

パート III. INTERNET SERVER API (ISAPI)

第11章 概要

この章では、Internet Server Application Programming Interface (ISAPI) について簡単に説明します。

11.1. INTERNET SERVER API とは

Internet Server Application Programming Interface (ISAPI) は、Microsoft Internet Information Services (IIS) Web サーバーと互換性のある他のサードパーティ Web サーバー用のマルチ階層アプリケーションプログラミングインターフェースです。

ISAPI アプリケーションには以下の 2 つのアプリケーションタイプが存在します。

- 拡張機能 (IIS で実行される完全なアプリケーション)
- フィルタ (IIS の機能に関連する要求を持続的にフィルタリングすることにより IIS 機能を変更または拡張するアプリケーション)。

拡張機能またはフィルタを Dynamic Link Library (DLL) ファイルにコンパイルすることにより ISAPI アプリケーションを実装します。DLL は DLL を使用できるようにする Web サーバーで登録されます。

第12章 WINDOWS での ISAPI コネクターの設定

この章では、Windows Server 2003 または 2008 マスターノード向けのワーカーノードとして JBoss Enterprise Application Platform を使用するよう ISAPI コネクタを設定する方法について説明します。

12.1. 前提条件と設定

後続タスクを続ける前に以下の前提条件を満たすようにしてください。

重要

この章のタスクでは、すべてのサーバーインスタンスが同じマシン上にあることを前提とします。各インスタンスに異なるマシンを使用するには、**-b** スイッチを使用して JBoss Enterprise Platform のインスタンスをパブリック IP アドレスにバインドします。IP アドレスの変更を反映するよう IIS マシン上の **workers.properties** ファイルを編集します。

- テクノロジーのいずれかの組み合わせとオペレーティングシステムとアーキテクチャ専用のネイティブバイナリをインストールすることにより、マスターノードを設定します。
  - Windows Server 2003 (32 ビット) と Microsoft IIS 6
  - Windows Server 2003 (64 ビット) と Microsoft IIS 6
  - Windows Server 2008 (32 ビット) と Microsoft IIS 7.0
  - Windows Server 2008 (64 ビット) と Microsoft IIS 7.0
- JBoss Enterprise Application Platform 5.1.0 以降をインストールすることによりワーカーノードを設定します。ネイティブコンポーネントはワーカーノードのオプションです。

注記

インストール手順については『インストールガイド (『Installation Guide』)』を参照し、ネイティブコンポーネントの詳細について学習してください。

12.2. ワーカーノードとしてのサーバーインスタンスの設定

タスク: ワーカーノードとしてサーバーインスタンスを設定

Microsoft Internet Information Services (IIS) と使用するために JBoss Enterprise Application Platform インスタンスをワーカーノードとして適切に設定するためにこのタスクを完了します。

前提条件

- 「前提条件と設定」

1. **各ワーカーノードに対してサーバープロファイルを作成する**
   ワーカーノードとして設定する **default** サーバープロファイルのコピーを作成し、**default-01** に名前を変更します。

2. **各インスタンスに一意の名前を付ける**
   新しい各ワーカーインスタンスの **deploy\jbossweb.sar\server.xml** ファイルの以下の行

を編集します。

```
<Engine name="jboss.web" defaultHost="localhost">
```

示しされたように一意の **jvmRoute** 値を追加します。この値はクラスタ内のこのノードの ID です。

default-01 サーバープロファイルの場合:

```
<Engine name="jboss.web" defaultHost="localhost"
jvmRoute="worker01">
```

default-02 サーバープロファイルの場合:

```
<Engine name="jboss.web" defaultHost="localhost"
jvmRoute="worker02">
```

3. **セッション処理を有効にする**
   各ワーカーノードの **deployers\jbossweb.deployer\META-INF\war-deployers-jboss-beans.xml** ファイルの以下の行をコメント解除します。

```
<property name="useJK">false</property>
```

   このプロパティは、mod_jk と他のコネクタバリアントを調整するために特別なセッション処理を使用するかどうかを制御します。両方のワーカーノードでこのプロパティを **true** に設定します。

```
<property name="useJK">true</property>
```

4. **ワーカーノードを起動します。**
   個別のコマンドラインインターフェースで各ワーカーノードを起動します。**--host** スイッチを使用して各ノードが異なる IP アドレスにバインドされるようにします。

```
JBOSS_EAP_DIST\bin\run.bat --host=127.0.0.1 -c default-01
```

```
JBOSS_EAP_DIST\bin\run.bat --host=127.0.0.100 -c default
```

12.3. MICROSOFT IIS 6 初期クラスタリング設定

Microsoft IIS 6 には、基本的な ISAPI フィルタと ISAPI マッピングがデフォルトインストールの一部として含まれます。

タスク: ISAPI フィルタの定義

このタスクでは、管理コンソールを使用してマスターサーバーで ISAPI フィルタを定義します。

1. マスターサーバーで、IIS マネージャを開きます。
   - **[スタート]** → **[実行]**、次に **inetmgr** と入力します。

- **[スタート]** → **[コントロール パネル]** → **[管理ツール]** → **[インターネット インフォメーション サービス (IIS) マネージャ]**

[IIS 6 Manager] ウィンドウが開きます。

2. ツリービューペインで、**[Web サイト]** を展開します。
3. **[既定の Web サイト]** を右クリックし、次に **[プロパティ]** をクリックします。

   [プロパティ] ウィンドウが開きます。
4. **[ISAPI フィルタ]** をクリックします。
5. **[追加]** ボタンをクリックし、**[フィルタのプロパティの追加と編集]** ウィンドウで以下の値を指定します。

   [フィルタ名]:

   jboss を (記述されたとおりに) 指定します。

   [実行可能ファイル]:

   C:\connectors\jboss-ep-5.1\native\sbin\isapi_redirect.dll を指定します。
6. **[OK]** をクリックして値を保存し、**[フィルタのプロパティの追加と編集]** ダイアログを閉じます。

注記

IIS はまだリソースの要求を受け取っていないため、**[ISAPI フィルタ]** タブでは **jboss** フィルタステータスと優先度が [不明] と表示されます。要求が実行されると、ステータスと優先度はそれぞれ [読み込み済み] と [高] に変わります。

タスク: ISAPI 仮想ディレクトリを定義する

このタスクでは、IIS 管理コンソールを使用して ISAPI 仮想ディレクトリを定義します。

1. **[既定の Web サイト]** を右クリックし、**[新規]** → **[仮想ディレクトリの追加]**をクリックします。

   [仮想ディレクトリの追加] ウィンドウが開きます。
2. **[仮想ディレクトリの追加]** ウィンドウで以下の値を指定します。

   [エイリアス]:

   jboss を (記述されたとおりに) 指定します。

   [物理パス]:

   C:\connectors\jboss-ep-5.1\native\sbin と指定します。
3. **[OK]** をクリックして値を保存し、**[仮想ディレクトリの追加]** ウィンドウを閉じます。
4. ツリービューペインで、**[Web サイト]** → **[既定の Web サイト]**を展開します。

5. **jboss** 仮想ディレクトリを右クリックし、**[プロパティ]** をクリックします。
6. **[仮想ディレクトリ]** タブをクリックし、以下の変更を行います。

   [実行アクセス許可]:
   Scrips and Executables を選択します。

   [Read (読み取り)] チェックボックス:
   読み取りアクセスを有効にします。

7. [OK] をクリックして値を保存し、**[jboss Properties]** ウィンドウを閉じます。

タスク: ISAPI Web サービス拡張を定義する

このタスクでは、管理コンソールを使用して ISAPI Web サービス拡張を定義します。

1. **[Web サービス拡張]** をクリックし、**[タスク]** グループで、**[新しい Web サービス拡張を追加]** を選択します。

   [新しい Web サービス拡張] ウィンドウが表示されます。

2. 以下の値を **[新しい Web サービス拡張]** ウィンドウに追加します。

   [Extension name (拡張名)]
   jboss を指定します。

   [必要なファイル]:
   パス **C:\connectors\jboss-ep-5.1\native\sbin\isapi_redirect.dll** を指定します。

   [拡張の状態を許可済みに設定する]:
   チェックボックスを選択します。

3. [OK] をクリックして値を保存し、**[新しい Web サービス拡張]** ウィンドウを閉じます。
4. **jboss** Web Service Extension がリストに表示されていることを確認します。

12.4. MICROSOFT IIS 7 初期クラスタリング設定

Microsoft IIS 7 は、管理コンソールを使用して管理したり、**APPCMD.EXE** コマンドツールを使用してコマンドプロンプトから管理したりできます。

用語

ISAPI 制限と CGI 制限

ISAPI 制限と CGI 制限はサーバーでの動的コンテンツの実行を可能にする要求ハンドラです。これらの制限は CGI ファイル (**.exe**) または ISAPI 拡張 (**.dll**) のいずれかです。IIS 設定システムがこれを許可する場合は、カスタム ISAPI 制限または CGI 制限を追加できます。
[http://technet.microsoft.com/en-us/library/cc730912(WS.10).aspx]

タスク: JBoss ネイティブ ISAPI 制限を定義する

このタスクでは、管理コンソールを使用して ISAPI 制限を定義します。

1. マスターサーバーで、IIS マネージャを開きます。
   - **[スタート]** → **[実行]**、次に **inetmgr** と入力します。
   - **[スタート]** → **[コントロール パネル]** → **[管理ツール]** → **[インターネット インフォメーション サービス (IIS) マネージャ]**

   [IIS 7 Manager] ウィンドウが開きます。
2. ツリービューペインで、**[IIS 7** (Server Home と呼ばれます) を選択します。

