Sun Java™ System

Sun Java Enterprise System
部署规划白皮书

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

文件号码：817-7594

目录

图形列表

表格列表

第 1 章

部署规划简介

本白皮书介绍如何基于 Sun Java™ Enterprise System 规划大规模部署。它提出了部署规划的一些基本概念和原理，并介绍了若干过程，可以此作为设计企业范围部署时的起点。

如果正要对 Java Enterprise System 进行评估或计划创建和部署基于 Java Enterprise System 的大规模应用程序，便可以本白皮书作为部署规划过程的指南。

本章对 Java Enterprise System 做了简要概述，并对将在以后各章予以阐述的有关部署规划的概念进行了介绍。本章包括以下节：

关于 Java Enterprise System

Java Enterprise System 是一种软件框架结构，为分布于网络或 Internet 环境中的企业级应用程序提供各种服务。下表列出了 Java Enterprise System 的各个组件及其各自所提供的框架结构服务。

表 1-1 Java Enterprise System 组件

系统组件	提供的服务
Application Server	为 Enterprise JavaBeans™ (EJB) 组件 （如会话 bean、实体 bean 和消息驱动 bean）提供各种 Java 2 Platform, Enterprise Edition (J2EE™ platform) 容器服务。由于容器可以提供紧耦合的分布式组件进行交互所需的各种框架结构服务，因而使其成为电子商务应用程序和 web 服务的开发和执行平台。Application Server 还提供各种 web 容器服务。
Calendar Server	向最终用户和最终用户组提供各种日历和调度服务。Calendar Server 自带可与服务器进行交互的基于浏览器的客户端。
Directory Proxy Server	从公司防火墙外为 Directory Server 提供各种安全服务。Directory Proxy Server 可提供增强型目录存取控制、模式兼容性、路由选择及多 Directory Server 实例间的负载平衡。
Directory Server	提供用于存储和管理内联网和 Internet 信息的中心信息库，这些信息包括身份配置文件 （员工、客户、供应商等）、用户凭证 （公共密钥证书、口令和个人识别号）、存取权限、应用程序资源信息及网络资源信息等。
Identity Server	提供各种存取管理和数字身份管理服务。存取管理服务包括存取应用程序和 / 或服务所需的验证 （包括单点登录）和基于角色的授权。管理服务包括对个人用户配置文件、角色、组及策略的集中化管理。
Instant Messaging	为最终用户提供安全、实时的通信，如即时消息传送 （聊天）、会议、警报、新闻、调查和文件传输。该服务自带一个在线管理器，可告知用户目前在线的人员；还自带一个基于浏览器的客户端，通过它可与服务器进行交互。

表 1-1 Java Enterprise System 组件 *(续)*

系统组件	提供的服务
Message Queue	在松散耦合的分布式组件和应用程序间提供可靠的异步消息传送。 Message Queue 实现了 Java 消息服务 (JMS) API 规范，并增加了安全性、可伸缩性和远程管理等企业功能。
Messaging Server	提供安全、可靠的大容量消息存储和转发服务，支持电子邮件、传真、寻呼、语音及视频。它可同时存取多个消息存储，并提供了内容过滤功能，可协助拒绝垃圾电子邮件和防止病毒攻击。
Portal Server	为需要存取业务应用程序或服务的基于浏览器的客户端提供内容集合和个性化等关键性门户服务。 Portal Server 还提供了可配置的搜索引擎。
Secure Remote Access	提供从企业防火墙外对 Portal Server 内容和服务 （包括内部门户和 Internet 应用程序）进行安全的 Internet 存取。
Web Server	为 Java web 组件 （如 Java Servlet 和 JavaServer Pages™ (JSP™) 组件）提供各种 J2EE 平台 web 容器服务。 Web Server 还支持可提供静态和动态 web 内容的其他 web 应用技术，如 CGI 脚本和 Active Server Pages。
Sun Cluster	为 Java Enterprise System 提供各种高可用性和可伸缩性服务，提供运行于 Java Enterprise System 框架结构之上的应用程序及部署这些服务和应用程序的硬件环境。

Java Enterprise System 服务套件

Java Enterprise System 部署通常分为两大类，即主要由 Java Enterprise System 提供的各种服务所组成的部署和集成了大量定制开发服务及第三方应用程序的部署。可将前者视作一种 *80:20 部署*（80% 的服务由 Java Enterprise System 提供）；同样，可将后者视作一种 *20:80 部署*。

注 实际企业部署所需的定制开发服务的数量会有很大差异。

Java Enterprise System 因拥有丰富的服务集而特别适用于进行 80:20 部署。例如，使用它进行企业范围通信系统或门户系统部署会相对容易。

而对于需要定制开发的部署， Java Enterprise System 提供了创建和集成定制开发服务和应用程序的功能。

下表将 Java Enterprise System 组件组合成可提供企业部署的各种套件。某些组件出现在不止一个套件中。

表 1-2 Java Enterprise System 服务套件

套件	Java Enterprise System 组件
网络身份管理服务	Identity Server Directory Server Web Server
企业门户服务	Portal Server Secure Remote Access Identity Server Directory Server Application Server 或 Web Server
企业通信和协作服务	Messaging Server Calendar Server Instant Messaging Identity Server Directory Server Application Server 或 Web Server
Web 和应用程序服务	Application Server Message Queue Web Server
可用性服务	Sun Cluster 3.1 4/04 Sun Cluster Agents

上面表 1-2 中的大多数套件均可提供 80:20 型部署。例如，可利用 “企业通信和协作”套件创建一个部署来为最终用户提供电子邮件、日历和即时消息传送服务，使用户能够集合和个性化内容。同样，可利用 “网络身份管理”和 “企业门户”套件安装和配置企业范围应用程序，而不必开发或集成各种定制服务。

“可用性服务”套件提供了进行大规模企业应用程序部署所需的高可用性。如果企业应用程序需要定制开发可在应用程序服务器或 web 服务器上运行的 J2EE 平台服务，请使用 “Web 和应用程序服务”套件。

由于各 Java Enterprise System 服务间可进行互操作，因此可根据企业的具体需要自行创建服务套件。

Java Enterprise System 的优点

成功的企业部署需具备三个关键要素。

- 交付时间
- 交付成本
- 功能

Java Enterprise System 提供了满足上述成功关键要素所需的工具，如下表所述。

表 1-3 Java Enterprise System 优点

优点	说明
易用	Java Enterprise System 提供通用安装程序，方便了安装、配置和升级。 用户可通过 Java Enterprise System 实现一种转变，即从原来的集成独立开发的单一产品和中间件转为采用包含各种集成平台服务的系统，这种系统的部署和配置几乎不需要进行定制。

表 1-3 Java Enterprise System 优点 *(续)*

优点	说明
可预知	Java Enterprise System 的发行周期考虑了各 Java Enterprise System 组件间的兼容性。升级到新版本时，不会出现组件间不兼容和不一致的情况。 Java Enterprise System 组件使用了一组共享平台组件，简化了各服务间的互操作。 Java Enterprise System 的定期发布交付模式使部署规划具有可预知性。
价格合理	Java Enterprise System 针对商业许可证的单件定价模式降低了部署安装和升级的复杂性和成本。单件价格包括支持、维护和咨询服务。 它还提供了针对 OEM 和教育许可证的定价模式。

关于部署规划

成功的部署规划来自各个阶段的认真准备、分析和设计，如下图所示。

图 1-1 部署规划的各个阶段

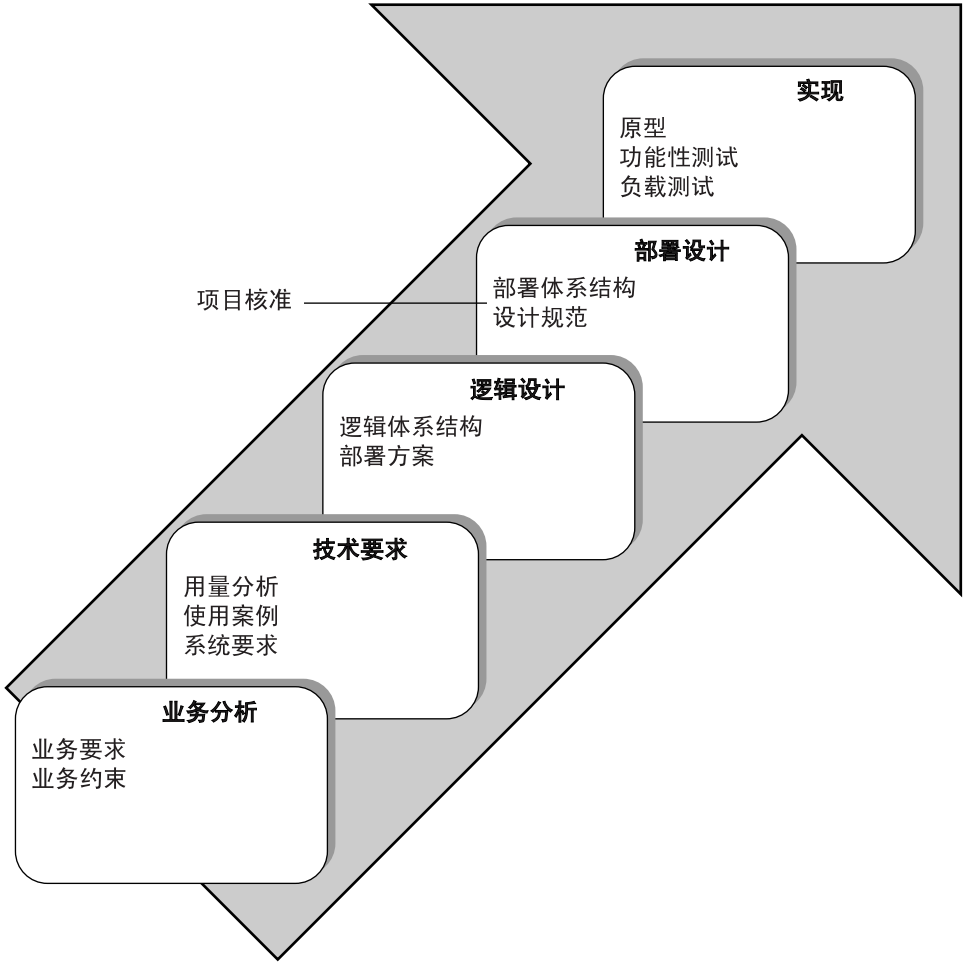


图 1-1 中所描述的每个阶段都有其各自的一套分析和程序，并通过它们将规范和设计推进到各后续阶段。本章以下各节对每个部署规划阶段做了摘要说明。

业务分析阶段

业务分析阶段的任务是，确定部署项目的业务目标和阐述实现该目标所必须满足的业务要求。阐述业务要求时，应将可能会对业务目标的实现能力产生影响的所有业务约束考虑在内。业务分析阶段结束时会形成一份*业务要求文档*，在后面的技术要求阶段会用到该文档，并会将其作为以后衡量部署设计是否成功的标准。

有关业务分析阶段的详细信息，参见第 21 页的第 2 章 “业务分析”。

技术要求阶段

技术要求阶段以业务分析阶段中形成的业务要求为起点，任务是将这些要求转化为可用来设计部署体系结构的技术规范。在技术要求阶段要准备下列信息：

- 用户任务和使用模式分析
- 模拟用户与计划部署间交互的使用案例
- 源自业务要求的系统要求，将用户任务和使用模式的分析结果考虑在内

使用分析、*使用案例*和*系统要求文档*所产生的信息集将作为逻辑设计阶段的信息来源。

在技术要求分析阶段，可能还要指定*服务级别要求*，该要求是指一些条件，满足这些条件时必须提供客户支持来修正部署系统故障，以满足系统要求。服务级别要求是项目核准阶段所签署的*服务级别协议*的基础。

