Sun GlassFish Communications Server 1.5 配備計画ガイド

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–7389–10

090310@21990

目次

図目次

表目次

例目次

はじめに

『配備計画ガイド』では、本番配備を構築する方法について説明します。

ここでは、Sun GlassFish™ Communications Server のマニュアルセット全体に関する情報と表記規則について説明しています。

Communications Server のマニュアルセット

Communications Server マニュアルの URL (Uniform Resource Locator) は、`http://docs.sun.com/coll/1343.8` です。Communications Server への導入としては、次の表に示されている順序でマニュアルを参照してください。

表 P–1　Communications Server のマニュアルセットの内容

マニュアル名	説明
『Documentation Center』	タスクや主題ごとに整理された Communications Server のマニュアルのトピック。
『リリースノート』	ソフトウェアとマニュアルに関する最新情報。サポートされているハードウェア、オペレーティングシステム、Java™ Development Kit (JDK™)、およびデータベースドライバの包括的な表ベースの概要を含みます。
『クイックスタートガイド』	Communications Server 製品の使用を開始するための手順。
『Installation Guide』	ソフトウェアとそのコンポーネントのインストール。
『アプリケーション配備ガイド』	アプリケーションおよびアプリケーション コンポーネントの Communications Server への配備。配備記述子に関する情報を含みます。
『開発者ガイド』	Communications Server 上で動作することを目的とし、Java EE コンポーネントおよび API のオープン Java スタンダードモデルに準拠した、Java Platform, Enterprise Edition (Java EE プラットフォーム) アプリケーションの作成と実装。開発者ツール、セキュリティー、デバッグ、ライフサイクルモジュールの作成に関する情報を含みます。
Java EE 5 Tutorial	Java EE 5 プラットフォームテクノロジと API を使用した Java EE アプリケーションの開発。

表 P–1 Communications Server のマニュアルセットの内容 (続き)

マニュアル名	説明
『Java WSIT Tutorial』	Web サービス相互運用性テクノロジ (WSIT) を使用した Web アプリケーションの開発。WSIT テクノロジを使用する方法、時期、および理由と、各テクノロジがサポートする機能およびオプションについて説明します。
『管理ガイド』	設定、監視、セキュリティー、資源管理、および Web サービス管理を含む Communications Server のシステム管理。
『高可用性 (HA) 管理ガイド』	クラスタの設定、ノードエージェントの使用、およびロードバランサの使用。
『Administration Reference』	Communications Server 設定ファイル `domain.xml` の編集。
『パフォーマンスチューニングガイド』	パフォーマンスを向上させるための Communications Server の調整。
『Reference Manual』	Communications Server で使用できるユーティリティーコマンド。マニュアルページのスタイルで記述されています。`asadmin` コマンド行インタフェースも含みます。

関連マニュアル

その他のスタンドアロンの Sun GlassFish サーバー製品については、次のマニュアルを参照してください。

- メッセージキュー documentation (`http://docs.sun.com/coll/1343.4`)
- Identity Server documentation (`http://docs.sun.com/app/docs/prod/ident.mgmt#hic`)
- Directory Server documentation (`http://docs.sun.com/coll/1224.1`)
- Web サーバー documentation (`http://docs.sun.com/coll/1308.3`)

Communications Server とともに提供されるパッケージの Javadoc™ ツールリファレンスの場所は `http://glassfish.dev.java.net/nonav/javaee5/api/index.html` です。さらに、次のリソースが役立つことがあります。

- Java EE 5 Specifications (`http://java.sun.com/javaee/5/javatech.html`)
- Java EE Blueprints (`http://java.sun.com/reference/blueprints/index.html`)

NetBeans™ 統合開発環境 (IDE) でのエンタープライズアプリケーション開発については、`http://www.netbeans.org/kb/55/index.html` を参照してください。

Communications Server に含まれる Java DB データベースの詳細は、`http://developers.sun.com/javadb/` を参照してください。

GlassFish Samples プロジェクトは、さまざまな Java EE テクノロジの使用方法の例を示すサンプルアプリケーションの集合です。GlassFish Samples は Java EE Software Development Kit (SDK) に付属しています。また、`https://glassfish-samples.dev.java.net/` の GlassFish Samples プロジェクトページからも入手できます。

デフォルトのパスおよびファイル名

次の表は、このマニュアルで使用されているデフォルトのパス名とファイル名について説明したものです。

表 P–2 デフォルトのパスおよびファイル名

プレースホルダ	説明	デフォルト値
as-install	Communications Server のベースインストールディレクトリを表します。	Solaris™ および Linux プラットフォームへのインストールで、ルートユーザーでない場合: *user's-home-directory*/SUNWappserver Solaris および Linux プラットフォームへのインストールで、ルートユーザーである場合: /opt/SUNWappserver Windows のすべてのインストールの場合: *SystemDrive*:\Sun\AppServer
domain-root-dir	すべてのドメインを含むディレクトリを表します。	すべてのインストールの場合: *as-install*/domains/
domain-dir	ドメインのディレクトリを表します。 設定ファイルには、次のように表される *domain-dir* があります。 ${com.sun.aas.instanceRoot}	*domain-root-dir*/*domain-dir*
instance-dir	サーバーインスタンスのディレクトリを表します。	*domain-dir*/*instance-dir*
samples-dir	サンプルアプリケーションを含むディレクトリを表します。	*as-install*/samples
docs-dir	マニュアルを含むディレクトリを表します。	*as-install*/docs

表記上の規則

この表は、このドキュメントで使用される表記の種類について説明したものです。

表 P–3 表記上の規則

書体	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、および画面上のコンピュータ出力	.login ファイルを編集します。 ls -a を使用してすべてのファイルを表示します。 machine_name% you have mail.
AaBbCc123	ユーザーが入力する文字 (画面上のコンピュータ出力と区別して表示)	machine_name% **su** パスワード:
AaBbCc123	実際の名前または値に置き換えられるプレースホルダ	ファイルを削除するコマンドは、rm *filename* です。
AaBbCc123	書名、新しい用語、および強調する用語 (オンラインでは、一部の強調項目は太字で表示される)	ユーザーズガイドの第 6 章を参照してください。 キャッシュとは、ローカルに格納されているコピーです。 ファイルを保存しないでください。

記号の表記ルール

この表は、このマニュアルで使用される記号について説明したものです。

表 P–4 記号の表記ルール

記号	説明	例	意味
[ ]	省略可能な引数やコマンドオプションが含まれます。	ls [-l]	-l オプションは必須ではありません。
{ | }	必須コマンドオプションの選択項目が含まれています。	-d {y|n}	-d オプションには、y 引数または n 引数のいずれかを使用する必要があります。
${ }	変数参照を示します。	${com.sun.javaRoot}	com.sun.javaRoot 変数の値を参照します。
-	同時に実行する複数のキーストロークを結び付けます。	Control-A	コントロールキーを押しながら A キーを押します。
+	連続で複数のキーストロークを行います。	Ctrl + A + N	Control キーを押して離してから、次のキーを押します。

表 P–4 記号の表記ルール (続き)

記号	説明	例	意味
→	グラフィカルユーザーインタフェースでのメニュー項目の選択を示します。	「ファイル」→「新規」→「テンプレート」	「ファイル」メニューから「新規」を選択します。「新規」サブメニューから、「テンプレート」を選択します。

マニュアル、サポート、およびトレーニング

Sun の Web サイトには、次に示す関連情報が示されています。

- ドキュメント (http://www.sun.com/documentation/)
- サポート (http://www.sun.com/support/)
- トレーニング (http://www.sun.com/training/)

第三者の Web サイト参照

このマニュアル内で参照している第三者の URL は、追加の関連情報を提供します。

注 – このマニュアル内で引用する第三者の Web サイトの可用性について Sun は責任を負いません。こうしたサイトやリソース上の、またはこれらを通じて利用可能な、コンテンツ、広告、製品、その他の素材について、Sun は推奨しているわけではなく、Sun はいかなる責任も負いません。こうしたサイトやリソース上の、またはこれらを経由して利用可能な、コンテンツ、製品、サービスを利用または信頼したことに伴って発生した (あるいは発生したと主張される) いかなる損害や損失についても、Sun は一切の責任を負いません。

