Sun Java System Application Server 9.1 部署规划指南

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

文件号码 820–4903

080429@19860

目录

图

表

示例

前言

部署规划指南说明如何构建生产部署。

本前言包含有关整个 Sun Java™ System Application Server 文档集的信息及其约定。

Application Server 文档集

Application Server 文档集介绍了部署规划和系统安装。Application Server 文档的统一资源定位器 (Uniform Resource Locator, URL) 为 `http://docs.sun.com/coll/1343.4`。 有关 Application Server 的简介，请按下表中列出的顺序参阅这些书籍。

表 P–1 Application Server 文档集中的书籍

书名	说明
Documentation Center	按照任务和主题进行组织的 Application Server 文档主题。
发行说明	软件和文档的最新信息。其中包括有关支持硬件、操作系统、Java Development Kit (JDK™) 以及数据库驱动程序的表式综合汇总。
Quick Start Guide	如何开始使用 Application Server 产品。
Installation Guide	安装软件及其组件。
部署规划指南	评估系统需求和企业状况，确保以最适合您的站点的方式部署 Application Server。此外还介绍了部署服务器时应该注意的常见问题。
Application Deployment Guide	Application Server 应用程序和应用程序组件部署。包含有关部署描述符的信息。
Developer's Guide	创建和实现要在 Application Server 上运行的 Java Platform, Enterprise Edition（Java EE 平台）应用程序，这些应用程序遵循针对 Java EE 组件和 API 的开放式 Java 标准模型。其中包括有关开发者工具、安全性、调试和创建生命周期模块的信息。
Java EE 5 Tutorial	使用 Java EE 5 平台技术和 API 开发 Java EE 应用程序。
Java WSIT Tutorial	使用 Web 服务互操作性技术 (Web Service Interoperability Technology, WSIT) 开发 Web 应用程序。介绍如何、何时以及为何使用 WSIT 技术以及每项技术所支持的功能和选项。
管理指南	Application Server 系统管理，包括配置、监视、安全、资源管理以及 Web 服务管理。

表 P–1 Application Server 文档集中的书籍 （续）

书名	说明
高可用性管理指南	有关高可用性数据库的安装后配置和管理说明。
Administration Reference	编辑 Application Server 配置文件 `domain.xml`。
Upgrade and Migration Guide	从 Application Server 的较早版本升级或从竞争性应用服务器迁移 Java EE 应用程序。该指南还介绍了相邻产品发行版之间的差异以及可导致与产品规范不兼容的配置选项。
Performance Tuning Guide	调优 Application Server 以提高性能。
Troubleshooting Guide	解决 Application Server 问题。
Error Message Reference	解析 Application Server 错误消息。
Reference Manual	可用于 Application Server 的实用程序命令，以手册页样式编写。其中包括 `asadmin` 命令行界面。

相关文档

可以单独购买 Application Server，也可以将其作为 Sun Java Enterprise System (Java ES) 的一个组件购买，Java ES 是一种软件基础结构，它支持分布在网络或 Internet 环境中的企业应用程序。如果将 Application Server 作为 Java ES 的一个组件购买，您应熟悉 `http://docs.sun.com/coll/1286.2` 中的系统文档。有关 Java ES 及其组件的所有文档的 URL 为 `http://docs.sun.com/prod/entsys.5`。

有关其他独立 Sun Java System 服务器产品的文档，请参阅以下资源：

- Message Queue 文档 (`http://docs.sun.com/coll/1343.4`)
- (`http://docs.sun.com/coll/1224.1`) 和 Directory Server 文档 (`http://docs.sun.com/coll/1606.1`)
- (`http://docs.sun.com/coll/1308.3`) 和 Web 服务器 文档 (`http://docs.sun.com/coll/1395.2`)

随 Application Server 提供的软件包的 Javadoc 工具参考位于 `http://glassfish.dev.java.net/nonav/javaee5/api/index.html` 中。此外，以下资源可能会有用：

- Java EE 5 规范 (`http://java.sun.com/javaee/5/javatech.html`)
- Java EE 5 教程 (`http://java.sun.com/javaee/5/docs/tutorial/doc/index.html`)
- Java EE Blueprints (`http://java.sun.com/reference/blueprints/index.html`)

默认路径和文件名

下表介绍了本书中使用的默认路径和文件名。

表 P–2 默认路径和文件名

占位符	说明	默认值

install-dir	表示 Application Server 的基安装目录。	Solaris™ 操作系统上的 Java ES 安装：
		`/opt/SUNWappserver/appserver`
		Linux 操作系统上的 Java ES 安装：
		`/opt/sun/appserver/`
		其他 Solaris 和 Linux 的安装（非超级用户）：
		user's-home-directory`/SUNWappserver`
		其他 Solaris 和 Linux 的安装（超级用户）：
		`/opt/SUNWappserver`
		Windows 的所有安装：
		SystemDrive`:\Sun\AppServer`

domain-root-dir	表示包含所有域的目录。	Java ES Solaris 安装：
		`/var/opt/SUNWappserver/domains/`
		Java ES Linux 安装：
		`/var/opt/sun/appserver/domains/`
		所有其他安装：
		install-dir`/domains/`

domain-dir	表示某个域的目录。 在配置文件中，您可能会看到 *domain-dir* 显示为以下内容： `${com.sun.aas.instanceRoot}`	*domain-root-dir*`/`*domain-dir*
instance-dir	表示某个服务器实例的目录。	*domain-dir*`/`*instance-dir*

印刷约定

下表介绍了本书中使用的印刷约定更改。

表 P–3 印刷约定

字样	含义	示例
`AaBbCc123`	命令、文件和目录的名称；计算机屏幕输出	编辑 `.login` 文件。 使用 `ls -a` 列出所有文件。 `machine_name% you have mail.`
`AaBbCc123`	用户键入的内容，与计算机屏幕输出的显示不同。	`machine_name%` **`su`** `Password:`
AaBbCc123	要使用实名或值替换的占位符	删除文件的命令为 `rm` *filename*。
AaBbCc123	书名、新词以及要强调的词（注：在联机状态下，有些需要强调的词以黑体显示）	阅读用户指南中的第 6 章。 **高速缓存**是存储在本地的副本。 请**勿**保存该文件。

符号约定

下表介绍了本书中可能使用的符号。

表 P–4 符号约定

符号	说明	示例	含义
`[ ]`	包含可选变量和命令选项。	`ls [-l]`	`-l` 选项不是必需的。
`{ | }`	包含为所需命令选项提供的一组选择。	`-d {y|n}`	`-d` 选项要求使用 `y` 参数或 `n` 参数。
`${ }`	表示一个变量引用。	`${com.sun.javaRoot}`	引用 `com.sun.javaRoot` 变量的值。
-	连接需同时按下的多个击键。	Control-A	同时按 Control 键和 A 键。
+	连接需连续按下的多个击键。	Ctrl+A+N	按 Control 键，然后松开并依次按后面的键。

表 P–4 符号约定 （续）

符号	说明	示例	含义
→	表示图形用户界面中的菜单项选定。	“文件” → “新建” → “模板”	从“文件”菜单中，选择“新建”。从“新建”子菜单中，选择“模板”。

文档、支持和培训

Sun web 站点提供有关以下附加资源的信息：

- 文档 (http://www.sun.com/documentation/)
- 支持 (http://www.sun.com/support/)
- 培训 (http://www.sun.com/training/)

搜索 Sun 产品文档

除了从 docs.sun.com℠ Web 站点中搜索 Sun 产品文档外，还可以通过在搜索字段中键入以下语法来使用搜索引擎进行搜索：

search-term `site:docs.sun.com`

例如，要搜索 "broker"，请键入以下内容：

```
broker site:docs.sun.com
```

要在搜索中包括其他 Sun Web 站点（例如，java.sun.com、www.sun.com 和 developers.sun.com），请使用 `sun.com` 替代搜索字段中的 `docs.sun.com`。

第三方 Web 站点引用

本文档引用了第三方 URL 以提供其他相关信息。

注 – Sun 对本文档中提到的第三方 Web 站点的可用性不承担任何责任。对于此类站点或资源中的（或通过它们获得的）任何内容、广告、产品或其他资料，Sun 并不表示认可，也不承担任何责任。对于因使用或依靠此类站点或资源中的（或通过它们获得的）任何内容、产品或服务而造成的、宣称的或连带产生的实际或名义损坏或损失，Sun 概不负责，也不承担任何责任。

Sun 欢迎您提出意见

Sun 致力于提高其文档的质量，并十分乐意收到您的意见和建议。为了共享您的意见，请访问 http://docs.sun.com，并单击 "Send Comments"（发送意见）。在联机表单中，提供完整的文档标题和文件号码。文件号码是一个七位或九位的数字，可以在书的标题页或文档的 URL 中找到。例如，本书的文件号码为 820-4903。

◆ ◆ ◆ 第 1 章

产品概念

Sun Java System Application Server 为 J2EE 应用程序开发、部署和管理提供了一个强健的平台。其主要功能包括可伸缩的事务管理、容器管理的持久性运行时、Web 服务性能、群集、高可用性会话状态、安全性和集成功能。

本节包含以下主题：

J2EE 平台概述

Application Server 实现了 Java 2 Enterprise Edition (J2EE) 1.4 技术。J2EE 平台是一组标准规范，它们描述了应用服务器的应用程序组件、API 以及运行时容器和服务。

J2EE 应用程序

J2EE 应用程序由一些组件组成，例如 JavaServer Pages (JSP)、Java Servlet 和 Enterprise JavaBeans (EJB) 模块。软件开发者可以通过这些组件来构建大型分布式应用程序。开发者将 J2EE 应用程序打包在 Java 归档 (Java Archive, JAR) 文件（类似于 zip 文件）中，这些文件可以分发到生产站点。管理员通过将 J2EE JAR 文件部署到一个或多个 Application Server 实例（或实例群集）中，在该服务器上安装 J2EE 应用程序。

下图展示了在以下各节中介绍的 J2EE 平台组件。

抱歉：图形当前不可用。

容器

每个服务器实例包含两个容器：Web 和 EJB。容器是一种运行时环境，它为 J2EE 组件提供服务（如安全性和事务管理）。Web 组件（如 Java Server Pages 和 Servlet）在 Web 容器内运行。Enterprise JavaBeans 在 EJB 容器内运行。

J2EE 服务

J2EE 平台为应用程序提供了一些服务，其中包括：

- **命名** - 命名和目录服务将对象绑定到名称。J2EE 应用程序可通过查找 Java Naming and Directory Interface (JNDI) 名称来找到对象。
- **安全性** - Java 容器授权约定 (Java Authorization Contract for Containers, JACC) 是为 J2EE 容器定义的一组安全约定。根据客户机的标识，容器可以限制对容器资源和服务的访问。
- **事务管理** - 事务是不可分的工作单元。例如，在银行帐户之间转帐就是一个事务。事务管理服务可确保完成事务或回滚事务。
- 消息服务 - 通过使用 Java™ 消息服务 (Java™ Message Service, JMS) 交换消息，位于不同系统上的应用程序可以彼此之间进行通信。JMS 是 J2EE 平台不可或缺的组成部分，它可以简化集成异构企业应用程序的任务。

Web 服务

除了通过 HTTP、RMI/IIOP 和 JMS 访问 J2EE 1.4 应用程序外，客户机还可以将其作为远程 Web 服务进行访问。Web 服务是使用 Java API for XML-based RPC (JAX-RPC) 实现的。J2EE 应用程序还可以作为 Web 服务的客户机，这在网络应用程序中是很常见的。

Web 服务描述语言 (Web Services Description Language, WSDL) 是一种描述 Web 服务接口的 XML 格式。Web 服务使用者可以动态地解析 WSDL 文档以确定 Web 服务提供的操作以及如何执行这些操作。Application Server 使用注册表分发 Web 服务接口描述，其他应用程序可通过 Java API for XML Registries (JAXR) 访问该注册表。

客户机访问

客户机可使用多种方法来访问 J2EE 应用程序。浏览器客户机使用超文本传输协议 (Hypertext Transfer Protocol, HTTP) 访问 Web 应用程序。对于安全通信，浏览器使用的是采用安全套接字层 (Secure Sockets Layer, SSL) 的 HTTP 安全 (HTTPS) 协议。

应用程序客户机容器中运行的丰富客户机应用程序可以使用对象请求代理 (Object Request Broker, ORB)、远程方法调用 (Remote Method Invocation, RMI) 和 Internet 对象请求代理间协议 (Internet Inter-ORB Protocol, IIOP) 或 IIOP/SSL（安全 IIOP）直接查找和访问 Enterprise JavaBeans；使用 HTTP/HTTPS、JMS 和 JAX-RPC 访问应用程序和 Web 服务；以及使用 JMS 向应用程序和消息驱动 Bean 发送消息并从中接收消息。

符合 Web 服务互操作性 (Web Services-Interoperability, WS-I) 基本配置文件的客户机可以访问 J2EE Web 服务。WS-I 是 J2EE 标准不可或缺的组成部分，它定义了可互操作的 Web 服务。用任何支持语言编写的客户机都可通过它访问 Application Server 上部署的 Web 服务。

最佳的访问机制取决于特定应用程序和预期通信量。Application Server 支持可为 HTTP、HTTPS、JMS、IIOP 和 IIOP/SSL 单独配置的侦听器。您可以为每种协议设置多个侦听器，以便提高可伸缩性和可靠性。

J2EE 应用程序还可以用作 J2EE 组件（如其他服务器上部署的 Enterprise JavaBeans 模块）的客户机，并且可以使用其中的任何访问机制。

外部系统和资源

在 J2EE 平台上，外部系统称为**资源**。例如，数据库管理系统就是一种 JDBC 资源。每种资源是按其 Java Naming and Directory Interface (JNDI) 名称进行唯一标识的。应用程序通过以下 API 和组件来访问外部系统：