   [IIS 7 Home Features View] が開きます。
3. **[ISAPI および CGI の制限]** をダブルクリックします。

   [ISAPI および CGI の制限] 機能ビューが開きます。
4. **[Actions (アクション)]** ペインで、**[追加]** をクリックします。

   [ISAPI または CGI の制限の追加] ウィンドウが開きます。
5. 以下の値を **[ISAPI または CGI の制限の追加]** ウィンドウで指定します。

   [ISAPI または CGI パス]:

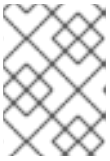
   c:\connectors\jboss-ep-5.1\native\sbin\isapi_redirect.dll と指定します。

   [説明]:

   jboss と指定します (記述されたとおり)。

   [拡張パスの実行を許可する]

   チェックボックスを選択します。
6. **[OK]** をクリックして値を保存し、**[ISAPI または CGI の制限の追加]** ウィンドウを閉じます。



注記

[ISAPI および CGI の制限] 機能ビューに、**jboss** 制限とパスが表示されます。

タスク: JBoss ネイティブ仮想ディレクトリを定義する

このタスクでは、管理コンソールを使用して、JBoss ネイティブバイナリの仮想ディレクトリを定義します。

1. **[既定の Web サイト]** を右クリックし、**[仮想ディレクトリの追加]** をクリックします。

   [仮想ディレクトリの追加] ウィンドウが開きます。

2. **[仮想ディレクトリの追加]** ウィンドウで以下の値を指定します。

   [エイリアス]:
   jboss を (記述されたとおりに) 指定します。

   [物理パス]:
   C:\connectors\jboss-ep-5.1\native\sbin と指定します。

3. **[OK]** をクリックして値を保存し、**[仮想ディレクトリの追加]** ウィンドウを閉じます。

タスク: JBoss ネイティブ ISAPI リダイレクトフィルタを定義する

このタスクでは、管理コンソールを使用して JBoss ネイティブ ISAPI リダイレクトフィルタを定義します。

1. ツリービューペインで、**[サイト]** → **[既定の Web サイト]**を展開します。
2. **[ISAPI フィルタ]** をダブルクリックします。

   [ISAPI フィルタ] 機能ビューが開きます。
3. **[Actions (アクション)]** ペインで、**[追加]** をクリックします。

   [ISAPI フィルタの追加] ウィンドウが開きます。
4. 以下の値を **[ISAPI フィルタの追加]** ウィンドウで指定します。

   [フィルタ名]:
   jboss を (記述されたとおりに) 指定します。

   [実行可能ファイル]:
   C:\connectors\jboss-ep-5.1\native\sbin\isapi_redirect.dll を指定します。

5. **[OK]** をクリックして値を保存し、**[ISAPI フィルタの追加]** ウィンドウを閉じます。

タスク: ISAPI-dll ハンドラを有効にする

このタスクでは、管理コンソールを使用して ISAPI-dll ハンドラを有効にします。

1. ツリービューペインで、**[IIS 7** (Server Home と呼ばれます) を選択します。

   [IIS 7 Home Features View] が開きます。
2. **[ハンドラ マッピング]** をダブルクリックします。

   [ハンドラ マッピング] 機能ビューが開きます。
3. **[グループ化]** ドロップダウンボックスで、**[State (状態)]** を選択します。

   ハンドラマッピングが Enabled グループと Disabled グループで表示されます。

4. **ISAPI-dll** が Disabled グループに存在する場合は、右クリックして **[機能のアクセス許可の編集]** を選択します。

5. **[Read (読み取り)]**、**[Script (スクリプト)]**、および **[Execute (実行)]** チェックボックスが選択されていることを確認します。

6. **[OK]** をクリックして値を保存し、**[機能のアクセス許可の編集]** ウィンドウを閉じます。

12.5. ISAPI を使用した基本的なクラスターの設定

タスク: 基本的なクラスタを提供するために ISAPI を設定する

このタスクでは、単一の IP アドレスですべてのサーバーで共通のアプリケーションを管理し、アプリケーション要求を適切なサーバーインスタンスにルーティングするために ISAPI を設定します。

ISAPI クラスタを設定する時にこの設定をサンプルとして使用します。

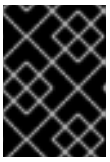
注記

このタスクは、負荷分散またはサーバー停止フェールオーバーに関する手順を提供しません。設定手順については、「ISAPI を使用した負荷分散クラスターの設定」 を参照してください。

前提条件

- 関連する Microsoft IIS クラスタリングセットアップ手順を完了します。詳細については、「Microsoft IIS 6 初期クラスタリング設定」 または 「Microsoft IIS 7 初期クラスタリング設定」 を参照してください。
- 以下の手順では、ログ、プロパティファイル、および NSAPI ロックを保存するために **C:\connectors** ディレクトリが使用されていることを前提としています。

1. **isapi_redirect.properties ファイルを作成する**
   C:\connectors\jboss-ep-5.1\native\sbin ディレクトリで **isapi_redirect.properties** という名前の新しいファイルを作成します。

   重要

   isapi_redirect.properties ファイルは、**isapi_redirect.dll** ファイルと同じディレクトリに存在する必要があります。

   isapi_redirect.properties に次の情報を追加します。

```
# Configuration file for the ISAPI Redirector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll

# Full path to the log file for the ISAPI Redirector
log_file=c:\connectors\isapi_redirect.log

# Log level (debug, info, warn, error or trace)
# Use debug only testing phase, for production switch to info
log_level=debug
```

```
# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkermap.properties file
worker_mount_file=c:\connectors\uriworkermap.properties

#Full path to the rewrite.properties file
rewrite_rule_file=c:\connectors\rewrite.properties
```

2. **オプション: rewrite.properties ファイルを作成する**
   rewrite.properties ファイルを使用すると、アプリケーションに固有の単純な URL 書き換えを指定できます。この設定ファイルはオプションであり、URL 書き換えが必要ない場合は **isapi_redirect.properties** から実行できます。

   提供される機能は Apache mod_rewrite に類似しますが、それほど強力ではありません。書き換えパスは名前と値のペアを使用して指定します。単純な例では、app_01 アプリケーションが、イメージを含む抽象的なディレクトリ名を持ち、そのディレクトリをもっと直感的なものに再マップします。

   ```
   #Simple example, images are accessible under abc path
   /app-01/abc/=/app-01/images/
   ```

3. **uriworkermap.properties ファイルを作成する**
   uriworkermap.properties ファイルには、デプロイされたアプリケーションのマッピング設定情報が含まれます。以下の行をこのファイルに追加します。

   ```
   # images and css files for path /status are provided by worker01
   /status=worker01
   /images/*=worker01
   /css/*=worker01

   # Path /web-console is provided by worker02
   # IIS (customized) error page is used for http errors with number
   greater or equal to 400
   # css files are provided by worker01
   /web-console/*=worker02;use_server_errors=400
   /web-console/css/*=worker01

   # Example of exclusion from mapping, logo.gif won't be displayed
   # !/web-console/images/logo.gif=*

   # Requests to /app-01 or /app-01/something will be routed to
   worker01
   /app-01|/*=worker01

   # Requests to /app-02 or /app-02/something will be routed to
   worker02
   /app-02|/*=worker02
   ```

4. **workers.properties ファイルを作成する**
   worker.properties ファイルには、ワーカーラベルとサーバーインスタンス間のマッピング定義が含まれます。以下の行をこのファイルに追加します。

```
# An entry that lists all the workers defined
worker.list=worker01, worker02

# Entries that define the host and port associated with these
workers

# First EAP server definition, port 8009 is standard port for AJP in
EAP
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

# Second EAP server definition
worker.worker02.host= 127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

5. **IIS を再起動する**
   変更内容を反映するために IIS サーバーを再起動します。実行している IIS バージョンに対して以下のコマンドを実行します。

   IIS 6

   ```
   C:\> net stop iisadmin /Y
   C:\> net start w3svc
   ```

   IIS 7

   ```
   C:\> net stop was /Y
   C:\> net start w3svc
   ```

6. **ログを検証する**
   IIS が再起動されたら ISAPI ログを確認します。このログは **`isapi_redirect.properties`** の **log_file** プロパティで指定されたファイルの場所に保存されます。また、他のイベントについて IIS ログとイベントビューアを確認する必要があります。

12.6. ISAPI を使用した負荷分散クラスターの設定

タスク: 負荷分散クラスタを提供するために ISAPI を設定する

このタスクでは、すべてのサーバーで共通のアプリケーションを管理し、要求を JBoss Enterprise Application Platform インスタンスにルーティングし、一部のノードがオンラインでない場合や接続障害が発生した場合にライブノードに要求をリダイレクトするために ISAPI を設定します。

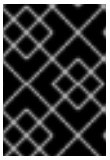
ISAPI クラスタを設定する時にこの設定をサンプルとして使用します。

前提条件

- 関連する Microsoft IIS クラスタリングセットアップ手順を完了します。詳細については、「Microsoft IIS 6 初期クラスタリング設定」 または 「Microsoft IIS 7 初期クラスタリング設定」 を参照してください。

- 以下の手順では、ログ、プロパティファイル、および NSAPI ロックを保存するために **C:\connectors** ディレクトリが使用されていることを前提としています。

1. **isapi_redirect.properties ファイルを作成する**
   C:\connectors\jboss-ep-5.1\native\sbin ディレクトリで **isapi_redirect.properties** という名前の新しいファイルを作成します。

   重要

   isapi_redirect.properties ファイルは、**isapi_redirect.dll** ファイルと同じディレクトリに存在する必要があります。

   以下の設定をファイルに追加します。

```
# Configuration file for the ISAPI Redirector
# Extension uri definition
extension_uri=/jboss/isapi_redirect.dll

# Full path to the log file for the ISAPI Redirector
log_file=c:\connectors\isapi_redirect.log

# Log level (debug, info, warn, error or trace)
# Use debug only testing phase, for production switch to info
log_level=debug