このマニュアルに関するコメント

弊社では、マニュアルの改善に努めており、お客様からのコメントおよびご忠告をお受けしております。

コメントを共有するには、http://docs.sun.com を参照し、「Feedback」をクリックしてください。このオンラインフォームでは、マニュアルのタイトルと Part No. もご記入ください。Part No. は、7 桁か 9 桁の番号で、マニュアルのタイトルページまたは最初のページに記載されています。

◆ ◆ ◆ 第 1 章

1 製品概念

Sun GlassFish Communications Server は、Java EE の統合および SIP アプリケーションの開発、融合、および管理のための堅牢なプラットフォームを提供します。主な機能には、スケーラブルトランザクション管理、Web サービスパフォーマンス、クラスタ化、セキュリティー、および統合機能があります。

この章の内容は次のとおりです。

Java EE プラットフォームの概要

Communications Server は、Java 2 Enterprise Edition (Java EE) 1.4 テクノロジを実装します。Java EE プラットフォームは、アプリケーションコンポーネント、API、およびアプリケーションサーバーの実行時コンテナとサービスについて記述した標準仕様のセットです。

Java EE アプリケーション

Java EE アプリケーションは、JavaServer Pages (JSP)、Java サーブレット、Enterprise JavaBeans (EJB) モジュールなどのコンポーネントで構成されます。ソフトウェア開発者はこれらのコンポーネントを利用して、大規模分散アプリケーションを構築できます。開発者は Java EE アプリケーションを、zip ファイルに似た Java アーカイブ (JAR) ファイルにパッケージ化して本番サイトに配布できます。管理者は、Java EE JAR ファイルを 1 つ以上のサーバーインスタンス (またはインスタンスのクラスタ) に配備することにより、Java EE アプリケーションを Application Server 上にインストールします。

次の図は、以降の節で説明する Java EE プラットフォームのコンポーネントを示したものです。

申し訳ございません: 現在、図が用意できておりません。

コンテナ

各サーバーインスタンスには、2 つのコンテナ (Web コンテナと EJB コンテナ) が含まれます。コンテナは、Java EE コンポーネントのセキュリティーやトランザクション管理などのサービスを提供する実行時環境です。JavaServer Pages (JSP) やサーブレットなどの Web コンポーネントは、Web コンテナ内で実行されます。Enterprise JavaBeans は EJB コンテナ内で実行されます。

Java EE サービス

Java EE プラットフォームは、次のようなサービスをアプリケーションに対して提供します。

- ネーミング -ネームサービスとディレクトリサービスは、オブジェクトを名前にバインドします。Java EE アプリケーションは、オブジェクトの Java Naming and Directory Interface (JNDI) 名を検索することによってオブジェクトを検出できます。
- セキュリティー - Java Authorization Contract for Containers (JACC) は、Java EE コンテナに関して定義された一連のセキュリティー規約です。クライアントの ID に基づいて、コンテナはコンテナのリソースおよびサービスに対するアクセスを制限できます。
- トランザクション管理 -トランザクションは作業の分割不能な単位です。たとえば、銀行口座間での資金の振り替えがトランザクションにあたります。トランザクション管理サービスは、トランザクションが完了するか、またはロールバックされるかの二者択一性を保証します。
- メッセージサービス - 別々のシステム上でホストされたアプリケーションどうしが、Java™ Message Service (JMS) を利用してメッセージを交換することによって互いに通信できます。JMS は Java EE プラットフォームの根幹的な部分であり、異機種システム混在のエンタープライズアプリケーションを統合する作業を簡略化します。

Web サービス

クライアントは HTTP、RMI/IIOP、JMS 経由でのアクセスに加えて、リモート Web サービスとして Java EE アプリケーションにアクセスできます。Web サービスは、Java API for XML-based RPC (JAX-RPC) を使用して実装されます。Java EE アプリケーションは Web サービスに対するクライアントとして動作することもでき、これはネットワークアプリケーションで典型的な構成です。

Web Services Description Language (WSDL) は、Web サービスのインタフェースを記述する XML 形式です。Web サービスのコンシューマは、WSDL ドキュメントを動的に解析して、Web サービスが提供する操作とその実行方法を特定できます。Application Server では、ほかのアプリケーションが Java API for XML Registries (JAXR) を経由してアクセス可能なレジストリを使用して、Web サービスインタフェースの記述を分散させています。

クライアントアクセス

クライアントは複数の方法で Java EE アプリケーションにアクセスできます。ブラウザクライアントは、ハイパーテキストトランスファープロトコル (HTTP) を使用して Web アプリケーションにアクセスします。セキュリティー保護された通信のために、ブラウザでは、Secure Sockets Layer (SSL) を使用する HTTPS プロトコルを利用します。

アプリケーションクライアントコンテナ内で動作するリッチクライアントアプリケーションは、オブジェクトリクエストブローカ (ORB)、リモートメソッド呼び出し (RMI) および IIOP (internet inter-ORB protocol)、または IIOP/SSL (セキュリティー保護された IIOP) を使用して Enterprise JavaBeans (EJB) を直接検索し、アクセスできます。そのようなアプリケーションは、HTTP/HTTPS、JMS、および JAX-RPC を使用してアプリケーションや Web サービスにアクセスできます。それらのアプリケーションでは JMS を使用して、アプリケーションおよびメッセージ駆動型 Beans との間でメッセージを送受信します。

Java EE Web サービスにアクセスできるのは、WS-I Basic Profile (Web サービス相互運用性基本プロファイル) に準拠したクライアントです。WS-I は Java EE 標準の根幹的な部分であり、Web サービスの相互運用性について定義しています。WS-I に準拠することにより、サポートされているすべての言語で記述されたクライアントが、Application Server に配備された Web サービスにアクセスできます。

最適なアクセス機構は、個別のアプリケーションおよび予想されるトラフィック量によって異なります。Application Server は、HTTP、HTTPS、JMS、IIOP、および IIOP/SSL のそれぞれに対応した設定が可能なリスナーをサポートします。各プロトコルに対して複数のリスナーを設定し、スケーラビリティーと信頼性を高めることができます。

Java EE アプリケーションは、ほかのサーバー上に配備された EJB モジュールなどの Java EE コンポーネントのクライアントとしても動作でき、これらのアクセス機構のうち任意のものを使用できます。

外部システムとリソース

Java EE プラットフォームでは、外部のシステムのことをリソースと呼びます。たとえば、データベース管理システムは JDBC リソースです。各リソースは、その Java Naming and Directory Interface (JNDI) 名によって一意に識別されます。アプリケーションは、次の API とコンポーネントを通して外部システムにアクセスします。

- *Java Database Connectivity (JDBC)* - データベース管理システム (DBMS) は、データを格納、組織化、および取得するための機能を提供します。大部分のビジネスアプリケーションは、アプリケーションが JDBC 経由でアクセスするリレーショナルデータベースにデータを格納します。Application Server に含まれる PointBase DBMS は、サンプルアプリケーションでの使用、アプリケーション開発、プロトタイプ作成などの用途に適していますが、配備用途には適していません。Application Server では、各種の主要リレーショナルデータベースに接続するための、動作確認済み JDBC ドライバを提供しています。これらのドライバは配備に適しています。
- Java Message Service (JMS) - メッセージングは、ソフトウェアコンポーネントまたはアプリケーション間で通信を行うための手段です。メッセージングクライアントは、Java Messaging Service (JMS) API を実装するメッセージングプロバイダを介して、任意のほかのクライアントとの間でメッセージを送受信します。Application Server には、高性能な JMS ブローカーである Sun Java System Message Queue が含まれています。Application Server Platform Edition には、Message Queue の無償版である Platform Edition が含まれます。Sun GlassFishCommunications Server には、クラスタ化とフェイルオーバーをサポートする Message Queue Enterprise Edition が含まれています。
- **Java EE** コネクタ - Java EE コネクタアーキテクチャーは、Java EE アプリケーションと既存の企業情報システム (EIS) との統合を可能にします。アプリケーションは、コネクタまたはリソースアダプタと呼ばれる、移植性のある Java EE コンポーネントを介して、JDBC ドライバを使用した RDBMS へのアクセスと同様の方法で EIS にアクセスします。リソースアダプタは、スタンドアロンのリソースアダプタアーカイブ (RAR) モジュールとして配布されるか、または Java EE アプリケーションアーカイブに組み込まれます。RAR 形式のリソースアダプタは、ほかの Java EE コンポーネントと同様に配備されます。Application Server には、一般的な EIS との統合が可能な評価用のリソースアダプタが含まれています。
- **JavaMail** - アプリケーションは JavaMail API を介して Simple Mail Transport Protocol (SMTP) サーバーにアクセスし、電子メールを送受信できます。