- *Java Database Connectivity (JDBC)* - 数据库管理系统 (Database Management System, DBMS) 提供了一些用于存储、组织和检索数据的工具。大多数商业应用程序将数据存储在关系数据库中，这些应用程序通过 JDBC 访问关系数据库。Application Server 包含 PointBase DBMS，用于提供样例应用程序、进行应用程序开发和建立原型，但并不适于进行部署。Application Server 提供认证的 JDBC 驱动程序以连接到主要关系数据库。这些驱动程序适于进行部署。
- *Java* **消息**服务 - 消息传送是一种在软件组件或应用程序之间进行通信的方法。通过使用实现 Java 消息传送服务 (Java Messaging Service, JMS) API 的消息传送提供者，消息传送客户机可以向任何其他客户机发送消息并从中接收消息。Application Server 包含一个高性能的 JMS 代理，即 Sun Java System Message Queue。Application Server Platform Edition 包含免费的 Message Queue Platform Edition。Application Server Enterprise Edition 包含支持群集和故障转移的 Message Queue Enterprise Edition。
- **J2EE 连接器** - J2EE 连接器体系结构允许将 J2EE 应用程序与现有的企业信息系统 (Enterprise Information System, EIS) 集成在一起。应用程序通过称为**连接器**或**资源适配器**的可移植 J2EE 组件访问 EIS，这类似于使用 JDBC 驱动程序访问 RDBMS。资源适配器作为独立的资源适配器归档 (Resource Adapter Archive, RAR) 模块进行分发，或者包含在 J2EE 应用程序归档中。与其他 J2EE 组件一样，它们是作为 RAR 进行部署的。Application Server 包含与常见 EIS 集成在一起的测试版资源适配器。
- **JavaMail** - 通过 JavaMail API，应用程序可以连接到简单邮件传输协议 (Simple Mail Transport Protocol, SMTP) 服务器以发送和接收电子邮件。

Application Server 组件

本节介绍了 Sun Java System Application Server 中的组件：

下图展示了这些 Application Server 组件如何使用提供高可用性的简单示例拓扑进行交互。在此示例拓扑中，一个管理员管理两台组成群集的计算机。HADB 和 Application Server 进程位于同一台计算机上。域管理服务器本身可能位于单独的计算机上，也可能位于承载应用服务器实例的任一计算机上。图表中的线表示通信或控制。

管理工具（如基于浏览器的管理控制台）与域管理服务器 (Domain Administration Server, DAS) 进行通信，后者又与节点代理和服务器实例进行通信。

服务器实例

服务器实例是在单个 Java 虚拟机 (Java Virtual Machine, JVM) 进程中运行的 Application Server。Application Server 已通过 Java 2 Standard Edition (J2SE) 5.0 和 1.4 认证。Application Server 安装中包含建议的 J2SE 分发包。

通常，在计算机上创建单个服务器实例就足够了，因为 Application Server 和附带的 JVM 均可扩展到多个处理器。但是，如果在一台计算机上创建多个实例，则将非常有益于进行应用程序隔离和滚动升级。在某些情况下，可以在多个管理域中使用具有多个实例的大型服务器。管理工具可简化在多台计算机中创建、删除和管理服务器实例的过程。

管理域

管理域（或简称**域**）是一组统一管理的服务器实例。服务器实例属于单个管理域。域中的实例可以在不同的物理主机上运行。

您可以从某个 Application Server 安装中创建多个域。通过将服务器实例分组到域中，不同的组织和管理员可以共享单个 Application Server 安装。每个域都有自己的独立于其他域的配置、日志文件和应用程序部署区域。更改某个域的配置不会影响其他域的配置。同样，在某个域上部署应用程序不会将其部署到任何其他域，或在任何其他域中显示该应用程序。在任何给定时间，管理员只能通过一个域的验证，因而只能在该域中执行管理操作。

域管理服务器 (Domain Administration Server, DAS)

每个域都具有一个域管理服务器 (Domain Administration Server, DAS)，这一特别指定的应用服务器实例用于承载管理应用程序。DAS 会对管理员进行验证，接受来自管理工具的请求，并与域中的服务器实例进行通信以执行请求。

管理工具是指 `asadmin` 命令行工具，即基于浏览器的管理控制台。Application Server 还提供了基于 JMX 的 API 以进行服务器管理。管理员每次只能查看和管理单个域，从而强制进行安全分离。

DAS 有时也称为**管理服务器**或**默认服务器**。之所以称为默认服务器是因为，它是某些管理操作的默认目标。

由于 DAS 是一个应用服务器实例，因此，它也可以承载用于测试的 J2EE 应用程序。但是，不要使用它来承载生产应用程序。例如，如果尚未创建用于承载生产应用程序的群集和实例，您可能希望将应用程序部署到 DAS 中。

DAS 保存一个系统信息库，其中包含其域和部署的所有应用程序的配置。如果 DAS 处于不活动状态或出现故障，则将不会影响活动服务器实例的性能或可用性，但无法进行管理更改。在某些情况下，为了安全起见，有意地停止 DAS 进程可能是非常有用的；例如，停止 DAS 进程以冻结生产配置。

系统提供了一些管理命令，用于备份和恢复域配置和应用程序。通过使用标准的备份和恢复步骤，您可以快速恢复工作配置。如果 DAS 主机出现故障，您必须创建新的 DAS 安装才能恢复以前的域配置。有关说明，请参见《Sun Java System Application Server 9.1 管理指南》中的“重新创建域管理服务器”。

Sun Cluster 数据服务通过转移 DAS 主机 IP 地址故障和使用全局文件系统来提供高可用性的 DAS。这种解决方案提供了几乎持续可用的 DAS 和系统信息库，从而避免出现多种类型的故障。Sun Cluster Data Services 随 Sun Java Enterprise System 附带提供，也可以与 Sun Cluster 一起另行购买。有关更多信息，请参见 Sun Cluster Data Services 文档。

群集

群集是命名的服务器实例集合，这些实例共享相同的应用程序、资源以及配置信息。您可以将不同计算机上的服务器实例分组到一个逻辑群集中并将其作为一个单元来管理。您可以使用 DAS 轻松控制多机群集的生命周期。

群集可以实现水平可伸缩性、负载平衡和故障转移保护。根据定义，群集中的所有实例都具有相同的资源和应用程序配置。当群集中的服务器实例或计算机出现故障时，负载平衡器检测到该故障，会将通信从出现故障的实例重定向至群集中的其他实例，并恢复用户会话状态。由于群集中所有实例上的应用程序和资源都相同，因此一个实例可以故障转移至群集中的任何其他实例。

群集、域和实例是相关的，如下所示：

- 管理域可以包含零个或多个群集。

- 群集包含一个或多个服务器实例。
- 群集属于单个域。

节点代理

节点代理是一个轻量级进程，它在每台承载服务器实例的计算机上运行，其中包括承载 DAS 的计算机。节点代理：

- 按照 DAS 的指令启动和停止服务器实例。
- 重新启动出现故障的服务器实例。
- 为出现故障的服务器提供日志文件视图，并帮助进行远程诊断。
- 将每个服务器实例的本地配置系统信息库与 DAS 的中心系统信息库进行同步，因为服务器实例是在其监视下启动的。
- 在最初创建实例时，创建实例所需的目录，并将实例配置与中心系统信息库进行同步。
- 在删除服务器实例时，执行相应的清除操作。

每个物理主机对其所属的每个域都必须至少具有一个节点代理。如果物理主机包含来自多个域的实例，则每个域需要有一个节点代理。虽然允许在主机上为每个域设置多个节点代理，但这样做并没有什么好处。

因为节点代理用于启动和停止服务器实例，所以它必须始终保持运行。因此，在操作系统引导时，将启动该代理。在 Solaris 和其他 Unix 平台上，可通过 `inetd` 进程来启动节点代理。在 Windows 上，可以将节点代理指定为 Windows 服务。

有关节点代理的更多信息，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的第 8 章 “配置节点代理”。

命名配置

命名配置是一个抽象概念，用于封装 Application Server 属性设置。群集和独立服务器实例引用命名配置以获取其属性设置。通过使用命名配置，J2EE 容器的配置独立于其所在的物理计算机，但特定细节除外，例如 IP 地址、端口号以及堆内存量。使用命名配置可以为 Application Server 管理提供强大的功能和较高的灵活性。

要应用配置更改，只需更改命名配置的属性设置，这样引用它的所有群集和独立实例便可获取更改。仅当删除了对命名配置的所有引用后，才能删除该命名配置。每个域可以包含多个命名配置。

Application Server 附带提供一个名为 default-config 的默认配置。该默认配置已进行了优化，以便在 Application Server Platform Edition 中提高开发者的生产效率以及在 Application Server Enterprise Edition 中提供较高的安全性和可用性。

您可以根据该默认配置创建自己的命名配置，并根据自身需要对其进行自定义。请使用管理控制台和 asadmin 命令行实用程序来创建和管理命名配置。

HTTP 负载平衡器插件

负载平衡器可以在多个物理计算机中分配工作负载，从而提高系统的整体吞吐量。Application Server Enterprise Edition 包含用于 Sun Java System Web Server、Apache Web Server 以及 Microsoft Internet Information Server 的负载平衡器插件。

负载平衡器插件接受 HTTP 和 HTTPS 请求，然后将请求转发到群集中的某个应用服务器实例。如果某个实例出现故障、变得不可用（由于网络故障）或无法响应，则会将请求重定向到现有的可用计算机。负载平衡器还可识别故障实例何时恢复并相应地重新分配负载。

对于简单的无状态应用程序，一个负载平衡群集可能就足够了。但是，对于具有会话状态的重点应用程序，请将负载平衡群集与 HADB 一起使用。

要为系统设置负载平衡，除了 Application Server 以外，还必须安装 Web 服务器和负载平衡器插件。然后，您必须：

- 创建要参与负载平衡的 Application Server 群集。
- 将应用程序部署到这些负载平衡群集中。

参与负载平衡的服务器实例和群集必须具有同构环境。通常，这意味着服务器实例均引用相同的服务器配置、可以访问相同的物理资源，以及具有部署到其上的相同的应用程序。同构环境确保了在出现故障前后，负载平衡器可以始终在群集中的活动实例之间平均分配负载。

可以使用 asadmin 命令行工具来创建负载平衡器配置，将对群集和服务器实例的引用添加到该配置中，启用负载平衡器对群集的引用，为应用程序启用负载平衡，选择创建运行状况检查器，生成负载平衡器配置文件，最后将负载平衡器配置文件复制到 Web 服务器的 config 目录中。管理员可以创建脚本以自动完成整个过程。

有关更多详细信息以及完整的配置说明，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的第 5 章 “配置 HTTP 负载平衡”。

会话持久性

J2EE 应用程序通常具有大量会话状态数据。Web 购物车是会话状态的一个典型示例。此外，应用程序可以高速缓存会话对象中需要频繁使用的数据。事实上，几乎所有包含大量用户交互的应用程序都需要保留会话状态。HTTP 会话和有状态会话 Bean (Stateful Session Bean, SFSB) 均包含会话状态数据。

虽然会话状态没有存储在数据库中的事务状态那样重要，但对于最终用户来说，在出现服务器故障时保留会话状态是非常有必要的。Application Server 提供了在系统信息库中保存或保留此会话状态的功能。如果承载用户会话的应用服务器实例出现故障，则可以恢复会话状态。会话可以继续进行，而不会丢失信息。

Application Server 支持以下类型的会话持久性存储：

- 内存
- 高可用性 (High Availability, HA)
- 文件

利用内存持久性，状态可始终保存在内存中，但会在出现故障时丢失。利用 HA 持久性，Application Server 可将 HADB 用作 HTTP 和 SFSB 会话的持久性存储。利用文件持久性，Application Server 可序列化会话对象，并将其存储到由会话管理器属性指定的文件系统位置。对于 SFSB，如果未指定 HA，则 Application Server 会将状态信息存储在此位置的 session-store 子目录中。

对需要保存的 SFSB 状态更改进行检查称为**检查点操作**。如果启用检查点操作，则通常会在完成任何涉及 SFSB 的事务之后（即使该事务回滚）执行该操作。有关开发有状态会话 Bean 的更多信息，请参见《Sun Java System Application Server 9.1 Developer's Guide》中的"Using Session Beans"。有关启用 SFSB 故障转移的更多信息，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的"有状态会话 Bean 故障转移"。

除了 Application Server 正在处理的请求数以外，会话持久性配置设置还会影响 HADB 每分钟收到的请求数以及每个请求中的会话信息。

有关配置会话持久性的更多信息，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的第 9 章 "配置高可用性会话持久性和故障转移"。

群集中的 IIOP 负载平衡

利用 IIOP 负载平衡，可以将 IIOP 客户机请求分配到不同的服务器实例或名称服务器。目标是将负载平均分布在群集中，从而提供可伸缩性。通过结合使用 IIOP 负载平衡以及 Sun Java System Application Server 中的 EJB 群集和可用性功能，不仅可以提供负载平衡，而且还可以提供 EJB 故障转移。

当客户机对某个对象执行 JNDI 查找时，命名服务会创建一个与特定服务器实例关联的 `InitialContext` (IC) 对象。从此时起，使用该 IC 对象进行的所有查找请求都会发送给相同的服务器实例。使用该 `InitialContext` 查找的所有 `EJBHome` 对象都托管在相同的目标服务器上。此后获得的所有 Bean 引用也创建在相同的目标主机上。这就有效地提供了负载平衡，原因是所有客户机都在创建 `InitialContext` 对象时随机使用动态目标服务器的列表。如果目标服务器实例发生故障，查找或 EJB 方法调用会将故障转移到另一个服务器实例。

例如，图中显示的 ic1、ic2 和 ic3 是在 Client2 的代码中创建的三个不同 `InitialContext` 实例。它们将被分配到群集的三个服务器实例中。因而，此客户机创建的 Enterprise JavaBeans 便会随之分布到这三个实例中。Client1 仅创建一个 InitialContext 对象，并且来自此客户机的 Bean 引用仅位于服务器实例 1 上。如果服务器实例 2 出现故障，ic2 上的查找请求会将故障转移到另一个服务器实例（不一定是服务器实例 3）。对于服务器实例 2 上以前承载的 Bean，还会自动将对其进行的任何 Bean 方法调用重定向到另一个实例（如果此操作安全）。虽然查找故障转移是自动完成的，但 Enterprise JavaBeans 模块将仅在安全时才重试方法调用。

IIOP 负载平衡和故障转移将透明地发生。在应用程序部署过程中无需特殊的步骤。在群集中添加或删除新实例将不会更新该群集的现有客户机视图。您必须在客户端手动更新端点列表。

消息队列和 JMS 资源

Sun Java System Message Queue (MQ) 为分布式应用程序提供了可靠的异步消息传送功能。MQ 是一个企业消息传送系统，它实现了 Java 消息服务 (Java Message Service, JMS) 标准。MQ 为 J2EE 应用程序组件提供了消息传送功能，例如，消息驱动 Bean (Message-Driven Bean, MDB)。