# Full path to the workers.properties file
worker_file=c:\connectors\workers.properties

# Full path to the uriworkermap.properties file
worker_mount_file=c:\connectors\uriworkermap.properties

#OPTIONAL: Full path to the rewrite.properties file
rewrite_rule_file=c:\connectors\rewrite.properties
```

2. **オプション: rewrite.properties ファイルを作成する**
   rewrite.properties ファイルを使用すると、アプリケーションに固有の単純な URL 書き換えを指定できます。この設定ファイルはオプションであり、URL 書き換えが必要ない場合は **isapi_redirect.properties** から実行できます。

   提供される機能は Apache mod_rewrite に類似しますが、それほど強力ではありません。書き換えパスは名前と値のペアを使用して指定します。単純な例では、app_01 アプリケーションが、イメージを含む抽象的なディレクトリ名を持ち、そのディレクトリをもっと直感的なものに再マップします。

```
#Simple example, images are accessible under abc path
/app-01/abc/=/app-01/images/
```

3. **uriworkermap.properties ファイルを作成する**
   uriworkermap.properties ファイルには、デプロイされたアプリケーションのマッピング設定情報が含まれます。以下の行をこのファイルに追加します。

```
# images, css files, path /status and /web-console will provided by
nodes defined in load-balancer
```

```
/css/*=router
/images/*=router
/status=router
/web-console|/*=router

# Example of exclusion from mapping, logo.gif won't be displayed
!/web-console/images/logo.gif=*

# Requests to /app-01 and /app-02 will be routed to nodes defined in
load-balancer
/app-01|/*=router
/app-02|/*=router

# mapping for management console, nodes in cluster can be enabled or
disabled here
/jkmanager|/*=status
```

4. **workers.properties ファイルを作成する**
   worker.properties ファイルには、ワーカーラベルとサーバーインスタンス間のマッピング定義が含まれます。以下の行をこのファイルに追加します。

```
# The advanced router LB worker
worker.list=router,status

# First EAP server definition, port 8009 is standard port for AJP in
EAP
#
# lbfactor defines how much the worker will be used.
# The higher the number, the more requests are served
# lbfactor is useful when one machine is more powerful
# ping_mode=A – all possible probes will be used to determine that
# connections are still working

worker.worker01.port=8009
worker.worker01.host=127.0.0.1
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

# Second EAP server definition
worker.worker02.port=8009
worker.worker02.host= 127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
worker.worker02.socket_timeout=10
worker.worker02.lbfactor=1

# Define the LB worker
worker.router.type=lb
worker.router.balance_workers=worker01,worker02

# Define the status worker for jkmanager
worker.status.type=status
```

注記

workers.properties ディレクティブについては、付録A *参照：workers.properties* を参照してください。

5. **IIS を再起動する**
   変更内容を反映するために IIS サーバーを再起動します。実行している IIS バージョンに対して以下のコマンドを実行します。

   IIS 6

   ```
   C:\> net stop iisadmin /Y
   C:\> net start w3svc
   ```

   IIS 7

   ```
   C:\> net stop was /Y
   C:\> net start w3svc
   ```

6. **ログを検証する**
   IIS が再起動されたら ISAPI ログを確認します。このログは **isapi_redirect.properties** の log_file プロパティで指定されたファイルの場所に保存されます。また、他のイベントについて IIS ログとイベントビューアを確認する必要があります。

パート IV. NETSCAPE SERVER API (NSAPI)

第13章 NETSCAPE SERVER API とは

この章では、Netscape Server API (NSAPI) について概説します。

NSAPI は、開発者がサーバープロセス内部で実行されるアプリケーション (プラグインと呼ばれます) を作成して Web サーバーソフトウェアの機能を拡張できるプログラミングインターフェースです。

NSAPI とプラグインの目的は、HTTP サーバーと HTTP サーバーで実行されるバックエンドアプリケーション間でさまざまな機能インターを作成する方法を提供することです。

NSAPI プラグインは Server Application Functions (SAF) を実装するよう設計されています。SAF は HTTP 要求を消費し、サーバー設定データベースから入力値を取得し、入力値に基づいてクライアントに応答を返します。各 SAF は特定のクラスにリンクされ、実装する要求応答手順に直接関連します。

要求応答手順 (クラス) の概要は以下のとおりです。

1. 権限変換
2. 名前変換
3. パスチェック
4. オブジェクトタイプ
5. 要求応答
6. ログトランザクション

各要求応答手順に対して SAF を提供する必要はありません。NSAPI を使用すると、中核的な要求応答手順の代わりに独自のカスタム機能を使用できます。また、SAF をグローバルに適用したり、SAF をディレクトリ、ファイルグループ、または個々のファイルに制限したりできます。

第14章 SOLARIS での NSAPI CONNECTOR の設定

以下のタスクは、JBoss Enterprise Platform を Sun Java System Web Server (SJWS) マスターノードのワーカーノードとして使用するよう NSAPI コネクタを設定する方法について説明しています。

注記

Sun Java System Web Server は、Oracle iPlanet Web Server に名前が変更されました。わかりやすくするために、このガイドでは以前の名前が使用されています。

この項では、すべてのサーバーインスタンスが同じマシンに存在します。各マシンで異なるマシンを使用するには、**-b** スイッチを使用して JBoss Enterprise Platform のインスタンスをパブリック IP アドレスにバインドします。IP アドレスのこれらの変更を反映させるために、SJWS マシンの **workers.properties** ファイルを編集してください。

14.1. 前提条件と設定

- ワーカーノードが JBoss Enterprise Platform 5.1 以降とともにインストールされていること。ネイティブコンポーネントは NSAPI コネクタの要件ではありません。この前提条件については、『『インストールガイド』』を参照してください。
- マスターノードが以下のいずれかのテクノロジの組み合わせと、オペレーティングシステムとアーキテクチャに適切なネイティブバイナリとともにインストールされていること。このインストールの前提条件については、『『インストールガイド』』を参照してください。
  - Solaris 9 x86 と Sun Java System Web Server 6.1 SP12
  - Solaris 9 SPARC 64 と Sun Java System Web Server 6.1 SP12
  - Solaris 10 x86 と Sun Java System Web Server 7.0 U8
  - Solaris 10 SPARC 64 と Sun Java System Web Server 7.0 U8

14.2. ワーカーノードとしてのサーバーインスタンスの設定

タスク: JBoss Enterprise Application Platform ワーカーノードの設定

JBoss Enterprise Application Platform インスタンスを SJWS ワーカーノードとして適切に設定する場合はこのタスクに従ってください。

前提条件

- 「前提条件と設定」

1. **各ワーカーノードに対してサーバープロファイルを作成する**
   ワーカーノードとして設定するサーバープロファイルのコピーを作成します (この手順では **default** サーバープロファイルを使用します)。

```
[user@workstation jboss-ep-5.1]$ cd jboss-as/server
[user@workstation server]$ cp -r default/ default-01
[user@workstation server]$ cp -r default/ default-02
```

2. **各インスタンスに一意の名前を付ける**

新しい各ワーカーインスタンスの **deploy/jbossweb.sar/server.xml** ファイルの以下の行を編集します。

```
<Engine name="jboss.web" defaultHost="localhost">
```

示しされたように一意の **jvmRoute** 値を追加します。この値はクラスタ内のこのノードの ID です。

default-01 サーバープロファイルの場合:

```
<Engine name="jboss.web" defaultHost="localhost"
jvmRoute="worker01">
```

default-02 サーバープロファイルの場合:

```
<Engine name="jboss.web" defaultHost="localhost"
jvmRoute="worker02">
```

3. **セッション処理を有効にする**
   各ワーカーノードの **deployers/jbossweb.deployer/META-INF/war-deployers-jboss-beans.xml** ファイルの以下の行をコメント解除します。

```
<property name="useJK">false</property>
```

このプロパティは、mod_jk と他のコネクタバリアントを調整するために特別なセッション処理を使用するかどうかを制御します。両方のワーカーノードでこのプロパティを **true** に設定します。

```
<property name="useJK">true</property>
```

4. **ワーカーノードを起動する**
   個別のコマンドインターフェースで各ノードワーカーを起動します。**-b** スイッチを使用して各ノードが異なる IP アドレスにバインドされるようにします。

```
[user@workstation jboss-eap-5.1]$ ./jboss-as/bin/run.sh -b 127.0.0.1
-c default-01
```

```
[user@workstation jboss-eap-5.1]$ ./jboss-as/bin/run.sh -b
127.0.0.100 -c default-02
```

14.3. 初期クラスタリング設定

タスク: 初期クラスタリング動作の設定

このタスクを完了し、Sun Java Web Server (SJWS) と NSAPI を使用してクラスタリングに必要な基本的な要素を設定します。

前提条件

- タスク: JBoss Enterprise Application Platform ワーカーノードの設定

- **/tmp/connectors/jboss-ep-native-5.1/** に抽出されたネイティブ Zip。この手順では、このパスは *NATIVE* と呼ばれます。
- ディレクトリ **/tmp/connectors** はログ、プロパティファイル、および NSAPI ロックのストレージ場所として使用されます。
- SJWS は、「ファイル命名規則」 の省略された *SJWS* ファイルパスで指定されたいずれかの場所でインストールされます。

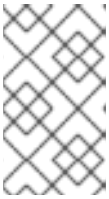
1. **サーブレットマッピングを無効にする**
   ***SJWS/PROFILE*/config/default-web.xml** ファイルの **Built In Servlet Mappings** 以下で、サンプルコードに示されているように以下のサーブレットのマッピングを無効にします。
   - default
   - invoker
   - jsp

```
<!-- ==================== Built In Servlet Mappings
===================== -->

<!-- The servlet mappings for the built in servlets defined above. -
->

<!-- The mapping for the default servlet -->
<!--servlet-mapping>
 <servlet-name>default</servlet-name>
 <url-pattern>/</url-pattern>
</servlet-mapping-->