Application Server のコンポーネント

この節では、Sun Java System Application Server のコンポーネントについて説明します。

まず、ブラウザベースの管理コンソールなどの管理ツールがドメイン管理サーバー (DAS) と通信し、その後 DAS がノードエージェントまたはサーバーインスタンスと通信します。

サーバーインスタンス

サーバーインスタンスは、単一の Java 仮想マシン (JVM) プロセス内で実行される Application Server です。Application Server は、Java 2 Standard Edition (J2SE) 5.0 および Java SE 6 での動作が検証されています。推奨される Java EE 配布は Application Server のインストールに含まれています。

Application Server と付属の JVM はともに、マルチプロセッサ構成で拡大縮小するように設計されているため、通常は 1 台のマシン上に 1 つのサーバーインスタンスを作成すれば十分です。ただし、アプリケーションの隔離と順次アップグレードのために、1 台のマシン上に複数のインスタンスを作成すると有利な場合もあります。場合によっては、複数のインスタンスが動作する 1 台の大規模サーバーを複数の管理ドメインで使用できます。管理ツールを使用することで、複数のマシンにまたがってサーバーインスタンスを容易に作成、削除、および管理できます。

管理ドメイン

管理ドメイン (または単にドメイン) は、まとめて管理されるサーバーインスタンスのグループです。1 つのサーバーインスタンスは単一の管理ドメインに属します。ドメイン内のインスタンスは、複数の異なる物理ホスト上で実行できます。

Application Server の 1 つのインストールから複数のドメインを作成できます。サーバーインスタンスをドメインにグループ化することにより、さまざまな組織および管理者が 1 つの Application Server インストールを共有できます。各ドメインには、固有の設定、ログファイル、およびアプリケーションの配備領域があり、こ

れらはほかのドメインとは無関係です。あるドメインの構成を変更しても、ほかのドメインの構成には影響しません。同様に、あるドメインにアプリケーションを配備しても、そのアプリケーションがほかのドメインに配備されたり、ほかのドメインから認識可能になることはありません。管理者は常に 1 つのドメインに対する認証しか受けられず、そのドメイン上での管理しか実行できません。

ドメイン管理サーバー (DAS)

ドメインには 1 つのドメイン管理サーバー (DAS) があります。これは、管理アプリケーションのホストとなる、特別に設計されたアプリケーションサーバーインスタンスです。DAS は管理者を認証し、管理ツールからの要求を受け付け、ドメイン内のサーバーインスタンスと通信して要求を実行します。

管理ツールには、コマンド行ツールの `asadmin` と、ブラウザベースの管理コンソールがあります。Application Server では、サーバー管理のための JMX ベースの API も提供されます。管理者が同時に表示および管理できるのは 1 つのドメインのみであり、これによってセキュリティーで保護された分離が実現されています。

DAS のことを管理サーバーまたはデフォルトサーバーと呼ぶ場合もあります。DAS をデフォルトサーバーと呼ぶ理由は、一部の管理操作のデフォルトターゲットであるためです。

DAS はアプリケーションサーバーインスタンスであるため、テスト目的で Java EE アプリケーションをホストすることもできます。ただし、本番アプリケーションのホストを目的として DAS を使用しないでください。たとえば、本番アプリケーションをホストする予定のクラスタやインスタンスをまだ作成していない場合に、アプリケーションを DAS に配備することができます。

DAS は、各ドメインおよびすべての配備済みアプリケーションの設定を格納するリポジトリを保持します。DAS がアクティブでないか停止している場合、アクティブなサーバーインスタンスのパフォーマンスまたは可用性への影響はありませんが、管理上の変更は実行できません。状況によっては、セキュリティー上の理由から、本番構成を凍結することなどを目的として意図的に DAS プロセスを停止すると便利な場合があります。

ドメイン設定やアプリケーションをバックアップおよび復元するための管理コマンドが用意されています。標準のバックアップ手順および復元手順を使用して、作業中の構成を迅速に復元できます。DAS ホストで障害が発生した場合、以前のドメイン設定を復元するために新しい DAS インストールを作成する必要があります。手順については、『Sun GlassFish Communications Server 1.5 管理ガイド』の「ドメイン管理サーバーの再作成」を参照してください。

Sun Cluster Data Services は、DAS ホストの IP アドレスのフェイルオーバーと、グローバルファイルシステムの使用によって高可用性を実現します。このソリューションにより、多くの種類の障害に対してほとんど停止しないレベルの可用性が、DAS およびリポジトリに対して提供されます。Sun Cluster Data Services

は、Sun Java Enterprise System に付属するか、または Sun Cluster とともに別途購入する形で提供されます。詳細は、Sun Cluster Data Services のマニュアルを参照してください。

クラスタ

クラスタとは、同じアプリケーション、リソース、および設定情報を共有するサーバーインスタンスの集まりに名前を付けたものです。異なるマシン上のサーバーインスタンスを 1 つの論理クラスタにまとめ、それらのインスタンスを 1 つの単位として管理できます。マルチマシンクラスタのライフサイクルは、DAS を使用して容易に制御できます。

クラスタにより、水平方向のスケーラビリティー、負荷分散、およびフェイルオーバー保護が使用可能になります。定義により、クラスタ内のすべてのインスタンスに対してリソースとアプリケーションの設定は同じになります。あるサーバーインスタンスまたはクラスタ内のあるマシンに障害が起きると、ロードバランサは障害を検出し、障害の起きたインスタンスからクラスタ内のほかのインスタンスにトラフィックをリダイレクトし、ユーザーセッションの状態を回復します。クラスタ内のすべてのインスタンス上には同一のアプリケーションとリソースがあるため、インスタンスはクラスタ内のほかのどのインスタンスにも処理を継続させることができます。

クラスタ、ドメイン、およびインスタンスの関係は次のようになります。

- 1 つの管理ドメイン内に任意の数 (0 個を含む) のクラスタを作成できます。
- 1 つのクラスタは 1 つ以上のサーバーインスタンスで構成できます。
- 1 つのクラスタは単一のドメインに属します。

ノードエージェント

ノードエージェントは、DAS をホストするマシンを含め、サーバーインスタンスをホストするすべてのマシン上で実行される軽量プロセスです。ノードエージェントは次の機能を実行します。

- DAS の指示に従い、サーバーインスタンスを起動および停止します。
- 障害の発生したサーバーインスタンスを再起動します。
- 障害が発生したサーバーのログファイルのビューを提供し、リモート診断を支援します。
- DAS がその監視下のサーバーインスタンスを起動するときに、各サーバーインスタンスのローカル設定リポジトリを DAS の集中リポジトリと同期します。
- インスタンスが最初に作成されるとき、インスタンスに必要なディレクトリを作成し、インスタンスの設定を集中リポジトリと同期します。

- サーバーインスタンスが削除されるときに適切なクリーンアップを実行します。

各物理ホストには、ホストが属する各ドメインに対して少なくとも 1 つのノードエージェントが必要です。1 台の物理ホストに、複数のドメインに属するインスタンスを配置する場合、そのホストにはそれぞれのドメインに対するノードエージェントが必要です。ホスト上のドメインごとに複数のノードエージェントを持つことは許可されていますが、利点はありません。

ノードエージェントはサーバーインスタンスを起動および停止するため、常に実行中である必要があります。したがって、ノードエージェントはオペレーティングシステムの起動時に起動されます。Solaris およびその他の Unix プラットフォームでは、ノードエージェントを `inetd` プロセスによって起動できます。Windows では、ノードエージェントを Windows のサービスにすることができます。

ノードエージェントの詳細は、『Sun GlassFish Communications Server 1.5 High Availability Administration Guide』の第 4 章「ノードエージェントの設定」を参照してください。

