making business intelligence easy

Yellowfin クラスタリングガイド

Version 7.2

June 2016

Yellowfin ®

Release 7.2

Clustering Guide

Version: 1.8

Published: June 2016

目次

Yellowfin クラスタリング 概要

Yellowfin は高可用性と負荷分散を可能にするため、複数サーバーでのクラスタリングが可能です。実際の負荷分散（外部からの多重リクエスト）にはハードウェア・ソフトウェア問わずロードバランサーが必要です。この資料ではクラスタリングを行うために必要な Yellowfin の設定変更について概説していますが、送信されたリクエストを多重化する外部環境については触れておりませんのでご了承ください。

クラスタリング構造にする場合、パケットが直接接続しているかのように、ネットワークトラフィックを透過的にアプリケーションサーバーに送信することが必要になります。もしセッションレプリケーションが有効でない場合には、ユーザーセッションのトラフィックが同じノードに送信されるように、”sticky-session”機能を使用する必要があります。

Yellowfin では二段階でのクラスタリングをサポートしています：

- アプリケーションレベルの連携（必須）
- コンテナレベルのセッションレプリケーション（任意）

アプリケーションレベルの連携（必須）

アプリケーション全体での連携（メッセージの送信）が必要な場合に、Yellowfin は他のノードと通信を行います。たとえばキャッシュの更新や同時ユーザーがログアウトしたといったメッセージを通信しています。これは、他のノードと通信を行っている Yellowfin 内のバックグラウンドサーブレットによってコントロールされています。

また、Yellowfin は cluster safe webservice を使用しています。アプリケーションから他のノードにシングルサインオンのリクエストを送り、トークン ID を返すことができます。このトークン ID はその後、受け取ったノード上で有効になります。

コンテナレベルのセッションレプリケーション（任意）

Yellowfin ではまた、ユーザーのセッション情報を複数のノード上で同時に複製することも可能です。ユーザーのセッション情報はメモリフットプリントとして、セッションごとにサーバーに格納されます。セッションレプリケーションを有効にしておくことで、あるノードで接続に失敗してもセッションを保持し続けることができるようになります。

セッションレプリケーションが無効の場合、ログインしていたノードがダウンするとそのユーザーのセッション情報は破棄され、別のノードに再ログインする必要があります。

セッションレプリケーションは Java Container の機能です。この資料では、Tomcat でセッションレプリケーションを行う場合の一例を記載しています。

Yellowfin データベースのクラスタリング

Yellowfin のクラスタリングでは、1 つのリポジトリデータベースを共有することを前提としています。データベースは 1 つのインスタンスか、データベース自体にクラスタリング設定したものを 1 つ用意し、全てのノードで共有するようにしてください。データベースのクラスタリングやレプリケーションについては、データベースの種類によって実現方法が異なりますので、使用しているデータベースに合わせて設定してください。また、Yellowfin からは常に 1 つのデータベース URL（接続情報）でデータベースに接続を行うため、データベースのクラスタリングやレプリケーションは Yellowfin からは透過的である必要があります。

ライセンスについて

Yellowfin のライセンス情報はリポジトリデータベースに格納されています。そのため、クラスタリングを行うためには全てのノードのホスト名を含んだライセンスファイルが必要です。クラスタリング用のライセンスファイルが必要な場合は弊社にお問合せください。

クラスタリング構成での Yellowfin のインストール

クラスタリングを行う場合、リポジトリデータベースは共通のものを 1 つだけ用意する必要があります。そのため、インストール手順が通常のインストール手順とは異なります。

1 つめのノードへのインストールは、通常のインストールと同じ手順で行ってください。コマンドラインでのインストール、GUI でのインストール、どちらでも構いません。

残りのノードではリポジトリデータベースを作成する必要はありません。以下のどちらかの方法でインストールを行ってください。

ファイルの複製

各ノードでフォルダー（ファイル）構成が同じの場合、Yellowfin のインストールディレクトリを残りのノードへ複製することでインストールすることができます。ただし、この資料で後述する web.xml の変更については各ノードで行う必要があります。