有关技术要求阶段的详细信息，参见第 27 页的第 3 章 “技术要求”。

逻辑设计阶段

部署设计始于逻辑设计阶段。此阶段的任务是设计一个逻辑体系结构，它应该体现能够满足技术要求阶段所确定的使用案例的各种 Java Enterprise System 服务和依赖性。

逻辑体系结构与系统要求文档共同构成了*部署方案*的特性。逻辑体系结构并不指定实现部署方案实际所需的硬件。

有关逻辑设计阶段的详细信息，参见第 41 页的第 4 章 “设计逻辑体系结构”。

部署设计阶段

部署设计阶段的任务是，创建一个反映部署方案与物理环境的映射关系的*部署体系结构*。物理环境是指部署的网络框架结构，它包括计算节点、每个节点的硬件要求、防火墙以及网络上的其他设备。

映射过程包括两节：部署估量，用于指定满足系统要求实际所需的硬件；策略确定，确定优化部署体系结构以满足预算因素的策略。

部署项目核准通常在部署体系结构创建后进行。*项目核准*阶段的任务是，评估部署成本，核准后签署部署实现合同及获得项目构建所需的资源。

部署设计阶段的另一项任务是制订详细的*设计规范*。设计规范提供实现部署体系结构所需的详细信息，如实际需要的硬件、操作系统、网络设计及物理环境的其他方面。详细的设计规范还包括指定*置备用户*存取系统服务所需的目录服务数据结构。设计规范的制订可在项目核准前或核准后进行，具体何时进行取决于部署项目的过程和策略。

有关部署设计阶段的详细信息，参见第 51 页的第 5 章 “设计部署体系结构”。

实现阶段

实现阶段的任务是拓展部署体系结构。此阶段包括以下节或全部步骤，具体包含的步骤取决于部署项目的性质：

- 在测试环境中创建和部署试验性和/或原型部署
- 设计和运行功能性测试来衡量与系统要求的符合度
- 设计和运行负载测试来衡量峰值负载下的性能
- 创建生产部署，可能需要分阶段部署到生产中

部署进入生产阶段后，仍需继续对其进行监视、测试和调整，确保其能够实现业务目标。

有关实现阶段的详细信息，参见第 73 页的第 6 章 “实现部署设计”。

第 2 章

业务分析

本章介绍了有关如何分析业务问题、确定其业务要求和约束及清晰阐明业务目标的一些原则。

业务分析的第一步是说明部署项目的业务目标，接下来是分析必须解决的业务问题和确定实现业务目标所必须满足的*业务要求*。还应考虑会限制目标实现能力的任何*业务约束*。所确定的业务要求和约束是在后来的技术要求阶段用于得出系统要求的*业务要求文档*的基础。

在确定业务要求上无简单公式可循，只能根据与客户的合作情况及自身所掌握的有关该业务领域的知识来确定业务要求。此处介绍的原则提供了一种可以用于开始业务分析的方法。

本章包括以下节：

业务要求

业务问题说明与项目综合摘要非常相似，其中对项目的最终目标进行了概述。业务问题说明的内容是，设定项目的业务情境 （项目的必要性或可取性）和界定项目的范围 （项目范围之内和之外的内容），还需确定对项目成功起决定性作用的项目特性。

业务要求分析的结果应当是一份详细说明部署如何满足业务目标需要的文档。下表列出了业务要求分析期间通常会涉及的主题。

表 2-1 业务要求分析主题

主题	说明
业务目标	明确阐述项目的目标。对目标有清晰的理解有助于突出设计决策的重心。 以下是几个目标示例： • 企业协作，包括消息传送、通讯录、即时消息传送及日历服务等功能 • 企业门户，允许用户集合和个性化内容和使用电子邮件、日历、即时消息传送及其他企业服务 • 企业资源调度程序，用于安排会议室、办公室及其他共享物理资源 • 实现在线商务 将计划部署的目标与当前业务进行对比有助于后期设计决策的确定。
部署类型	确定预想的是以下哪一部署类型： • 企业对客户 • 企业对员工 • 企业对企业 • 企业员工间通信 • 上述类型的某种组合 理解部署类型有助于突出类型所固有的特定设计问题。
范围	明确阐述项目范围。确保所确定的领域能够得到解决，并避免做使目标不明晰或无法达到的 “开放式”阐述。 范围定义不明确会导致部署设计不能充分满足业务需要。
风险承担者	确定能够从部署成功中获取既得利益的个人和组织。 所有风险承担者均应积极参与业务目标和要求的确定。
关键特质	确定成功所必需的关键领域。这方便了根据最重要的标准进行设计分析。

表 2-1 业务要求分析主题 *(续)*

主题	说明
目标用户	确定部署所面向的用户类型。例如： • 当前和先前员工 • 活动客户 • 成员站点 • 公众 • 管理员
用户益处	阐述预期可为部署用户带来的益处。例如： • 远程存取公司资源 • 企业协作 • 缩短响应时间 • 降低错误率 • 简化日常任务 • 使远程团队得以共享资源 • 提高生产率 明确阐述预期益处有助于推动设计决策。
服务级别协议	定义部署未能满足特定系统要求时必须提供的客户支持级别和程度。 通常，服务级别协议根据在技术要求分析期间定义的服务级别要求在项目核准时签署。
安全问题	先前确定的目标可能存在隐性的安全问题，这些问题不必在问题阐述中列出。不过，突出对部署至关重要的某些安全目标可能会有益处。例如： • 存取授权用户所有的专有信息 • 以基于角色的方式存取机密信息 • 远程位置间的安全通信 • 在本地系统上调用远程应用程序 • 与第三方企业的安全交易
优先级	阐述目标的优先级。 大规模、复杂的部署可能需要分阶段实现。受资源限制，可能需要取消或修改某些目标。为使部署设计能够通过验收，可能需要作出某些决策，而通过清晰阐述优先级，便可为决策制定提供指导。

业务约束

业务约束在确定部署项目的性质上发挥着重要作用。部署设计取得成功的关键是在受已知业务约束的情况下找到可以满足业务要求的最佳方法。

下表列出了通常可能影响部署设计的业务约束。具体部署项目可能存在其自身状况所特有的业务约束。

表 2-2 业务约束分析主题

主题	说明
期限或时间表	部署的时间表会影响所作出的设计决策。进取性的时间表可能导致降低目标、更改优先级或采用渐增式解决方法。 时间表内可能还存在值得考虑的重大事件点。
预算注意事项	大多数部署都必须遵守特定预算。在整个设计过程中应始终考虑预算，以避免超支。 考虑预算时，不但要计算完成项目所需的成本，还要计算在特定生命周期内维护项目所需的资源。
资源	考虑成功部署所需的全部资源，而不仅仅是资本支出。这包括以下资源： • 现有硬件和网络基础结构 对现有基础结构的依赖会影响系统的设计。 • 实现部署设计所需的开发资源 如果开发资源（包括硬件、软件和人力资源）有限，便可能意味着要进行渐增式部署。可能必须在每个渐增阶段重复使用相同的资源或开发团队。 • 维护、管理和支持 分析可用于管理、维护和支持系统用户的资源。如果此类资源有限，可能会影响所作的设计决策。
拥有成本	除维护、管理和支持外，还应分析影响拥有成本的其他因素。 例如，可能必要的硬件和软件升级、电力网的覆盖区域、电信成本及影响现金支出的其他因素。

表 2-2 业务约束分析主题 *(续)*

主题	说明
公司标准和策略	确保理解请求部署的组织的标准和策略。 这些标准和策略可能会影响设计的技术层面、产品选择及部署方法。
公司变化管理	公司变化管理程序可能会对部署方法和时间表产生极大影响。
投资收益	每项部署都应为其投资客户带来收益。投资收益分析通常涉及资本支出所获财务收益的计算。 估算部署的财务收益涉及对以部署方式实现目标与以其他方法实现这些目标，或与保持现状所需成本的详细比较分析。
监管要求	监管要求的差异极大，具体要求取决于部署的性质。

渐增式部署方法

通常将部署视为一个完整的综合系统。不过，实际运作中往往会将综合系统的部署分为几个可衡量的步骤，以渐增方式完成。

这种渐增式方法具有以下优势：

- 能够根据业务增长所带来的要求变化作出调整
- 可在向最终部署实现过渡的过程中充分利用现有基础结构
- 能够适应资本支出要求
- 能够充分发挥少量人力资源的作用
- 能够为集成合作伙伴方案做准备

采用渐增式方法时，通常要设计一个路线图，其中给出通往最终的综合解决方案的重大事件点。此外，可能还必须考虑制定针对随后将实施的各规划阶段的短期解决方案。

无论采取什么方法，应始终确保部署设计有变动和增长的余地。

第 3 章

技术要求

本章讨论技术要求分析阶段进行的其中一些过程和步骤。

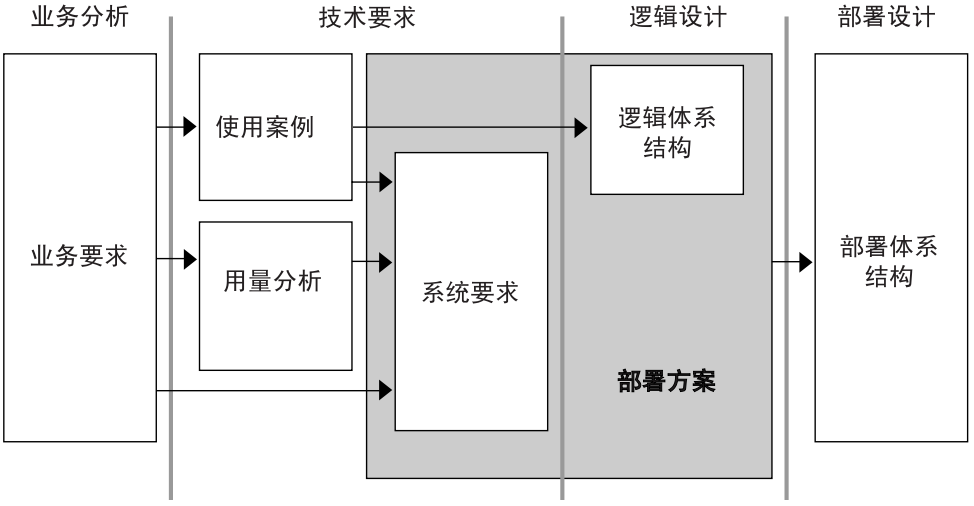
技术要求分析以业务分析阶段所创建的业务要求文档为起点。需要以这些业务要求为基础，执行下列步骤：

- 执行*用量分析*，以协助确定预计的部署负载
- 创建一组模拟典型的用户与部署间交互的*使用案例*
- 创建一组源自业务要求、使用案例和用量分析的*系统要求*

使用案例还是设计阶段进行*逻辑体系结构*设计的基础。逻辑体系机构和系统要求一起组成了*部署方案*，以后作为部署设计阶段的信息来源。

下图显示的是技术要求阶段与业务分析、逻辑设计及部署设计各阶段的相互关系。

图 3-1 技术要求阶段和其他部署规划阶段



与业务分析阶段一样，技术要求分析阶段也不存在可生成用量分析、使用案例和系统要求的魔方。进行技术要求分析要求对业务领域、业务目标及基础系统技术有了解。

本章包括以下节：

- “用量分析”
- 第 30 页的 “使用案例”
- 第 31 页的 “系统要求”

用量分析

用量分析的任务是标识要设计的部署的各个用户及确定这些用户的使用模式。可通过收集到的用量分析信息对预期的负载情况有一个大致了解，随后将使用这些信息来确定性能要求和其他系统要求。在给使用案例指定加权时，用量分析信息同样有用，如第 30 页的 “使用案例”中所述。