名前付き設定

名前付き設定は、Application Server のプロパティー設定をカプセル化する抽象化オブジェクトです。クラスタおよびスタンドアロンのサーバーインスタンスは、名前付き設定を参照してそれぞれのプロパティー設定を取得します。名前付き設定を使用することにより、IP アドレス、ポート番号、ヒープメモリー容量など一部の設定を除く Java EE コンテナの設定が、そのコンテナが位置する物理マシンから分離されます。名前付き設定を使用することにより、Application Server の管理をより強力に、また柔軟に行えるようになります。

設定変更を適用するには、単に名前付き設定のプロパティー設定を変更します。変更すると、その設定を参照するすべてのクラスタおよびスタンドアロンインスタンスが変更内容を取得します。名前付き設定の削除は、その設定へのすべての参照が削除済みの場合にしか行えません。1 つのドメインに複数の名前付き設定を含めることができます。

Application Server には、default-config という名称のデフォルト設定が付属します。デフォルト設定は、Application Server Platform Edition の開発者の生産性を高めるように、また、セキュリティーと可用性を高めるように最適化されています。

デフォルト設定をベースにして独自の名前付き設定を作成し、その設定を目的に合わせてカスタマイズできます。名前付き設定の作成と管理には、管理コンソールおよび `asadmin` コマンド行ユーティリティーを使用します。

融合ロードバランサ

ロードバランサは、複数の物理マシン間で作業負荷を分散させることによって、システムの全体的なスループットを向上させます。

ロードバランサプラグインは、SIP、SIPS、HTTP および HTTPS 要求を受け付け、それをクラスタ内のアプリケーションサーバーインスタンスのうちの 1 つに転送します。(ネットワーク障害により) インスタンスが停止して使用不能または応答不能になると、要求は既存の使用可能なマシンにリダイレクトされます。ロードバランサはまた、障害が起きたインスタンスが復旧したことを認識し、それに応じて負荷を再配分することもできます。

クラスタ内の **IIOP** 負荷分散

IIOP 負荷分散では、IIOP クライアントの要求が複数の異なるサーバーインスタンスまたはネームサーバーに分散されます。目標は、負荷をクラスタ間に均等に拡散して、スケーラビリティーを実現することです。IIOP 負荷分散と、EJB クラスタ化および Sun Java System Application Server の 可用性機能を組み合わせることにより、負荷分散に加えて EJB フェイルオーバーも実現されます。

クライアントがオブジェクトに対して JNDI 検索を実行すると、ネームサービスは、特定のサーバーインスタンスに関連付けられた `InitialContext` (IC) オブジェクトを作成します。それ以降、その IC オブジェクトを使用して作成された検索要求はすべて、同じサーバーインスタンスに送信されます。その `InitialContext` を使用して検索された `EJBHome` オブジェクトはすべて、同じターゲットサーバーにホストされます。また、それ以降に取得された Bean 参照もすべて、同じターゲットホスト上に作成されます。`InitialContext` オブジェクトの作成時に、ライブターゲットサーバーのリストがすべてのクライアントによってランダムに選択されるため、これにより負荷分散が効果的に実現されます。ターゲットサーバーインスタンスが停止すると、検索または EJB メソッド呼び出しは、別のサーバーインスタンスに処理が引き継がれます。

たとえば、この図で示されている ic1、ic2、および ic3 は、クライアント 2 のコードで作成される 3 つの異なる `InitialContext` インスタンスです。それらはクラスタ内の 3 つのサーバーインスタンスに分散されます。そのため、このクライアントによって作成される EJB は 3 つのインスタンス間に分散されます。クライアント 1 は InitialContext オブジェクトを 1 つだけ作成したため、このクライアントからの Bean 参照はサーバーインスタンス 1 のみです。サーバーインスタンス 2 がダウンした場合、ic2 の検索要求は別のサーバーインスタンス (サーバーインスタンス 3 とは限らない) にフェイルオーバーします。以前にサーバーインスタンス 2 でホストされていた Bean に対するすべての Bean メソッド呼び出しも、そのように処理するのが安全であれば、別のインスタンスに自動的にリダイレクトされます。検索のフェイルオーバーは自動的ですが、EJB モジュールは再試行が安全なときに限りメソッド呼び出しを再試行します。

RMI-IIOP 負荷分散とフェイルオーバーは、透過的に発生します。アプリケーションの配備中に、特別な手順は必要ありません。クラスタに新しいインスタンスを追加したり削除したりしても、そのクラスタに関する既存のクライアントの表示は更新されません。クライアント側で、端点の一覧を手動で更新する必要があります。

Message Queue と JMS リソース

Sun Java System Message Queue (MQ) は、分散アプリケーションのための、信頼性のある非同期メッセージングを提供します。MQ は Java Message Service (JMS) 標準を実装するエンタープライズメッセージングシステムです。MQ は、メッセージ駆動型 Beans (MDB) などの Java EE アプリケーションコンポーネントに対してメッセージングを提供します。

Application Server は Sun Java System Message Queue を Application Server に統合することにより Java Message Service (JMS) API を実装します。Communications Server には、フェイルオーバー、クラスタ化、および負荷分散機能を備えた、Enterprise バージョンが含まれています。

基本的な JMS 管理タスクには、Application Server の管理コンソールおよび `asadmin` コマンド行ユーティリティーを使用します。

Message Queue クラスタの管理など、高度なタスクの場合は、*install_dir*`/imq/bin` ディレクトリに用意されたツールを使用します。Message Queue の管理の詳細は、『Sun Java System Message Queue 管理ガイド』を参照してください。

メッセージフェイルオーバーのための JMS アプリケーションの配備および MQ クラスタ化の詳細は、35 ページの「Message Queue ブローカの配備の計画」を参照してください。

◆ ◆ ◆ 第 2 章

2 配備の計画

Application Server を配備する前に、まずパフォーマンスと可用性の目標を決定したあと、それに応じてハードウェア、ネットワーク、およびストレージの要件に関する決定を行います。

この章で説明する内容は次のとおりです。

パフォーマンス目標の確立

高パフォーマンスのもっとも単純な意味は、スループットを最大化し、応答時間を短縮することです。これらの基本的な目標を超えて特定の目標を確立するには、次の内容を決定します。

- 配備するアプリケーションやサービスの種類と、クライアントからのアクセス方法。
- 高可用性を必要とするアプリケーションやサービス。
- アプリケーションはセッション状態を持つか、または状態を持たないか。
- システムでサポートする必要のある要求の容量またはスループット。
- システムでサポートする必要のある並行ユーザーの数。
- ユーザー要求に対する許容可能な平均応答時間。
- 要求の間の平均思考時間。

これらのメトリックスの一部は、リモートブラウザエミュレータ (RBE) ツールか、または予測されるアプリケーションアクティビティーのシミュレーションを行う Web サイトのパフォーマンスおよびベンチマークソフトウェアを使用して計算できます。一般に、RBE やベンチマーク製品は並行 HTTP 要求を生成し、次に 1 分あたりの特定数の要求に対する応答時間を報告します。これらの数値を使用することによって、サーバーアクティビティーを計算できます。

この章で説明されている計算の結果は絶対ではありません。Application Server や独自のアプリケーションのパフォーマンスを微調整するときに照合するための参照点として扱ってください。

この節では、次の項目について説明します。

スループットの見積もり

広義では、スループットは Communications Server によって実行された作業の量を測定します。Communications Server の場合、スループットは、サーバーインスタンスあたり、1 分あたりに処理された要求の数として定義できます。

次の節で説明するように、Communications Server のスループットは、ユーザー要求の性質やサイズ、ユーザーの数、Communications Server インスタンスやバックエンドデータベースのパフォーマンスを含む、多くの要因から成る関数です。シミュレートされた作業負荷のベンチマークによって、1 台のマシンでのスループットを見積もることができます。

Application Server インスタンスへの負荷の見積もり

Communications Server インスタンスへの負荷を見積もるには、次の要因を考慮してください。

並行ユーザーの最大数

ユーザーは、Web ブラウザや Java プログラムなどのクライアントを介してアプリケーションと対話します。ユーザーのアクションに基づいて、クライアントは Communications Server に要求を定期的に送信します。ユーザーは、そのユーザーのセッションが期限切れでなく、終了されてもいないかぎり、アクティブと見なされます。並行ユーザーの数を見積もる場合は、すべてのアクティブユーザーを含めてください。