個別インストール

同じインストーラーを使用して各ノードに Yellowfin をインストールすることも可能です。ただし、この方法ではノードごとにリポジトリデータベースが作成されますので、各ノードへのインストールが終了したら、不要なリポジトリデータベースを削除してください。1 つめのノードへのインストールで作成されたリポジトリデータベースを参照できるよう、各ノードの web.xml の変更を忘れずに行ってください。

Yellowfin インストールディレクトリ/appsever/webapps/ROOT/WEB-INF/web.xml にリポジトリデータベースとの接続設定が記載されていますので、 2 つめ以降のノードは 1 つめのノードと設定を合わせてください。

ここでは、リポジトリデータベースに Postgresql を使用した場合の例を記載します。

JDBCDriverClass や JDBCUrl のパラメーターなどはお使いのデータベース製品に合わせてください。

```
<!-- Initiate JDBC Connection Pooling - Setup/Configuration -->
  <servlet>
    <servlet-name>InitConnectionPool</servlet-name>
    <servlet-class>com.hof.servlet.InitConnectionPool</servlet-class>
    <init-param>
      <param-name>Description</param-name>
      <param-value>Yellowfin Connection Pool</param-value>
    </init-param>
    <init-param>
      <param-name>JDBCDriverClass</param-name>
      <param-value>org.postgresql.Driver</param-value>
    </init-param>
    <init-param>
      <param-name>JDBCUrl</param-name>
      <param-value>jdbc:postgresql://xxx.xxx.xxx.xxx:5432/yellowfin</param-value>
    </init-param>
    <init-param>
        <param-name>JDBCUser</param-name>
        <param-value>DBuser</param-value>
    </init-param>
    【コメント省略】
    <init-param>
      <param-name>JDBCPassword</param-name>
      <param-value>Password</param-value>
    </init-param>
    <init-param>
```

```
      <param-name>JDBCPasswordEncrypted</param-name>
      <param-value>true</param-value>
    </init-param>
    <init-param>
      <param-name>JDBCMaxCount</param-name>
      <param-value>25</param-value>
    </init-param>
    <init-param>
      <param-name>JDBCMinCount</param-name>
      <param-value>2</param-value>
    </init-param>
    <load-on-startup>4</load-on-startup>
  </servlet>
```

アプリケーションレベルで連携するための設定

Yellowfin の各ノードは、”Cluster Aware"を使用するため ClusterManagement サーブレットを有効にする必要があります。ClusterManagement サーブレットは、各ノードの web.xml ファイルに設定を追加することで有効になります。

アプリケーションの連携は webservice を介して実行され、異なるノードがキャッシュをクリアしたりライセンス詳細を変更したりすることを可能にします。

また、アプリケーションの連携方法は 2 通りの設定方法があります。

- Multicast Cluster Messaging（DYNAMIC モード）
- Webservice Cluster Messaging （LEGACY モード）

Multicast Cluster Messaging (DYNAMIC モード)

Yellowfin のノード間の連携は JGroups と呼ばれる高度な設定が可能なライブラリーによってハンドリングされます。この方法を使うことで、同じリポジトリデータベースを共有している他のノードを自動で探し出すことができます。また、Dynamic モードにすることで、同一サーバー内に複数のインスタンスを保持することが可能になります。

JGroups のデフォルト構成では、グループメンバーシップを確定し、新しいノードを見つけるために UDP マルチキャストメッセージを使用します。環境によってはこのタイプのメッセージの送信ができない場合があります。たとえば、AWS ではノード間の内部ネットワーク上でマルチキャストパケットを許可していません。Multicast Cluster Messaging アダプターを使用すると、ノード検索に他の方法を使用するよう、JGroups を設定する xml ファイルを渡すことができます。このファイルは、ClusterManagement サーブレット内の BroadcastConfiguration サーブレットのパラメーターにパスを渡すことで参照することができます。

※パラメーターの詳細については、web.xml ファイルのパラメーターを参照ください。

Webservice Cluster Messaging　(LEGACY モード)