<!-- The mapping for the invoker servlet -->
<!--servlet-mapping>
 <servlet-name>invoker</servlet-name>
 <url-pattern>/servlet/*</url-pattern>
</servlet-mapping-->

<!-- The mapping for the JSP servlet -->
<!--servlet-mapping>
 <servlet-name>jsp</servlet-name>
 <url-pattern>*.jsp</url-pattern>
</servlet-mapping-->
```

2. **必要なモジュールとプロパティをロードする**
   以下の行を ***SJWS/PROFILE*/config/magnus.conf** ファイルに追加します。

```
Init fn="load-modules" funcs="jk_init,jk_service"
shlib="NATIVE/lib/nsapi_redirector.so" shlib_flags="(global|now)"
Init fn="jk_init" worker_file="/tmp/connectors/workers.properties"
log_level="debug" log_file="/tmp/connectors/nsapi.log"
shm_file="/tmp/connectors/jk_shm"
```

   これらの行は **jk_init** 関数と **jk_service** 関数により使用される **nsapi_redirector.so** モジュールの場所とワーカーノードおよびその属性を定義する **workers.properties** ファイルの場所を定義します。



注記

***NATIVE*/lib/nsapi_redirector.so** パスの **lib** ディレクトリは 32 ビットマシンにだけ適用されます。64 ビットマシンでは、このディレクトリは **lib64** という名前になります。

14.4. NSAPI で基本的なクラスを設定

以下の手順に従います。

タスク: NSAPI を使用した基本的なクラスターの設定

このタスクを完了すると、特定のパスの要求が特定のワーカーノードに転送される基本的なクラスターが設定されます。この手順では、worker02 が **/nc** パスを提供し、worker01 が **/status** を提供し、他のすべてのパスが **obj.conf** ファイルの最初の部分で定義されます。

前提条件

- タスク: 初期クラスタリング動作の設定
- SJWS は、「ファイル命名規則」 の省略された *SJWS* ファイルパスで指定されたいずれかの場所でインストールされます。

1. **NSAPI を介して提供するパスを定義する**
   ***SJWS*/*PROFILE*/config/obj.conf** ファイルを編集します。示されたように、NSAPI を介して提供するパスを **default** オブジェクト定義の最後で定義します。

```
<Object name="default">
      [...]
   NameTrans fn="assign-name" from="/status" name="jknsapi"
   NameTrans fn="assign-name" from="/images(|/*)" name="jknsapi"
   NameTrans fn="assign-name" from="/css(|/*)" name="jknsapi"
   NameTrans fn="assign-name" from="/nc(|/*)" name="jknsapi"
   NameTrans fn="assign-name" from="/jmx-console(|/*)"
name="jknsapi"
</Object>
```

   この **obj.conf** ファイルで JBoss Enterprise Platform インスタンスにデプロイされた任意のアプリケーションのパスをマップできます。サンプルコードでは、**/nc** パスが **nc** という名前以下にデプロイされたアプリケーションにマップされます。

2. **各パスを提供するワーカーを定義する**
   ***SJWS*/*PROFILE*/config/obj.conf** ファイルを編集し、以下の **jknsapi** オブジェクト定義を **default** オブジェクト定義の後に追加します。

```
<Object name="jknsapi">
   ObjectType fn=force-type type=text/plain
   Service fn="jk_service" worker="worker01" path="/status"
   Service fn="jk_service" worker="worker02" path="/nc(/*)"
   Service fn="jk_service" worker="worker01"
</Object>
```

   この **jknsapi** オブジェクトは **default** オブジェクトの **name="jknsapi"** に割り当てられた各パスを提供するために使用されるワーカーノードを定義します。

サンプルコードでは、3 つ目のサービス定義で **path** 値が指定され、定義されたワーカーノード (**worker01**) がデフォルトで **jknsapi** に割り当てられたすべてのパスを提供します。この場合、**/status** パスを **worker01** に割り当てるサンプルコードの最初のサービス定義は余分です。

3. **ワーカーとその属性を定義する**
   ステップ 2 で定義した場所で **workers.properties** ファイルを作成します。

   このファイルでワーカーノードと各ワーカーノードのプロパティのリストを定義します。

```
# An entry that lists all the workers defined
worker.list=worker01, worker02

# Entries that define the host and port associated with these
workers
worker.worker01.host=127.0.0.1
worker.worker01.port=8009
worker.worker01.type=ajp13

worker.worker02.host=127.0.0.100
worker.worker02.port=8009
worker.worker02.type=ajp13
```

4. **サーバーを再起動する**
   Sun Java System Web Server インスタンスを設定した後で、インスタンスを再起動して変更内容を反映します。

   Sun Java System Web Server 6.1 の場合:

```
SJWS/PROFILE/stop
SJWS/PROFILE/start
```

   Sun Java System Web Server 7.0 の場合:

```
SJWS/PROFILE/bin/stopserv
SJWS/PROFILE/bin/startserv
```

14.5. NSAPI を使用した負荷分散されたクラスターの設定

タスク: NSAPI を使用した負荷分散されたクラスターの設定

このタスクを完了すると、2 つのワーカーノードから構成される負荷分散されたクラスタが設定されます。

前提条件

- タスク: 初期クラスタリング動作の設定
- SJWS は、「ファイル命名規則」 の省略された *SJWS* ファイルパスで指定されたいずれかの場所でインストールされます。

1. **NSAPI を介して提供するパスを定義する**

***SJWS/PROFILE*/config/obj.conf** を開き、示されたように、NSAPI を介して提供するパスを **default** オブジェクト定義の最後に定義します。

```
<Object name="default">
      [...]
   NameTrans fn="assign-name" from="/status" name="jknsapi"
   NameTrans fn="assign-name" from="/images(|/*)" name="jknsapi"
   NameTrans fn="assign-name" from="/css(|/*)" name="jknsapi"
   NameTrans fn="assign-name" from="/nc(|/*)" name="jknsapi"
   NameTrans fn="assign-name" from="/jmx-console(|/*)"
name="jknsapi"
   NameTrans fn="assign-name" from="/jkmanager/*" name="jknsapi"
</Object>
```

この **obj.conf** ファイルで JBoss Enterprise Platform インスタンスにデプロイされた任意のアプリケーションのパスをマップできます。サンプルコードでは、**/nc** パスが **nc** という名前以下にデプロイされたアプリケーションにマップされます。

2. **各パスを提供するワーカーを定義する**
   ***SJWS/PROFILE*/config/obj.conf** を開き、以下の **jknsapi** オブジェクト定義を **default** オブジェクト定義の後に追加します。

```
<Object name="jknsapi">
   ObjectType fn=force-type type=text/plain
   Service fn="jk_service" worker="status" path="/jkmanager(/*)"
   Service fn="jk_service" worker="router"
</Object>
```

この **jknsapi** オブジェクトは **default** オブジェクトの **name="jknsapi"** に割り当てられた各パスを提供するために使用されるワーカーノードを定義します。

3. **ワーカーとその属性を定義する**
   ***SJWS/PROFILE*/config/workers.properties** を作成します。

   このファイルでワーカーノードと各ワーカーノードのプロパティのリストを定義します。

注記

workers.properties ディレクティブについては、付録A *参照：workers.properties* を参照してください。

```
# The advanced router LB worker
worker.list=router,status

#First EAP server definition, port 8009 is standard port for AJP in
EAP
#
#lbfactor defines how much the worker will be used.
#The higher the number, the more requests are served
#lbfactor is useful when one machine is more powerful
#ping_mode=A – all possible probes will be used to determine that
#connections are still working
worker.worker01.port=8009
worker.worker01.host=127.0.0.1
```

```
worker.worker01.type=ajp13
worker.worker01.ping_mode=A
worker.worker01.socket_timeout=10
worker.worker01.lbfactor=3

#Second EAP server definition
worker.worker02.port=8009
worker.worker02.host=127.0.0.100
worker.worker02.type=ajp13
worker.worker02.ping_mode=A
worker.worker02.socket_timeout=10
worker.worker02.lbfactor=1