最初は、ユーザーの数が増えると、それに対応してスループットが増加します。ただし、並行要求の数が増えるにつれてサーバーパフォーマンスが飽和し始めるため、スループットは低下し始めます。

並行ユーザーを追加した場合に 1 分あたりに処理できる要求の数が減る点を特定します。この点は、最適なパフォーマンスに達しており、それを超えるとスループットが低下し始める時点を示します。一般には、システムをできるだけ最適なスループットで動作させるようにしてください。追加の負荷を処理し、スループットを向上させるには、処理能力の追加が必要になる場合があります。

思考時間

ユーザーが要求を連続して送信することはありません。ユーザーが要求を送信すると、サーバーがその要求を受信し、処理したあと、結果を返します。ユーザーはその時点で、新しい要求を送信する前に一定の時間を費やします。ある要求から次の要求までの時間を、思考時間と呼びます。

思考時間は、ユーザーの種類に依存します。たとえば、Web サービスの場合のようなマシン同士の対話では一般に、思考時間は人間同士の対話より短くなります。思考時間を見積もるために、マシンと人間の対話の混在を考慮することが必要な場合があります。

平均思考時間の特定は重要です。この所要時間を使用すると、1 分あたりに完了する必要のある要求の数や、システムでサポートできる並行ユーザーの数を計算できます。

平均応答時間

応答時間とは、Communications Server が、要求の結果をユーザーに返すために費やす時間のことを指します。応答時間は、ネットワーク帯域幅、ユーザーの数、送信される要求の数とタイプ、平均思考時間などの要因によって影響されます。

ここでは、応答時間は平均の応答時間を指します。要求のタイプごとに、独自の最小応答時間があります。ただし、システム性能を評価する場合は、すべての要求の平均応答時間に基づいて分析します。

応答時間が速ければ速いほど、1 分あたりに処理される要求が増えます。ただし、システム上のユーザーの数が増えるにつれ、1 分あたりの要求の数が低下するにもかかわらず応答時間も増え始めます。

この図のようなシステム性能のグラフは、特定の点を過ぎると、1 分あたりの要求数が応答時間に反比例することを示しています。点線の矢印で表されているように、1 分あたりの要求数の低下が激しければ激しいほど、応答時間の増加も急になります。

この図の場合、ピーク負荷の点は、1 分あたりの要求数が低下し始める時点になります。この点より前は、数式にピークの数値が使用されていないため、応答時間の計

算は必ずしも正確ではありません。この点よりあとは、1 分あたりの要求数と応答時間の間に反比例の関係があるため、管理者は、ユーザーの最大数と 1 分あたりの要求数を使用して応答時間をより正確に計算できます。

ピーク負荷時の応答時間 (秒単位) である $T_{response}$ を特定するには、次の数式を使用します。

$T_{response} = n/r - T_{think}$

次に、各引数について説明します。

- *n* は、並行ユーザーの数です。
- *r* は、サーバーが受信する 1 秒あたりの要求の数です。
- T_{think} は、平均思考時間 (秒単位) です。
  正確な応答時間結果を得るには、必ず式に思考時間を含めてください。

例 2–1 応答時間の計算

次の条件が存在する場合、

- ピーク負荷時にシステムでサポートできる並行ユーザーの最大数 (*n*) は 5,000。
- ピーク負荷時にシステムで処理できる要求の最大数 (*r*) は 1 秒あたり 1,000。

平均思考時間 (T_{think}) は 1 要求あたり 3 秒。

応答時間の計算は次のようになります。

$T_{response} = n/r - T_{think} = (5000/1000) - 3 秒 = 5 - 3 秒$

したがって、応答時間は 2 秒です。

システムの (特に、ピーク負荷時の) 応答時間を計算したら、それを、アプリケーションで許容可能な応答時間と比較します。応答時間は、スループットとともに、Application Server のパフォーマンスにとって重要な主要要因の 1 つです。

1 分あたりの要求数

任意の時点での並行ユーザーの数、その要求の応答時間、およびユーザーの平均思考時間がわかっている場合は、1 分あたりの要求数を計算できます。一般には、システム上に存在する並行ユーザーの数の見積もりから始めます。

たとえば、Web サイトパフォーマンスソフトウェアの実行後、管理者が、オンラインバンキング Web サイトで要求を送信する並行ユーザーの平均数を 3000 と結論付けます。この数は、オンラインバンクの会員登録を申し込んだユーザーの数、バンキングトランザクション動作、要求を送信する曜日または日時などに基づきます。

したがって、これらの情報がわかれば、この節で説明した 1 分あたりの要求数の数式を使用して、このユーザーベースに対してシステムが処理できる 1 分あたりの要求数を計算できます。ピーク負荷時には 1 分あたりの要求数と応答時間が反比例関

係になるため、より優れた応答時間を得るためのトレードオフとして 1 分あたりの要求数を減らすことが許容可能かどうか、あるいは、1 分あたりの要求数を増やすためのトレードオフとして応答時間を遅くすることが許容可能かどうかを判断してください。

システム性能を微調整するための開始点として許容可能な 1 分あたりの要求数と応答時間のしきい値で試してください。そのあと、システムのどの領域に調整が必要かを判断してください。

前の節の式で *r* を求めると、次のようになります。

$r = n/(T_{response} + T_{think})$

例 2–2　1 秒あたりの要求数の計算

次の値の場合、

- $n = 2{,}800$ の並行ユーザー
- $T_{response} = 1$ (1 要求あたりの平均応答時間は 1 秒)
- $T_{think} = 3$ (平均思考時間は 3 秒)

1 秒あたりの要求の数の計算は次のようになります。

```
r = 2800 / (1+3) = 700
```

したがって、1 秒あたりの要求の数は 700、1 分あたりの要求の数は 42000 です。

ネットワーク構成の計画

Communications Server をネットワークに統合する方法を計画する場合は、帯域幅の要件を見積もり、ユーザーのパフォーマンス要件を満たすような方法でネットワークを計画します。

ここで説明する内容は次のとおりです。

- 31 ページの「帯域幅の要件の見積もり」
- 32 ページの「必要な帯域幅の計算」
- 33 ページの「ピーク負荷の見積もり」
- 34 ページの「障害クラスの識別」

帯域幅の要件の見積もり

ネットワークの望ましいサイズと帯域幅を決定するには、まずネットワークトラフィックを特定し、そのピークを識別します。全体の量がピークになる特定の時間、曜日、または月の特定の日が存在するかどうかを確認したあと、そのピークの所要時間を特定します。

ピーク負荷の時間帯に、ネットワーク内のパケットの数はもっとも高いレベルに達します。一般に、ピーク負荷の設計を行う場合は、ピーク量の 100 パーセントを処理するという目標でシステムを拡張させます。ただし、どのようなネットワークでも予期しない動作をすること、またどれだけ拡張したとしても、必ずしもピーク量の 100 パーセントを処理できるとは限らないことを念頭においてください。

たとえば、ピーク負荷時に、ユーザーの 5 パーセントが Communications Server 上に配備されているアプリケーションへのアクセス中にネットワークに直接アクセスできない場合があるとします。その 5 パーセントのうち、最初の試行のあとにアクセスを再試行するユーザーがどれだけいるかを見積もってください。ここでも、それらのユーザーがすべて実行するとは限らず、その失敗したユーザーのうちの一定の割合が再試行します。その結果、ユーザーがアクセスを試行し続けるに伴って徐々にピーク使用が拡大するため、ピークが現れる時間は長くなります。

必要な帯域幅の計算

27 ページの「パフォーマンス目標の確立」で行なった計算に基づいて、サイトに Application Server を配備するために必要な追加の帯域幅を決定します。

アクセス方法 (T-1 回線、ADSL、ケーブルモデム、その他) に応じて、見積もった負荷を処理するために必要な増加した帯域幅の量を計算します。たとえば、サイトで T-1 回線または高速な T-3 回線を使用しているとします。回線の帯域幅がわかったら、サイトで 1 秒あたりに生成される要求の平均数および最大ピーク負荷に基づいて、ネットワークに必要な回線数を見積もります。Web サイトの分析および監視ツールを使用して、これらの数値を計算します。