Yellowfin の Legacy Cluster Messaging は AXIS Webservice によってハンドリングされます。全てのノードが起動時に定義されている必要があり、サービスのエンドポイント、ポート、ユーザーとパスワードが各ノードの web.xml ファイルで定義されている必要があります。Legacy モードではクラスターのインスタンスが同じホスト上に存在することはできません。

web.xml の変更（DYNAMIC モードの場合）

各ノードの、Yellowfin インストールディレクトリ/appserver/webapps/ROOT/WEB-INF/web.xml に対して、以下の ClusterManagement サーブレット定義を追加してください。

```
<!-- Cluster Management -->
<servlet>
        <servlet-name>ClusterManagement</servlet-name>
        <servlet-class>com.hof.mi.servlet.ClusterManagement</servlet-class>
        <init-param>
                <param-name>ClusterType</param-name>
                <param-value>DYNAMIC</param-value>
        </init-param>
        <init-param>
                <param-name>SerialiseWebserviceSessions</param-name>
                <param-value>true</param-value>
        </init-param>
        <init-param>
                <param-name>CheckSumRows</param-name>
                <param-value>true</param-value>
        </init-param>
        <init-param>
                <param-name>EncryptSessionId</param-name>
                <param-value>true</param-value>
        </init-param>
        <init-param>
                <param-name>EncryptSessionData</param-name>
                <param-value>true</param-value>
        </init-param>
        <init-param>
                <param-name>AutoTaskDelegation</param-name>
                <param-value>true</param-value>
        </init-param>
        <load-on-startup>11</load-on-startup>
</servlet>
```

※パラメーターの詳細については、web.xml ファイルのパラメーターを参照ください。

web.xml の変更（LEGACY モードの場合）

各ノードの、Yellowfin インストールディレクトリ/appserver/webapps/ROOT/WEB-INF/web.xml に対して、以下の ClusterManagement サーブレット定義を追加してください。

```
<!-- Cluster Management -->

<servlet>
        <servlet-name>ClusterManagement</servlet-name>
        <servlet-class>com.hof.mi.servlet.ClusterManagement</servlet-class>
        <init-param>
                <param-name>ServiceUser</param-name>
                <param-value>admin@yellowfin.com.au</param-value>
        </init-param>
        <init-param>
                <param-name>ServicePassword</param-name>
                <param-value>test</param-value>
        </init-param>
        <init-param>
                <param-name>ServiceAddress</param-name>
                <param-value>/services/AdministrationService</param-value>
        </init-param>
        <init-param>
                <param-name>ServicePort</param-name>
                <param-value>80</param-value>
        </init-param>
        <init-param>
                <param-name>ClusterHosts</param-name>
                <param-value>
                        192.168.4.184
                </param-value>
        </init-param>
        <init-param>
                <param-name>SerialiseWebserviceSessions</param-name>
                <param-value>true</param-value>
        </init-param>
        <init-param>
                <param-name>CheckSumRows</param-name>
                <param-value>true</param-value>
```

```
        </init-param>
        <init-param>
                <param-name>EncryptSessionId</param-name>
                <param-value>true</param-value>
        </init-param>
        <init-param>
                <param-name>EncryptSessionData</param-name>
                <param-value>true</param-value>
        </init-param>
        <load-on-startup>11</load-on-startup>
</servlet>
```

※パラメーターの詳細については、web.xml ファイルのパラメーターを参照ください。

web.xml ファイルのパラメーター

ここでは、ClusterManagement サーブレットに定義するパラメーターの概要について説明します。

プロパティ	パラメーター
ClusterType	DYNAMIC または LEGACY のどちらかを定義します。 DYNAMIC はマルチキャストメッセージングを使用し、クラスタリング内の他のノードを検出します。LEGACY は webservice を使用してあらかじめ定義したノード間で通信を行います。デフォルトは、LEGACY に設定されています。
BroadcastConfiguration	JGroups の構成ファイルです。マルチキャストネットワーキングが無効な環境で使用できるよう、JGroups の構成をカスタマイズすることが可能です。このパラメーターは DYNAMIC モードでのみ有効なオプションで、デフォルトでは udp.xml で定義されている構成を使用します。
ServiceUser	他のノードへの webservice 接続時に使用するユーザーを定義します。LEGACY モードでのみ使用します。
ServicePassword	ServiceUser のパスワードを定義します。LEGACY モードでのみ使用されます。
ServicePasswordEncrypted	True または False のどちらかを定義します。 ServicePassword の値が暗号化されている場合は True、暗号化されていない場合は False を設定します。
ServiceAddress	Yellowfin webservice のパスを定義します。 LEGACY モードでのみ使用します。
ServicePort	Yellowfin の起動ポートを定義します。 LEGACY モードでのみ使用します。
ClusterHosts	クラスタリング内全てのノードを IP アドレスかホスト名で定義します。複数のノードがあるときはカンマで区切ります。LEGACY モードでのみ使用します。