# Define the LB worker
worker.router.type=lb
worker.router.balance_workers=worker01,worker02

# Define the status worker
worker.status.type=status
```

4. **サーバーを再起動する**
   Sun Java System Web Server インスタンスを設定した後で、インスタンスを再起動して変更内容を反映します。

   Sun Java System Web Server 6.1 の場合:

   ```
   SJWS/PROFILE/stop
   SJWS/PROFILE/start
   ```

   Sun Java System Web Server 7.0 の場合:

   ```
   SJWS/PROFILE/bin/stopserv
   SJWS/PROFILE/bin/startserv
   ```

パート V. 一般的な負荷分散タスク

第15章 HTTP セッションステートレプリケーション

ソフトウェアロードバランサ

複数のコンピュータサーバー (クラスタ) に HTTP クライアントセッション要求を分散するよう設計された専用のソフトウェアベースのサービス。ソフトウェアロードバランサの主要な目的はリソース使用率を最大化し、要求応答時間を短縮し、サーバーの過負荷を防ぐことです。ロードバランサは、サーバーの負荷と可用性に基づいてクライアントセッション要求をサーバークラスタに転送します。

クライアントセッション

クライアント (アプリケーション) とサーバー間の半永久的な接続。ロードバランサは、永続化でクライアントセッションが作成されるかどうか、またはサーバーの負荷および可用性い基づいてクライアントセッションが再分散されるかどうかを決定します。

セッション永続化

単一のサーバーインスタンスにだけ割り当てられるクライアントセッション。ロードバランサはクライアントセッションに関連付けられたすべての HTTP 要求を割り当てられたサーバーインスタンスにだけルーティングします。セッションの永続化は、一般的に*スティッキーセッション*と呼ばれます。

スティッキーセッション

セッションの永続化を参照してください。

セッション内のクライアントが常に同じサーバーノードにルーティングされるようにロードバランサでセッションステート永続化を設定する方法については、「mod_jk のワーカーノードの設定」を参照してください。

独自のセッション永続化は、サーバーで障害が発生した場合に、すべてのセッションステートデータが失われるため、最良のソリューションではありません。たとえば、顧客が Web サイトで買い物をするときにショッピングカートインスタンスをホストしているサーバーで障害が発生した場合、ショッピングカートに関連付けられたセッションステートデータは永久的に失われます。

クライアントセッションデータの損失を回避する方法の 1 つは、クライアント内のサーバーでセッションデータをレプリケートすることです。サーバーノードで障害が発生した場合やサーバーノードがショットダウンされた場合、ロードバランサは任意のサーバーノードに次のクライアント要求をフェールオーバーし、同じセッションステートを取得できます。

Web アプリケーションでセッションレプリケーションを設定せずに、セッション永続化をサポートするロードバランサを使用すると、セッションステートレプリケーションのコストを回避することにより実装のスケーラビリティが実現されます。セッションの各要求は常に同じノードで処理されます。

セッションステートレプリケーションは、基本的なセッション永続化よりもコストが高くなりますが、セッションステートデータに提供される安定性は、負荷分散されたクラスタを作成する場合に重要になります。

15.1. アプリケーションでのセッションレプリケーションの有効化

Web アプリケーションのレプリケーションを有効にするには、**web.xml** 記述子でアプリケーションを分散可能としてタグ付けする必要があります。以下に例を示します。

```
<?xml version="1.0"?>
<web-app  xmlns="http://java.sun.com/xml/ns/j2ee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
                              http://java.sun.com/xml/ns/j2ee/web-
app_2_4.xsd"
```

```
          version="2.4">

    <distributable/>

</web-app>
```

また、**jboss-web.xml** ファイルで **replication-config** 要素を使用することにより、セッションレプリケーションを使用できます。ただし、**replication-config** 要素は、以下で説明された 1 つ以上のデフォルト値を使用できる場合にのみ設定する必要があります。例は以下のとおりです。

```
<!DOCTYPE jboss-web PUBLIC
    "-//JBoss//DTD Web Application 5.0//EN"
    "http://www.jboss.org/j2ee/dtd/jboss-web_5_0.dtd">

<jboss-web>

   <replication-config>
      <cache-name>custom-session-cache</cache-name>
      <replication-trigger>SET</replication-trigger>
      <replication-granularity>ATTRIBUTE</replication-granularity>
      <replication-field-batch-mode>true</replication-field-batch-mode>
      <use-jk>false</use-jk>
      <max-unreplicated-interval>30</max-unreplicated-interval>
      <snapshot-mode>INSTANT</snapshot-mode>
      <snapshot-interval>1000</snapshot-interval>
      <session-notification-
policy>com.example.CustomSessionNotificationPolicy</session-notification-
policy>
   </replication-config>

</jboss-web>
```

すべての設定要素はオプションであり、デフォルト値が使用可能な場合は省略できます。この組み合わせは一般的に使用され、残りの値はデフォルト値からほとんど変更されません。一般的に使用される値について最初に説明します。

<replication-trigger> 要素は、セッションデータをクラスタでレプリケートすることをコンテナがいつ考慮するかを決定します。この設定の理由は、セッション属性として保存された変更可能なオブジェクトがセッションからアクセスされた後に、**setAttribute** 呼び出しが存在しない場合、オブジェクト (したがって、セッションステートも) が変更され、レプリケートする必要があるかどうかをコンテナが明確に認識できないことです。この要素は 3 つの有効な値を持ちます。

- **SET_AND_GET** は保守的であり、最適ではありません (パフォーマンスの点で)。内容が変更されず、単にアクセスされた場合であってもセッションデータは常にレプリケートされます。この設定は JBoss Enterprise Application Platform 4 では (ある程度は) 適切です。なぜなら、この設定を使用すると、要求ごとに、セッションのタイムスタンプのレプリケーションがトリガされるためです。**max_unreplicated_interval** を 0 に設定すると、同じことを非常に小さいコストで実現できるため、Enterprise Application Platform 5 で **SET_AND_GET** を使用することは適切ではありません。

- **SET_AND_NON_PRIMITIVE_GET** は保守的であり、非プリミティブタイプがアクセスされた場合にのみレプリケートされます (つまり、このオブジェクトは **Integer**、**Long**、**String** などのよく知られた不変な JDK タイプではありません)。これはデフォルト値です。

- **SET** を使用する場合は、データをレプリケートする必要があるときに開発者がセッションで

setAttribute を明示的に呼び出すことが前提とされます。この設定では、不必要なレプリケーションが回避され、パフォーマンスが大幅に向上しますが、セッションで保存された変更可能なオブジェクトが変更されたときに常に **setAttribute** が呼び出されるようにするために非常に優れたコーディングが必要になります。

すべてのケースで、**setAttribute** を呼び出すと、セッションがレプリケーションを必要としているとマークされます。

<replication-granularity> 要素は、セッションレプリケーションが必要なことをコンテナが決定する場合に、レプリケート対象の粒度を決定します。サポートされる値は以下のとおりです。

SESSION

属性が変更されたと見なされた場合に全体のセッション属性マップをレプリケートすることを指定します。レプリケーションは要求側で行われます。このオプションにより、ほとんどのデータがレプリケートされ、レプリケーションコストが最大になります。ただし、すべての属性値は常に一緒にレプリケートされるため、セッションがシリアル化解除されたときに属性値間の参照は破壊されなくなります。このため、これはデフォルト設定になります。

ATTRIBUTE

セッションが変更可能と見なす属性のみがレプリケートされることを指定します。レプリケーションは要求側で行われます。大量のデータを扱うセッション (一部のデータが頻繁に更新されない) では、このオプションによりレプリケーションパフォーマンスが大幅に向上することがあります。ただし、これは、お互いに参照を共有する異なる属性のオブジェクトを格納するアプリケーションには適していません (たとえば、**Address** オブジェクトへの参照を "wife" 属性の別の **Person** オブジェクトと共有する "husband" 属性の **Person** オブジェクト)。これは、属性が別々にレプリケートされ、セッションがリモートノードでシリアル化解除された場合は、共有された参照が破壊されるからです。

replication-config 要素下の他の要素はほとんど使用されません。

<cacheName>

分散可能なセッションを格納し、クラスタでレプリケートするために使用する必要がある JBoss Cache 設定の名前を指定します。この要素を使用すると、さまざまなキャッシュ特性が必要な Web アプリケーションで、個別の別々に設定された JBoss Cache インスタンスの使用を指定できます。JBoss Enterprise Application Platform 4 では、使用するキャッシュは Web アプリケーションごとに変更できず、サーバー全体の設定でした。デフォルト値は **standard-session-cache** です。Web 階層クラスタリングに関する JBoss Cache 設定の詳細については、「セッションステートレプリケーションに使用される JBoss Cache インスタンスの設定」 を参照してください。

<replication-field-batch-mode>

要求に関連付けられたすべてのレプリケーションメッセージを 1 つのメッセージにまとめるかどうかを指定します。これは、**replication-granularity** が **FIELD** の場合にのみ適用できます。**replication-field-batch-mode** が **true** に設定されている場合は、セッション属性マップに格納されたオブジェクトに加えられた粒度の細かい変更が HTTP 要求が完了したときにレプリケートされます。それ以外の場合、これらの変更は発生したときにレプリケートされます。この値を **false** に設定することは推奨されません。デフォルト値は**true** です。

<useJK>

コンテナーで、この Web アプリケーションの負荷分散に JK ベースのソフトウェアロードバランサー (mod_jk、mod_proxy、mod_cluster など) を使用することを前提とするかどうかを指定します。**true** に設定されている場合は、コンテナーが各要求に関連付けられたセッション ID を調べ、フェールオーバーを検出した場合にセッション ID の **jvmRoute** 部分を置き換えます。

これは、URL を JK ロードバランサで処理できない Web アプリケーションに対してのみ **false** に設定する必要があります。

<max-unreplicated-interval>

要求の最大間隔を秒単位で設定します。この時間後、要求によりセッションがダーティになったかどうかに関係なくセッションのタイムスタンプのレプリケーションが要求によりトリガされます。このようなレプリケーションにより、クラスタの他のノードはセッションのタイムスタンプの最新の値を認識し、フェールオーバー時にレプリケートされないセッションが間違って時間切れになることがなくなります。また、フェールオーバー後に、**HttpSession.getLastAccessedTime()** コールの値が適切になります。

デフォルト値は **null** (つまり、未指定) です。この場合、セッションマネージャーは組み込まれた JBoss Web **Engine** に関する **jvmRoute** 設定があるかどうかを確認して (「mod_jk と動作するよう JBoss を設定する」 を参照)、JK が使用されているかどうかを調べます。

値が **0** の場合は、セッションがアクセスされたときに必ずタイムスタンプがレプリケートされます。値が **-1** の場合は、要求中の他のアクティビティ (属性の変更など) により、セッションに関連する他のレプリケーション作業が発生したときにだけタイムスタンプがレプリケートされます。**HttpSession.getMaxInactiveInterval()** 値よりも大きい正の値は設定ミスとして扱われ、**0** に変換されます (つまり、各要求時にメタデータがレプリケートされます)。デフォルト値は **60** です。

<snapshot-mode>

セッションが他のノードにレプリケートされるタイミングを指定します。可能な値は **INSTANT** (デフォルト値) と **INTERVAL** です。

一般的な値 **INSTANT** は、レプリケーションを実行する要求処理スレッドを使用して要求終了時に変更を他のノードにレプリケートします。この場合、**snapshot-interval** プロパティは無視されます。

INTERVAL モードでは、**snapshot-interval** ミリ秒ごとに実行されるバックグラウンドタスクが作成され、変更されたセッションをチェックしてレプリケートします。

replication-granularity が **FIELD** に設定されている場合、このプロパティは無効です。**FIELD** である場合は、**instant** モードが使用されます。

<snapshot-interval>

この Web アプリケーションに対して変更されたセッションをレプリケートするバックグラウンドタスクを開始する頻度 (ミリ秒単位) を定義します。**snapshot-mode** が **INTERVAL** に設定されている場合のみ意味を持ちます。

<session-notification-policy>

登録された任意の **HttpSessionListener**、**HttpSessionAttributeListener**、または **HttpSessionBindingListener** にサーブレット仕様通知を送出するかどうかを管理するために使用する必要がある **ClusteredSessionNotificationPolicy** インターフェースの実装の完全修飾名を指定します。

重要

非クラスタ環境で適切なイベント通知がクラスタ環境で必ずしも適切であるとは限りません。通知が適切でない理由の例については、https://jira.jboss.org/jira/browse/JBAS-5778 を参照してください。適切な **ClusteredSessionNotificationPolicy** を設定すると、アプリケーション作成者は発行される通知を細かく制御できます。

15.2. HTTPSESSION パッシブ化およびアクティブ化

パッシブ化

比較的未使用のセッションを永続ストレージに保存し、メモリから削除することによりメモリ使用を制御するプロセス。

パッシブ化されたセッションがクライアントにより要求された場合、セッションはメモリにアクティブ化され、永続ストアから削除されることがあります。JBoss Enterprise Application Platform 5 は、**web.xml** ファイルに **distributable** ディレクティブが含まれるクラスタ化された Web アプリケーションからの HttpSessions パッシブ化をサポートします。

パッシブ化は、Web アプリケーションのライフサイクルの 3 つの地点で行われます。

- コンテナが新しい要求の作成を要求するとき。現在アクティブなセッションの数が設定可能な制限を超えると、セッションをパッシブ化し、メモリのスペースを空けることが試行されます。
- JBoss Web バックグラウンドタスクスレッドが実行される周期 (デフォルトでは 10 秒ごと)。
- Web アプリケーションがデプロイされ、他のサーバーでアクティブなセッションのバックアップコピーが新しいデプロイ Web アプリケーションのセッションマネージャにより取得されたとき。

セッションは、以下の条件のいずれかに該当する場合にパッシブ化されます。

- セッションが設定可能な最大アイドル時間より長い時間使用されていない場合。
- アクティブセッションの数が設定可能な最大値を超え、セッションが設定可能な最小アイドル時間より長い時間使用されていない場合。

どちらの場合でも、セッションは Least Recently Used (LRU) に基づいてパッシブ化されます。

15.2.1. HttpSession パッシブ化の設定

セッションパッシブ化の動作は、Web アプリケーションの **WEB-INF** ディレクトリの **jboss-web.xml** デプロイメント記述子により設定されます。