用量分析阶段的任务是：尽可能与用户面谈，对与使用模式有关的现有数据进行研究及与先前系统的制造者和管理员会谈。下表列出了进行用量分析时应考虑的主题。

表 3-1 用量分析主题

主题	说明
用户数量及类型	确定部署必须支持的用户数量，并在必要时对用户进行分类。 例如： • “企业对客户”部署可能会有大量存取者，但可能只有少数用户会进行注册并参与商业交易。 • “企业对员工”部署通常必须可以为每位员工提供服务，但有些员工可能需要从公司网外部进行存取。 • 在“企业对员工”部署中，经理可能需要获得对某些区域的存取权限，而这种权限是不应提供给普通员工的。
活动和非活动用户	确定活动和非活动用户的使用模式和使用比率。 活动用户是指登录系统并与系统组件进行交互的用户。非活动用户是指不登录的用户，或虽然登录但不与系统组件进行交互的用户。
管理用户	确定对部署系统进行存取，以对部署进行监控、更新和支持的用户。 确定可能会对系统要求产生影响的任何特定管理使用模式，例如，从防火墙外对部署进行管理。
使用模式	确定各类用户如何存取系统，并提供预期用量目标。 例如： • 是否存在因用量高而产生的高峰期？ • 正常营业时间是什么？ • 用户是否遍布全球？ • 预计的用户连接持续时间是多少？
用户增长	确定用户群体规模是否固定，或部署是否预期有用户数量增长。 如果预期用户群体会有增长，请尽量作出合理的增长预测。

表 3-1 用量分析主题 *(续)*

主题	说明
用户事务	确定必须支持的用户事务类型。可将这些用户事务转化为使用案例。 例如： • 用户将执行哪些任务？ • 用户登录后是保持登录状态，还是通常会执行若干任务，然后注销？ • 用户间是否会进行一些重要协作，而这种协作需要利用共用日历、召开 web 会议及部署内部 web 页才能实现？
用户研究和统计数据	利用现有用户研究和其他资源来确定用户行为模式。 企业或行业组织往往都会进行一些用户研究，可以从中汲取有用的用户信息。现有应用程序的日志文件中可能会包含一些统计数据，这些数据在做系统估量时会用到。

使用案例

使用案例模拟典型的用户与要设计的部署间的交互，以最终用户的视角说明操作的完整流程。在设计中对整个一组使用案例给予优先考虑可确保设计不偏离提供预期功能这一中心。

每个使用案例均可包括用户行为的定量估计，随后可利用该估计结果来确定系统对性能、可用性及其他服务质量的要求。使用案例还是逻辑体系结构设计的起点，如第 41 页的第 4 章 “设计逻辑体系结构”中所述。

为使用案例指定相对加权，加权最高的使用案例代表最常见的用户任务，这是一种常见的做法。对使用案例加权有助于确定系统要求。

可分两级对使用案例进行说明。

- 使用案例图

  参与者和使用案例关系的图形描述。

- 使用案例报告

  对各种使用案例 （包括主要及备用事件流）的说明。

系统要求

系统要求说明的是，部署系统为满足通过业务分析而得出的业务要求而必须提供的服务质量。通常使用用量分析和使用案例，并结合业务要求来得出系统要求。

下表列出了常用来定义系统要求的系统特性。

表 3-2 影响部署设计的系统特性

系统特性	说明
可用性	一种对系统资源和服务可供最终用户使用比率的衡量指标，通常以系统的*正常运行时间*来表示。
潜在容量	在不增加资源的情况下，系统处理异常峰值负载用量的能力。
性能	对用户负载情况响应时间和等待时间的衡量指标。
可伸缩性	随时间推移为部署系统增加容量 （和用户）的能力。可伸缩性通常涉及向系统添加资源，但不应要求对部署体系结构进行更改。
安全性	对系统及其用户的完整性进行说明的复杂因素组合。安全性包括用户的认证和授权以及信息的安全传输。
可维护性	对部署系统进行管理的容易度，其中包括监视系统、修复出现的故障及升级硬件和软件组件等任务。

影响部署设计的各系统特性间关联紧密。对一个系统特性的要求可能会影响到对其他系统特性的要求和设计。例如，提高安全性级别可能会影响到性能，而性能又会影响到可用性。增加服务器来解决可用性问题可能会影响到维护成本 （可维护性）。

能否设计出既可满足业务要求，又能兼顾业务约束的系统的关键在于，了解各系统特性间的关联方式及必须作出的权衡。

以下各节将对影响部署设计的各种系统特性做更深入的探讨，并就确定系统要求时应考虑的因素提供相关指导。其中有一节专门探讨服务级别要求，它是一组用于创建服务级别协议的特殊系统要求。

可用性

可用性是定义部署系统*正常运行时间*的一种方法，通常以系统可供用户存取的时间占总时间的百分比来表示。系统不可存取时间 （*停机时间*）可能是因硬件、软件、网络故障或任何其他因素 （如断电）所致，这些因素会使系统停机。在大多数情况下，不将计划的服务 （维护和升级）时间视为停机时间。

通常以 “九”的个数来衡量可达到的可用性。例如， 99% 的可用性为两个九。指定更多的九会对部署的可用性设计产生很大影响。下表显示的是为某系统增加代表可用性的九后的结果，该系统以 24x7 方式全年不间断运行，运行时间共计 8,760 小时。

表 3-3 全年运行 （8,760 小时）系统的停机时间

九	可用性百分比	停机时间
两个	99%	88 小时
三个	99.9%	9 小时
四个	99.99%	45 分钟
五个	99.999%	5 分钟

容错系统

对可用性的要求达四个或五个九通常要求必须系统是一个*容错*系统。容错系统必须能够在硬件或软件出现故障时继续运行。通常，容错的实现手段是为提供关键服务的硬件 （如 CPU、内存和网络设备）及软件配置冗余组件。

*单一故障点*是指没有备用的冗余组件的硬件或软件组件。该组件出现故障会使系统无法继续提供服务。设计容错系统时，必须确定潜在的单一故障点并将其消除。

容错系统的实现和维护成本高昂，请确保先了解业务可用性要求的本质，然后再考虑能够满足这些要求的可用性解决方案的策略和成本。

Sun Cluster 3.1 4/04

Sun Cluster 3.1 4/04 软件可为要求具备高可用性容错系统的部署提供高可用性解决方案。 Sun Cluster 3.1 4/04 将服务器、存储器及其他网络资源相结合，使系统用户能够快速完成故障转移，而几乎不用中断服务。

排定各类服务的可用性优先级

在用户看来，可用性更多牵涉到的往往是部署系统所提供的每项服务，而非整个系统的可用性。例如，如果即时消息传送服务不可用，通常情况下对其他服务的可用性几乎没有影响或无任何影响。不过，许多其他服务 （如 Directory Server）所依存的服务的可用性则对系统有较大影响。

根据一组有序的优先级列出可用性需求会有帮助。下表按优先级顺序列出了不同服务类型的可用性。

表 3-4 排定各类服务的可用性优先级

优先级	服务类型	说明
1	策略	运行所不可或缺的服务。例如，许多服务依存于 Directory Server。
2	关键任务	峰值负载时必须可用的服务。例如，被定义为关键任务的应用程序数据库服务。
3	必须可用	必须可用，但可以较低性能获得的服务。例如，在某些业务环境中， Messaging Server 可用性可能并不关键。
4	可延迟	在特定时间段内必须可用的服务。例如，在某些业务环境中，Instant Messaging 可用性可能并非不可或缺。
5	可选	可无限期延迟的服务。

有关用于实现可用性要求的各种设计策略的信息，请参阅第 61 页的 “可用性估量”。

潜在容量

潜在容量是指在不增加资源的情况下，部署处理异常峰值负载用量的能力。通常不直接围绕潜在容量定义系统要求，但该系统特性是确定可用性、性能和可伸缩性要求的一个因素。

性能

确定性能要求是将业务要求的性能预期转化为系统要求的过程。业务要求通常以指定响应时间的非技术性术语来表示性能。例如，基于 web 存取的业务要求可能会做以下说明：

> 正常情况下，用户登录后会有一段合理的响应时间，这段时间通常不超过四秒。

从该业务要求入手，对所有使用案例进行研究，以确定在系统级体现该要求的方式。请将用量分析阶段所确定的用户负载情况考虑在内。以术语*指定负载情况下的响应时间*或*响应时间加吞吐量*来表示每个使用案例的性能要求。可能还要指定容错数。

下面是有关如何指定系统性能要求的一个示例。

> 以 15 分钟为间隔测量的用户登录响应时间在全天各时段均不得超过四秒，每百万事务错误数须少于 3.4 个。

性能要求与可用性要求 （故障转移对性能的影响）及潜在容量 （可用于处理异常峰值负载的容量）关系密切。

可伸缩性

可伸缩性说明的是随时间推移为系统增加容量和用户的能力。可伸缩性通常要求增加资源，但不应要求对部署体系结构的设计进行更改，或因增加额外资源需要时间而中断服务。

与可用性一样，可伸缩性更多牵涉到的是系统所提供的各项服务，而非整个系统。不过，对于其他服务所依存的服务 （如 Directory Server），可伸缩性的影响可能会波及整个系统。

不必在系统要求中指定可伸缩性要求，除非业务要求中对预测的部署增长做了明确说明。在部署设计阶段，即使不指定可伸缩性要求，部署体系结构也应将系统扩展考虑在内。

确定可伸缩性要求并不是一门精确的科学，进行系统增长估计是做一些可能无法达成的预测、估计和推测。以下是构建可伸缩系统的三个关键点。

- 采用高性能设计策略。

  在性能要求的确定和设计阶段加入潜在容量，以处理可能会随时间推移而增长的负载。还要在预算限度内尽可能提高可用性。采用这一策略可使系统能够承担增长的负载，并可更从容地制订系统扩展的重大事件点。

- 分阶段实现部署。

  采用渐增式实现方法有助于资源增加计划的制订。

- 实现大范围性能监视。

  对部署的性能进行监视有助于确定向部署中增加资源的时机。

下表列出了在可伸缩性方面应考虑的一些主题。

表 3-5 可伸缩性方面应予考虑的一些因素

主题	说明
用量分析	通过研究现有数据了解当前 （或预测）用户群体的使用模式。如果缺少现时数据，可对行业数据或市场估计进行分析。

表 3-5 可伸缩性方面应予考虑的一些因素 *(续)*

主题	说明
以合理的最大标度为目标进行设计	设计时以能够满足已知和潜在需求的最大必需标度为目标。 这往往是根据现有用户负载的性能评估和对未来负载的合理预期而作出的 24 个月估计。估计周期的长短在很大程度上取决于预测的可靠性。
设置合适的重大事件点	以渐增方式实现部署设计来满足短期要求，同时设立缓冲区来应对意外增长。设置增加系统资源的重大事件点。 例如： • 资本获得 如每季度或每年 • 硬件研制周期 例如，一到六个星期 • 缓冲区 （10% 到 100%，具体取决于增长预期）
融入新兴技术	了解新兴技术 （如速度更快的 CPU 和 Web 服务器），及该技术会对基础体系结构的性能产生怎样的影响。

安全性要求

安全性是指会对系统及其用户的完整性 （包括用户事务及相关数据的完整性）产生影响的系统特性。与其他系统要求一样，业务要求、用量分析和使用案例会推动针对安全性要求的分析。

针对安全性要求的分析分为以下几类：

- 验证
- 授权
- 身份认证管理

进行验证、授权、身份认证管理及在整个企业范围内贯彻可靠安全性惯例实施策略，可保障事务安全及确保存储在站点的数据不受损害。

注 在系统要求分析阶段，通常对影响基础结构 （例如，防火墙软件和网络设计）完整性的安全性要求不予考虑。但这些安全性问题会在部署设计阶段产生影响。

验证

验证是指用户如何向系统表明身份及系统如何向用户表明自身身份。验证是系统完整性的一个关键节，它能够防止系统受到未授权存取。

应了解用户要求，以为部署选择最佳验证方案。例如，“企业对客户”部署可允许用户使用用户名 / 口令组合进行注册，这些用户凭借信任的证书授权机构 （如 VeriSign）所颁发的服务器证书，通过安全传输来验证销售系统。