SerialiseWebserviceSessions	True または False のどちらかを定義します。 クラスタリングでシングルサインオンを使用する際に必要なパラメーターです。どのノードからもアクセスできるように、データベースへのトークンをシリアライズします。
CheckSumRows	True または False のどちらかを定義します。 データベースにシリアライズされた webservice のセッションレコードの合計数を確認するセキュリティオプションです。Yellowfin で未認証のセッションが作成されてしまうようなデータベースへの変更を防ぐことができます。
EncryptSessionId	True または False のどちらかを定義します。 データベース内のシリアライズされた webservice のセッション ID を暗号化するセキュリティオプションです。Yellowfin で未認証のセッションが作成されてしまうようなデータベースへの変更を防ぐことができます。
EncryptSessionData	TrueまたはFalseのどちらかを定義します。 データベース内のシリアライズされたwebserviceのセッションレコードを暗号化するセキュリティオプションです。Yellowfinで未認証のセッションが作成されてしまうようなデータベースへの変更を防ぐことができます。
AutoTaskDelegation	TrueまたはFalseのどちらかを定義します。 このパラメーターはDYNAMICモードでのみ有効なオプションです。Trueに設定すると、バックグラウンドタスクを実行するノードが自動で割り当てられるため、手動でノードを設定する必要がなくなります。既に他のノードにシステムタスクが割り当てられていた場合、起動時のシステムタスクは実行されません。
SessionReplication	TrueまたはFalseのどちらかを定義します。 セッションレプリケーションを有効にする際には、このパラメーターをTrueにします。クラスタリング内のセッションを破棄する際に使われるロジックが変更されます。

追加のパラメーター

環境によっては、DYNAMIC モードが動作しない場合があります。その理由は主にサーバーのネットワーク構成によるものです。JGroups は、利用可能であれば IPv6 を使用するようにデフォルト設定されています。IPv4 で正常に動作させるためには、以下のように catalina.sh、または catalina.bat に設定を追加します。

OS が Linux 環境の場合は、Catalina.sh に以下を設定してください。

Catalina.sh

```
JAVA_OPTS="$JAVA_OPTS -Djava.net.preferIPv4Stack=true"
```

OS が Windows 環境（コンソール起動）の場合は、Catalina.bat に以下を設定してください。

Catalina.bat

```
set JAVA_OPTS=%JAVA_OPTS% -Djava.net.preferIPv4Stack=true
```

OS が Windows 環境（サービス起動）の場合は、Java Options に以下を設定してください。

Java Options の設定方法は「付録 1 ：Windows サービスの Java オプションの変更方法」を参照してください。

```
-Djava.net.preferIPv4Stack=true
```

デフォルトの JGroups 構成（udp.xml）を使用している場合、マルチキャストアドレスとポートも設定することができます。クラスタリングノードがお互いを検出できない環境で有効な設定です。

Yellowfin のデフォルトの構成では udp.xml は使用していません。

追加で udp.xml を使用する場合に以下の設定が有効になります。

OS が Linux 環境の場合は、Catalina.sh に以下を設定してください。

Catalina.sh

```
JAVA_OPTS="$JAVA_OPTS -Djgroups.udp.mcast_addr=228.0.0.5 -Djgroups.udp.mcast_port=47885"
```

OS が Windows 環境（コンソール起動）の場合は、Catalina.bat に以下を設定してください。

Catalina.bat

```
set JAVA_OPTS=%JAVA_OPTS% -Djgroups.udp.mcast_addr=228.0.0.5 -Djgroups.udp.mcast_port=47885
```

OS が Windows 環境（サービス起動）の場合は、Java Options に以下を設定してください。

Java Options の設定方法は「付録 1 ：Windows サービスの Java オプションの変更方法」を参照してください。

```
-Djgroups.udp.mcast_addr=228.0.0.5

-Djgroups.udp.mcast_port=47885
```