```
<!DOCTYPE jboss-web PUBLIC
    "-//JBoss//DTD Web Application 5.0//EN"
    "http://www.jboss.org/j2ee/dtd/jboss-web_5_0.dtd">

<jboss-web>

   <max-active-sessions>20</max-active-sessions>
   <passivation-config>
```

```
      <use-session-passivation>true</use-session-passivation>
      <passivation-min-idle-time>60</passivation-min-idle-time>
      <passivation-max-idle-time>600</passivation-max-idle-time>
   </passivation-config>


</jboss-web>
```

- **max-active-session**

  許可されるアクティブセッションの最大数を決定します。セッションマネージャにより管理されるセッションの数がこの値を超え、パッシブ化が有効化される場合、設定された **passivation-min-idle-time** に基づいて超過はパッシブ化されます。パッシブ化が完了した後でも (または、パッシブ化が無効な場合)、アクティブセッションの数がこの制限を超えるは、新しいセションの作成が拒否されます。**-1** (デフォルト値) に設定された場合、制限はありません。

- **use-session-passivation**

  Web アプリケーションに対してセッションパッシブ化が有効であるかどうかを決定します。デフォルト値は **false** です。

- **passivation-min-idle-time**

  max-active-sessions により定義された値に従うアクティブセッション数を削減するために、コンテナがパッシブ化を考慮する前にセッションを無効にする最小時間 (秒単位) を決定します。値が **-1** (デフォルト値) の場合は、**passivation-max-idle-time** より前にセッションのパッシブ化が無効にされます。**max-active-sessions** が設定されている場合は、**-1** の値や大きい値は推奨されません。

- **passivation-max-idle-time**

  コンテナがセッションをパッシブ化してメモリを保存する前にセッションを無効にできる最大時間 (秒数) を決定します。このようなセッション のパッシブ化は、アクティブセッション数が **max-active-sessions** を超えるかどうかに関係なく、実行されます。**web.xml session-timeout** 設定よりも小さい必要があります。値が**-1** (デフォルト値) である場合は、無効の最大時間に基づいてパッシブ化が無効になります。

メモリ内のセッションの合計数には、このノードでアクセスされない他のクラスタノードからレプリケートされたセッションが含まれます。**max-active-sessions** を設定する場合はこれを考慮してください。他のノードからレプリケートされたセッションの数は、**buddy replication** が有効であるかどうかによっても異なります。

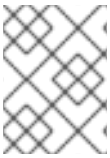
たとえば、8 つのノードクラスタが存在し、各ノードが 100 人のユーザーからの要求を処理するとします。**total replication** を使用する場合、各ノードはメモリ内に 800 個のセッションを保存します。**buddy replication** が有効であり、デフォルトの **numBuddies** 設定 (**1**) を使用する場合、各ノードはメモリ内に 200 個のセッションを保存します。

15.3. セッションステートレプリケーションに使用される JBOSS CACHE インスタンスの設定

分散可能な Web アプリケーションのコンテナは JBoss Cache を使用してクラスタに HTTP セッションレプリケーションサービスを提供します。コンテナは JBoss Cache インスタンスへの参照を取得するために CacheManager と統合されます。詳細については、『『管理設定ガイド』』の章「『JBoss Cache による分散キャッシング』」を参照してください。

使用する JBoss Cache 設定の名前は、アプリケーションの **jboss-web.xml** の **cacheName** 要素によって制御されます (「アプリケーションでのセッションレプリケーションの有効化」を参照)。ほとんどの場合、**standard-session-cache** のデフォルト値が適切であるため、これを設定する必要はありません。

CacheManager サービスの JBoss Cache 設定では、複数のオプションが利用可能です。



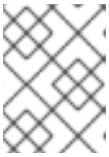
注記

詳細については、『『管理設定ガイド』』の章「『JBoss キャッシュ』」を参照してください。

standard-session-cache 設定は Web セッションレプリケーションを使用する場合のためにすでに最適化されており、ほとんどの設定は変更する必要がありません。管理者は以下の設定を変更できます。

- **cacheMode**

  デフォルト値は **REPL_ASYNC** であり、クラスタに送信されたセッションレプリケーションメッセージが、メッセージが受信され、処理されたことを確認する応答を他のクラスタノードから受け取るまで待たないことを指定します。代わりのモードである **REPL_SYNC** は、セッションステートが受信されたが、パフォーマンスが大幅に低下することを確認します。

  #### 注記

  cacheMode に対して他の利用可能なパラメータの詳細については、『『管理設定ガイド』』の節「『キャッシュモード』」を参照してください。

- **buddyReplicationConfig** セクションの **enabled** プロパティ

  バディレプリケーションを有効にする場合は、**true** に設定します。デフォルト値は **false** です。

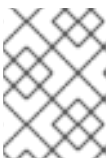
  

  #### 注記

  cacheMode に対して他の利用可能なパラメータの詳細については、『『管理設定ガイド』』の節「『バディレプリケーション』」を参照してください。

- **buddyReplicationConfig** セクションの **numBuddies** プロパティ

  セッションがレプリケートされるバックアップノードの数を増加させる場合はデフォルト値 **(1)** よりも大きい値に設定します。バディレプリケーションが有効な場合にのみ重要です。

  #### 注記

  cacheMode に対して他の利用可能なパラメータの詳細については、『『管理設定ガイド』』の節「『バディレプリケーション』」を参照してください。

- **buddyReplicationConfig** セクションの **buddyPoolName** プロパティ

バディレプリケーションが有効な場合に推奨されるレプリケーショングループを指定する方法。JBoss Cache は同じプール名を共有するバディを選択しようとします (利用可能でない場合は他のバディを選択します)。バディレプリケーションが有効な場合のみ重要です。

注記

cacheMode に対して他の利用可能なパラメータの詳細については、『『管理設定ガイド』』の節「『バディレプリケーション』」を参照してください。

- **multiplexerStack**

  キャッシュが使用する必要がある JGroups プロトコルスタックの名前。

注記

cacheMode に対して他の利用可能なパラメータの詳細については、『『管理設定ガイド』』の節「『チャネルファクトリサービス』」を参照してください。

- **clusterName**

  JGroups がこのキャッシュのチャネルに使用する名前の指定。これは、このプロパティが他のすべてのキャッシュ設定からの異なる値を持つ必要がある、新しいキャッシュ設定を作成する場合にのみ変更します。

既存のものを設定するよりも完全に新しい JBoss Cache 設定を使用したい場合は、『』『管理設定ガイド』の節「『CacheManager サービスによるデプロイメント』」を参照してください。

第16章 クラスター化されたシングルサインオン (SSO) の使用

JBoss はクラスタ化されたシングルサインオン (SSO) をサポートします。ユーザーはある Web アプリケーションに対して認証を行い、同じ仮想ホストでデプロイされたすべての Web アプリケーションで認識できます。この場合、アプリケーションが同じマシンまたはクラスタの別のノードにデプロイされているかどうかは関係ありません。

認証レプリケーションは JBoss Cache により処理されます。クラスタ化されたシングルサインオンサポートは、Tomcat および JBoss Web の標準的な部分である非クラスタ **`org.apache.catalina.authenticator.SingleSignOn`** バルブの JBoss 固有の拡張機能です。

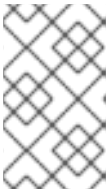
16.1. 設定

クラスタ化されたシングルサインオンを有効にするには、**`JBOSS_HOME/server/PROFILE/deploy/jbossweb.sar/server.xml`** ファイルの適切な **`Host`** 要素に **`ClusteredSingleSignOn`** バルブを追加する必要があります。バルブ要素はすでに標準的なファイルに含まれています。単にコメント解除することが必要なだけです。バルブ設定を以下に示します。