而 “企业对员工”部署可从现有用户群体中置备员工。在公司防火墙内，允许对已知安全位置进行存取。在防火墙以外，对安全位置的存取则通过代理来实现，代理会在公司防火墙内执行验证和重定向操作。

授权

授权是对已验证用户拥有特定权限的认可。例如，拥有管理员授权的用户可以存取普通用户无法存取的部署系统的某些节。

授权在实现单点登录 (SSO) 的部署中也发挥着作用。某部署的已验证用户只需登录一次，便可使用多项服务。

身份认证管理

部署系统必须提供一种途径来添加、修改或删除存取系统服务的用户。身份认证管理可由授权管理员执行或由用户自己借助*委派管理*接口来完成，具体采用何种方式取决于实际需要。大中型企业部署应考虑采用委派管理设计。采用委派管理可提高客户满意度及降低系统管理成本。

可维护性要求

可维护性是指对部署系统进行管理的容易度，其中包括监视系统、修复出现的故障及升级硬件和软件组件等任务。

计划可维护性要求时应对下表所列的主题予以考虑。

表 3-6 可维护性要求主题

主题	说明
停机时间计划	确定必须使特定服务完全或部分不可用的维护任务。 某些维护和升级任务可在用户不知不觉中完成，而要执行其他类似任务则必须中断服务。尽可能与用户一起为要求停机的维护活动制订计划，使用户能够为停机时间作出准备。
使用模式	确定部署的使用模式，以确定可执行维护任务的时段。 例如，在通常于正常营业时间内出现用量高峰的系统中，适合在傍晚或周末执行维护任务。对于在地域上分散的系统而言，确定这些时段的难度可能更大。
可用性	可维护性往往是可用性设计思想的反映，因为最大限度缩短维护和升级停机时间策略是围绕可用性策略来制订的。要求具有高可用性的系统的维护、升级和检修期较短。 处理可用性要求的策略会影响到对维护和升级的处理方式。例如，在地域上分散的系统中，维护方式可能取决于维护期内将工作负载发送到远程服务器的能力。 要求具有高可用性的系统可能还需要采用更为复杂的解决方案，使系统重新启动可以自动进行，而几乎不需要人为介入。
诊断和监视	定期运行诊断和监视工具来确定故障区域，可以提高系统的稳定性。 这样可防患于未然，有助于根据可用性策略来平衡工作负载及改进维护和停机计划。

服务级别要求

服务级别要求 是一组系统要求，用于指定必须提供客户支持的一些情况。服务级别要求是服务级别协议的基础，通常在项目核准阶段签署该协议。

与系统要求一样，服务级别要求源自业务要求，并代表着对客户的一种担保，即部署必须达到的整体系统质量。由于服务级别协议是与客户间签订的一份合同，因此在服务级别要求的具体规定上不应有模糊之处。服务级别要求对要求的测试条件及不合要求的构成条件均有明确规定。

第 4 章

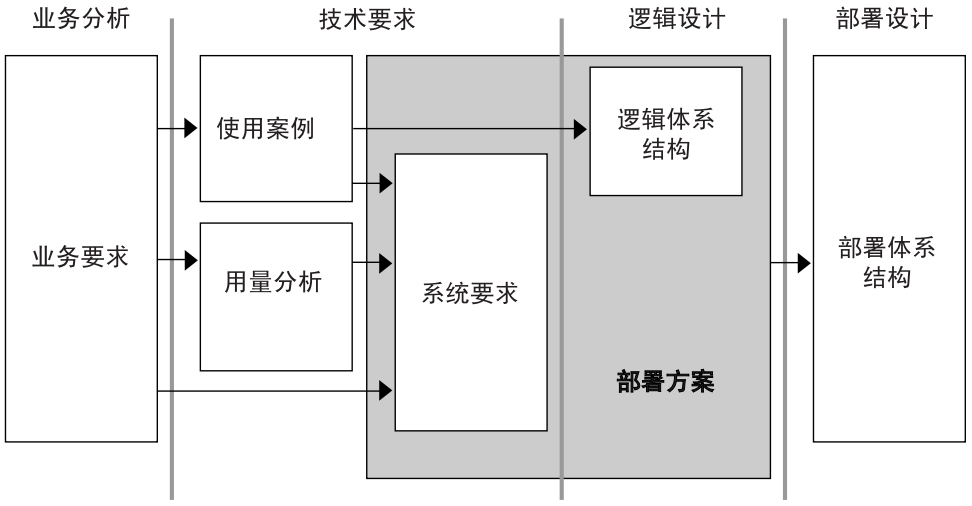
设计逻辑体系结构

本章利用中型企业通信部署的一组典型使用案例，阐述了逻辑体系结构的创建过程，并提供了该过程的一个示例。

逻辑体系结构确定各 Java Enterprise System 组件（及其依赖性），这些组件提供了满足部署业务目标所需的软件服务。技术要求阶段开发的使用案例通常会指出所需的软件服务。不过，往往可从业务分析阶段形成的业务要求中直接获得有关软件服务的信息。

逻辑体系结构加上要求分析阶段所确定的系统要求，便代表了一种部署方案。部署方案是进行部署体系结构设计的基础。下图显示的是逻辑设计阶段与业务分析、技术要求及部署设计各阶段间的关系。

图 4-1 涉及其他部署规划阶段的逻辑设计



本章包括以下节：

部署规划示例

为协助说明部署规划过程，本节介绍了一个以典型的中型企业通信需要为基础的示例部署的各种使用案例。本白皮书的后续各章会沿用该示例部署来说明部署规划的各个步骤。

警告 该示例部署的使用案例、逻辑体系结构、部署体系结构及设计规范是部署规划过程各步骤的简化版本。

为便于说明，该示例经过了简化。该示例的设计并不完整，也从未实际构建或测试过。请勿将该示例用作任何欲计划的部署的蓝图。

该示例部署以一组使用案例为起点，这些使用案例源自通信部署的典型业务要求。下表对这些使用案例进行了总结。

表 4-1 示例部署的使用案例

使用案例	说明
#1 单点登录	用户通过 Web 浏览器登录系统（用户名 / 口令）来存取企业服务，该服务可以是以下类型的服务： • 定制门户 Web 页 • 基于 Web 的电子邮件页 • 日历界面 • 安全 Web 页
#2 打开个人门户屏幕	用户通过 Web 浏览器导航到个人门户屏幕。
#3 用户通过门户检查电子邮件	用户通过门户界面检查是否有新的电子邮件。
#4 用户通过门户检查安全 web 页	用户通过个人门户界面检查安全项目状态页。
#5 用户通过门户检查日历	用户通过门户界面检查每日日程。
#6 管理日历	用户通过基于 Web 的日历客户机安排日程。
#7 管理电子邮件	用户通过电子邮件客户端阅读和发送电子邮件。

可从这些使用案例中获得逻辑体系结构所需的服务，如以下节中所述。

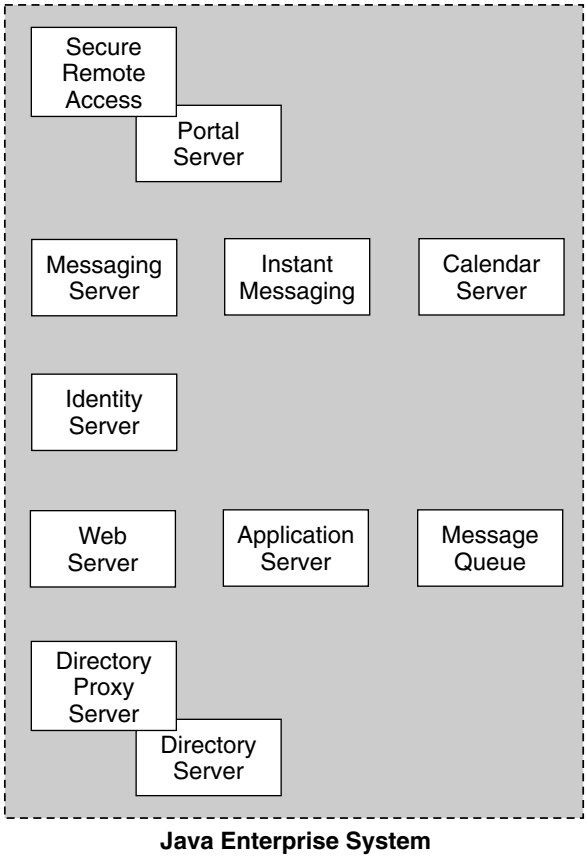
Java Enterprise System 服务

逻辑体系结构设计自分析使用案例开始，这应有助于确定部署所需的服务。利用所掌握的 Java Enterprise System 知识和已往的设计经验，对各 Java Enterprise System 组件进行初步逻辑设计，使用案例所确定的服务是由这些组件提供的。

设计组件时，请考虑系统内的逻辑数据流动及提供服务的各组件间的依赖性。逻辑设计应反映这些依赖性，它们会影响设计中各组件间的数据流。

下图显示的是 Java Enterprise System 提供的组件。请利用该图和第 45 页的表 4-2来了解各 Java Enterprise System 组件间的相关性。一般而言，图中下方的组件为其上方的组件提供支持。

图 4-2 Java Enterprise System 组件



下表列出了 Java Enterprise System 各组件间的实际相关性。

表 4-2 Java Enterprise System 组件相关性

Java Enterprise System 组件	所支持的组件	所依赖的组件
Application Server	Identity Server Portal Server	Message Queue
Calendar Server	Portal Server （用于日历通道）	Directory Server Identity Server （用于单点登录） Messaging Server （用于 Calendar Server 电子邮件通知服务）
Directory Proxy Server	无	Directory Server
Directory Server	Administration Server Calendar Server Directory Proxy Server Identity Server Instant Messaging Messaging Server Portal Server	无
Identity Server	Portal Server 如果配置了单点登录： Calendar Server Instant Messaging Messaging Server	Directory Server Application Server 或 Web Server
Instant Messaging	Portal Server	Directory Server
Message Queue	Application Server	Directory Server （可选）
Messaging Server	Calendar Server Portal Server （用于消息传送通道）	Directory Server Web Server Identity Server
Portal Server	Secure Remote Access	Directory Server Application Server 或 Web Server 如果配置为使用 Portal Server 通道： Calendar Server Messaging Server Instant Messaging
Secure Remote Access	无	Portal Server
Web Server	Identity Server Portal Server	无

例如，要针对示例通信部署设计各 Java Enterprise System 组件，请对第 43 页的表 4-1 中所列的各种使用案例进行分析。下表列出了该部署直接需要的组件，如各使用案例所示。

表 4-3 支持示例使用案例的 Java Enterprise System 组件

Java Enterprise System 组件	使用案例
Portal Server	#1 单点登录
	#2 打开个人门户屏幕
	#3 用户通过门户检查电子邮件
	#4 用户通过门户检查安全 web 页
	#5 用户通过门户检查日历
Calendar Server	#1 单点登录
	#5 用户通过门户检查日历
	#6 管理日历
Messaging Server	#1 单点登录
	#3 用户通过门户检查电子邮件
	#7 管理电子邮件