Application Server 通过集成 Sun Java System Message Queue 来实现 Java 消息服务 (Java Message Service, JMS) API。Application Server Enterprise Edition 包含 MQ 企业版，该版本具有故障转移、群集和负载平衡功能。

对于基本 JMS 管理任务，请使用 Application Server 管理控制台和 `asadmin` 命令行实用程序。

对于高级任务（包括管理 Message Queue 群集），请使用 *install_dir*`/imq/bin` 目录中提供的工具。有关管理 Message Queue 的详细信息，请参见 *Sun Java System Message Queue* **管理指南**。

有关部署 JMS 应用程序和 MQ 群集以进行消息故障转移的信息，请参见第 49 页中的“Message Queue 代理部署规划”。

高可用性数据库

本节包括以下主题：

概述

前面的第 25 页中的 “会话持久性”中介绍了 J2EE 应用程序的会话持久性需求。Application Server 将高可用性数据库 (High-Availability Database, HADB) 用作高可用性的会话存储。HADB 包含在 Application Server Enterprise Edition 中，但部署后，可以在单独的主机上运行。HADB 为 HTTP 会话和有状态会话 Bean 数据提供了高可用性的数据存储。

这种分离的体系结构具有以下优点：

- 高可用性群集中的服务器实例是松散耦合的，并用作高性能 J2EE 容器。
- 停止和启动服务器实例不会影响其他服务器或其可用性。
- HADB 可以在另一组成本较低的计算机（例如，具有一个或两个处理器）上运行。几个群集可以共享这些计算机。根据部署需求，您既可以在与 Application Server 相同的计算机（同位）上运行 HADB，也可以在不同的计算机（分层）上运行 HADB。有关这两个选项的更多信息，请参见第 57 页中的 “同位拓扑”。
- 当状态管理要求发生变化时，可以在 HADB 系统中添加资源，而不会影响现有群集或其应用程序。

注 – HADB 进行了优化以便 Application Server 使用，但不能作为通用数据库供应用程序使用。

有关 HADB 硬件和网络系统要求，请参见《Sun Java System Application Server 9.1 发行说明》中的“硬件和软件要求”。有关 HADB 所需的其他系统配置步骤，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的第 2 章 “安装和设置高可用性数据库”。

系统要求

HADB 主机的系统要求如下：

- 每个 HADB 节点至少有一个 CPU。
- 每个节点至少有 512 MB 的内存。

有关网络配置要求，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的第 2 章 “安装和设置高可用性数据库”。有关高可用性的其他要求，请参见第 31 页中的 “减小双故障风险”。

HADB 体系结构

HADB 是由**节点**对组成的分布式系统。节点将划分到两个数据冗余单元 (Data Redundancy Unit, DRU) 中，每个 DRU 包含每对节点中的一个节点，如第 29 页中的 “数据冗余单元”中所述。

每个节点包含：

- 一组用于事务状态复制的进程。
- 用于在进程之间进行通信的专用共享内存区域。
- 一个或多个辅助存储设备（磁盘）。

一组 HADB 节点可以承载一个或多个**会话数据库**。每个会话数据库与一个特定的应用服务器群集相关联。在删除群集时，还会删除与其关联的会话数据库。

有关 HADB 硬件要求，请参见《Sun Java System Application Server 9.1 发行说明》中的“硬件和软件要求”。

节点和节点进程

共有两种类型的 HADB 节点：

- **活动节点**，用于存储数据。
- **备用节点**，最初不包含任何数据，但在活动节点变为不可用时，它将作为活动节点使用。备用节点是可选的，但对获得较高可用性非常有用。

每个节点包含一个父进程和几个子进程。父进程称为节点监控 (NSUP)，由管理代理启动。它负责创建子进程并使它们保持运行。

子进程可以是：

- 事务服务器进程 (TRANS)，用于协调分布式节点上的事务以及管理数据存储。
- 关系代数服务器进程 (RELALG)，用于协调并执行复杂的关系代数查询，如排序和连接。
- SQL 共享内存服务器进程 (SQLSHM)，用于维护 SQL 字典高速缓存。
- SQL 服务器进程 (SQLC)，用于接收客户机查询，将查询编译为本地 HADB 指令，将这些指令发送到 TRANS，接收结果并将结果传送到客户机。对于每个客户机连接，每个节点包含一个主 SQL 服务器和一个子服务器。
- 节点管理器服务器进程 (NOMAN)，管理代理使用该进程来执行由 hadbm 管理客户机发出的管理命令。

数据冗余单元

如前面所述，HADB 实例包含一对 DRU。每对中的两个 DRU 具有相同的活动节点数和备用节点数。一个 DRU 中的每个活动节点在另一个 DRU 中具有**镜像节点**。由于进行了镜像，每个 DRU 都包含完整的数据库副本。

下图显示了一个包含 6 个节点的示例 HADB 体系结构：四个活动节点和两个备用节点。节点 0 和 1 是一个镜像对，节点 2 和 3 也是一个镜像对。在此示例中，每个主机有一个节点。通常，如果主机具备足够的系统资源（请参见第 28 页中的 “系统要求”），则它可以有多个节点。

注 – 必须成对添加承载 HADB 节点的计算机，每个 DRU 中各有一台计算机。

HADB 可通过复制数据和服务来获得高可用性。可以将镜像节点上的数据副本指定为**主副本**和**热备用副本**。主副本执行插入、删除、更新以及读取等操作。热备用副本接收主副本操作的日志记录，并在事务生命周期内重复执行这些操作。读取操作仅由主节点执行，因而不会记录这些操作。每个节点同时包含主副本和热备用副本，并担当这两种角色。数据库将进行分段并分配到 DRU 的活动节点中。镜像对中的每个节点包含一组相同的数据分段。复制镜像节点上的数据称为**复制**。HADB 可通过复制来提供高可用性：当某个节点出现故障时，其镜像节点几乎立即接管该节点的功能（在几秒钟内）。复制可确保高可用性，并且屏蔽节点故障或 DRU 故障，而不会丢失数据或服务。

当镜像节点接管出现故障的节点的功能时，它必须完成双倍的任务量：其自己的任务和出现故障的节点的任务。如果镜像节点不具备足够的资源，过多的负载会降低其性能，并增大出现故障的可能性。当节点出现故障时，HADB 会尝试重新启动它。如果无法重新启动出现故障的节点（例如，由于硬件故障），则系统将继续运行，但会降低可用性。

HADB 容许整个 DRU、一个或多个节点出现故障，但不容许出现“双故障”，即节点及其镜像均出现故障。有关如何减小出现双故障的可能性的信息，请参见第 31 页中的“减小双故障风险”。

备用节点

当某个节点出现故障时，其镜像节点将接管该节点的功能。如果出现故障的节点没有备用节点，则此时它将没有镜像。备用节点自动接替出现故障的节点的镜像。通过设置备用节点，可以缩短系统在没有镜像节点的情况下运行的时间。

备用节点通常不包含数据，但会持续监视 DRU 中活动节点的故障情况。如果某个节点出现故障并且在指定的超时期限内未能恢复，备用节点将从镜像节点中复制数据并与其进行同步。所花的时间取决于复制的数据量以及系统和网络能力。在进行同步后，备用节点将自动接替镜像节点，而无需手动介入，从而减轻了镜像节点的过载压力并平衡了镜像上的负载。这称为**故障恢复**或**自我修复**。

在修复出现故障的主机（通过更换硬件或升级软件）并重新启动后，该主机中运行的一个或多个节点将作为备用节点加入到系统中，因为原始备用节点现在是活动节点。

备用节点并不是必需的，但系统可以通过这些节点保持其整体服务级别，即使在某台计算机出现故障的情况下也是如此。备用节点还可以简化在承载活动节点的计算机上执行计划维护的过程。请为每个 DRU 分配一台计算机以作为备用计算机，以便在某台计算机出现故障时，HADB 系统可继续运行，而不会对性能和可用性造成负面影响。

注 – 通常，应使用具有足够多 Application Server 实例和 HADB 节点的备用计算机接替变为不可用的任何计算机。

示例备用节点配置

以下示例说明了如何在 HADB 部署中使用备用节点。共有两种可能的部署拓扑：**同位**（HADB 和 Application Server 位于相同的主机上）和**分层**（它们位于不同的主机上）。有关部署拓扑的更多信息，请参见第 3 章。

示例：同位配置

作为一个备用节点配置示例，假定四个 Sun Fire™ V480 服务器采用的是同位拓扑，每个服务器具有一个 Application Server 实例和两个 HADB 数据节点。

对于备用节点，额外分配两个服务器（每个 DRU 一台计算机）。每台备用计算机运行一个应用服务器实例和两个备用 HADB 节点。

示例：分层配置

假定使用以下分层拓扑：HADB 层具有两个 Sun Fire™ 280R 服务器，每个服务器运行两个 HADB 数据节点。为保持此系统的完整能力（即使一台计算机变为不可用），对 Application Server 实例层以及 HADB 层各配置一台备用计算机。

Application Server 实例层的备用计算机所具有的实例数必须与该层中的其他计算机相同。同样，HADB 层的备用计算机所具有的 HADB 节点数也必须与该层中的其他计算机相同。

减小双故障风险

HADB 的内置数据复制功能使其容许单个节点或整个 DRU 出现故障。默认情况下，在出现**双故障**时，即镜像节点对或两个 DRU 均出现故障时，HADB 将无法继续运行。在这种情况下，HADB 会变为不可用。

除了使用上一节中介绍的备用节点外，还可以采取以下措施来最大限度地减小出现双故障的可能性：

- **提供独立电源**：为实现最佳的容错能力，支持一个 DRU 的服务器必须使用独立的电源（通过不间断电源）、处理单元和存储器。如果一个 DRU 出现电源故障，另一个 DRU 中的节点可以继续处理请求，直至电源恢复正常。
- **提供双互连**：要容许单个网络出现故障，可复制 DRU 之间的线路和交换机。

这些措施是可选的，但会提高 HADB 实例的整体可用性。

HADB 管理系统

HADB 管理系统提供了内置安全功能并简化了多平台管理。如下图所示，HADB 管理体系结构包含以下组件：

如图所示，在运行 HADB 服务的每台计算机上都运行了一个 HADB 管理代理。每台计算机通常承载一个或多个 HADB 节点。与 Application Server 域类似，HADB 管理域中包含多台计算机。要容许数据库出现故障，域中必须至少有两台计算机；通常计算机的个数必须为偶数以组成 DRU 对。因此，每个域中包含多个管理代理。

如图所示，每个域可以包含一个或多个数据库实例。每台计算机可以包含一个或多个节点，这些节点属于一个或多个数据库实例。

管理客户机

HADB **管理客户机**是命令行实用程序 `hadbm`，用于管理 HADB 域及其数据库实例。HADB 服务可以连续运行（即使已停止关联的 Application Server 群集），但如果要删除这些服务，则必须小心将其关闭。有关使用 `hadbm` 的更多信息，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的第 3 章 “管理高可用性数据库”。

可以使用 asadmin 命令行实用程序来创建和删除与高可用性群集关联的 HADB 实例。有关更多信息，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的第 9 章 “配置高可用性会话持久性和故障转移”。

管理代理

管理代理是一个名为 `ma` 的服务器进程，它可以访问主机上的资源；例如，它可以创建设备并启动数据库进程。管理代理协调并执行管理客户机命令，如启动或停止数据库实例。

管理客户机通过指定管理代理的地址和端口号来连接到该代理。在连接后，管理客户机通过管理代理向 HADB 发送命令。代理将接收并执行请求。因此，管理代理必须正在主机上运行，才能向该主机发出任何 `hadbm` 管理命令。可以将管理代理配置为自动启动的系统服务。

确保管理代理的可用性

管理代理进程通过重新启动出现故障的 HADB 节点监控进程来确保其可用性。因此，要进行部署，您必须确保 `ma` 进程的可用性才能保持 HADB 的整体可用性。在重新启动后，管理代理通过域中的其他代理来恢复域和数据库配置数据。

可以使用主机操作系统 (Operating System, OS) 来确保管理代理的可用性。在 Solaris 或 Linux 上，`init.d` 可确保 `ma` 进程在出现故障并重新引导操作系统后的可用性。在 Windows 上，管理代理作为 Windows 服务运行。因此，如果管理代理出现故障或重新引导操作系统，操作系统将重新启动该代理。

管理域

HADB 管理域是一组主机，每个主机在相同的端口号上运行管理代理。域中的主机可以包含一个或多个 HADB 数据库实例。管理域是由代理使用的通用端口号以及在创建域或在其中添加代理时生成的标识符（称为 *domainkey*）定义的。domainkey 为域提供唯一的标识符，这一点至关重要，因为管理代理使用多址广播进行通信。可以将 HADB 管理域设置为与 Application Server 域相匹配。

在开发环境中，在一个域中包含多个数据库实例会非常有用，因为这样可以使不同的开发者小组使用其自己的数据库实例。在某些情况下，这在生产环境中也可能非常有用。

所有属于某个域的代理将协调其管理操作。在通过 `hadbm` 命令更改数据库配置时，所有代理将相应地更改配置。除非节点主机上的管理代理正在运行，否则无法停止或重新启动该节点。不过，即使某些代理不可用，您也可以执行 `hadbm` 命令以读取 HADB 状态或配置变量值。

可以使用以下管理客户机命令来执行与管理域相关的操作：

- **hadbm createdomain**：使用指定的主机创建管理域。
- **hadbm extenddomain**：将主机添加到现有管理域中。
- **hadbm deletedomain**：删除管理域。
- **hadbm reducedomain**：从管理域中删除主机。
- **hadbm listdomain**：列出在管理域中定义的所有主机。

系统信息库

管理代理将数据库配置存储在**系统信息库**中。系统信息库具有较高的容错能力，因为将在所有管理代理上复制该系统信息库。通过将配置保存在服务器上，您可以从任何装有管理客户机的计算机中执行管理操作。

要对系统信息库进行任何更改，域中的大多数管理代理必须正在运行。因此，如果域中有 M 个代理，则必须至少有 M/2 + 1 个代理（向下舍入为整数）正在运行才能对系统信息库进行更改。

如果域中的某些主机不可用（例如，由于硬件故障），并且由于未达到法定数目而无法执行某些管理命令，请使用 `hadbm disablehost` 命令从域中删除出现故障的主机。

设置和配置规划

▼ 为 Application Server 设置并配置高可用性

1 **确定性能以及 QoS 要求和目标，如第 1 章中所述。**

2 **确定系统大小，如第 1 章中的第 46 页中的 “ 设计决策 ” 所述。**

- Application Server 实例数
- HADB 节点和主机数
- HADB 存储容量

3 **确定系统拓扑，如第 3 章中所述。**

这可确定是将 HADB 安装在与 Application Server 相同的主机上，还是安装在不同的主机上。

4 **安装 Application Server 实例以及相关的子组件，如 HADB 和 Web 服务器。**

5 **创建域和群集。**

6 **配置 Web 服务器软件。**

7 **安装负载平衡器插件。**

8 **设置并配置负载平衡。**

9 **设置并配置 HADB 节点和 DRU。**

10 **为 AS Web 容器和 EJB 容器配置 HA 会话持久性。**

11 **部署应用程序并为其配置高可用性和会话故障转移。**

12 **如果大量使用消息传送功能，则为 JMS 群集配置故障转移。**

有关更多信息，请参见《*Sun Java System Message Queue Adminstration Guide*》。

◆ ◆ ◆ 第 2 章

2 部署规划

在部署 Application Server 之前，请先确定性能和可用性目标，然后再相应地确定硬件、网络和存储要求。

本章包括以下几个部分：

设定性能目标

简单地说，高性能是指最大限度地增加吞吐量并缩短响应时间。除了这些基本目标外，还可以通过确定以下内容来设定特定目标：

- 要部署哪些类型的应用程序和服务以及客户机如何访问它们?
- 哪些应用程序和服务需要具有较高的可用性?
- 应用程序是具有会话状态还是无状态?
- 系统必须支持的请求数量或吞吐量是多少?
- 系统必须支持多少个并发用户?
- 可接受的用户请求的平均响应时间是多少?
- 请求之间的平均延迟时间是多少?