バックグラウンドタスク

レポートのブロードキャスト配信を含むバックグラウンドタスクは、Yellowfin の各ノードで実行されるようにデフォルトで設定されています。この場合、エンドユーザーは複数回同じレポートを受け取ることになります。これを防ぐためには、ここで説明しているバックグラウンドタスク（SystemTasks および Task Scheduler）は 1 つのノードでのみ有効にし、2 つめ以降のノードでは無効にすることを推奨しています。なお、アプリケーションの連携方法が DYNAMIC モードに設定されていて、AutoTaskDelegation のパラメーターが True に設定されている場合、バックグラウンドタスクを無効にする設定は不要です。

システムタスク

Yellowfin インストールディレクトリ/appserver/webapps/ROOT/WEB-INF/web.xml から以下のブロックを削除、またはコメントアウトしてバックグラウンドシステムタスクを無効にします。これにより、LDAP との同期、Event テーブルのアーカイブ、レポートの履歴の削除、平均実行時間の算出などのシステムタスクが無効になります。

```
<servlet>
        <servlet-name>SystemTaskManager</servlet-name>
        <servlet-class>com.hof.servlet.SystemTaskManager</servlet-class>
        <load-on-startup>8</load-on-startup>
</servlet>
```

タスクスケジューラー

Yellowfin インストールディレクトリ/appserver/webapps/ROOT/WEB-INF/web.xml の MIStartup サーブレットブロックに以下を追加することで、タスクスケジューラーを無効にします。これにより、ブロードキャスト、キャッシュフィルターの更新、レポートのスケジュール更新、ビューのキャッシュといった、Yellowfin のスケジュール管理画面に表示される全てのタスクが無効になります。

```
<init-param>
      <param-name>DisableTaskScheduler</param-name>
      <param-value>TRUE</param-value>
</init-param>
```

コンテナレベルのセッションレプリケーション

コンテナレベルのセッションレプリケーションは、クラスタリング内のあるノードがダウンした際、セッションを継続できる他のノードに配置することを可能にします。これはアプリケーションサーバーレベルでのセッションレプリケーションを実現するものであり、Yellowfin そのものの機能ではありません。したがって、Yellowfin のサポートコンポーネントではありません。また、送信されたリクエストを異なるノードに分散させるためのロードバランサーが必要になりますが、本資料内では触れておりませんのでご了承ください。

アプリケーションサーバーによって実現方法が異なります。ここでは Yellowfin に同梱している Tomcat を使用した例を記載しています。

また、コンテナレベルのセッションレプリケーションを設定するためにはアプリケーションレベルで連携するための設定が完了している必要があります。

以下にコンテナレベルのセッションレプリケーションの設定手順を記載します。

1. web.xml の変更
2. server.xml の変更
3. ROOT.xml の変更

Tomcat のセッションレプリケーションに関する詳しい情報は、 http://tomcat.apache.org で見つけることができます

注意事項

Yellowfin（や、他の Java Web アプリケーション）でセッションレプリケーションを行う際、いくつか注意点があります。それはセッションレプリケーションのオーバーヘッドとメモリの使用量です。

クラスタリング内の全てのノードにそれぞれのセッションを複製する際、追加でオーバーヘッドがあります。これは各ユーザーのリクエスト後に発生し、デフォルトでは変更があったセッション要素だけが他のノードにコピーされます。つまり、ユーザーが接続先のノードで何かするたびに、全ノードで追加の処理が発生することになります。

セッションレプリケーション機能とは、クラスタリング内の全てのノードにユーザーのメモリフットプリントを複製することです。つまり、クラスタリング内の全てのユーザーが、追加のメモリを占有しながら各ノードに格納されることを意味します。ノード追加によるクラスタリングの拡張は、各ノードのメモリ量によって制限されます。

この設定が本当に必要かどうかは熟考する必要があります。もしセッションレプリケーションが無効の場合にノードがダウンしたら、ユーザーのセッションは破棄されますが、他のノードにログインし直すことで作業を続けることができます。

web.xml の変更

全てのノードの Yellowfin インストールディレクトリ/appserver/webapps/ROOT/WEB-INF/web.xml に対して、2 つ設定を追加します。