还需要确定需要哪些 Java Enterprise System 组件来支持上面表 4-3 中所列的各个组件。下表列出了这些附加组件。

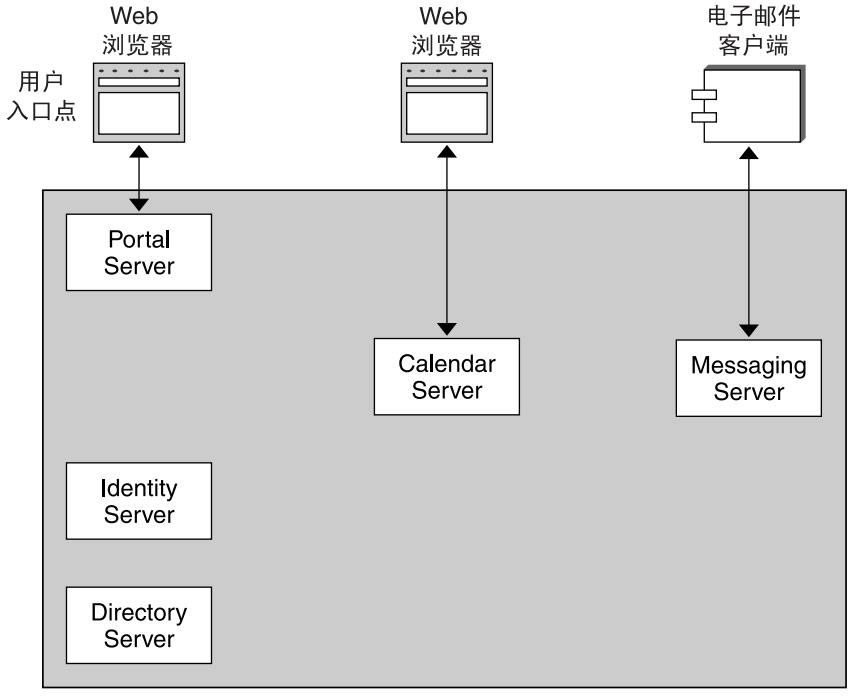
表 4-4 支持示例使用案例的附加组件

Java Enterprise System 组件	所提供的支持
Identity Server	支持 Portal Server。
	对 Calendar Server 和 Messaging Server 提供单点登录支持。
Directory Server	支持 Identity Server 和 Portal Server。
Application Server 或 Web Server	支持 Identity Server 和 Portal Server。（Identity Server 和 Portal Server 必须在 web 容器内运行。）

示例部署的逻辑体系结构

下方的图 4-3 显示了示例部署的组件布局，并标明了该部署的用户入口点。该图将需要最多支持的服务 (Portal Server) 置于顶端，并在其下方列出提供支持的各个组件。它大致反映了各组件间的依赖性（如第 45 页的表 4-2 中所述）。该图对提供 web 容器以支持 Portal Server 和 Identity Server 的组件未予描述，原因是这种依赖性并不反映部署中的数据流向。

图 4-3 逻辑体系结构中的 Java Enterprise System 组件



示例部署的逻辑体系结构

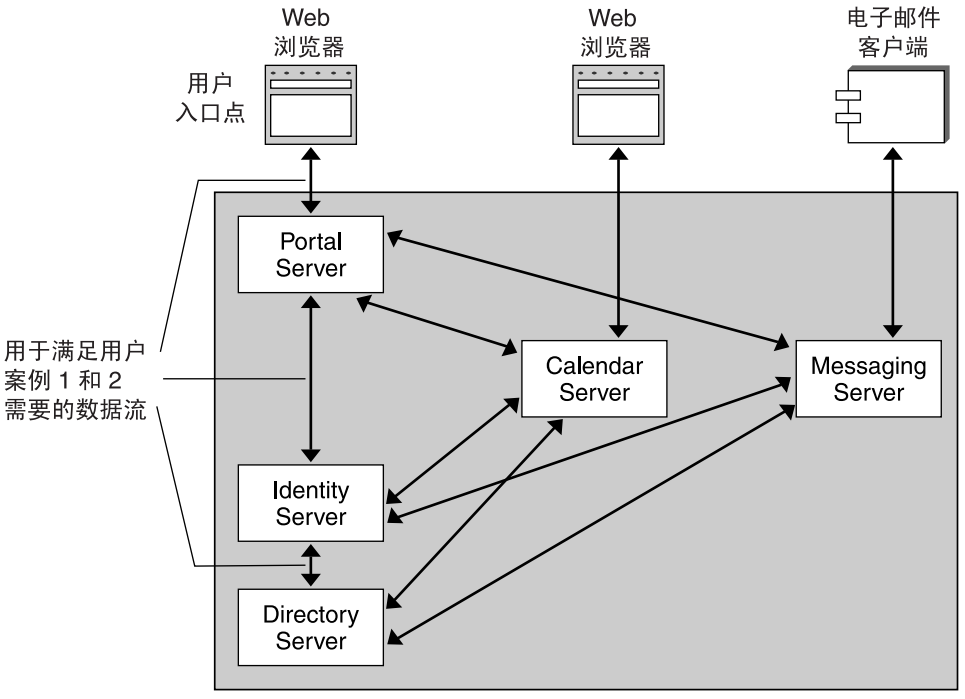
☐ 未展示支持 Portal Server 和 Identity Server 的 Web 容器

示例部署的数据流动

对使用案例进行研究，以确定逻辑体系结构中各服务间的逻辑数据流动，并在布局图中标明这种流动。系统中各服务间的数据流动在进行性能和可用性估量时发挥着重要作用，如第 52 页的 “计划部署估量”中所述。

下图描述的是示例部署的数据流动。数据流动通过部署的使用案例以及各 Java Enterprise System 服务的依赖性来确定。

图 4-4 示例部署的逻辑数据流动



示例部署的逻辑体系结构

图 4-4 突出了能够同时满足使用案例 1 和 2 的数据流动，该数据流动代表以下内容：

- 用户通过基于 Web 的客户端发出登录请求
- Portal Server 依赖 Identity Server 提供验证服务

- Directory Server 向 Identity Server 提供的 LDAP 信息

图 4-4 中的其余数据流动同样源自使用案例和服务器依赖性。

部署方案

完成的逻辑体系结构设计和要求分析阶段得出的系统要求一同构成了部署方案。部署方案是部署体系结构设计的起点，如第 5 章 “设计部署体系结构”中所述。

第 5 章

设计部署体系结构

本章介绍针对性能、安全性、可用性及其他系统特性设计部署的方法。本章还介绍部署设计优化的相关信息。

部署体系结构描述逻辑体系结构与物理环境的映射关系。物理环境包括内联网或 Internet 环境中的计算节点、 CPU、内存、存储器设备及其他硬件和网络设备。

设计部署体系结构涉及*部署估量*，以确定满足技术要求阶段指定的系统要求所必需的物理资源。还要通过分析部署估量的结果来*优化资源*，使所形成的设计能够在受业务约束的情况下最充分地利用资源。

部署体系结构设计完成后，*项目核准*阶段的任务是评估部署的实际成本。项目一经核准，即需签署部署完成合同和获取项目实现所需的资源。

应在项目核准前或核准后制订*详细设计规范*。详细设计规范用于在实现阶段扩展设计。

本章继续使用第 4 章的示例部署来说明部署体系结构设计过程中的各个步骤。

本章包括以下节：

计划部署估量

计划部署估量是确定满足系统要求并最终实现业务目标所必需的硬件资源的过程。与部署规划和设计的其他方面一样，估量并不是一门精确的科学，不能用公式和配方来规定。只有参照以往的设计经验，凭借所掌握的系统体系结构知识和特定领域知识，并发挥创造性思维，才能成功完成估量工作。

估量要围绕先前为以下系统特性确定的系统要求（第 31 页的 “系统要求”中所述的系统要求）进行。此外，部署设计早期阶段的业务要求、用量分析和使用案例在系统估量中也发挥着一定的作用。

执行估量工作时，使用案例和用量分析有助于确定使用案例所必需的支持资源。估量通常从最大加权的使用案例（代表最常见的事务量）开始，最后估量的是最小加权的使用案例。使用加权使用案例有助于根据预期的系统负载分配资源。

以下各节就如何针对下列系统特性进行部署估量提供了一般性指导：

- 性能
- 安全性
- 可用性
- 可维护性

性能导向估量

性能和负载要求导向估量是一个反复进行的过程，它对支持部署系统中的服务所需的 CPU 数量和内存大小进行估计。估计支持服务所需的 CPU 数量时，请考虑下列各项：

- 应用于服务的使用案例和相应的用量分析
- 技术要求分析阶段确定的系统要求
- 对提供服务的 Java Enterprise System 组件的以往应用经验

- 咨询 Sun 专业服务机构，他们有估量各类部署方案的经验

性能导向估量过程通常由下列步骤组成。这些步骤的执行顺序并非关键所在，这一顺序只是提供了考虑可对最终结果产生影响的因素的一种方式。

1. 给确定为用户的系统入口点的组件确定 CPU 数量估计底线。
2. 对 CPU 数量估计进行调整，将组件间的依赖性考虑在内。
3. 对 CPU 数量估计进行调整，将安全性、可用性、可伸缩性和潜在容量要求考虑在内。

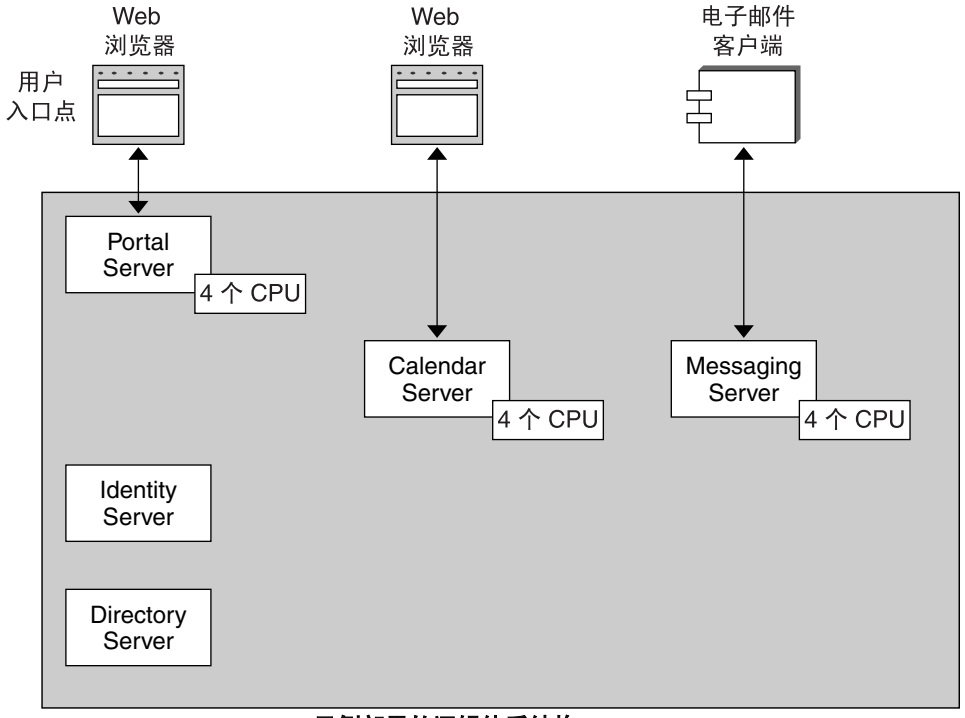
为用户入口点确定 CPU 数量估计底线

先估计处理每个作为用户入口点的组件的预期负载所需的 CPU 数量，然后在逻辑体系结构的布局设计图上标注估计值。

下图使用的是第 42 页的 “部署规划示例”中介绍的示例部署，它描述作为用户入口点的组件的初始 CPU 数量估计。这些估计值所代表的是通过系统要求分析、使用案例和用量分析而可能得到的数字。

警告 本白皮书不会在性能导向估量细节上给予指导。本手册中使用的 CPU 和内存数字都只是说明性的随意估计，它们用于说明设计系统时可能使用的一种过程，并不代表任何具体的实现建议。