例 2–3　必要な帯域幅の計算

1 本の T-1 回線は、1.544 Mbps を処理できます。したがって、T-1 回線 4 本から成るネットワークは約 6 Mbps のデータを処理できます。クライアントに送り返される平均的な HTML ページが 30K バイトであると仮定すると、T-1 回線 4 本から成るこのネットワークは 1 秒あたり次のトラフィックを処理できます。

6,176,000 ビット/8 ビット = 1 秒あたり 772,000 バイト

1 秒あたり 772,000 バイト/30K バイト = 1 秒あたり約 25 の並行応答ページ

1 秒あたり 25 ページのトラフィックとすると、このシステムは 1 時間あたり 90,000 ページ (25 × 60 秒× 60 分) を処理できるため、負荷が 1 日中均一であると仮定した場合、1 日あたり最大 2,160,000 ページになります。最大ピーク負荷がこれより高い場合は、それに応じて帯域幅を増やします。

ピーク負荷の見積もり

負荷を 1 日中均一にすることは、おそらく現実的ではありません。ピーク負荷がいつ発生し、どれだけ持続するか、および負荷全体の何パーセントがピーク負荷になるかを特定する必要があります。

例 2–4　ピーク負荷の計算

ピーク負荷が 2 時間持続し、2,160,000 ページの負荷全体の 30 パーセントに相当する場合は、1 日のうちの 2 時間の間に T-1 回線上で 648,000 ページを処理する必要があることを示します。

したがって、この 2 時間の間にピーク負荷に対応するには、T-1 回線の数を次の計算に従って増やします。

648,000 ページ/120 分 = 1 分あたり 5,400 ページ

1 分あたり 5,400 ページ/60 秒 = 1 秒あたり 90 ページ

4 回線で 1 秒あたり 25 ページを処理できるとすると、その約 4 倍のページにはその 4 倍の回線 (この場合は 16 回線) が必要になります。この 16 回線は、30 パーセントのピーク負荷という現実的な最大量の処理を考慮したものです。明らかに、これらの多数の回線を使用すれば、その他の 70 パーセントの負荷を 1 日の残り時間にわたって処理できます。

可用性のための計画

ここでは、次の内容について説明します。

- 33 ページの「可用性の規模の適正化」
- 34 ページの「可用性を向上させるためのクラスタの使用」
- 34 ページの「システムへの冗長性の追加」

可用性の規模の適正化

システムやアプリケーションの可用性を計画するには、異なるアプリケーションにアクセスするユーザーグループの可用性ニーズを評価します。たとえば、料金を支払う外部のユーザーやビジネスパートナーはたいてい、サービスの品質 (QoS) に内部のユーザーより高い期待を持っています。そのため、アプリケーション機能、アプリケーション、サーバーなどが使用不可になっても、支払いを行う外部の顧客に比べて内部のユーザーの方が許容性が高い可能性があります。

次の図は、発生確率が低い出来事ほど、影響を軽減するためのコストと複雑さが増加するようすを示しています。この連続曲線の一端では、単純な負荷分散クラスタ

を使用して、ローカライズされたアプリケーション、ミドルウェア、およびハードウェアの障害に耐えることができます。曲線のもう一方の端では、地理的に孤立したクラスタを使用することにより、データセンター全体に影響する大災害を軽減できます。

高い投資収益率を実現するには、多くの場合、アプリケーション内の機能の可用性の要件を特定することが有効です。たとえば、保険見積もりシステムが使用不可になることは許容されない (それによって新しい企業を失う) 可能性がありますが、既存の顧客が現在の対象範囲を表示できるアカウント管理機能が短期間使用不可になっても、既存の顧客を失うことはほとんどありません。

可用性を向上させるためのクラスタの使用

もっとも基本的なレベルで言えば、クラスタとは、クライアントには 1 つのインスタンスとして見えるアプリケーションサーバーインスタンスのグループであり、たいていは複数の物理サーバー上にホストされています。これにより、水平方向のスケーラビリティーや、1 台のマシン上の 1 つのインスタンスより高い可用性が提供されます。この基本的なレベルのクラスタ分布が融合ロードバランサと連携して動作します。このロードバランサは、HTTP/HTTPS および SIP/SIPS 要求を受け付け、それをクラスタ内のインスタンスの 1 つに転送します。また、ORB や、統合されている JMS ブローカも、アプリケーションサーバークラスタへの負荷分散を実行します。ネットワーク障害のためにインスタンスが失敗して使用不可になるか、または応答しなくなると、要求は既存の使用可能なマシンにのみリダイレクトされます。ロードバランサはまた、失敗したインスタンスが復旧したことを認識し、それに応じて負荷を再配分することもできます。

システムへの冗長性の追加

高可用性を実現するための 1 つの方法は、システムにハードウェアやソフトウェアの冗長性を追加することです。あるユニットに障害が発生すると、冗長なユニットが引き継ぎます。これは、耐障害性とも呼ばれます。一般に、高可用性を最大化するには、システム内に存在する可能性のあるすべてのシングルポイント障害を特定して取り除きます。

障害クラスの識別

冗長性のレベルは、システムが耐える必要のある障害クラス (障害の種類) によって決定されます。障害クラスのいくつかの例を次に示します。

- システムプロセス
- マシン
- 電源装置
- ディスク

- ネットワーク障害
- ビル火災またはその他の予防可能な災害
- 予測できない天災

重複したシステムプロセスによって、単一のシステムプロセス障害や単一のマシン障害に耐えることができます。重複した、ミラー化された (ペアになった) マシンを異なる電源装置に接続することにより、単一の電源障害に耐えることができます。ミラー化されたマシンを個別のビル内に保持することにより、単一のビル火災に耐えることができます。ミラー化されたマシンを地理的に離れた場所に保持することにより、地震などの天災に耐えることができます。

Message Queue ブローカの配備の計画

Java Message Service (JMS) API は、Java EE アプリケーションおよびコンポーネントに対して、メッセージの作成、送信、受信、および読み取りを可能にするメッセージング標準です。この API によって、緩やかに結合され、信頼性が高く、非同期の分散通信が可能となります。Sun Java System Message Queue は JMS を実装し、Communications Server と統合されているため、MQ を使用してメッセージ駆動型 Beans (MDB) などのコンポーネントを作成できます。

Sun Java System Message Queue (MQ) は、Java EE コネクタアーキテクチャー仕様 (JCA) 1.5 で規定されているように、コネクタモジュール (リソースアダプタともいう) を使用して Communications Server に統合されています。コネクタモジュールは、Communications Server に機能を追加するための標準化された方法です。Communications Server に配備された Java EE コンポーネントは、コネクタモジュールによって統合された JMS プロバイダを使用して JMS メッセージを交換します。この JMS プロバイダは、デフォルトでは Sun Java System Message Queue ですが、JCA 1.5 が実装されているかぎり、必要に応じて別の JMS プロバイダを使用できます。

Communications Server で JMS リソースを作成すると、バックグラウンドでコネクタリソースが作成されます。そのようにして、JMS 操作のたびにコネクタランタイムが呼び出され、バックグラウンドで MQ リソースアダプタが使用されます。

リソースアダプタ API の使用に加えて、Communications Server は、MQ とのより緊密な統合を実現するために MQ API を使用します。この緊密な統合によって、コネクタのフェイルオーバー、アウトバウンド接続の負荷分散、MDB へのインバウンドメッセージの負荷分散などの機能が可能になります。これらの機能を使用すると、メッセージングトラフィックの耐障害性や高可用性を実現できます。

マルチブローカクラスタ

MQ は、ブローカクラスタと呼ばれる、複数の相互接続されたブローカインスタンスの使用をサポートしています。ブローカクラスタによって、クライアント接続は

クラスタ内のすべてのブローカに分散されます。クラスタ化することで、水平方向のスケーラビリティーが提供され、可用性が向上します。

1 つのメッセージブローカは約 8 個の CPU まで拡大縮小し、標準的なアプリケーションのための十分なスループットを提供します。ブローカプロセスが異常終了した場合、そのプロセスは自動的に再起動されます。ただし、ブローカーに接続するクライアントの数や配信されるメッセージの数が増えると、ブローカーは最終的に、ファイル記述子の数やメモリーなどの制限を超えてしまいます。