可以使用远程浏览器仿真器 (Remote Browser Emulator, RBE) 工具或 Web 站点性能和基准软件（用于模拟预期的应用程序活动）来计算其中的某些度量。通常，RBE 和基准产品将生成并发的 HTTP 请求，然后报告每分钟处理给定请求数的响应时间。然后，您可以使用这些数据来计算服务器的活动。

本章介绍的计算结果并不是一成不变的。在微调 Application Server 和应用程序的性能时，请将这些结果作为工作的参考点。

本节包括以下主题：

估算吞吐量

从广义而言，**吞吐量**测量的是 Application Server 完成的工作量。对于 Application Server，可以将吞吐量定义为每个服务器实例每分钟处理的请求数。高可用性应用程序还会对 HADB 的吞吐量有要求，因为它们会定期保存会话状态数据。对于 HADB，可以将其吞吐量定义为每分钟存储的会话数据量，它是每分钟的 HADB 请求数与每个请求的平均会话大小的乘积。

如下一节所述，Application Server 吞吐量是多种因素的作用结果，其中包括用户请求的特性和大小、用户数以及 Application Server 实例和后端数据库的性能。可以使用模拟的工作负载作为基准来估算单个计算机上的吞吐量。

高可用性应用程序会产生额外的开销，因为它们会定期将数据保存到 HADB 中。开销量取决于数据量、数据的更改频率和数据的保存频率。前两种因素取决于所使用的应用程序；其中后一种还受服务器设置的影响。

可以将 HADB 吞吐量定义为每分钟的 HADB 请求数与每个请求的平均数据量的乘积。较大的 HADB 吞吐量意味着，需要更多的 HADB 节点以及更大的存储大小。

估算 Application Server 实例上的负载

应考虑以下因素来估算 Application Server 实例上的负载：

最大并发用户数

用户通过客户机（如 Web 浏览器或 Java 程序）与应用程序进行交互。根据用户的操作，客户机会定期向 Application Server 发送请求。只要用户会话既未到期也没有终止，系统就认为用户处于**活动**状态。在估算并发用户数时，应将所有活动用户包括在内。

下图对每分钟处理的请求数（吞吐量）与用户数之间的典型关系进行了图形说明。最初，随着用户数的增加，吞吐量也会随之增加。不过，随着并发请求数的增加，服务器性能开始趋于饱和，并且吞吐量开始下降。

请确定一个临界点，当超过此限制后，您再添加并发用户时，每分钟可处理的请求数将会下降。该点表示此时已达到最佳的性能，之后吞吐量将开始下降。一般来说，应尽可能地使系统在最佳吞吐量下运行。您可能需要增加处理能力以处理额外的负载并提高吞吐量。

延迟时间

用户并不是连续地提交请求。当用户提交请求后，服务器将接收并处理该请求，然后返回结果，此时，用户需要等待一段时间，然后再提交新请求。两个请求的间隔时间称为**延迟时间**。

延迟时间取决于用户类型。例如，用于 Web 服务的计算机间交互的延迟时间通常比用户交互的延迟时间短。您可能需要综合考虑计算机和用户交互的情况来估算延迟时间。

确定平均延迟时间是至关重要的。可以使用此时段来计算每分钟需要完成的请求数以及系统可支持的并发用户数。

平均响应时间

响应时间是指 Application Server 为用户返回请求结果所花的时间。响应时间受很多因素的影响，例如，网络带宽、用户数、提交的请求数和类型以及平均延迟时间。

在本节中，响应时间是指平均响应时间。每种类型的请求都具有其自己的最小响应时间。不过，在评估系统性能时，请根据所有请求的平均响应时间进行分析。

响应时间越快，每分钟处理的请求就会越多。不过，随着系统上的用户数的增加，即使每分钟的请求数有所下降，响应时间也会开始增加，如下图所示：

抱歉：此版本的手册未提供相关图形。

与此图形类似的系统性能图形表明，达到某个临界点后，每分钟的请求数与响应时间将成反比。每分钟的请求数下降得越快，响应时间（由虚线箭头表示）就会增加得越多。

在该图中，峰值负载点为每分钟的请求数开始下降的点。在该点之前，响应时间的计算并不一定准确，因为在其公式计算中不使用峰值数字。在该点之后（由于每分钟的请求数与响应时间成反比关系），管理员可以使用每分钟的最大用户数和请求数更准确地计算响应时间。

请使用以下公式来确定 $T_{response}$ 的值，即峰值负载时的响应时间（以秒为单位）：

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

其中

- *n* 是并发用户数
- *r* 是服务器每秒收到的请求数

- T_{think} 是平均延迟时间（以秒为单位）

  要获得准确的响应时间结果，应始终在等式中使用延迟时间。

示例 2–1 计算响应时间

如果以下条件成立：

- 系统在达到峰值负载时可支持的最大并发用户数 *n* 为 5,000 个。
- 系统在达到峰值负载时可处理的最大请求数 *r* 为每秒 1,000 个。

平均延迟时间 T_{think} 为每个请求 3 秒。

因此，响应时间的计算公式为：

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

因此，响应时间为 2 秒。

在计算出系统的响应时间后（尤其是峰值负载时的响应时间），请将其与应用程序可接受的响应时间进行比较。响应时间以及吞吐量是两个影响 Application Server 性能的主要因素。

每分钟的请求数

如果知道任意给定时间的并发用户数、用户请求的响应时间以及平均用户延迟时间，则可以计算出每分钟的请求数。通常，是从估算系统上的并发用户数开始的。

例如，在运行 Web 站点性能软件后，管理员可推断出在联机银行的 Web 站点上提交请求的平均并发用户数为 3,000 个。此数字取决于已注册成为联机银行会员的用户数、其银行交易行为以及他们选择提交请求的日期或星期的时间等。

因此，如果知道了这些信息，则可以使用本节中介绍的每分钟的请求数公式，计算出系统每分钟可为此用户群处理的请求数。由于峰值负载时的每分钟请求数与响应时间成反比，因此，您可以决定选择每分钟较少的请求数来换取较快的响应时间，或者选择较慢的响应时间来换取每分钟较多的请求数。

微调系统性能首先要对不同的每分钟请求数和响应时间阈值进行试验以选出最佳的值。此后，将确定需要调整的系统区域。

计算上一节的等式中的 *r* 可得出：

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

示例 2–2 计算每秒的请求数

对于以下值：

- $n = 2,800$ 个并发用户
- $T_{response} = 1$（每个请求的平均响应时间为 1 秒）

示例 2–2 计算每秒的请求数 （续）

- $T_{think} = 3$（平均延迟时间为 3 秒）

每秒请求数的计算结果为：

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

因此，每秒的请求数为 700 个，每分钟的请求数为 42000 个。

估算 HADB 上的负载

要计算 HADB 上的负载，请考虑以下因素：

- 第 39 页中的 “HTTP 会话持久性频率”
- 第 39 页中的 “HTTP 会话大小和范围”
- 第 41 页中的 “有状态会话 Bean 检查点”

有关配置会话持久性的说明，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的第 9 章 “配置高可用性会话持久性和故障转移”。

HTTP 会话持久性频率

HADB 每分钟收到的请求数取决于**持久性频率**。持久性频率决定了 Application Server 将 HTTP 会话数据保存到 HADB 中的频率。

持久性频率选项为：

- **web-method**（默认）：服务器将会话数据与每个 HTTP 响应存储在一起。此选项可确保存储的会话信息是最新的，但会产生较高的 HADB 通信量。
- **time-based**：按指定的时间间隔存储会话。此选项可减少 HADB 通信量，但无法确保会话信息是最新的。

下表简要说明了持久性频率选项的优缺点。

表 2–1 持久性频率选项的比较

持久性频率选项	优点	缺点
web-method	确保提供最新的会话信息。	可能会增加响应时间并降低吞吐量。
time-based	响应时间较短，并且可能会提高吞吐量。	不能完全确保在应用服务器实例出现故障后提供最新的会话信息。

HTTP 会话大小和范围

每个请求的会话大小取决于会话中存储的会话信息量。

提示 – 要提高整体性能，应尽可能减少会话中的信息量。

可通过**持久性范围**设置微调每个请求的会话大小。请从以下 HTTP 会话持久性范围选项中进行选择：

- **session**：每次服务器将会话信息保存到 HADB 时，将序列化并保存整个会话对象。
- **modified-session**：服务器仅会保存已修改的会话。它通过截获对该 Bean 的 `setAttribute()` 方法的调用来检测修改。此选项将不检测对内部对象的直接修改，因此，在这种情况下，必须对 SFSB 进行编码才能显式调用 `setAttribute()`。
- **modified-attribute**：服务器仅保存自上次会话存储以来修改（插入、更新或删除）的属性。它与 modified-session 的缺点相同，但如果正确应用，可以显著降低 HADB 写入吞吐量要求。

要使用此选项，应用程序必须：

- 在每次修改会话状态时调用 `setAttribute()` 或 `removeAttribute()`。
- 确保各属性之间没有交叉引用。
- 在多个属性之间分布会话状态，或者至少在只读属性和可修改属性之间分布会话状态。

下表简要说明了持久性范围选项的优缺点。

表 2–2 持久性范围选项的比较

持久性范围选项	优点	缺点
modified-session	为没有修改会话状态的请求提供改进的响应时间。	在执行 Web 方法（通常为 `doGet()` 或 `doPost()`）期间，应用程序必须调用一种会话方法： ■ 如果更改了属性，则调用 `setAttribute()`。 ■ 如果删除了属性，则调用 `removeAttribute()`。
session	对应用程序没有限制。	与 `modified-session` 和 `modified-attribute` 选项相比，可能具有更差的吞吐量和响应时间。
modified-attribute	如果为请求修改的会话状态百分比较低，则请求具有更佳的吞吐量和响应时间。	当为给定请求修改的会话状态百分比接近 60% 时，吞吐量将会下降，响应时间将会延长。在这种情况下，其性能比使用其他选项的性能要低，因为将属性分隔为各个记录会产生一些开销。

有状态会话 Bean 检查点

对于 SFSB 会话持久性，HADB 上的负载取决于以下内容：

- 为检查点启用的 SFSB 数。
- 为检查点选择的 SFSB 方法以及使用这些方法的频率。
- 会话对象的大小。
- 哪些方法是事务性的。

通常在完成任何涉及 SFSB 的事务之后（即使该事务回滚）执行检查点操作。

为了获得最佳性能，请为检查点指定较少的一组方法。所检查的数据大小以及检查频率决定了在给定客户机交互的响应时间方面产生的额外开销。

网络配置规划

在规划如何将 Application Server 集成到网络中时，请估算带宽要求并使用某种方法规划网络以使其满足用户的性能要求。

本节包括以下主题：

估算带宽要求

要确定所需的网络大小和带宽，请先确定网络通信量和峰值。检查是否在特定时间、星期或日期出现总体流量峰值，然后确定该峰值的持续时间。

在峰值负载期间，网络中的数据包数达到最大级别。通常，如果根据峰值负载进行设计，则需要扩展系统以实现处理 100% 的峰值流量目标。但要记住，任何网络都具有一些无法预测的行为，即便进行了扩展，也并非始终能够处理 100% 的峰值流量。

例如，假定在出现峰值负载时，有 5% 的用户在访问 Application Server 中部署的应用程序时，偶尔会出现无法立即访问网络的情况。在这 5% 的用户中，请估算一下有多少个用户在第一次访问尝试后会再次尝试进行访问。此外，不是所有的用户都能够进行访问，而未能成功访问的用户中，又会有一部分用户再次尝试进行访问。因此，峰值会持续较长的时间，因为在用户继续尝试访问时，峰值情况也会持续下去。

计算所需的带宽

根据第 35 页中的 “设定性能目标”中的计算结果，确定在您的站点中部署 Application Server 所需的额外带宽。

根据访问方法（T-1 线路、ADSL 和电缆调制解调器等），计算需要增加多少带宽就可以处理估算的负载。例如，假定您的站点使用 T-1 或高速 T-3 线路。在给定带宽的情况下，请根据您所在站点每秒平均生成的请求数以及最大峰值负载来估算网络上所需的线路数。请使用 Web 站点分析和监视工具来计算这些数字。

示例 2–3 计算所需的带宽

一条 T-1 线路可以处理 1.544 Mbps 的数据。因此，由 4 条 T-1 线路组成的网络可以处理大约 6 Mbps 的数据。假定发回到客户机的平均 HTML 页大小为 30 KB，则由 4 条 T-1 线路组成的网络每秒可以处理的通信量如下所示：

6,176,000 位/8 位 = 每秒 772,000 字节

每秒 772,000 字节/30 KB = 每秒大约 25 个并发响应页。

对于每秒 25 页的通信量，此系统每小时可处理 90,000 页（25 x 60 秒 x 60 分钟），因此，每天最多可处理 2,160,000 页（假定全天各时段的负载是均等的）。如果最大峰值负载大于此数值，请相应地增加带宽。