```
<Valve className="org.jboss.web.tomcat.service.sso.ClusteredSingleSignOn"
/>
```

この要素は以下の属性をサポートします。

- **className** は、使用するバルブ実装の Java クラス名を設定するために必要な属性です。これは **`org.jboss.web.tomcat.service.sso.ClusteredSingleSign`** に設定する必要があります。
- **cacheConfig** クラスタ化された SSO キャッシュに使用するキャッシュ設定の名前です。デフォルト値は **`clustered-sso`** です。

  注記

  キャッシュ設定の詳細については、『『管理設定ガイド』』の節「『JBoss Enterprise Application Platform の CacheManager サービス』」を参照してください。

- **treeCacheName** は廃止されました。**`cacheConfig`** を使用してください。クラスタ化された SSO キャッシュに使用する JBoss Cache MBean の JMX ObjectName を指定します。**`cacheConfig`** の値を使用して CacheManager サービスからキャッシュを見つけることができない場合は、この ObjectName 以下の JMX で登録された mbean を見つけようとします。デフォルト値は **`jboss.cache:service=TomcatClusteringCache`** です。
- **cookieDomain** は、SSO クッキーに使用されるホストドメインを設定するために使用されます。詳細については、「クッキードメインの設定」を参照してください。デフォルト値は **`"/"`** です。
- **maxEmptyLife** は、アクティブセッションがない SSO が要求により使用可能になる最大秒数です。クラスター化された SSO バルブは SSO に関連するセッションを管理するクラスターノードを追跡します。この属性に正の値を使用すると、SSO に関連付けられた任意のセッションを処理した唯一のノードのシャットダウンが適切に処理されます。このシャットダウンにより、セッションのローカルコピーが無効になり、SSO からすべてのセッションが削除されます。maxEmptyLife がゼロの場合は、SSO がローカルセッションコピーとともに終了します。ただし、セッションのバックアップコピー (クラスター化された Web アプリケーションからの

場合) は他のクラスタノードで利用可能です。管理されたセッションの存続期間を超えて SSO が存在するようにすると、ユーザーには異なるクラスタノードにフェールオーバーできる他の要求を行う時間が与えられ、セッションのバックアップコピーが有効化されます。デフォルト値は **1800** (つまり、30 分) です。

- **processExpiresInterval** は、バルブが 'maxEmptyLife' を超えた SSO を検出し、無効化する処理間の最小秒数です。'processExpiresInterval' ごとにクリーンアップなどの処理が行われるのではなく、その値よりも頻繁に処理が行われないことを意味します。デフォルト値は **60** です。

- **requireReauthentication** はセキュリティ **Realm** に対して各要求を再認証する必要があるかどうかを決定するフラグです。"true" の場合、このバルブはキャッシュされたセキュリティクレデンシャル (ユーザー名とパスワード) を使用して JBoss Web セキュリティ **Realm** に対して SSO セッションに関連付けられた各要求を再認証します。**false** の場合は、**Realm** で再チェックせずにバルブ自体が有効な SSO クッキーの存在に基づいて要求を認証できます。**true** に設定すると、異なる **security-domain** 設定を持つ Web アプリケーションが SSO を共有できるようになります。デフォルト値は **false** です。

16.2. SSO の動作

ユーザーは、仮想ホストにある任意の Web アプリケーションの保護されていないリソースのみにアクセスする限り、拒否されません。

任意の Web アプリケーションの保護されたリソースにアクセスする場合は、Web アプリケーションに対して設計されたログイン方法を使用して認証を行うよう求められます。

認証されると、関連付けられたすべての Web アプリケーションに対するアクセス制御決定にこのユーザーに関連付けられたロールが使用されます。この場合、ユーザーは各アプリケーションに対して個別に認証することを求められません。

Web アプリケーションがセッションを無効化する場合 (**javax.servlet.http.HttpSession.invalidate()** メソッドを呼び出すことにより)、すべての Web アプリケーションのユーザーのセッションは無効化されます。

他のセッションがまだ有効な場合、セッションタイムアウトにより SSO は無効になりません。

16.3. 制限

この Tomcat バルブベースの SSO 実装には、複数の既知の制限が存在します。

- JBoss サーバーのクラスタ内でのみ役に立ちます。SSO は他のリソースに伝播しません。

- コンテナにより管理された認証の使用が必要です (**web.xml** の **<login-config>** 要素を使用)。

- クッキーを必要とします。SSO はクッキーを使用して保守され、URL の書き換えはサポートされていません。

- **requireReauthentication** が **true** に設定されない限り、同じ SSO バルブに設定されたすべての Web アプリケーションは同じ JBoss Web **Realm** と JBoss Security **security-domain** を共有する必要があります。つまり、以下のようになります。

  - **server.xml** で、**Realm** 要素を **Host** 要素 (または周りの **Engine** 要素) 内部にネストできます。ただし、関連するいずれかの Web アプリケーションでパッケージ化される **context.xml** 内部ではネストできません。

- **jboss-web.xml** または **jboss-app.xml** で設定された **security-domain** はすべての Web アプリケーションで同一である必要があります。
- **requireReauthentication** を **true** に設定し、異なる Web アプリケーションに対して異なる **security-domain** (または、可能性的には低いですが、異なる **Realm**) を使用する場合であっても、さまざまなセキュリティ統合が同じ認証情報 (ユーザー名やパスワードなど) をすべて受け入れる必要があります。

16.4. クッキードメインの設定

SSO バルブは **cookieDomain** 設定属性をサポートします。この属性により、SSO クッキーのドメイン (ブラウザがクッキーを提供する一連のホスト) を設定することができます。デフォルトでは、ドメインは **"/"** であり、ブラウザはクッキーを発行したホストにだけクッキーを提供します。**cookieDomain** 属性により、クッキーの範囲となるドメインは拡張されます。

たとえば、URL **http://app1.xyz.com** と **http://app2.xyz.com** を持つ 2 つのアプリケーションがあり、SSO コンテキストを共有したいとします。これらのアプリケーションはクラスタ内のさまざまなサーバーで実行できます。または、関連付けられた仮想ホストは複数のエイリアスを持つことができます。これは、以下の設定でサポートされます。

```
<Valve className="org.jboss.web.tomcat.service.sso.ClusteredSingleSignOn"
                 cookieDomain="xyz.com" />
```

第17章 完全な使用例

完全な使用例の一連の設定例を以下に示します。

プロキシサーバー

localhost でリッスンしているプロキシサーバー:

```
<LoadModule slotmem_module modules/mod_slotmem.so
LoadModule manager_module modules/mod_manager.so
LoadModule proxy_cluster_module modules/mod_proxy_cluster.so
LoadModule advertise_module modules/mod_advertise.so

Listen 127.0.0.1:6666
<VirtualHost 127.0.0.1:6666>

    <Directory />
        Order deny,allow
        Deny from all
        Allow from 127.0.0.1
    </Directory>

    KeepAliveTimeout 60
    MaxKeepAliveRequests 0

    ManagerBalancerName mycluster
    ServerAdvertise On
    AdvertiseFrequency 5

</VirtualHost>

<Location /mod_cluster-manager>
    SetHandler mod_cluster-manager
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
</Location>
```

JBoss Web クライアントリスナー

***JBOSS_EAP_DIST*/server/*PROFILE*/deploy/jbossweb.sar/server.xml** のリスナー定義を以下に示します。

```
<!-- Non-clustered mode -->
<Listener
className="org.jboss.web.tomcat.service.deployers.MicrocontainerIntegratio
nLifecycleListener" delegateBeanName="ModClusterService"/>
<!-- Clustered mode
 Listener
className="org.jboss.web.tomcat.service.deployers.MicrocontainerIntegratio
nLifecycleListener" delegateBeanName="HAModClusterService"/-->
```

JBoss Web クライアントサービスの依存関係

Following are the required dependencies for the WebServer bean in ***JBOSS_EAP_DIST*/server/*PROFILE*/deploy/jbossweb.sar/META-INF/jboss-beans.xml**. Add them to the existing dependencies.