クラスタ内に 1 つのブローカではなく複数のブローカを配置すると、次のことが可能になります。

- 1 台のマシンでハードウェア障害が発生した場合でもメッセージングサービスを提供する。
- システム保守を実行している間の停止時間を最小限に抑える。
- 異なるユーザーリポジトリを持つワークグループを格納する。
- ファイアウォールの制限に対応する。

ただし、複数のブローカを配置しても、ブローカ障害の時点で進行中であったトランザクションが代替ブローカに引き継がれることは保証されません。MQ は、失敗した接続をクラスタ内の別のブローカを使用して再確立しますが、トランザクションメッセージングを失い、進行中のトランザクションをロールバックします。完了できなかったトランザクションを除き、ユーザーアプリケーションは影響されません。接続は引き続き使用可能であるため、サービスのフェイルオーバーは保証されます。

そのため、MQ はクラスタ内の高可用性持続メッセージをサポートしていません。障害のあとにブローカを再起動すると、ブローカは自動的に回復し、持続メッセージの配信を完了します。持続メッセージはデータベース内か、またはファイルシステムに格納される可能性があります。ただし、ブローカをホストしているマシンがハード障害から回復しない場合は、メッセージが失われる可能性があります。

Sun Cluster Data Service for Sun Message Queue を備えた Solaris プラットフォームは、持続メッセージの透過的なフェイルオーバーをサポートしています。この構成は、真の高可用性を実現するために Sun Cluster のグローバルファイルシステムと IP フェイルオーバーを利用しており、また Java Enterprise System にも含まれています。

マスターブローカとクライアントの同期

マルチブローカ構成では、各送信先がクラスタ内のすべてのブローカにレプリケートされます。各ブローカは、ほかのすべてのブローカ上の送信先として登録されているメッセージコンシューマを認識しています。そのため、各ブローカは、自身に直接接続されているメッセージプロデューサから遠隔メッセージコンシューマにメッセージを経路指定したり、遠隔プロデューサから自身に直接接続されているコンシューマにメッセージを配信したりすることができます。

クラスタ構成では、各メッセージプロデューサが直接接続されているブローカが、そのプロデューサから送信されるメッセージの経路指定を実行します。そのため、持続メッセージの格納と経路指定の両方を、そのメッセージのホームブローカが実行します。

管理者がブローカ上の送信先を作成または破棄した場合は常に、この情報がクラスタ内のほかのすべてのブローカに自動的に伝播されます。同様に、メッセージコンシューマがホームブローカに登録された場合、またはコンシューマがホームブローカから切り離された (明示的か、クライアントまたはネットワークの障害のためか、ホームブローカがダウンしたためかにかかわらず) 場合は常に、そのコンシューマに関連する情報がクラスタ全体にわたって伝播されます。同様の方法で、持続性サブスクリプションに関する情報もクラスタ内のすべてのブローカに伝播されます。

Message Queue ブローカを使用するための Application Server の設定

Communications Server の Java Message Service は、Message Queue のコネクタモジュール (リソースアダプタ) を表します。Java Message Service は、管理コンソールまたは `asadmin` コマンド行ユーティリティーから管理することができます。

MQ ブローカ (JMS ホスト) は、Communications Server プロセスとは別の JVM で動作します。これにより、複数の Communications Server インスタンスまたはクラスタが MQ ブローカの同じセットを共有できます。

Communications Server では、*JMS* ホストは MQ ブローカを指します。Communications Server の Java Message Service 設定には、使用されるすべての JMS ホストを含む JMS ホストリスト (AddressList とも呼ばれる) が含まれています。

管理コンソールを使用した JMS の管理

管理コンソールでは、特定の構成の「Java メッセージサービス」ノードを使用して JMS プロパティーを設定できます。「再接続間隔」や「再接続試行」などのプロパティーを設定できます。詳細は、『Sun GlassFish Communications Server 1.5 管理ガイド』の第 4 章「Java Message Service (JMS) リソースの設定」を参照してください。

「Java メッセージサービス」ノードの下の「JMS ホスト」ノードには、JMS ホストのリストが含まれています。リストにホストを追加することも、リストからホストを削除することもできます。ホストごとに、ホスト名、ポート番号、および管理ユーザーの名前とパスワードを設定できます。JMS ホストリストには、デフォルトで、Communications Server に統合されたローカルの MQ ブローカを表す、"default_JMS_host" という 1 つの MQ ブローカが含まれています。

JMS ホストリストを、クラスタ内のすべての MQ ブローカを含むように設定します。たとえば、3 つの MQ ブローカを含むクラスタを設定するには、それぞれに対して Java Message Service 内に 1 つの JMS ホストを追加します。Message Queue クライアントは、Java Message Service 内の設定情報を使用して MQ ブローカと通信します。

asadmin を使用した JMS の管理

管理コンソールに加えて、`asadmin` コマンド行ユーティリティーでも Java Message Service と JMS ホストを管理できます。次の `asadmin` コマンドを使用します。

- Java Message Service 属性の設定: `asadmin set`
- JMS ホストの管理:
  - `asadmin create-jms-host`
  - `asadmin delete-jms-host`
  - `asadmin list-jms-hosts`
- JMS リソースの管理:
  - `asadmin create-jms-resource`
  - `asadmin delete-jms-resource`
  - `asadmin list-jms-resources`

    これらのコマンドの詳細は、『Sun GlassFish Communications Server 1.5 Reference Manual』または対応するマニュアルページを参照してください。

Java Message Service タイプ

Communications Server と MQ ブローカの間の統合には、ローカルとリモートの 2 つのタイプがあります。このタイプ属性は、管理コンソールの「Java メッセージサービス」ページで設定できます。

ローカルの Java Message Service

「タイプ」属性が LOCAL の場合は、Communications Server が MQ ブローカを起動および停止します。Communications Server は起動時に、デフォルト JMS ホストとして指定されている MQ ブローカを起動します。同様に、Communications Server インスタンスは停止時に、MQ ブローカを停止します。LOCAL タイプはスタンドアロンの Communications Server インスタンスに最適です。

LOCAL タイプでは、「起動引数」属性を使用して MQ ブローカの起動パラメータを指定します。

リモートの Java Message Service

「タイプ」属性が REMOTE の場合、Communications Server は、外部に設定されているブローカまたはブローカクラスタを使用します。この場合、MQ ブローカの起動

と停止は Communications Server とは別個に行い、MQ ツールを使用してブローカまたはブローカクラスタを設定および調整する必要があります。REMOTE タイプは Communications Server クラスタに最適です。

REMOTE タイプでは、MQ ツールを使用して MQ ブローカ起動パラメータを指定する必要があります。「起動引数」属性は無視されます。

デフォルト JMS ホスト

デフォルト JMS ホストは、管理コンソールの「Java メッセージサービス」ページで指定できます。Java Message Service タイプが LOCAL の場合、Communications Server は、Communications Server インスタンスが起動したときにデフォルト JMS ホストを起動します。

MQ ブローカクラスタを使用するには、デフォルト JMS ホストを削除したあと、クラスタ内のすべての MQ ブローカを JMS ホストとして追加します。この場合、デフォルト JMS ホストは JMS ホストリスト内の最初の JMS ホストになります。

また、デフォルト JMS ホストを、いずれかの JMS ホストに明示的に設定することもできます。Communications Server が Message Queue クラスタを使用する場合、デフォルト JMS ホストは MQ 固有のコマンドを実行します。たとえば、MQ ブローカクラスタの物理送信先が作成される場合、デフォルト JMS ホストはその物理送信先を作成するためにコマンドを実行しますが、クラスタ内のすべてのブローカがその物理送信先を使用します。

配備シナリオの例

メッセージングのニーズに対応するには、Java Message Service と JMS ホストリストを、配備、パフォーマンス、および可用性のニーズを満たすように変更します。以降の節では、いくつかの標準的なシナリオについて説明します。