估算峰值负载

全天各时段具有均等的负载可能不太现实。您需要确定峰值负载的出现时间、持续时间以及峰值负载占总负载的百分比。

示例 2–4 计算峰值负载

如果峰值负载占总负载 2,160,000 页的 30% 且持续时间为两小时，这意味着必须在当天的两小时内通过 T-1 线路传输 648,000 页。

因此，为了在这两个小时内处理峰值负载，请根据以下计算结果增加 T-1 线路：

648,000 页/120 分钟 = 每分钟 5,400 页

每分钟/60 秒 5,400 页 = 每秒 90 页

如果 4 条线路每秒可处理 25 页，则大约 4 倍的页数需要 4 倍的线路数，此处为 16 条线路。这 16 条线路用于处理实际上最大为 30% 的峰值负载。显而易见，另外的 70% 负载可在当天的其余时间通过这些线路进行处理。

配置子网

如果使用分层拓扑（应用服务器实例和 HADB 节点位于不同的主机上），可通过将所有 HADB 节点放在单独的子网上来提高性能。这是因为 HADB 使用用户数据报协议 (User Datagram Protocol, UDP)。使用单独的子网可降低该子网外部的计算机上的 UDP 通信量。但请注意，所有 HADB 节点必须位于同一个子网上。

您仍然可以从其他子网中运行管理客户机，但前提是所有节点和管理代理位于同一个子网上。应该可以在所有节点代理内访问任何主机和端口，并且防火墙以及 UDP 阻塞等都不能阻止节点访问。

HADB 使用 UDP 多址广播，因此，必须为包含 HADB 节点的任何子网配置多址广播。

选择网卡

为获得更大的带宽和最佳的网络性能，请至少使用 100 Mbps 的以太网卡，或者最好在承载 Application Server 和 HADB 节点的服务器之间使用 1 Gbps 的以太网卡。

HADB 的网络设置

注 – HADB 使用 UDP 多址广播，因此，必须在系统路由器和主机网络接口卡上启用多址广播。如果 HADB 跨多个子网，还必须在子网之间的路由器上启用多址广播。为获得最佳结果，请将 HADB 节点全部放在同一个网络中。Application Server 实例可以位于不同的子网上。

如果采用以下建议，可使 HADB 在网络中以最佳方式进行工作：

- 使用交换路由器，以便每个网络接口具有专用的 100 Mbps 或更高的以太网通道。
- 在承载 4 个或更多 HADB 节点的多 CPU 计算机上运行 HADB 时，请使用 1 Gbps 的以太网卡。如果平均会话大小大于 50 KB，即使每台计算机的 HADB 节点数少于 4 个，也应该使用 1 Gbps 的以太网卡。
- 如果怀疑 HADB 节点内存在网络瓶颈：
  - 在 HADB 服务器上运行网络监视软件来诊断问题。
  - 考虑将网络中的所有 100 Mbps 以太网卡更换为 1 Gbps 以太网卡。

可用性规划

本节包括以下主题：

设置最佳的可用性

要规划系统和应用程序的可用性，请评估访问不同应用程序的用户组的可用性需求。例如，外部付费用户和业务合作伙伴要求的服务质量 (Quality Of Service, QoS) 通常比内部用户高。因此，与外部付费客户相比，内部用户更容易接受应用程序功能、应用程序或服务器不可用的事实。

下图展示了在事件的发生概率不断下降时，防范这些事件的成本和复杂性却不断增加的情况。在图的一端，简单的负载平衡群集可实现本地化应用程序、中间件以及硬件的容错能力。在图的另一端，位于不同地理位置的群集可以防范影响整个数据中心的重大灾难。

要实现较好的投资回报，确定应用程序中的功能可用性要求通常是非常有必要的。例如， 保险报价系统不可用（可能会丢失新业务）是不可接受的，而帐户管理功能（现有客户可查看其当前的保险项目）暂时不可用，则不太可能使现有客户流失。

使用群集提高可用性

在最基本的情况下，群集是一组应用服务器实例（通常位于多个物理服务器上），对于客户机而言，这些实例显示为单个实例。这可提供水平伸缩性以及比单个计算机上的单个实例更高的可用性。这种基本级别的群集与 Application Server 的 HTTP 负载平衡器插件结合使用，它接受 HTTP 和 HTTPS 请求，并将其转发到群集中的某个应用服务器实例。ORB 和集成的 JMS 代理也会为应用服务器群集执行负载平衡。如果某个实例出现故障、变得不可用（由于网络故障）或无法作出响应，则会将请求仅重定向到现有的可用计算机中。负载平衡器还可以识别有故障的实例何时得到恢复并相应地重新分配负载的情况。

HTTP 负载平衡器还提供一个**运行状况检查器**程序，它可以监视服务器和特定 URL 以确定它们是否可用。必须小心地控制运行状况检查的开销，以免其本身成为较大的处理负载。

对于无状态应用程序或仅包含较小值的简单用户事务的应用程序，通常只需要使用简单的负载平衡群集。对于有状态的重要应用程序，请考虑使用 HADB 保存会话持久性。有关 HADB 的概述，请参见 Application Server 管理指南第 1 章中的第 27 页中的 “高可用性数据库”。

要联机执行应用程序的升级，最好将应用服务器实例划分到多个群集中。Application Server 可以将应用程序和实例设置为**休眠状态**。休眠是指以可控制的方式将实例（或实例组）或特定应用程序设置为脱机状态，而不会影响用户当前使用实例或应用程序。将某个实例设置为休眠状态后，新用户将使用另一个实例上已升级的应用程序。这种类型的应用程序升级称为**滚动升级**。有关升级实时应用程序的更多信息，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的"升级应用程序而不使可用性受到损失"。

在系统中添加冗余功能

一种实现高可用性的方法是，在系统中添加硬件和软件冗余功能。当一个单元出现故障时，冗余单元将接管该单元。这也称为容错。通常，要最大限度提高可用性，请确定并消除系统中每个可能的故障点。

确定故障类

冗余级别是由系统需要容错的故障类（故障类型）确定的。下面是一些故障类示例：

- 系统进程
- 计算机
- 电源
- 磁盘
- 网络故障
- 建筑物火灾或其他可预防的灾难
- 无法预测的自然灾害

重复的系统进程可实现单个系统进程和单个计算机的容错能力。通过将重复的镜像（成对）计算机连接到不同的电源，可以实现单个电源的容错能力。通过在不同的建筑物中配置镜像计算机，可以实现单个建筑物火灾的容错能力。通过在不同的地理位置配置这些计算机，可以实现自然灾害（如地震）的容错能力。

使用 HADB 冗余单元提高可用性

为提高可用性，应始终在数据冗余单元 (Data Redundancy Unit, DRU) 中使用 HADB 节点，如第 35 页中的 "设定性能目标"中所述。

使用 HADB 备用节点提高容错能力

使用备用节点可提高容错能力。虽然并不强制要求使用备用节点，但它们可提供最大限度的可用性。

故障转移容量规划

故障转移容量规划是指，确定需要在 Application Server 部署中添加多少个额外的服务器和进程，以便在出现服务器或进程故障时，系统可以无缝地恢复数据并继续进行处

理。如果系统出现过载，则可能导致进程或服务器出现故障，从而使响应时间变慢，甚至造成服务全部中断。为此类情况做好准备对成功部署至关重要。

要保持容量（尤其是峰值负载时），请在现有部署中添加运行 Application Server 实例的备用计算机。

例如，请考虑一个包含两台计算机的系统，每台计算机运行一个 Application Server 实例。这两台计算机共同处理每秒 300 个请求的峰值负载。如果其中的一台计算机变得不可用，则系统只能处理 150 个请求，假定负载是在两台计算机之间均等分配的。因此，有一半的请求在峰值负载期间无法处理。

设计决策

设计决策包括是根据峰值负载还是稳定状态负载来设计系统的，并确定具有不同角色和大小容量的计算机数量。

根据峰值负载或稳定状态负载进行设计

在典型部署中，稳定状态和峰值工作负载是有差别的：

- 如果系统设计用于处理峰值负载，则它可以承受预期的最大用户和请求负载，而不会出现响应时间变慢的情况。这意味着，系统可以处理预期系统负载的极端情况。如果峰值负载和稳定状态负载的差别很大，则根据峰值负载进行设计可能意味着，花费大量资金购置的资源经常会闲置起来。
- 如果系统设计用于处理稳定状态负载，则它不需要处理预期峰值负载所需的所有资源。因此，在出现峰值负载时，系统的响应时间会变慢。

估计系统处理峰值负载的频率，将决定您要根据峰值负载还是稳定状态负载进行设计。

如果经常出现峰值负载（比如，每天出现多次），则需要扩充容量来处理峰值负载。如果系统在 90% 的时间内都是平稳运行的，而仅有 10% 的时间在峰值状态下运行，则最好部署一个根据稳定状态负载设计的系统。这意味着，仅在 10% 的时间里，系统的响应时间才会变慢。确定系统在峰值状态下运行的频率或持续时间，可以判断是否有必要在系统中添加资源。

设定系统规模

根据应用服务器实例上的负载、HADB 上的负载以及故障转移要求，您可以确定：

- 第 47 页中的 “Application Server 实例数”
- 第 47 页中的 “HADB 节点数”
- 第 48 页中的 “HADB 主机数”

- 第 48 页中的 “HADB 存储容量”

Application Server 实例数

要确定所需的应用服务器实例（主机）数，请根据第 36 页中的 “估算 Application Server 实例上的负载”中介绍的各种因素评估每个应用服务器实例的环境，即使每个实例可以使用多个中央处理单元 (Central Processing Unit, CPU) 也要进行评估。

HADB 节点数

作为一般准则，计划为系统中的每个 CPU 分配一个 HADB 节点。例如，包含两个 CPU 的计算机使用两个 HADB 节点。

注 – 如果每台计算机（例如，如果使用较大的计算机）有多个 HADB 节点，则必须确保计算机上有足够的冗余和可伸缩性；例如，配备了多个不间断电源和独立磁盘控制器。

或者，使用以下步骤。

▼ 确定所需的 HADB 节点数

1 **确定以下参数：**

- 最大并发用户数 n_{users}。
- 平均 BLOB 大小 s。
- 每个用户的最大事务率，称为 NTPS。

2 **确定最大主数据量的大小 V_{data}（以千兆字节为单位）。**

请使用下面的公式：

$$V_{data} = n_{users} \cdot s$$

3 **确定最大 HADB 数据传输速率 R_{dt}。**

这反映了从应用程序端传入 HADB 的数据量。请使用下面的公式：

$$R_{dt} = n_{users} \cdot s \cdot NTPS$$

4 **确定节点数 N_{NODES}。**

请使用下面的公式：

$$N_{NODES} = V_{data} / 5GB$$

将该值舍入为偶数值，因为节点是成对使用的。

HADB 主机数

根据数据传输要求确定 HADB 主机数。此计算假定所有主机具有类似的硬件配置和操作系统，并且为它们运行的节点分配了所需的资源。

▼ 计算主机数

1 确定最大主机数据传输速率 $R_{max.}$。

根据经验确定该值，因为它是由网络和主机硬件决定的。请注意，它与上一节中确定的最大 HADB 数据传输速率 R_{dt} 是不同的。

2 确定包含此数据所需的主机数

更新一些主机 (N_{HOSTS}) 上分配的数据量 V 可使每个主机接收大约 $4V/N_{HOSTS}$ 的数据。请使用以下公式确定包含此数据量所需的主机数：

$$N_{HOSTS} = 4 \cdot R_{dt} / R_{max}$$

将该值舍入为最接近的偶数值，以使每个 DRU 具有相同数量的主机。

3 在每个 DRU 上添加一个主机作为备用节点。

如果其他的每个主机运行 N 个数据节点，则允许此主机运行 N 个备用节点。这可防范单个计算机出现故障而将 N 个数据节点关闭的情况。

每个主机需要运行至少一个节点，因此，如果节点数小于主机数 ($N_{NODES} < N_{HOSTS}$)，则调整 N_{NODES} 以使其等于 N_{HOSTS}。如果节点数大于主机数 ($N_{NODES} > N_{HOSTS}$)，则可以在同一个主机上运行多个节点。

HADB 存储容量

HADB 容量随节点数的增加呈近乎线性增长，直至超过网络容量为止。必须在一个或多个专用磁盘上为每个节点配置存储设备。必须在存储设备上为所有节点分配相同的容量空间。请确保在本地磁盘上分配存储设备。

假定预期大小的会话数据为 *x* MB。HADB 需将数据复制在镜像节点上，因此，它需要 2*x* MB 的存储容量。此外，HADB 使用索引来实现对数据的快速访问。两个节点需要额外的 2*x* MB 以用于索引，所需的总存储容量为 4*x*。因此，HADB 的预期存储容量要求是预期数据量的 4 倍。

考虑到以后要进行扩容以防止数据从 HADB 中丢失，您必须提供额外的存储容量以便联机进行升级，因为您可能需要在添加新节点后对数据进行重新分段。在这种情况下，数据设备上需要类似数量 (4*x*) 的额外空间。因此，预期的存储容量为预期数据量的 8 倍。

此外，HADB 将磁盘空间用作以下空间：

- 日志缓冲区的临时存储空间。此空间是日志缓冲区大小的 4 倍。日志缓冲区跟踪与数据相关的操作。日志缓冲区大小的默认值为 48 MB。

- 用于内部管理的空间。此空间占存储设备大小的 1%。

下表简要说明了 *x* MB 会话数据的 HADB 存储空间要求。

表 2–3 会话大小为 X MB 的 HADB 存储空间要求

条件	需要的 HADB 存储空间
在**不**需要联机的情况下添加或删除 HADB 节点。	4*x* MB +（4*日志缓冲区大小）+ 设备大小的 1%
在需要联机的情况下添加或删除 HADB 节点。	8*x* MB +（4*日志缓冲区大小）+ 设备大小的 1%

如果 HADB 使用的设备空间不足，它不会接受客户机的插入或更新数据请求。不过，它将接受删除操作。如果 HADB 使用的设备空间不足，它将返回错误代码 4593 或 4592，并在历史记录文件中写入相应的错误消息。有关这些消息的更多信息，请参见《Sun Java System Application Server 9.1 Error Message Reference》中的第 14 章 “HADB Error Messages”。