图 5-1 提供用户入口点组件的 CPU 数量估计底线



示例部署的逻辑体系结构

针对服务依赖性调整 CPU 数量估计

提供用户入口点的组件需要其他 Java Enterprise System 服务的支持。为继续指定性能要求，请调整性能估计，将必需的其他组件支持考虑在内。

在本例中，检查第 48 页的图 4-4 所示的逻辑数据流动，为给其他组件提供支持的组件作出调整。下表总结了对 CPU 数量估计所做的调整。可指定含小数的 CPU 数量估计。完成性能估计后，系统会对 CPU 计数求和并向上舍入。

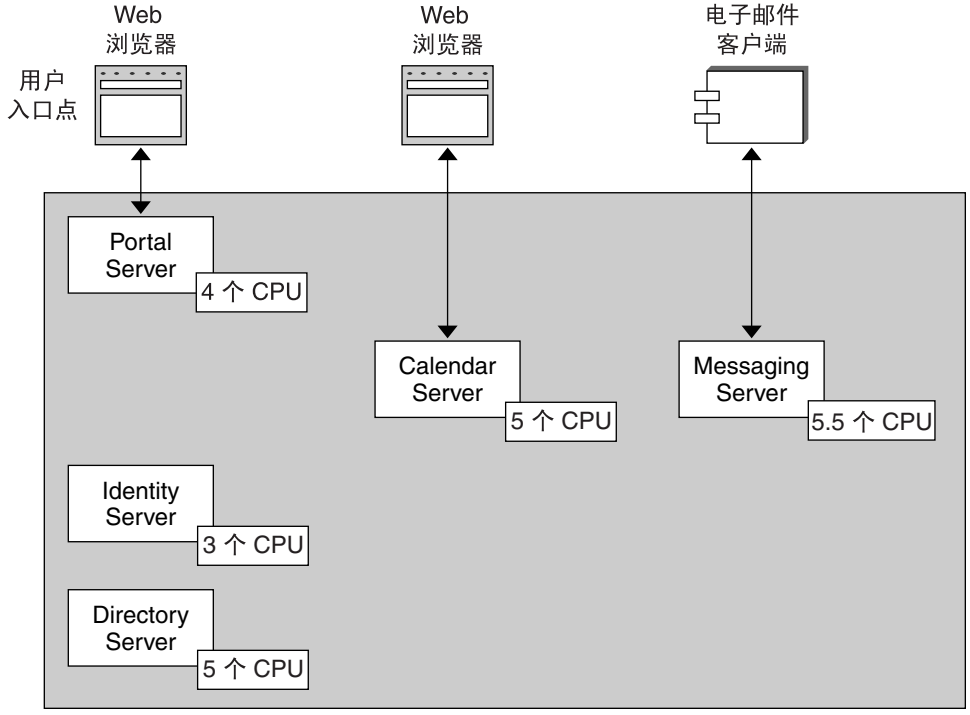
与前一节中的估计一样，下表中的性能估计也只是说明性的随意值。

表 5-1 支持服务的 CPU 数量估计

服务	数量估计	说明
Portal Server	无	不为其他服务提供支持。
Calendar Server	1 个 CPU	支持： • Portal Server 的日历通道
Messaging Server	1.5 个 CPU	支持： • Portal Server 的消息传送通道 • Calendar Server 的电子邮件通知服务
Identity Server	3 个 CPU	支持： • Portal Server • Calendar Server • Messaging Server
Directory Server	5 个 CPU	支持： • Identity Server • Calendar Server • Messaging Server

下图根据表 5-1 中的信息更新性能导向估计数量。

图 5-2 针对支持服务调整后的 CPU 数量估计



示例部署的逻辑体系结构

针对潜在容量、可伸缩性和可用性要求调整 CPU 数量估计

性能导向估量完成后，需向上舍入 CPU 数量数字。通常将 CPU 数量向上舍入到最近的偶数。对 CPU 数量估计进行向上舍入时，请考虑以下因素：

- 潜在容量

  增加 CPU 数量估计以增强系统处理峰值负载的能力。

- 可伸缩性

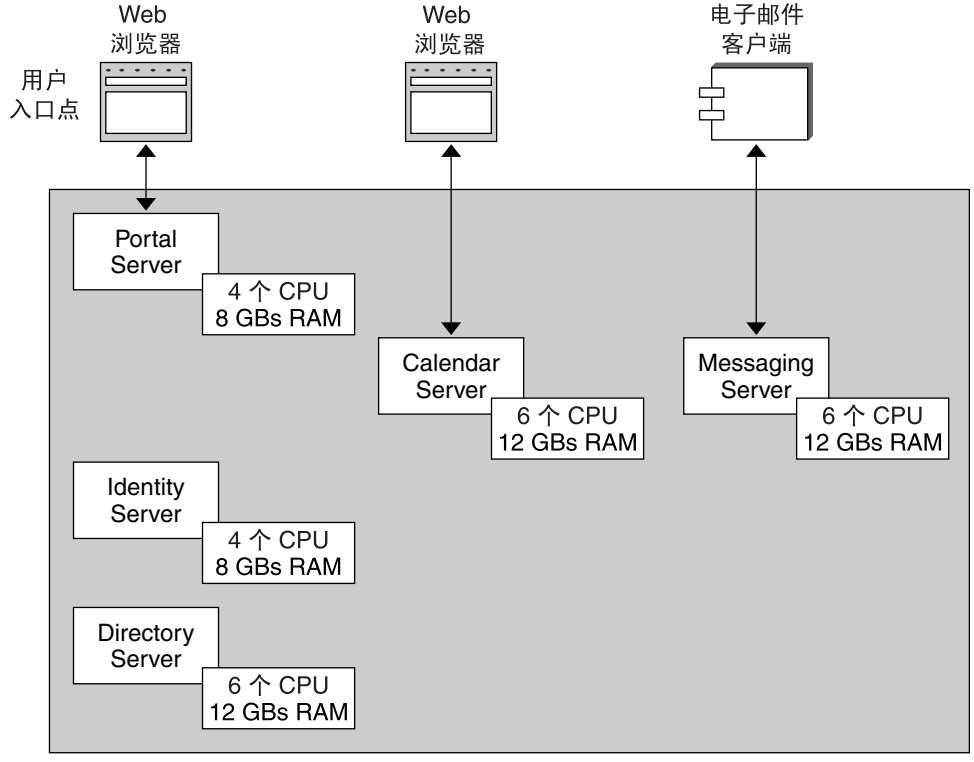
  增加 CPU 数量估计，以确保不必过早扩展部署规模。查看预计的扩展重大事件点及相应时段的预计负载增量，确保潜在容量足以满足重大事件点的要求。

- 可用性

  调整 CPU 数量估计，将可用性或故障转移可能需要复制服务的情况考虑在内。

下图针对示例部署调整了 CPU 数量估计。图中还标明了每个 CPU 的内存要求。本例假设每个 CPU 需要 2GB 内存。本例中的这些内存数字只是说明性的随意数字。计算每个 CPU 所需的内存不在本白皮书讨论的范围之内。

图 5-3 包括内存要求在内的性能数字



示例部署的逻辑体系结构

安全性估量

进行部署估量时，应在以下方面将安全性问题作为考虑的因素：

- 安全数据传输
- 用户验证

安全数据传输涉及通过安全传输协议 （如安全套接字层 (SSL)）处理事务。用户验证也可能需要通过安全传输处理事务。

通过安全传输处理的事务通常需要额外的计算能力先建立安全会话 （称为*握手*），然后对传输的数据进行加密和解密。额外的计算能力要求可能相当大，这取决于所使用的加密算法 （例如， 40 位还是 128 位加密算法）。

为使安全事务具有与非安全事务相同水平的性能，必须对额外计算能力进行计划。安全事务可能需要四倍 （甚至更多倍）的计算能力，具体取决于事务的性质和处理事务的 Java Enterprise System 服务。

估计处理安全事务的性能要求时，首先要分析使用案例来确定需要安全传输的事务所占的百分比。如果安全事务的性能要求与非安全事务相同，则需修改 CPU 数量估计，将安全事务所需的额外计算能力考虑在内。

在某些使用方案中，可能只有在验证时才需要进行安全传输。用户通过系统验证后，便不需要对数据传输采取额外安全措施。但在其他方案中，可能所有事务都需要安全传输。在许多情况下，合理的估计是有百分之五到十的事务需要安全传输。

例如，浏览在线电子商务站点的产品目录时，客户完成商品挑选并准备 “付帐” 前，所有事务都可以是非安全的。而且，许多这类电子商务站点均放宽了对安全事务的潜在响应要求。不过，某些使用方案 （如银行或证券经纪行部署）要求大多数 （即便不是全部）事务必须为安全事务，并对安全与非安全事务有着相同的性能标准。

计算安全事务性能

本节继续使用示例部署来说明一个工作单，它用于计算一个既包括安全事务又包括非安全事务的使用案例的 CPU 数量要求。

要计算 CPU 数量要求，请在工作单中进行以下计算：

1. 以 CPU 数量要求底线 （如在上一节第 52 页的 “性能导向估量”中计算的数字）作为起始数量。
2. 计算需要 SSL 的事务的百分比，然后计算 SSL 事务的 CPU 数量要求。
3. 针对非安全事务调整 CPU 计算。
4. 将安全和非安全数量要求相加，计算出总 CPU 数量要求。

图 5-4 中的工作单的依据是 Portal Server 的额外使用案例和用量分析。以下是额外使用案例和用量分析的前提：

- 所有登录均要求安全验证
- 所有登录占 Portal Server 总负载的 10%
- 安全事务的性能要求与非安全事务相同。

就本例而言，由于要将处理 SSL 事务的额外计算能力考虑在内，处理这些事务的 CPU 数量将增加五倍。与本例中其他 CPU 数字一样，该数字也只是说明性的随意数字。

图 5-4 计算安全事务 CPU 数量估计的工作单

所有 Portal Server 事务的估计底线：4 个 CPU

1. 计算 SSL 事务的 CPU 数量估计： 百分之十需要 SSL .10 x 4 = 4 SSL 事务需要 5 倍的 CPU 能力 5 x .4 = 2 个 CPU	2 个 CPU
2. 调整非 SSL 事务的 CPU 数量估计： 百分之九十为非安全事务 .9 x 4 = 至少 3.6 个 CPU	3.6 个 CPU
3. CPU 数量估计总数： SSL 2 非 SSL 3.6 5.6 个 CPU	5.6 个 CPU

处理 SSL 事务的专用硬件

可以利用专用硬件设备 （如 SSL 加速卡和其他装置）提供建立安全会话和 / 或加密和解密数据所需的计算能力。使用专用硬件进行 SSL 运算时，计算能力专用于 SSL 计算的特定节，通常是建立安全会话的 “握手”运算。

这种硬件可能会对最终部署体系结构有益。不过，由于此类硬件的专用化性质，最好先以 CPU 能力估计出安全事务的性能要求，然后再考虑使用专用硬件处理额外负载的益处。

使用专用硬件时应考虑的一些因素有：使用案例 （例如，要求大量 SSL “握手”运算的使用案例）是否支持使用该硬件及使用此类硬件给设计增添的复杂性。这种复杂性包括这些设备的安装、配置、测试和管理。

可用性估量

完成性能估量后，便可开始进行系统可用性估量。这一过程的任务是，指定逻辑体系结构中的组件的宿主服务器及为各种 Java Enterprise System 组件设计负载平衡、冗余和故障转移策略。

研究使用案例和用量分析，以确定应予考虑的可用性解决方案。下列项目是为帮助确定可用性策略而需收集的信息类型的示例：

- 指定的可用性中有多少个九？
- 是否指定了故障转移情况下的性能要求 （例如，故障转移期间至少保持 50% 的性能）？
- 使用案例和用量分析是否区分高峰和非高峰使用时间？
- 有哪些潜在性能要求？
- 是否有地域考虑因素？