メッセージングのニーズの考慮対象が Communications Server のみでない場合は、最高の可用性を得られるように、MQ ブローカと Communications Server を別のマシンに配備します。別のオプションとして、十分なメッセージング容量が存在するようになるまで、Communications Server インスタンスと MQ ブローカインスタンスを各マシンで実行する方法があります。

デフォルト配備

Communications Server をインストールすると、ドメイン管理サーバー (DAS) が自動的に作成されます。デフォルトでは、DAS の Java Message Service タイプは LOCAL です。そのため、DAS を起動すると、そのデフォルト MQ ブローカも起動されます。

新しいドメインを作成すると、新しいブローカも作成されます。デフォルトでは、ドメインにスタンドアロンのサーバーインスタンスまたはクラスタを追加すると、その Java Message Service は REMOTE として設定され、デフォルト JMS ホストは DAS によって起動されるブローカになります。

39 ページの「デフォルト配備」は、3 つのインスタンスを含む Communications Server クラスタが含まれたデフォルト配備の例を示しています。

Application Server クラスタでの MQ ブローカクラスタの使用

MQ ブローカクラスタを使用するように Communications Server クラスタを設定するには、Communications Server の Java Message Service で、すべての MQ ブローカを JMS ホストとして追加します。それにより、作成された JMS 接続ファクトリや、配備された MDB はすべて、指定された JMS 設定を使用するようになります。

次の図は、ブローカクラスタ内に 3 つの MQ ブローカが含まれ、クラスタ内に 3 つの Communications Server インスタンスが含まれる配備の例を示しています。

アプリケーション固有の MQ ブローカクラスタの指定

場合によっては、アプリケーションが、Communications Server クラスタで使用されているものとは別の MQ ブローカクラスタを使用しなければならないことがあります。40 ページの「アプリケーション固有の MQ ブローカクラスタの指定」は、このようなシナリオの例を示しています。これを行うには、JMS 接続ファクトリの `AddressList` プロパティー、または MDB 配備記述子内の `activation-config` 要素を使用して MQ ブローカクラスタを指定します。

接続ファクトリの設定の詳細については、『Sun GlassFish Communications Server 1.5 管理ガイド』の「JMS 接続ファクトリ」を参照してください。MDB の詳細については、『Sun GlassFish Communications Server 1.5 Developer's Guide』の「Using Message-Driven Beans」を参照してください。

アプリケーションクライアント

アプリケーションクライアントまたはスタンドアロンのアプリケーションが、JMS 管理によるオブジェクトにはじめてアクセスすると、クライアント JVM はサーバーから Java Message Service 設定を取得します。JMS サービスへのそれ以上の変更は、JMS サービスが再開されるまで、クライアント JVM には使用できません。

◆ ◆ ◆ 第 3 章

3 配備のためのチェックリスト

この付録では、Communications Server を使用した評価および本番運用を始めるためのチェックリストを示します。

配備のためのチェックリスト

表 3–1 チェックリスト

コンポーネント/機能	説明
アプリケーション	配備するアプリケーションについて、次の要件を決定します。 ■ 必要または許容可能な応答時間。 ■ ピーク負荷特性。 ■ 必要な持続性の範囲と頻度。 ■ `web.xml` でのセッションタイムアウト設定。 ■ フェイルオーバーおよび可用性の要件。 詳細は、『Sun GlassFish Communications Server 1.5 Performance Tuning Guide』を参照してください。
ハードウェア	■ 必要なハードディスク領域およびメモリー容量が備わっています。 ■ サイジングの演習を使用して、配備の要件を識別します。 詳細は、『Sun GlassFish Communications Server 1.5 リリースノート』を参照してください。

表 3–1 チェックリスト (続き)

コンポーネント/機能	説明
オペレーティングシステム	■ 製品がサポート対象のプラットフォームにインストールされていることを確認します。 ■ パッチレベルが最新かつ適切であることを確認します。詳細は、『Sun GlassFish Communications Server 1.5 リリースノート』を参照してください。
ネットワークインフラストラクチャー	■ シングルポイント障害を識別して対処します。 ■ NIC およびその他のネットワークコンポーネントが正しく設定されていることを確認します。 ■ `ttcp` ベンチマークテストを実行し、スループットが要件または期待値を満たしているかどうかを調べます。 ■ HADB ノードが正しくインストールされるように、必要に応じて `rsh/ssh` を設定します。詳細は、『Sun GlassFish Communications Server 1.5 Installation Guide』を参照してください。
バックエンドおよびその他の外部データソース	その分野の専門家またはベンダーの協力を得て、これらのデータソースが適切に設定されていることを確認します。
システムの変更/設定	■ パフォーマンステストまたはストレステストを実行する前に、`/etc/system` および Linux でこれに相当するファイルの変更が完了していることを確認します。 ■ TCP/IP 設定の変更が完了していることを確認します。 ■ システムのデフォルトでは、多くのサービスが事前に設定されています。これらすべてのサービスが実行のために必要とは限りません。システムリソースを節約するために、必要でないサービスは無効にします。 ■ Solaris では、`Setoolkit` を使用してシステムの動作を調べます。出現したフラグをすべて解決します。詳細は、『Sun GlassFish Communications Server 1.5 Performance Tuning Guide』を参照してください。
Installation	■ NFS マウントされたボリュームにこれらのサーバーをインストールしていないことを確認します。

表 3–1 チェックリスト (続き)

コンポーネント/機能	説明
Communications Server の設定	■ ログ: アクセスログのローテーションを有効にします。 ■ 適切なログレベルを選択します。通常は WARNING が適切です。 ■ 管理コンソールを使用して Java EE コンテナを設定します。 ■ 管理コンソールを使用して HTTP リスナーを設定します。 ■ 管理コンソールを使用して SIP リスナーを設定します。 ■ 管理コンソールを使用して ORB スレッドプールを設定します。 ■ ネイティブコードを必要とする Type2 ドライバまたは呼び出しを使用している場合、LD_LIBRARY_PATH に mtmalloc.so が指定されていることを確認します。 ■ 適切な持続性の範囲および頻度を使用しており、これらが個別の Web/EJB モジュールでオーバーライドされないことを確認します。 ■ SFSB の重要なメソッドのみがチェックポイント設定されることを確認します。チューニングの詳細は、『Sun GlassFish Communications Server 1.5 Performance Tuning Guide』を参照してください。 設定の詳細は、『Sun GlassFish Communications Server 1.5 管理ガイド』を参照してください。
融合ロードバランサの設定	■ 自動適用オプションを適切に設定していることを確認します。 ■ ロードバランサの設定ファイルに適切な値があることを確認します。
Java 仮想マシンの設定	■ 最初は、最小および最大のヒープサイズを同じ値に設定し、各インスタンスには 1G バイト以上を設定します。 ■ 詳細は、 Java Hotspot VM Options を参照してください。 ■ Communications Server の複数のインスタンスを実行する場合、プロセッサセットを作成して、Communications Server をそれに結合することを検討します。これは、古い世代のスイープに CMS コレクタが使用される場合に効果的です。
Communications Server のタイムアウト設定;	■ Max-wait-time-millis - プールからの接続取得を待機し、超過した場合に例外をスローする時間。デフォルトは 6s です。持続されるデータのサイズが 50K バイトを超える高負荷システムでは、この値の変更を検討します。 ■ Cache-idle-timeout-in-seconds - 非活性化されるまでに EJB がキャッシュ内でアイドル状態でいられる時間。エンティティー Bean およびステートフルセッション Bean のみに適用されます。 ■ Removal-timeout-in-seconds - EJB が非活性化状態 (バックアップストア内でアイドル状態) にとどまる時間。デフォルト値は 60 分です。この値は、SFSB フェイルオーバーの必要性に基づいて調整します。

表 3–1 チェックリスト *(続き)*

コンポーネント/機能	説明
VM ガベージコレクション **(GC)** のチューニング	ガベージコレクションを 4 秒以上一時停止すると、断続的に問題が発生することがあります。この問題を避けるには、VM ヒープを調整します。データ持続の失敗を 1 回も許容できない場合や、システムが高負荷状態でないときは、CMS コレクタまたはスループットコレクタを使用します。 これらのコレクタを有効にするには、次のオプションを追加します。 `<jvm-options>-XX:+UseConcMarkSweepGC</jvm-options>` このオプションはスループットを低下させる場合があります。