Message Queue 代理部署规划

Java 消息服务 (Java Message Service, JMS) API 是一种消息传送标准，它允许 J2EE 应用程序和组件创建、发送、接收和读取消息。并启用了松散耦合的可靠异步分布式通信。Sun Java System Message Queue（用于实现 JMS）与 Application Server 相集成，使您可以创建消息驱动 Bean (Message-Driven Bean, MDB) 之类的组件。

Sun Java System Message Queue (MQ) 使用**连接器模块**（也称为资源适配器）与 Application Server 集成在一起，此模块是由 J2EE 连接器体系结构规范 (J2EE Connector Architecture, JCA) 1.5 定义的。连接器模块是一种在 Application Server 中添加功能的标准方法。部署到 Application Server 上的 J2EE 组件使用通过连接器模块集成的 JMS 提供者交换 JMS 消息。默认情况下，JMS 提供者是 Sun Java System Message Queue，但如果愿意，您也可以使用不同的 JMS 提供者，只要它可以实现 JCA 1.5。

在 Application Server 中创建 JMS 资源将会在后台创建连接器资源。因此，每个 JMS 操作将调用连接器运行时并在后台使用 MQ 资源适配器。

除了使用资源适配器 API 以外，Application Server 还使用其他 MQ API 以提供与 MQ 更好的集成。这种紧密集成实现了如下功能：连接器故障转移、传出连接的负载平衡以及传入到 MDB 中的消息的负载平衡。这些功能可实现消息传送通信容错能力和高可用性。

多代理群集

MQ Enterprise Edition 支持使用多个互连的代理实例（称为**代理群集**）。使用代理群集的情况下，客户机连接将分布在群集的所有代理中。群集可以提供水平可伸缩性并提高可用性。

单个消息代理可扩展到大约 8 个 CPU，并为典型的应用程序提供足够的吞吐量。如果代理进程失败，将会自动重新启动该进程。但是，随着连接到代理的客户机数以及传送的消息数的增加，代理最终将超过诸如文件描述符数量和内存容量之类的限制。

通过在群集中设置多个代理而不是一个代理，您可以：

- 提供消息传送服务，而不管单个计算机上是否出现了硬件故障。
- 在执行系统维护时，最大限度地减少停机时间。
- 满足具有不同用户系统信息库的工作组的需要。
- 规避防火墙限制。

不过，具有多个代理并不能确保正在处理事务的代理发生故障时，可以在备用代理上继续处理这些事务。虽然 MQ 将与群集中的其他代理重新恢复连接，但会中断事务消息传送并回滚正在进行的事务。这不会影响用户应用程序，但会影响无法完成的事务。由于可以继续使用这些连接，因此可确保服务的故障转移。

因此，MQ 在群集中不支持高可用性持久性消息传送。如果在出现故障后重新启动代理，它将自动恢复并完成持久性消息的传送。持久性消息可以存储在数据库或文件系统中。不过，如果承载代理的计算机不能从硬盘故障中恢复，则可能会丢失消息。

具有 Sun Cluster Data Service for Sun Message Queue 的 Solaris 平台支持透明的持久性消息故障转移。此配置利用 Sun Cluster 的全局文件系统和 IP 故障转移功能提供真正的高可用性，此配置包含在 Java Enterprise System 中。

主代理和客户机同步

在多代理配置中，将在群集的所有代理上复制每个目的地。每个代理知道为所有其他代理上的目的地注册的消息使用方。因此，每个代理可以将消息从其自己直接连接的消息生成方路由到远程消息使用方，然后将消息从远程生成方传送到其自己直接连接的使用方。

在群集配置中，每个消息生成方直接连接的代理将路由该生成方发送给它的消息。因此，持久性消息是由消息的**主代理**存储和路由的。

每当管理员在代理上创建或销毁目的地时，此信息将会自动传播到群集中的所有其他代理。同样，每当在主代理中注册消息使用方时，或者使用方与其主代理断开连接时（显式断开、由于客户机或网络故障或其主代理出现故障），就会在整个群集中传播该使用方的相关信息。还会以类似方式将持久订阅的相关信息传播到群集中的所有代理。

配置 Application Server 以使用 Message Queue 代理

Application Server 的 Java 消息服务表示的是 Message Queue 的连接器模块（资源适配器）。您可以通过管理控制台或 `asadmin` 命令行实用程序管理 Java 消息服务。

MQ 代理（JMS 主机）在 Application Server 进程的单独 JVM 中运行。这允许多个 Application Server 实例或群集共享一组相同的 MQ 代理。

在 Application Server 中，*JMS* **主机**指的是 MQ 代理。Application Server 的 Java 消息服务配置包含了一个 JMS 主机列表（也称为 AddressList），其中列出了将使用的所有 JMS 主机。

使用管理控制台管理 JMS

在管理控制台中，可以使用 Java 消息服务节点为特定配置设置 JMS 属性。您可以设置诸如“重新连接时间间隔”和“重新连接尝试”之类的属性。有关更多信息，请参见《Sun Java System Application Server 9.1 管理指南》中的第 4 章“配置 Java 消息服务资源”。

Java 消息服务节点下的 JMS 主机节点中包含 JMS 主机列表。可以在该列表中添加和删除主机。对于每个主机，您可以设置主机名、端口号以及管理用户名和密码。默认情况下，JMS 主机列表中包含一个名为 "default_JMS_host" 的 MQ 代理，它表示的是与 Application Server 集成的本地 MQ 代理。

可以配置 JMS 主机列表以包含群集中的所有 MQ 代理。例如，要建立一个包含 3 个 MQ 代理的群集，请在 Java 消息服务中为每个代理添加一个 JMS 主机。Message Queue 客户机使用 Java 消息服务中的配置信息与 MQ 代理进行通信。

使用 asadmin 管理 JMS

除了管理控制台以外，您还可以使用 `asadmin` 命令行实用程序来管理 Java 消息服务和 JMS 主机。请使用以下 `asadmin` 命令：

- 配置 Java 消息服务属性：`asadmin set`
- 管理 JMS 主机：
  - `asadmin create-jms-host`
  - `asadmin delete-jms-host`
  - `asadmin list-jms-hosts`
- 管理 JMS 资源：
  - `asadmin create-jms-resource`
  - `asadmin delete-jms-resource`
  - `asadmin list-jms-resources`

    有关这些命令的更多信息，请参见《Sun Java System Application Server 9.1 Reference Manual》或相应的手册页。

Java 消息服务类型

Application Server 与 MQ 代理之间共有两种类型的集成：本地和远程。您可以在管理控制台的 Java 消息服务页中设置此类属性。

本地 Java 消息服务

如果 Type 属性为 LOCAL，Application Server 将启动和停止 MQ 代理。当 Application Server 启动时，它将启动指定为默认 JMS 主机的 MQ 代理。同样，在关闭 Application Server 实例时，它将关闭 MQ 代理。LOCAL 类型是适用于独立 Application Server 实例的最佳类型。

在类型为 LOCAL 的情况下，请使用“启动参数”属性指定 MQ 代理的启动参数。

远程 Java 消息服务

如果 Type 属性为 REMOTE，Application Server 将使用在外部配置的代理或代理群集。在这种情况下，您必须在 Application Server 之外启动和停止 MQ 代理，并使用 MQ 工具来配置和调整代理或代理群集。REMOTE 类型是适用于 Application Server 群集的最佳类型。

在类型为 REMOTE 的情况下，您必须使用 MQ 工具指定 MQ 代理启动参数。忽略“启动参数”属性。

缺省 JMS 主机

可以在管理控制台的 Java 消息服务页中指定默认 JMS 主机。如果 Java 消息服务类型为 LOCAL，则在 Application Server 实例启动时，Application Server 将启动默认 JMS 主机。

要使用 MQ 代理群集，请删除默认 JMS 主机，然后将群集中的所有 MQ 代理添加为 JMS 主机。在这种情况下，默认 JMS 主机将变为 JMS 主机列表中的第一个 JMS 主机。

也可以显式地将默认 JMS 主机设置为某个 JMS 主机。当 Application Server 使用 Message Queue 群集时，默认 JMS 主机将执行特定于 MQ 的命令。例如，为 MQ 代理群集创建物理目的地时，默认 JMS 主机将执行该命令来创建物理目的地，但群集中的所有代理将使用该物理目的地。

示例部署方案

为满足消息传送需求，请根据您的部署、性能和可用性需求修改 Java 消息服务和 JMS 主机列表。以下各节介绍了一些典型方案。

为获得最佳可用性，如果 Application Server 无法满足消息传送需求，请将 MQ 代理和 Application Server 部署到不同的计算机上。另一种方法是，在每台计算机上运行 Application Server 实例和 MQ 代理实例，直至有足够的消息传送能力。

默认部署

在安装 Application Server 时，将会自动创建域管理服务器 (Domain Administration Server, DAS)。默认情况下，DAS 的 Java 消息服务类型为 LOCAL。因此，在启动 DAS 时，还会启动其默认 MQ 代理。

在创建新的域时，还会创建新的代理。默认情况下，在域中添加独立服务器实例或群集时，会将其 Java 消息服务配置为 REMOTE，并且其默认 JMS 主机是由 DAS 启动的代理。

第 52 页中的 “默认部署”列举了一个 Application Server 群集中包含 3 个实例的默认部署示例 。

将 MQ 代理群集与 Application Server 群集结合使用

要配置 Application Server 群集以使用 MQ 代理群集，请在 Application Server 的 Java 消息服务中将所有 MQ 代理添加为 JMS 主机。然后，创建的任何 JMS 连接工厂和部署的 MDB 将使用指定的 JMS 配置。

下图展示了一个部署示例，其中代理群集中包含 3 个 MQ 代理，某一群集中包含 3 个 Application Server 实例。

指定特定于应用程序的 MQ 代理群集

在某些情况下，应用程序可能需要使用一个不同于 Application Server 群集所使用的 MQ 代理群集。第 53 页中的 “指定特定于应用程序的 MQ 代理群集”说明了这样一个示例方案。为此，请使用 JMS 连接工厂的 `AddressList` 属性或 MDB 部署描述符中的 `activation-config` 元素指定 MQ 代理群集。

有关配置连接工厂的更多信息，请参见《Sun Java System Application Server 9.1 管理指南》中的“JMS 连接工厂”。有关 MDB 的更多信息，请参见《Sun Java System Application Server 9.1 Developer’s Guide》中的“Using Message-Driven Beans”。

应用程序客户机

在应用程序客户机或独立应用程序首次访问 JMS 管理对象时，客户机 JVM 将从服务器中检索 Java 消息服务配置。只有重新启动客户机 JVM 后，客户机 JVM 才能获取有关 JMS 服务的其他更改。

◆ ◆ ◆ 第 3 章

3 选择拓扑

在按照第 1 章中的所述估计与性能相关的因素后，请为 Application Server 选择拓扑。拓扑是指计算机、Application Server 实例、HADB 节点以及这三者之间的通信流的排列方式。

共有两种基本的部署拓扑。两种拓扑具有一些通用的构件：群集中的多个 Application Server 实例、一组镜像 HADB 节点以及 HADB 备用节点。两种拓扑都需要一组通用的配置设置才能正常工作。

本章讨论了以下内容：

- 第 55 页中的 “通用要求”（适用于两种拓扑）。
- 两种拓扑：
  - 第 57 页中的 “同位拓扑” - Application Server 实例和 HADB 节点位于同一台计算机上。
  - 第 61 页中的 “分层拓扑” - Application Server 实例和 HADB 节点位于不同的计算机上。
- 第 65 页中的 “确定要使用的拓扑”

通用要求

本节介绍了两种拓扑的通用要求：

- 第 56 页中的 “一般要求”
- 第 56 页中的 “HADB 节点和计算机”
- 第 57 页中的 “负载平衡器配置”

一般要求

两种拓扑必须满足以下一般要求：

- 承载 HADB 节点的计算机必须成对存在。也就是说，计算机个数必须为偶数。
- 每个数据冗余单元 (Data Redundancy Unit, DRU) 必须具有相同数量的计算机。创建 HADB 数据库时，应确保镜像（成对）节点位于与主节点不同的 DRU 上。
- 每台承载 HADB 节点的计算机必须具备本地磁盘存储器，用于存储 HADB 中所有永久保留的信息。
- 承载 HADB 节点的计算机必须运行相同的操作系统。最好使用在配置和性能方面相同或几乎相同的计算机。
- 要在 HADB 中永久保留 HTTP 和 SFSB 会话信息，Application Server 实例必须位于群集中并满足所有相关要求。有关配置群集的更多信息，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的第 6 章 “使用 Application Server 群集”。
- 承载 Application Server 实例的计算机在配置和性能方面必须尽可能相同。这是因为负载平衡器插件使用循环策略进行负载平衡；如果使用不同类型的计算机来承载实例，则无法以最佳方式在这些计算机之间平衡负载。
- 最好为每个 DRU 配备单独的不间断电源 (Uninterruptible Power Supply, UPS)。

HADB 节点和计算机

每个 DRU 包含完整的 HADB 数据副本；如果一个 DRU 变得不可用，另一个 DRU 可以继续处理请求。但是，如果某个 DRU 中的节点及其在另一个 DRU 中的镜像同时出现故障，则会丢失一部分数据。为此，在设置系统时，请务必确保两个 DRU 不会受到单个故障的影响，例如电源故障或磁盘故障。

注 – 每个 DRU 必须在完全独立的冗余系统上运行。

在设置 HADB 节点和计算机时，请遵循以下准则：

- 要增加容量和吞吐量，应成对添加节点，每个 DRU 一个节点。
- 应为每个 DRU 设置一定数量的备用节点，该数量与每台计算机上运行的节点数相同。这是因为，如果该配置中的每台计算机运行 *n* 个数据节点，当一台计算机出现故障时，这 *n* 个节点也会随之关闭。
- 应在所有计算机上运行相同数量的 HADB 节点，以便尽可能均等地平衡负载。

注意 – 不要在同一台计算机上运行来自不同 DRU 的节点。如果必须在同一台计算机上运行来自不同 DRU 的节点，请确保计算机可以处理任何单点故障（与磁盘、内存、CPU、电源和操作系统崩溃等相关的故障）。

负载平衡器配置

两种拓扑均将 Application Server 实例放在群集中。这些实例将会话信息永久保留到 HADB 中。请将负载平衡器配置为包含群集中所有 Application Server 实例的配置信息。

有关建立群集以及在群集中添加 Application Server 实例的更多信息，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的第 6 章 “使用 Application Server 群集”。