1 つめは<web-app>行の下に<distributable/>を追加してください。

```
<web-app>
        <distributable/>
        <!-- System Event and Debug classes -->
        <listener>
                <listener-class>com.hof.servlet.SysSessionListener</listener-class>
        </listener>
```

2 つめは ClusterManagement サーブレット内に以下を追加してください。

```
<init-param>
        <param-name>SessionReplication</param-name>
        <param-value>true</param-value>
</init-param>
```

server.xml の変更

全てのノードの Yellowfin インストールディレクトリ/appserver/conf/server.xml に対して、<host>ブロック内に以下を追加してください。

```
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
```

IP アドレス「224.0.0.0」は、セッションレプリケーションメッセージが受信されたマルチキャストのアドレスです。特に変更する必要はありませんが、Linux サーバーまたは UNIX サーバーは、このアドレスにブロードキャストを受信するための有効なマルチキャストルートを登録する必要があります。

コンソールから、以下のコマンドを実行して登録することができます。

```
sudo route add -net 224.0.0.0 netmask 224.0.0.0 dev eth0
```

※上記のコマンドはルートに使用されるネットワークデバイスとして、「eth0」を使用しています。

ROOT.xml の変更

全てのノードの Yellowfin インストールディレクトリ/appserver/conf/Catalina/localhost/ROOT.xml に記載されている以下の行を削除してください。

```
<Manager className="org.apache.catalina.session.PersistentManager" debug="0" distributable="false" saveOnRestart="false">
        <Store className="org.apache.catalina.session.FileStore" />
</Manager>
```

付録 1

Windows サービスの Java オプションの変更方法

1. Yellowfin サービスを停止します。
2. 管理者でコマンドプロンプトを起動します。
3. コマンドプロンプトで cd <Yellowfin インストールフォルダー>¥appserver¥bin と入力して実行します。

```
cd C:¥Yellowfin¥appserver¥bin
```

4. コマンドプロンプトで次のコマンドを実行します。

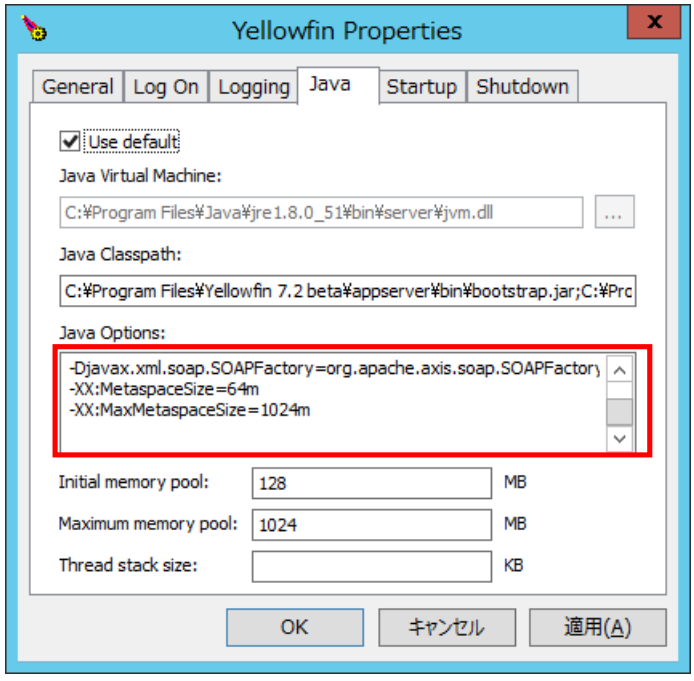
```
tomcat7w.exe //ES//Yellowfin
```

5. Tomcat のサービス設定画面が表示されるので、画面の「Java」タブの「Java Options」の欄に Java オプションを追記、変更します。

   以下は MetaspaceSize の設定の例です。

```
-XX:MetaspaceSize=64m

-XX:MaxMetaspaceSize=1024m
```



付録 2

UDP マルチキャストメッセージが使用できない場合の設定例

Yellowfin のノード間の連携は Dynamic モードが推奨ですが、UDP マルチキャストが使用できないサーバー環境の場合があります。代表的なクラウドサービスである AWS EC2、Microsoft Azure、Google Cloud は全て UDP マルチキャストを使用することができません。