为每个组件分析使用案例，以确定最适合组件的、符合故障转移和负载平衡要求的解决方案。还要考虑使用案例和用量分析，以确定实现服务负载平衡的最佳方式。

所选的可用性策略还必须考虑第 66 页的 “可维护性问题”中阐述的可维护性要求。尽量避免采用以易管理系统组成的复杂解决方案，此类方案需要花费相当大的管理和维护成本。

复杂系统的目录设计

涉及大量用户的复杂部署可能需要为 Directory Server 进行目录设计，而这可能会影响可用性策略。这是因为 LDAP 目录设计可能会影响 Identity Server 和 Messaging Server 的可用性策略，进而可能影响其他系统特性。

如果要设计复杂部署，请考虑创建一个初步目录设计来协助可用性设计，在后续的详细设计制订或开发阶段再提供完整的目录设计。

硬件和软件故障

可用性设计应提供硬件和软件故障保护。软件故障的代价通常高于硬件故障。软件故障的平均间隔时间长于硬件故障的平均间隔时间。此外，软件故障诊断和修复的难度更大，故障防范所需投入的管理和维护成本也更高。

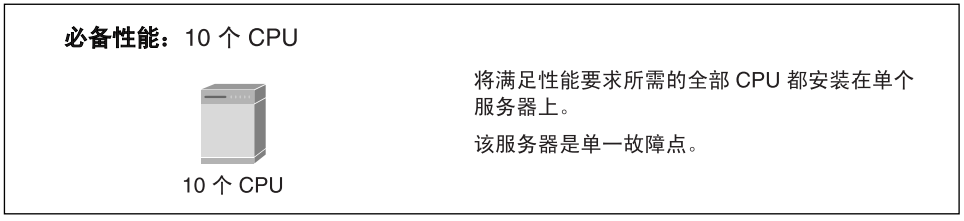
可用性的一般设计方法

本节提供可用性要求的一般设计方法。具体可用性设计不在本白皮书讨论的范围之内。

单服务器系统

将服务的所有计算资源置于单个服务器上。如果服务器发生故障，整个服务便终止运行。

图 5-5 单服务器



Sun 提供具有下列优点的高端服务器：

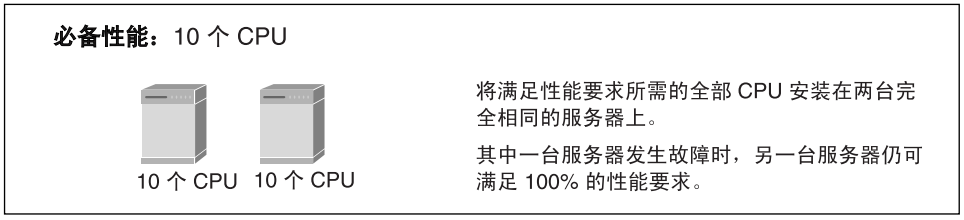
- 系统运行中更换和重新配置硬件组件
- 在服务器的故障隔离域中运行多个应用程序
- 不必重新引导系统即可升级容量、执行速度和 I/O 配置

一台高端服务器的价格通常高于具有可比性的多服务器系统。不过，单个服务器可节省对数据中心多台服务器的管理、监视和驻扎成本。但多服务器系统在负载平衡、故障转移和解除单个故障点方面的灵活性更强。

水平冗余系统

利用可提供负载平衡和故障转移两种功能的平行冗余服务器提高可用性的方法有若干种。下图显示的是组成一个 *N+1 可用性*系统的两台复制服务器。其中一台服务器发生故障时， N+1 系统通过一个额外组件继续提供 100% 容量。

图 5-6 两台复制服务器

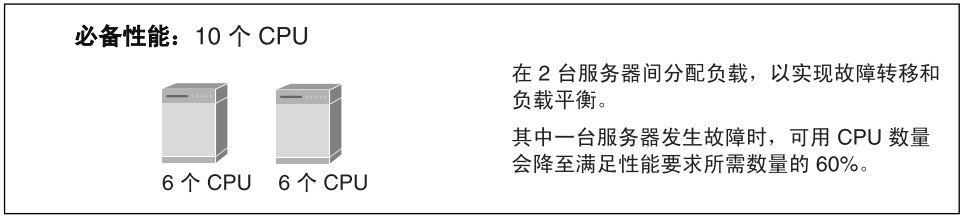


上面图 5-6 中每台服务器的计算能力都完全相同。一台服务器即可满足性能要求，另一台服务器作为备份服务器调入服务时，也可提供 100% 的性能。

复制服务器这种设计的优点是，故障转移情况下仍可达到 100% 的性能； 缺点是增加了硬件成本，而总体性能却未得到相应提升。

下图所显示的是这样一种方案：通过在两台服务器间分配性能负载来满足负载平衡和故障转移要求。

图 5-7 在两台服务器间分配负载

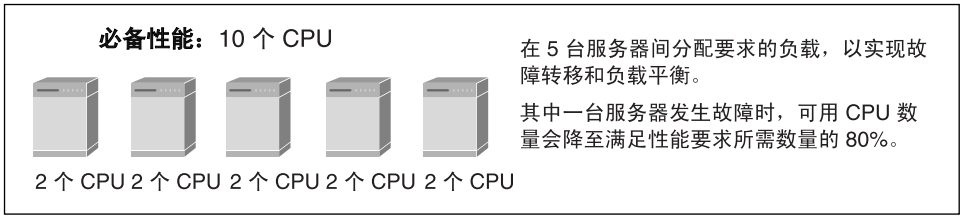


在上面的图 5-7 中，如果一台服务器发生故障，所有服务仍然可用，尽管性能只能达到完全性能的某一百分比。另一台服务器提供 6 CPU 计算能力，为 10 CPU 要求的 60%。

这种设计的一个优点是，两台服务器均可用时有额外 2 CPU 的潜在容量。而且，如果一台服务器发生故障，所有服务均可用，只不过性能可能有所下降。

下图显示的是，在多台服务器间分配负载来满足性能和负载平衡要求。

图 5-8 在 *n* 台服务器间分配负载



由于图 5-8 所示的设计中有五台服务器，如果一台服务器发生故障，其余服务器可继续提供总计 8 CPU 的计算能力，达 10 CPU 性能要求的 80%。如果在设计中增加一个具有 2 CPU 计算能力的服务器，实际得到的便是 N+1 设计。如果一台服务器发生故障，其余服务器可满足 100% 的性能要求。

这种设计具有下列优点：

- 单台服务器发生故障时的性能得到提升
- 即使不止一台服务器停机，仍然具有可用性
- 可轮换将服务器停机，以进行维护和升级
- 多台低端服务器的价格通常低于单台高端服务器

不过，增加服务器数量会使管理和维护成本大幅增加，同时还会带来数据中心服务器驻扎成本。达到某一数量后，再增加服务器所得到的性能提升会越来越少。

Sun Cluster

对于要求高可用性 （如四或五个九）的情况，可以考虑将 Sun Cluster 纳入可用性设计。群集系统是服务器、存储器及其他网络资源相结合的产物。群集中的服务器彼此间不间断地通信，如果其中一台服务器脱机，群集中的其余设备会将该服务器隔离，并将故障节点上的任何应用程序或数据故障转移到另一节点。这一故障转移过程所需时间较短，几乎不用中断为系统用户提供的服务。

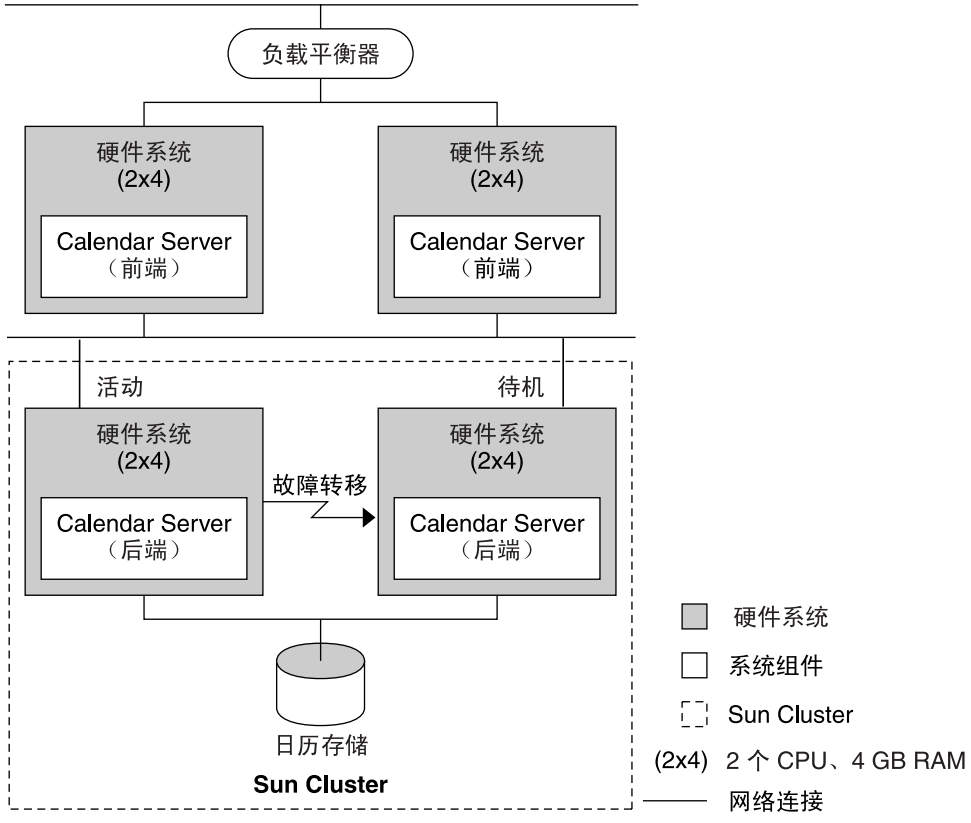
配置、管理和维护 Sun Cluster 需要额外的专用硬件和专门技能。

示例部署的可用性设计

下图显示的是示例部署日历服务节的可用性设计，第 4 章 “设计逻辑体系结构”。中有对该节的介绍。该图描述的是示例部署逻辑体系结构 Calendar Server 节的可用性解决方案。示例部署完整可用性解决方案的分析不在本白皮书讨论的范围之内。

本章前文中的估量确定，Calendar Server 需要 6 个 CPU 和 12 GB 内存，如第 57 页的图 5-3 所示。下图显示的是，为满足负载平衡的收到和外出请求而在两台服务器上部署的 Calendar Server 的前端。 Calendar Server 的后端部署在另外一台服务器上，并为实现故障转移而在 Sun Cluster 3.1 4/04 中进行了复制。为实现故障转移，Calendar Server 后端所需的 CPU 和内存在 Sun Cluster 3.1 4/04 中进行了复制。

图 5-9 示例部署中 Calendar Server 的可用性设计



可维护性问题

针对可用性进行设计时，还必须考虑解决方案的管理和维护成本。设计中往往会忽略这些成本，因为它们与硬件采购并无特定关联。不过，它们却可能成为反衬出设计复杂性的隐性、持续成本。

例如，设计中可能包括能够提供高可用性的大量水平冗余服务器。但是，如果不将设置和配置服务器、不断升级软件和监视系统运行状况的成本计算在内，便可能达不到预期的可用性提升。

针对可维护性进行设计时，请考虑下列管理和维护成本：

- 设置和配置

- 监视
- 升级服务器硬件
- 升级服务器软件
- 故障转移自动化

可伸缩性估量

可伸缩性是指为系统增加容量的能力，通常是以增加系统资源，但不改变部署体系结构的方式进行。本节讨论的是针对可伸缩性进行设计时需考虑的主题。

在要求分析阶段，通常是根据业务要求和后续用量分析对预期增长作出预测。这些对系统用户数量和满足他们需要的系统容量的预测往往只是估计值，可能与部署系统的实际数字大相径庭。设计应具备足够的灵活性，考虑到预测中存在的偏差。