同位拓扑

在同位拓扑中，Application Server 实例和 HADB 节点位于同一台计算机上（因此称为**同位**）。这种拓扑需要的计算机数量比分层拓扑少。同位拓扑可以更有效地使用 CPU：Application Server 实例和 HADB 节点位于同一台计算机上，并在它们之间平均分配处理负载。

这种拓扑最少需要两台计算机。要提高吞吐量，请成对添加计算机。

注 – 同位拓扑非常适于大型对称多重处理 (Symmetric Multiprocessing, SMP) 计算机，因为您可以充分利用这些计算机的处理能力。

示例配置

下图展示了一个示例同位拓扑配置。

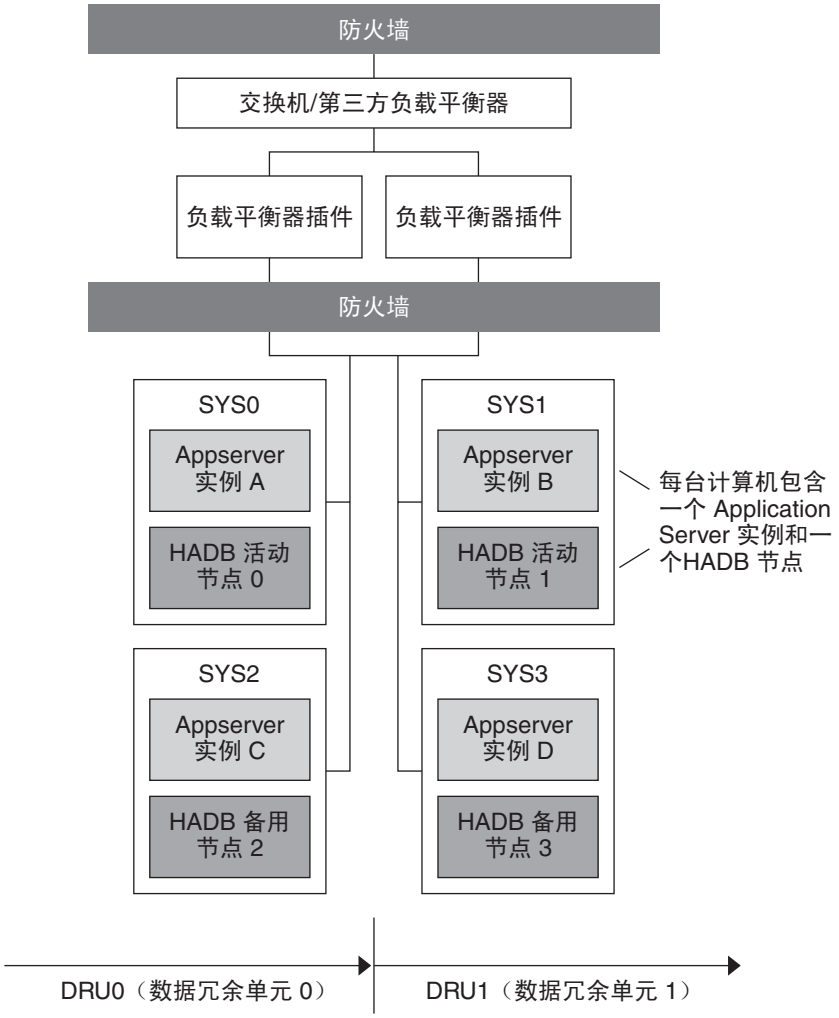

图 3–1 示例同位拓扑

计算机 SYS0 承载 Application Server 实例 A，计算机 SYS1 承载 Application Server 实例 B，计算机 SYS2 承载 Application Server 实例 C，计算机 SYS3 承载 Application Server 实例 D。

这四个实例组成一个群集，用于将信息永久保留到两个 DRU 上：

- **DRU0** 包含两台计算机，即 SYS0 和 SYS2。HADB 活动节点 0 位于计算机 SYS0 上。HADB 备用节点 2 位于计算机 SYS2 上。
- **DRU1** 包含两台计算机，即 SYS1 和 SYS3。HADB 活动节点 1 位于计算机 SYS1 上。HADB 备用节点 3 位于计算机 SYS3 上。

同位拓扑的变化形式

要获得更好的可伸缩性和更大的吞吐量，请添加更多的计算机以增加 Application Server 实例和 HADB 节点数。例如，您可以添加两台计算机，每台计算机包含一个 Application Server 实例和一个 HADB 节点。请确保成对添加 HADB 节点，并为每个 DRU 分配一个节点。第 59 页中的 “同位拓扑的变化形式”对这种配置进行了说明。

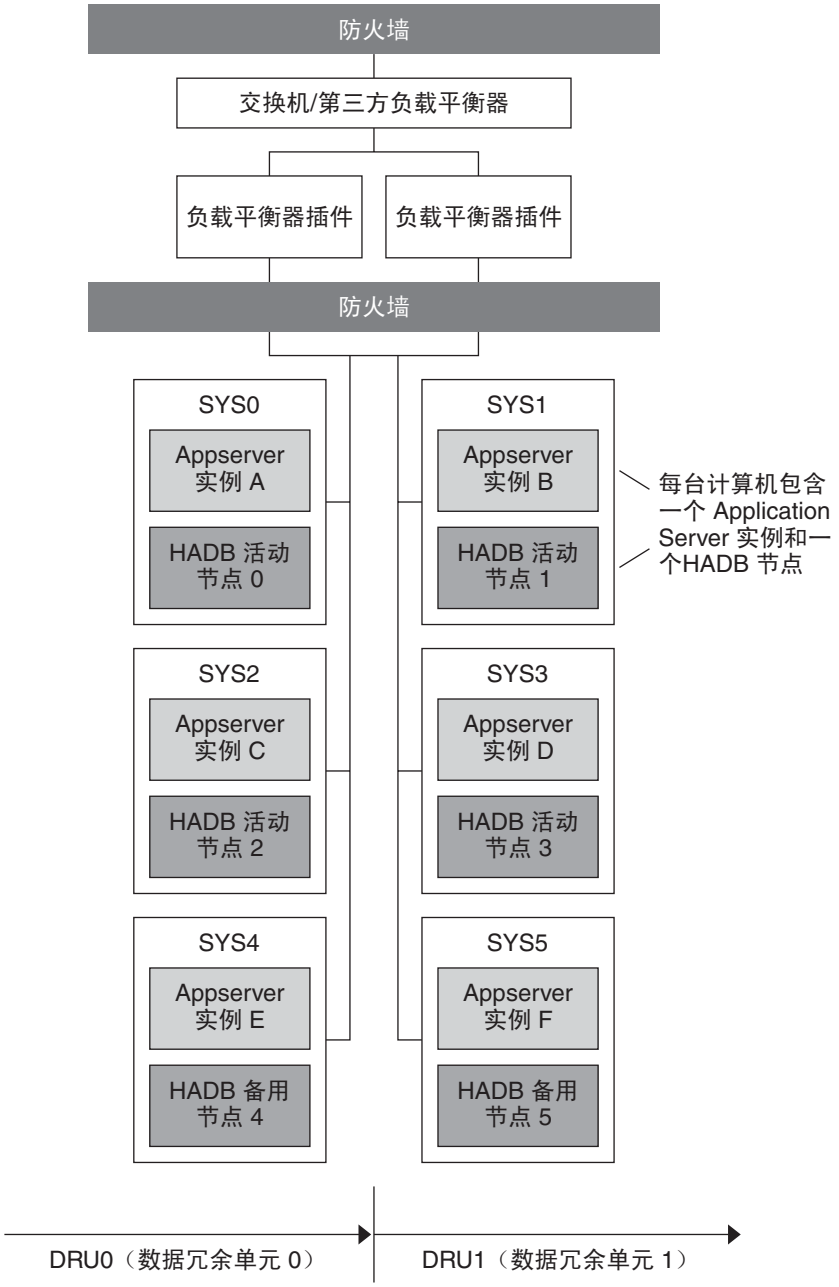

图 3–2 同位拓扑的变化形式

在这种变化形式中，在第 57 页中的 “示例配置”所述的同位拓扑中添加了计算机 SYS4 和 SYS5。

Application Server 实例的承载方式如下：

- 计算机 SYS0 承载实例 A
- 计算机 SYS1 承载实例 B
- 计算机 SYS2 承载实例 C
- 计算机 SYS3 承载实例 D
- 计算机 SYS4 承载实例 E
- 计算机 SYS5 承载实例 F

这些实例组成一个群集，用于将信息永久保留到两个 DRU 上：

- **DRU0** 包含计算机 SYS0、SYS2 和 SYS4。HADB 活动节点 0 位于计算机 SYS0 上。HADB 活动节点 2 位于计算机 SYS2 上。HADB 备用节点 4 位于计算机 SYS4 上。
- **DRU1** 包含计算机 SYS1、SYS3 和 SYS5。HADB 活动节点 1 位于计算机 SYS1 上。HADB 活动节点 3 位于计算机 SYS3 上。HADB 备用节点 5 位于计算机 SYS5 上。

分层拓扑

在这种拓扑中，Application Server 实例和 HADB 节点位于不同的计算机上（因此称为**分层**）。

这种拓扑需要的硬件比同位拓扑多。如果使用不同类型的计算机，这种拓扑可能会比较适合：您可以分配一组计算机来承载 Application Server 实例，而分配另一组计算机来承载 HADB 节点。例如，可以为 Application Server 实例配备功能较强的计算机，而为 HADB 配备功能次之的计算机。

示例配置

下图展示了分层拓扑。

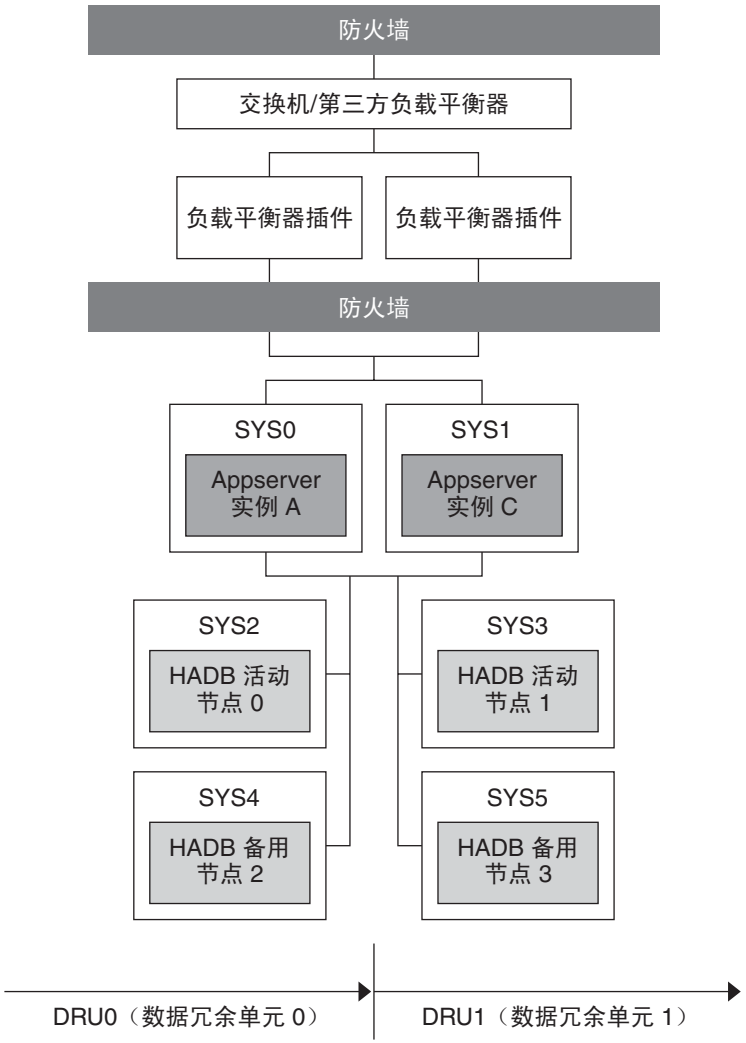

图 3–3 示例分层拓扑

在这种拓扑中，计算机 SYS0 承载 Application Server 实例 A，计算机 SYS1 承载 Application Server 实例 B。这两个实例组成一个群集，用于将会话信息永久保留到两个 DRU 上：

- **DRU0** 包含两台计算机，即 SYS2 和 SYS4。HADB 活动节点 0 位于计算机 SYS2 上，HADB 备用节点 2 位于计算机 SYS4 上。
- **DRU1** 包含两台计算机，即 SYS3 和 SYS5。HADB 活动节点 1 位于计算机 SYS3 上，HADB 备用节点 3 位于计算机 SYS5 上。

DRU 上的所有节点位于不同的计算机上，因此，即使一台计算机出现故障，仍可在其他计算机上获得任何 DRU 的完整数据。

分层拓扑的变化形式

分层拓扑的一种变化形式是，在配置中水平添加更多的计算机以增加 Application Server 实例数。例如，通过创建一个新 Application Server 实例，在示例配置中添加另一台计算机。类似地，通过添加更多的计算机来承载 HADB 节点以增加 HADB 节点数。需要重申的是，必须成对添加 HADB 节点，每个 DRU 一个节点。