JGroups でサポートされている他のプロトコルを使用することで Dynamic モードを実現することができます。サポートされるプロトコルは複数ありますが、ここでは TCPPING を用いて実装を行います。

※サポートされるプロトコルの種類及び設定プロパティの詳細は下記の URL を参照してください。

http://www.jgroups.org/manual/html/protlist.html

web.xml の変更

各ノードの、Yellowfin インストールディレクトリ/appserver/webapps/ROOT/WEB-INF/web.xml に対して、以下の ClusterManagement サーブレット定義を追加してください。
先述の Dynamic モードの設定に BroadcastConfiguration パラメーターが追加されています。
BroadcastConfiguration パラメーターに TCPPING の設定を記載した xml ファイルのフルパスを指定します。

```
<!-- Cluster Management -->
<servlet>
	<servlet-name>ClusterManagement</servlet-name>
	<servlet-class>com.hof.mi.servlet.ClusterManagement</servlet-class>
	<init-param>
		<param-name>ClusterType</param-name>
		<param-value>DYNAMIC</param-value>
	</init-param>
	<init-param>
		<param-name>SerialiseWebserviceSessions</param-name>
		<param-value>true</param-value>
	</init-param>
	<init-param>
		<param-name>CheckSumRows</param-name>
		<param-value>true</param-value>
	</init-param>
	<init-param>
		<param-name>EncryptSessionId</param-name>
		<param-value>true</param-value>
	</init-param>
	<init-param>
		<param-name>EncryptSessionData</param-name>
		<param-value>true</param-value>
	</init-param>
	<init-param>
		<param-name>AutoTaskDelegation</param-name>
		<param-value>true</param-value>
```

```
        </init-param>
        <init-param>
                <param-name>BroadcastConfiguration</param-name>
                <param-value>
                        /usr/local/yellowfin/appserver/webapps/ROOT/WEB-INF/tcp.xml
                </param-value>
        </init-param>
        <load-on-startup>11</load-on-startup>
</servlet>
```

※パラメーターの詳細については、web.xml ファイルのパラメーターを参照ください。

tcp.xml の作成

TCPPING を使用するためには設定を記載した xml 形式のファイルが必要です。
TCPPING の設定ファイルはサンプルが用意されています。それを利用することにより作業を大幅に短縮することができます。

Yellowfin インストールディレクトリ/appserver/webapps/ROOT/WEB-INF/lib/ jgroups-3.2.8.Final.jar を任意の作業フォルダにコピーし拡張子を zip に変更します。
3.2.8.Final.zip を解凍し、ルートフォルダの tcp.xml を編集します。

TPCPING 要素の initial_hosts 属性に「IP アドレス[待受けポート]」のフォーマットで全てのクラスタメンバをカンマ区切りで列挙します。

```
    <TCPPING timeout="3000"
                initial_hosts="${jgroups.tcpping.initial_hosts:192.168.1.10[7800],192.168.1.20[7800]}"
                port_range="1"
                num_initial_members="10"/>
```

tcp.xml を各ノードの、Yellowfin インストールディレクトリ/appserver/webapps/ROOT/WEB-INF にコピーします。

Java 起動オプションの設定

JGroups は、利用可能であれば IPv6 を使用するようにデフォルト設定されています。TCPPING は IPv6 では正常に動作しないため、IPv4 だけを使用する必要があります。そのため、以下のように catalina.sh、または catalina.bat に設定を追加します。

OS が Linux 環境の場合は、Catalina.sh に以下を設定してください。

Catalina.sh

```
JAVA_OPTS="$JAVA_OPTS -Djava.net.preferIPv4Stack=true"
```

OS が Windows 環境（コンソール起動）の場合は、Catalina.bat に以下を設定してください。

Catalina.bat

```
set JAVA_OPTS=%JAVA_OPTS% -Djava.net.preferIPv4Stack=true
```

OS が Windows 環境（サービス起動）の場合は、Java Options に以下を設定してください。

Java Options の設定方法は「付録 1 ：Windows サービスの Java オプションの変更方法」を参照してください。

```
-Djava.net.preferIPv4Stack=true
```