潜在容量

潜在容量是可伸缩性的一个方面，即在系统中增加额外的性能和可用性资源，以便在出现异常高峰负载时能够从容应对。潜在容量是给设计注入安全机制的一种方法。

对使用案例进行仔细分析有助于确定可能产生异常峰值负载的情况（例如，安排强制性 webcast 的企业对员工的部署）。利用对异常高峰负载的分析，并对可应对意外增长的因素加以考虑，便能够设计出可给系统注入安全机制的潜在容量。

还可以对部署系统中潜在容量的使用案例进行监视，以协助确定何时有必要通过增加资源来扩展系统。

升级系统容量

系统设计应能处理系统运行前 6 到 12 个月的预测容量。可利用维护周期，根据需要增加资源或扩大容量。理想情况下应能安排定期对系统进行升级，但预测需要增加的容量往往很难。根据仔细的资源监视和业务预测来确定升级系统的时间。

如果要执行的是渐增式部署，即出于业务或技术原因推迟系统某些节的部署，则可相应制定系统容量升级时间表，使之与为系统增加新功能的其他升级同时进行。

优化资源

部署估量不只是对满足系统要求的资源进行估计，它还涉及风险管理和资源管理。设计对风险管理和资源管理的处理方式往往是能否实现业务目标的关键。

风险管理

估量所依据的许多信息，如业务要求和用量分析，并不是基于经验的数据，而是基于估计和预测的数据。完成计划部署估量前，请复查数据，确保估量设计已将估计或预测值的任何合理偏差考虑在内。

例如，如果业务要求预测低估了系统的实际用量，便要面临这样的风险：所构建的系统无法应付其遇到的流量。未达到性能要求的设计当然是失败的设计。

反之，如果所构建系统的能力远超所需能力，便白白浪费了本可用在他处的资源。关键在于，在满足要求的基础上留出一定的保险余量，但要避免资源浪费。

资源浪费也可导致设计失败，因为这种设计未能将某些资源加以利用，而这些资源本可用于对成功至关重要的其他一些领域。此外，浪费资源的解决方案还可能给人以未诚信履行合同的印象。

管理资源

管理资源是对所有可用的估量方案进行分析，选择可最大限度降低成本而又能满足系统要求的最适用解决方案的过程。这一过程包括了解每个设计决策的平衡点，确保在某一方面获得的益处不会被在另一方面产生的成本抵消。

例如，针对可用性进行水平扩展可能会提升总体可用性，但代价是需要增加维护和维修成本。针对性能进行垂直扩展可能会以经济的方式提高计算能力，但某些服务对这些附加能力的使用效率不高。

完成估量策略前，请对决策进行检查，确保平衡利用资源，使设计总体受益。这种检查通常是检查某一方面的系统特性如何对其他系统特性产生影响。下表列出了在资源管理上可能需要考虑的一些主题。

表 5-2 资源管理主题

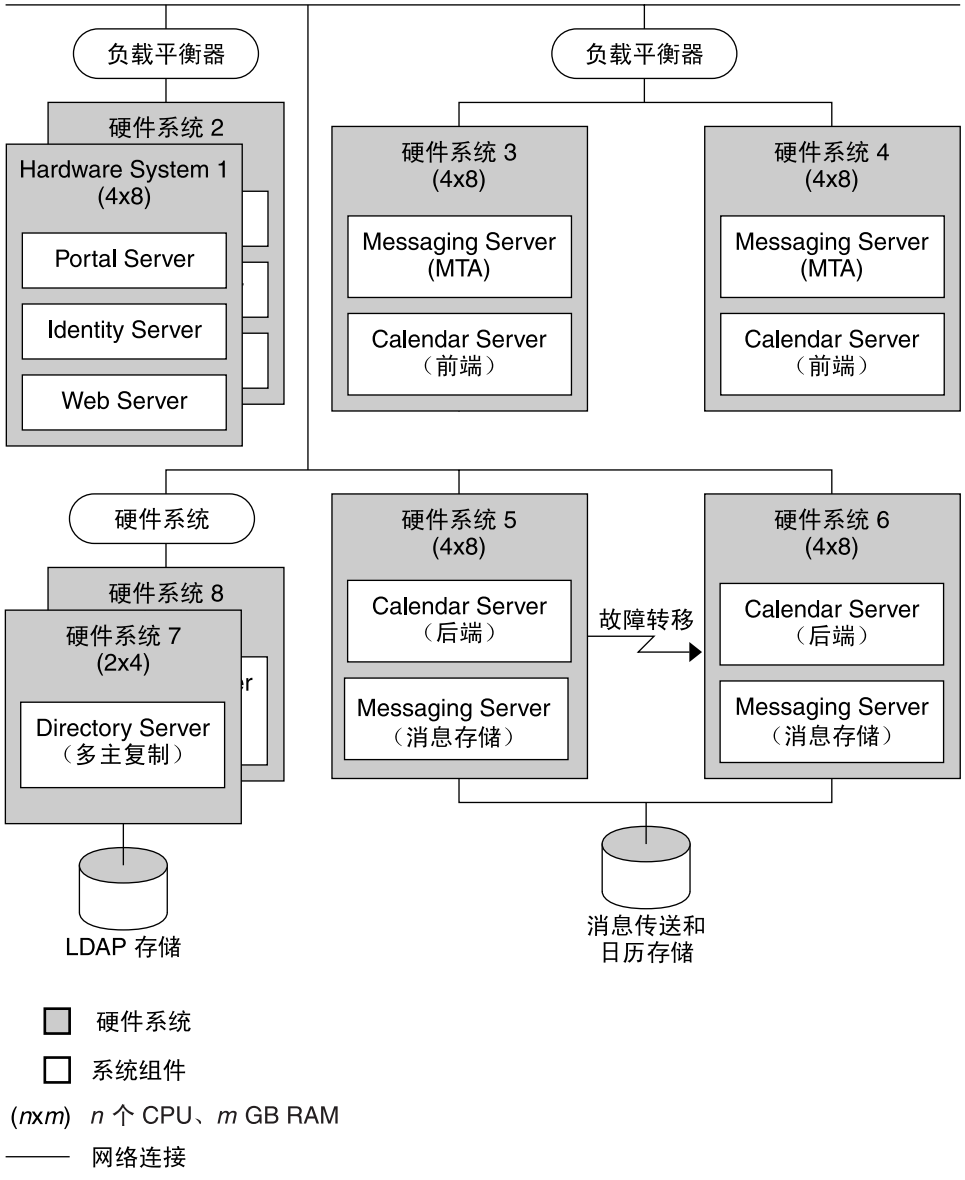
主题	说明
性能	对于将 CPU 集中分布在个别几台服务器上的性能解决方案，服务能否对计算能力加以高效利用。（例如，某些服务对可高效利用的 CPU 数量有上限。）
潜在容量	是否有处理超出性能估计的负载的策略？
	对于过载，是以在服务器上进行垂直扩展的方式，以负载平衡到其他服务器的方式，还是以这两种方式兼用的方式进行处理？
	在下一部署扩展重大事件点前，潜在容量是否足以处理出现的异常高峰负载？
安全性	是否对处理安全事务所需的性能开销给予了充分考虑？
可用性	对于水平冗余解决方案，是否对长期维护开支给予了充分估计？
	是否已将系统维护所需的计划停机考虑在内？
	是否在高端服务器和低端服务器间求得了成本平衡？
可伸缩性	是否对部署扩展的重大事件点进行了估计？
	是否制定了可在部署扩展重大事件点前提供足够容量来处理预测的负载增长的策略？
可维护性	是否在可用性设计中考虑了管理、监视和维护成本？
	是否考虑了采用委派管理解决方案 （允许用户自己执行某些管理任务）来降低管理成本？

示例部署体系结构

下图显示的是本白皮书前文中介绍的示例部署的完整部署体系结构。通过此图可大概了解部署体系结构的表示方法。

警告 下图中的部署体系结构只作说明之用，它不代表实际设计、构建或测试的部署，不应将其视为部署规划建议。

图 5-10 示例部署体系结构



详细设计规范

部署体系结构完成后是客户审核阶段，顺利的话，之后便是项目核准阶段。在某些情况下，客户可能会再次给出指示，要求对部署体系结构进行某些更改，然后才能核准。

项目核准后，便要制订详细的设计规范，而这是部署实现的起点。设计规范包括具体硬件资源和网络设备的详细信息以及详细的 LDAP 目录设置。

第 6 章

实现部署设计

本章概述实现部署设计所需的各个步骤。

部署体系结构获核准且已完成详细的设计规范后，便进入了部署规划的实现阶段。实现阶段的任务是拓展部署体系结构。实现部署设计包括以下某些或全部步骤，具体包含的步骤取决于部署项目的性质：

- 在测试环境中创建和部署试验性或原型系统
- 设计和运行功能性测试来衡量与系统要求的符合度
- 设计和运行负载测试来衡量峰值负载下的性能
- 创建生产部署，可能需要分阶段部署到生产中

本章阐述以下几节内容：

开发试验性和原型系统

Java Enterprise System 部署通常分为两类：一类是主要基于 Java Enterprise System 所提供服务的部署；另一类是需要大量与 Java Enterprise System 服务集成的定制服务的部署。可将前者视作一种 *80:20 部署*（80% 的服务由 Java Enterprise System 提供），同样可将前者视作一种 *20:80 部署*。

对于 80:20 部署，通常需在实现阶段开发用于测试的*试验性部署*。由于 80:20 部署使用的是提供即用性功能的成熟的 Java Enterprise System 服务，因此相对而言，试验性部署完成开发、测试和修改步骤，最终进至生产部署阶段的速度较快。

相反， 20:80 部署引入了新的定制服务，这些服务不具有 80:20 部署所具有的互操作性历史记录。因此需要创建*原型部署*，它们是概念验证式部署，通常需要经历更为严格的开发、测试、修改过程，才能进至生产阶段。

注 实际企业部署所需的定制服务开发的数量会有很大差异。使用试验性部署还是原型部署进行测试取决于部署的复杂程度和性质。

测试试验性部署和原型部署

测试试验性部署和原型部署的目的是，在测试条件下尽一切可能确定部署是否既能满足系统要求，又可实现业务目标。

理想情况下，*功能性测试*应基于所有已确定使用案例模拟各种方案 — 应开发一套标准来衡量符合性。功能性测试还可包含将系统有限地部署给选定的一组试用版用户，以确定其能否满足业务要求。

*负载测试*衡量在峰值负载下的性能。这些测试通常使用一系列模拟环境和负载发生器来衡量数据吞吐量和性能。部署的系统要求通常是设计和通过负载测试的基础。

注 对于系统要求未经明确定义、没有可作为估量基础的先前实现且需要进行大量全新开发的大型部署，功能性测试和负载测试尤其重要。

通过测试能够发现部署设计规范存在的问题，并可能要经过若干次反复设计、生成和测试，才能向生产环境展开部署。不过，测试决不应是发现部署体系结构问题的所在。如果在测试阶段发现部署体系结构存在设计问题，便可将分析、规划和设计视作失败。

展开生产部署

试验性或概念验证部署符合测试标准后，即可向生产环境展开部署。向生产环境展开部署通常分阶段进行。分阶段展开对会影响大量用户的大型部署具有尤其重要的意义。

分阶段部署可以先向一小节用户部署，然后逐步扩大用户范围，直至将其部署给所有用户。分阶段部署也可这样进行：先部署一定类型的服务，然后逐步引入其余类型的服务。分阶段升级服务有助于隔离、确定和解决服务可能会在生产环境中遇到的任何问题。

由于测试永远不可能完全模拟生产环境，因此应继续对部署的系统进行监控，以确定是否有需要调节、维护或维修的节。