第 63 页中的 “分层拓扑的变化形式”对这种配置进行了说明。

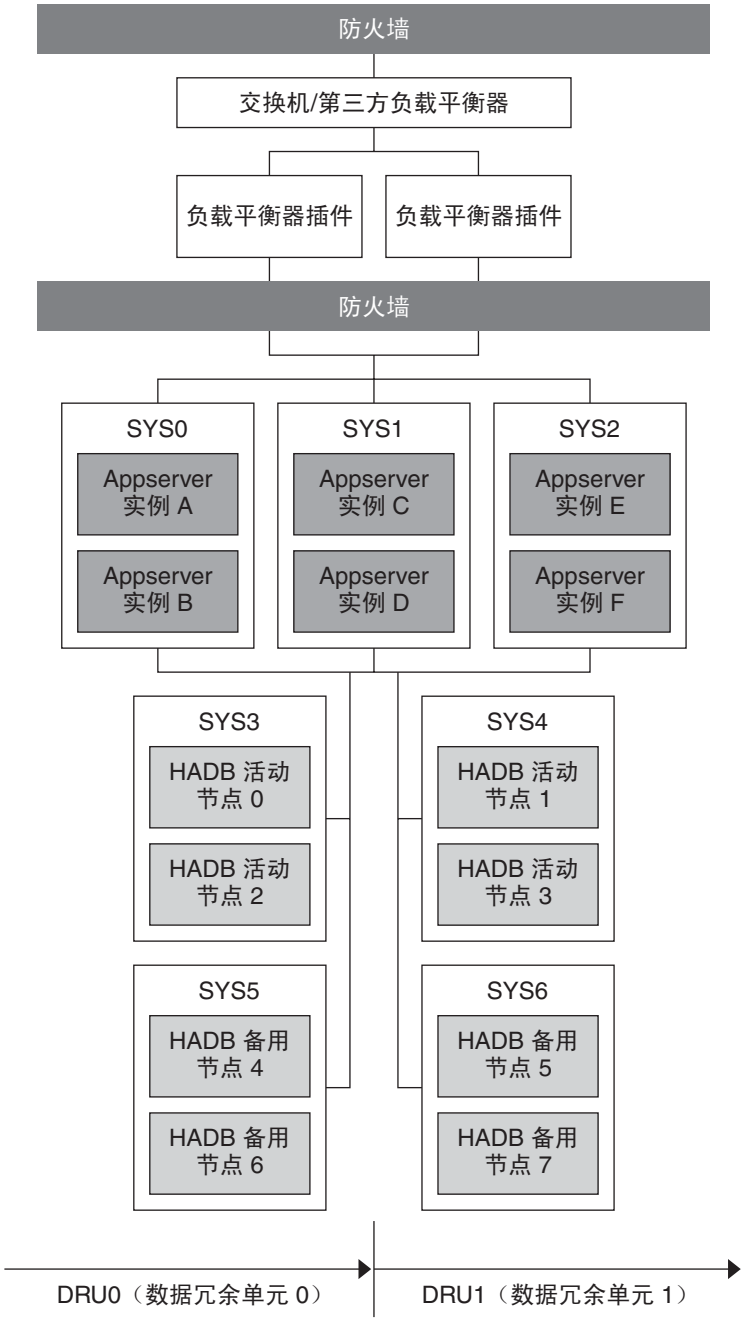

图 3–4 分层拓扑的变化形式

在这种配置中，每台承载 Application Server 实例的计算机包含两个实例。因此，群集中共有六个 Application Server 实例。

HADB 节点位于计算机 SYS3、SYS4、SYS5 和 SYS6 上。

DRU0 包含两台计算机：

- SYS3，用于承载 HADB 活动节点 0 和 HADB 活动节点 2。
- SYS5，用于承载 HADB 备用节点 4 和 HADB 备用节点 6。

DRU1 包含两台计算机：

- SYS4，用于承载 HADB 活动节点 1 和 HADB 活动节点 3。
- SYS6，用于承载 HADB 备用节点 5 和 HADB 备用节点 7。

承载 HADB 节点的每台计算机承载两个节点。因此，共有八个 HADB 节点：四个活动节点和四个备用节点。

确定要使用的拓扑

要确定最符合性能和可用性要求的拓扑（或变化形式），请对拓扑进行测试并试验不同的计算机和 CPU 组合。

确定为满足目标而需要做出的折衷处理。例如，如果维护的简便性对您至关重要，则分层拓扑更适合一些。但不利的一面是，这种拓扑需要的计算机比同位拓扑多。

在选择拓扑时，一个重要因素是可用的计算机类型。如果系统包含大型对称多重处理 (Symmetric Multiprocessing, SMP) 计算机，则同位拓扑更具吸引力，因为您可以充分利用这些计算机的处理能力。如果系统包含不同类型的计算机，则分层拓扑会更有用，因为您可以为 Application Server 层和 HADB 层分配不同的计算机组。例如，您可能希望将功能较强的计算机用于 Application Server 层，而将功能次之的计算机用于 HADB 层。

拓扑比较

下表对同位拓扑和分层拓扑进行了比较。左列显示的是拓扑名称，中间列显示的是拓扑优点，右列显示的是拓扑缺点。

表 3–1 拓扑比较

拓扑	优点	缺点
同位拓扑	需要较少的计算机。由于 HADB 节点和 Application Server 实例位于同一层中，因此，您可以在每个备用节点上创建 Application Server 实例以处理额外负载。 提高了 CPU 利用率。在位于同一台计算机上的 Application Server 实例和 HADB 节点之间平均分配处理负载。 对于大型对称多重处理 (Symmetric Multiprocessing, SMP) 计算机非常有用，因为它可充分利用这些计算机的处理能力。	增加了维护复杂性。例如，当需要关闭承载 HADB 节点的计算机以执行维护时，该计算机上的应用服务器实例也会变得不可用。
分层拓扑	易于维护。例如，无需关闭 HADB 节点便可在承载 Application Server 实例的计算机上执行维护。 对于不同类型的计算机非常有用。可以为 Application Server 层和 HADB 层分配不同的计算机组。例如，可以将功能较强的计算机用于 Application Server 层，而将功能次之的计算机用于 HADB 层。	需要的计算机比同位拓扑多。由于应用服务器实例和 HADB 节点位于不同的层中，因此，应用服务器实例不能位于承载 HADB 备用节点的计算机上。 降低了 CPU 利用率。应用服务器层和 HADB 层的负载可能不均等。当计算机数较少时（4 到 6 台），此缺点会更加显著。

◆ ◆ ◆ 第 4 章

部署核对表

本附录提供了一个核对表，可借助此表来着手评估并在生产环境中使用 Application Server。

部署核对表

表 4–1 核对表

组件/功能	说明
应用程序	确定要部署的应用程序的以下要求： ■ 所需/可接受的响应时间。 ■ 峰值负载特性。 ■ 所需的持久性范围和频率。 ■ `web.xml` 中的会话超时。 ■ 故障转移和可用性要求。 有关更多信息，请参见《Sun Java System Application Server 9.1 Performance Tuning Guide》。
硬件	■ 使用相同类型的硬件来承载 HADB 节点。 ■ 拥有所需的硬盘空间和内存量。 ■ 通过调整大小来确定部署要求。 有关更多信息，请参见《Sun Java System Application Server 9.1 发行说明》。
操作系统	■ 确保将产品安装在支持的平台上。 ■ 确保修补级别是最新且准确的。 有关更多信息，请参见《Sun Java System Application Server 9.1 发行说明》。

表 4–1 核对表 （续）

组件/功能	说明
网络基础设施	■ 确定单点故障并对其进行修复。 ■ 确保正确配置了 NIC 和其他网络组件。 ■ 运行 `ttcp` 基准测试，以确定吞吐量是否符合要求/预期结果。 ■ 根据偏好安装 `rsh/ssh`，以便正确安装 HADB 节点。 有关更多信息，请参见《Sun Java System Application Server 9.1 Installation Guide》。
后端和其他外部数据源	与相关领域专家或供应商进行核实，以确保这些数据源配置正确。
系统更改/配置	■ 在运行任何性能/压力测试之前，请确保对 `/etc/system` 及其在 Linux 上的等效文件的更改已完成。 ■ 确保对 TCP/IP 设置的更改已完成。 ■ 默认情况下，系统上预配置了许多服务。并非所有服务都需要运行。请关闭不需要的服务以节省系统资源。 ■ 在 Solaris 上，可使用 `Setoolkit` 来确定系统的行为。解析显示的所有标志。 有关更多信息，请参见《Sun Java System Application Server 9.1 Performance Tuning Guide》。
Application Server 和 HADB 安装	■ 确保不要将这些服务器安装在 NFS 挂载卷中。 ■ 在同一台计算机上安装 Application Server 和 HADB 节点时，请检查是否拥有足够的磁盘空间和 RAM。 ■ 在同一系统上安装多个 HADB 节点时，请检查是否拥有足够的独立磁盘。 有关更多信息，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的第 2 章 “安装和设置高可用性数据库”。
HADB 配置	■ 设置 HADB 数据设备的大小。 ■ 定义 DataBufferPoolSize。 ■ 定义 LogBufferSize。 ■ 定义 InternalBufferSize。 ■ 设置 NumberOfLocks。 ■ 为各种 Application Server 组件设置最佳的超时值。 ■ 在文件系统上创建 HADB 节点的物理布局。 有关更多信息，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的“配置 HADB”。

表 4–1 核对表 （续）

组件/功能	说明
Application Server 配置	■ 日志记录：启用访问日志轮转。 ■ 选择正确的日志记录级别。通常会使用 "WARNING"。 ■ 使用管理控制台配置 J2EE 容器。 ■ 使用管理控制台配置 HTTP 侦听器。 ■ 使用管理控制台配置 ORB 线程池。 ■ 如果使用包含本机代码的 Type2 驱动程序或调用，请确保在 LD_LIBRARY_PATH 中指定了 mtmalloc.so。 ■ 确保使用适当的持久性范围和频率，并且在各个 Web/EJB 模块中没有将它们覆盖。 ■ 确保仅对 SFSB 中的关键方法执行检查点操作。 有关性能调节的更多信息，请参见《Sun Java System Application Server 9.1 Performance Tuning Guide》。 有关配置的更多信息，请参见《Sun Java System Application Server 9.1 管理指南》。
负载平衡器配置	■ 确保安装了 Web Server。 ■ 确保安装了 Web Server 的负载平衡器插件。 ■ 确保禁用了修补程序检查。 ■ 降低 `KeepAliveQuery` 参数的值。该值越小，负载较低的系统上的延迟时间越短。该值越大，负载较高的系统上的吞吐量越大。 有关更多信息，请参见《Sun Java System Application Server 9.1 Performance Tuning Guide》中的“Keep Alive”。
Java 虚拟机配置	■ 最初，将最小和最大堆大小设置为相同的值，每个实例至少为 1 GB。 ■ 有关更多信息，请参见 Java Hotspot VM Options（Java Hotspot VM 选项）。 ■ 在运行多个 Application Server 实例时，请考虑创建一个处理器集，并将 Application Server 绑定到该处理器集。如果使用 CMS 收集器清除以前的内容，这是非常有用的。

表 4–1 核对表 （续）

组件/功能	说明
在负载平衡器中配置超时	■ Response-time-out-in-seconds - 负载平衡器在声明 Application Server 实例运行不正常之前等待的时间。根据应用程序的响应时间设置该值。如果设置的值太大，Web Server 和负载平衡器插件将等待很长的时间，才会将 Application Server 实例标记为运行不正常。如果设置的值太小，并且 Application Server 的响应时间超过该阈值，则会将该实例错误地标记为运行不正常。 ■ Interval-in-seconds - 检查运行不正常的实例是否恢复为正常状态之前等待的时间（以秒为单位）。如果该值太小，则会产生从负载平衡器插件到 Application Server 实例的额外通信；如果该值太大，则会延迟向已恢复正常状态的实例路由请求。 ■ Timeout-in-seconds - 为运行状况检查请求获取响应的持续时间。请根据群集中的各个系统的通信情况调整该值，以确保运行状况检查成功完成。 有关更多信息，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的第 5 章 “配置 HTTP 负载平衡”。

表 4–1 核对表 （续）

组件/功能	说明
在 HADB 中配置超时	■ sql_client_timeout - SQLSUB 等待空闲客户机的时间。例如，已登录、发送了一些请求后等待用户输入的客户机。假定空闲时间超过 30 分钟的客户机已停用并终止会话。如果设置的值太小，则可能会提前终止 SQL 会话。如果设置的值太大，则可能会导致未处于空闲状态但已退出的 SQL 会话占用资源。而这又会阻止其他 SQL 客户机进行登录。在调整此变量时，还要考虑 `nsessions` 设置。如果 HADB JDBC 连接池的 steady-pool-size 大于 max-pool-size，则可以将 idle-timeout-in-seconds 设置为小于 sql_client_timeout，以便 Application Server 在 HADB 关闭连接之前自行将其关闭。默认值为 1800 秒。 ■ lock_timeout - 事务等待访问数据的最长时间（以毫秒为单位）。如果超过该时间，事务将生成以下错误消息：“事务超时”。此类超时是由于事务等待其他事务所保留的锁定（死锁）引起的，这会产生很高的服务器负载。不要将该值设置为低于 500 毫秒。如果在服务器日志中看到“事务超时”消息，请增大该值。可通过在 HADB 的 JDBC 连接池中添加属性来设置锁定超时值，如下所示：`<property name=lockTimeout value="x"\>`。默认值为 5000 毫秒。 ■ Querytimeout - HADB 等待执行查询的最长时间（以毫秒为单位）。如果服务器日志中不断出现指明查询超时的异常，请考虑增大该值。可通过在 HADB 的 JDBC 连接池中添加以下属性来设置该值：`<property name=QueryTimeout value="x"\>`。默认值为 30 秒。 ■ loginTimeout - 客户机等待登录到 HADB 的最长时间（以秒为单位）。可通过在 HADB 的 JDBC 连接池中添加以下属性来设置该值：`<property name=loginTimeout value="x"\>`。默认值为 10 秒。 ■ MaxTransIdle - 在向客户机发送回复和接收下一个请求之间，事务可以处于空闲状态的最长时间（以毫秒为单位）。可通过在 HADB 的 JDBC 连接池中添加属性来更改该值，如下所示：`<property name=maxtransIdle value="x"\>`。默认值为 40 秒。 有关更多信息，请参见：《Sun Java System Application Server Performance Tuning Guide》。
在 Application Server 中配置超时	■ Max-wait-time-millis - 在抛出异常之前等待从池中获取连接的时间。默认值为 6 秒。对于要保留的数据大小超过 50 KB 的高负载系统，请考虑更改该值。 ■ Cache-idle-timeout-in-seconds - 在 EJB 钝化之前允许其在高速缓存中处于空闲状态的时间。仅适用于实体 Bean 和有状态会话 Bean。 ■ Removal-timeout-in-seconds - EJB 保持钝化状态的时间（在备份存储中处于空闲状态）。默认值为 60 分钟。请根据 SFSB 故障转移需求调整该值。 在调整所有这些值时，应考虑到 HADB 的 JDBC 连接池设置 max-wait-time-in-millis。有关更多信息，请参见《Sun Java System Application Server 9.1 高可用性管理指南》中的“配置 JDBC 连接池”。

表 4–1 核对表 （续）

组件/功能	说明
调整 VM 垃圾收集 (Garbage Collection, GC)	如果垃圾收集暂停 4 秒或更长时间，则可能会导致在 HADB 中保留会话状态的过程中出现时断时续的问题。为避免出现此问题，请调整 VM 堆。如果在保留数据时不容许出现丝毫故障，或者系统并未满负荷运行，请使用 CMS 收集器或吞吐量收集器。 可通过添加以下内容来启用这些收集器： `<jvm-options>-XX:+UseConcMarkSweepGC</jvm-options>` 此选项可能会降低吞吐量。

索引