```
<bean name="WebServer"
class="org.jboss.web.tomcat.service.deployers.TomcatService">
  <!-- ... -->
  <depends>ModClusterService</depends><!-- Non-clustered mode -->
  <!--depends>HAModClusterService</depends--><!-- Clustered mode -->
  <!-- ... -->
</bean>
```

iptables ファイアウォールルールの例

以下は、192.168.1.0/24 サブネットのクラスタノードに対して **iptables** を使用した場合のファイアウォールルールの例です。

```
/sbin/iptables -I INPUT 5 -p udp -d 224.0.1.0/24 -j ACCEPT -m comment --
comment "mod_cluster traffic"
/sbin/iptables -I INPUT 6 -p udp -d 224.0.0.0/4 -j ACCEPT -m comment --
comment "JBoss Cluster traffic"
/sbin/iptables -I INPUT 9 -p udp -s 192.168.1.0/24 -j ACCEPT -m comment --
comment "cluster subnet for inter-node communication"
/sbin/iptables -I INPUT 10 -p tcp -s 192.168.1.0/24 -j ACCEPT -m comment -
-comment "cluster subnet for inter-node communication"
/etc/init.d/iptables save
```

付録A 参照：WORKERS.PROPERTIES

Apache httpd Server ワーカーノードは **mod_jk** ロードバランサにマップされる Servlet コンテナです。ワーカーノードは ***HTTPD_DIST*/conf/workers.properties** で定義されます。このファイルは、さまざまな Servlet コンテナが存在する場所とコールが負荷分散される方法を指定します。

workers.properties ファイルには、以下の 2 つのセクションが含まれます。

Global Properties

このセクションには、すべてのワーカーに適用されるディレクティブが含まれます。

Worker Properties

このセクションには、個別のワーカーに適用されるディレクティブが含まれます。

各ノードは Worker Properties の命名規則を使用して定義されます。ワーカー名には、英数字のみを使用でき、文字は [a-z][A-Z][0-9][_-/] に制限されます。

ワーカープロパティの構造は **worker.*worker_name*.*directive*** です。

worker

すべてのワーカープロパティで同一の接頭辞。

worker_name

ワーカーに与えられた任意の名前 (node1、node_01、Node_1 など)。

directive

必要な特定のディレクティブ。

ワーカーノードを設定するのに必要な主なディレクティブについて以下で説明します。

注記

worker.properties 設定ディレクティブの完全なリストについては、『Apache Tomcat Connector - Reference Guide』を参照してください。

worker.properties グローバルディレクティブ

worker.list

mod_jk により使用されたワーカー名のリストを指定します。このリストのワーカーは要求をマップするために使用できます。

注記

ロードバランサにより管理されない単一ノード設定は **worker.list=*[worker name]*** に設定する必要があります。

workers.properties 必須ディレクティブ

type

ワーカーのタイプを指定し、ワーカーに適用可能なディレクティブを決定します。デフォルト値は ***ajp13*** であり、Web サーバーと Apache httpd サーバー間の通信に選択する場合に推奨されるワーカータイプです。

他の値には ***ajp14***、***lb***、***status*** などがあります。

ajp13 の詳細については、『The Apache Tomcat Connector - AJP Protocol Reference 』を参照してください。

workers.properties 接続ディレクティブ

host

ワーカーのホスト名または IP アドレス。ワーカーノードは ajp13 プロトコルスタックをサポートする必要があります。デフォルト値は **localhost** です。

ホスト名または IP アドレスの後にポート番号を付加することによりホストディレクティブの一分として ***port*** ディレクティブを指定できます (たとえば、**worker.node1.host=192.168.2.1:8009** や **worker.node1.host=node1.example.com:8009** など)。

port

定義されたプロトコル要求をリッスンするリモートサーバーインスタンスのポート番号。デフォルト値は、AJP13 ワーカーのデフォルトリッスンポートである **8009** です。AJP14 ワーカーを使用する場合は、このバルブを **8011** に設定する必要があります。

ping_mode

現在のネットワークの状態を調べるために接続がプローブされる条件を指定します。

プローブは CPing に空の AJP13 パケットを使用し、指定されたタイムアウト内に返答として CPong を期待します。

ディレクティブフラグの組み合わせを使用して条件を指定します。フラグはカンマ区切りではありません。たとえば、適切なディレクティブフラグセットは **worker.node1.ping_mode=CI** です。

C (connect)

サーバーの接続後に条件がプローブされることを指定します。***connect_timeout*** ディレクティブを使用してタイムアウトを指定します。指定しない場合は、***ping_timeout*** の値が使用されます。

P (prepost)

サーバーに各要求を送信した後に条件がプローブされることを指定します。***prepost_timeout*** ディレクティブを使用してタイムアウトを指定します。指定しない場合は、***ping_timeout*** の値が使用されます。

I (interval)

通常の内部保守サイクルの間に接続がプローブされることを指定します。***connection_ping_interval*** ディレクティブを使用して各間隔の間のアイドル時間を指定します。指定しない場合は、***ping_timeout*** の値が使用されます。

A (all)

すべてのディレクティブが適用されることを指定する最も一般的な設定。高度な ***_timeout*** ディレクティブについては、『Apache Tomcat Connector - Reference Guide 』を参照してください。

ping_timeout

CPing 接続プローブに対する CPong 回答を待機する時間を指定します (***ping_mode*** を参照)。デフォルト値は 10000 (ミリ秒) です。

worker.properties 負荷分散ディレクティブ

lbfactor

個々のワーカーに対する負荷分散要素を指定します。ロードバランサのメンバーワーカーに対してのみ指定されます。

このディレクティブは、ワーカーに分散された HTTP 要求負荷とクラスタ内の他のワーカーの相対的な量を定義します。

このディレクティブが適用される一般的な例は、クラスタ内の他のサーバーよりも処理パワーが大きいサーバーを区別する場合です。たとえば、ワーカーが他のワーカーよりも 3 倍の負荷を引き受ける必要がある場合は、**worker.*worker name*.lbfactor=3** を指定します。

balance_workers

ロードバランサが管理する必要があるワーカーノードを指定します。ディレクティブは同じロードバランサに対して複数回使用でき、workers.properties ファイルで指定されたカンマ区切りのワーカー名リストから構成されます。

sticky_session

SESSION ID を持つワーカーに対する要求が同じワーカーに再びルーティングされるかどうかを指定します。デフォルト値は ***0*** (false) です。***1*** (true) に設定されると、ロードバランサの永続性が有効になります。

たとえば、**worker.loadbalancer.sticky_session=0** を指定する場合、各要求はクラスタ内の各ノード間で負荷分散されます。つまり、同じセッションの異なる要求はサーバーの負荷に基づいて異なるサーバーに送信されます。

worker.loadbalancer.sticky_session=1 の場合、各セッションはセッションが終了するまで 1 つのサーバーに永続化 (ロック) され、そのサーバーが利用可能になります。

付録B 参照：JAVA プロパティ

この付録では、JBoss Enterprise Application Platform サーバーノードに適用される JBoss HTTP Connector mod_cluster 設定プロパティについて説明します。

B.1. プロキシ設定

設定値は以下の条件に該当する場合にプロキシに送信されます。

- サーバーの起動時
- アドバタイズメカニズムを介してプロキシが検出されるとき
- エラーリカバリ中にプロキシの設定がリセットされるとき

プロキシ設定値

stickySession (デフォルト値は `true`)

可能な場合に、該当するセッションの後続要求を同じノードにルーティングするかどうかを指定します。

stickySessionRemove (デフォルト値は `false`)

バランサが固定されたノードに要求をルーティングできない場合に、httpd プロキシがセッションスティッキネスを削除するかどうかを指定します。**`stickySession`** が **`false`** の場合、このプロパティは無視されます。

stickySessionForce (デフォルト値は `true`)

バランサが固定されたノードに要求をルーティングできない場合に、httpd プロキシがエラーを返すかどうかを指定します。**`stickySession`** が **`false`** の場合、このプロパティは無視されます。

workerTimeout (デフォルト値は `-1`)

ワーカーが要求を処理できるようになるまで待機する時間を秒数で指定します。バランサのすべてのワーカーが使用できない場合、しばらくしてから (workerTimeout/100) mod_cluster が使用可能なワーカーを見つけようとします。

値が **`-1`** の場合、httpd はワーカーが利用可能になるまで待機せず、ワーカーが利用可能でない場合はエラーを返します。

maxAttempts (デフォルト値は `1`)

異常終了するまで httpd プロキシが該当する要求をワーカーに送信しようとする回数を指定します。最小値は 1 であり、異常終了する前に 1 度だけ送信を試行します。

flushPackets (デフォルト値は `false`)

パケットフラッシュが有効であるか、無効であるかを指定します。

flushWait (デフォルト値は `-1`)

パケットを破棄するまで待機する時間を指定します。値が **`-1`** の場合は永遠に待機します。

ping (デフォルト値は `10`)

ping に対して pong 回答を受けとるまで待機する時間 (秒単位)

smax

ソフト最大アイドル接続数を指定します。最大値は **httpd** スレッド設定 (**ThreadsPerChild** または**1**) によって決定されます。

ttl (デフォルト値は 60)

アイドル接続が持続する時間 (秒単位) を指定します。**smax** しきい値より大きくします。

nodeTimeout (デフォルト値は -1)

エラーを返すまで **mod_cluster** がバックエンドサーバーの応答を待機する時間 (秒単位) を指定します。

mod_cluster は要求を転送する前に常に **cping/cpong** を使用します。**mod_cluster** により使用される **connectiontimeout** 値は **ping** 値です。

balancer (デフォルト値は mycluster)

ロードバランサの名前を指定します。

domain (デフォルト値ではありません)

同じドメイン内の **jvmRoutes** で負荷が分散される方法を指定するオプションのパラメータ。分割されたセッションレプリケーション (バディレプリケーションなど) とともに **domain** が使用されます。

付録C 改訂履歴

改訂 5.1.2-2.400 **2013-10-30** **Rüdiger Landmann**
Rebuild with publican 4.0.0

改訂 5.1.2-2 **2012-07-18** **Anthony Towns**
Rebuild for Publican 3.0

改訂 5.1.2-100 **Thu Dec 8 2011** **Jared Morgan**
JBoss Enterprise Application Platform 5.1.2 GA の変更が含まれます。本書の変更箇所については、『『リリースノート 5.1.2』』を参照してください。

改訂 5.1.1-100 **Mon Jul 18 2011** **Jared Morgan**
JBoss Enterprise Application Platform 5.1.1 GA の変更が含まれます。本書の変更箇所については、『『リリースノート 5.1.1』』を参照してください。