第3章 软件体系结构风格

内容

- 3.1 概述
- 3.2 数据流风格
- 3.3 过程调用风格
- 3.4 独立构件风格
- 3.5 层次风格
- 3.6 虚拟机风格

- 3.7 客户/服务器风格
- 3.8 表示分离风格
- 3.9 插件风格
- 3.10 微内核风格
- 3.11 SOA风格

客户/服务器风格

- 3.7.1 计算模式的演化史
- 3.7.2 客户机/服务器(C/S)架构
- 3.7.3 二层C/S架构
- 3.7.4 三层C/S架构
- 3.7.5 浏览器/服务器(B/S)架构
- 3.7.6 C/S+B/S混合体系结构
- 3.7.7 案例分析

3.7.1 计算模式的演化史

计算机历史发展


计算机历史发展
大型机
小型机
工作站
PC
笔记本
PDA
手机
年代
大规模数据存储和计算
人机交互
真实世界的数据流

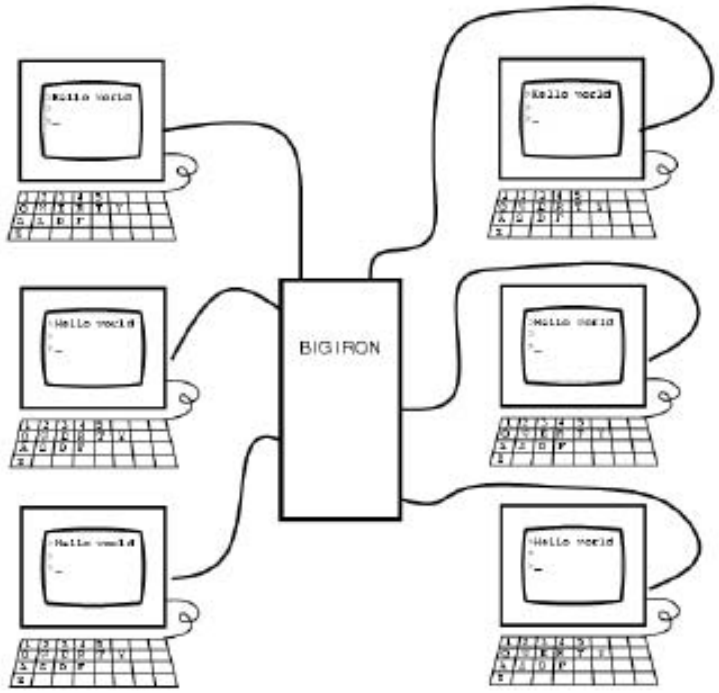
计算模式的演化

- 计算模式大致经历了如下几代：
  - 1965-1985：以大型机为核心的集中式处理模式；
  - 1986-1990：以PC/文件服务器为核心的文件共享计算模式；
  - 1990-1996：以C/S结构为主流的分布式计算模式；
  - 1996-：以Web为核心、B/S结构为主流的分布式计算模式；
  - 2000-：以各类移动设备为核心的普适计算模式(无所不在的计算，无所不在的通讯)；
  - 2005-：以Web3.0技术为核心的分布式计算模式；
  - 目前和未来一段时间，以移动互联网（**Mobile Internet**）、云计算（**Cloud Computing**）、物联网（**Internet of Things**）技术为核心的新型计算模式。

集中式计算模式
(Mainframe Computer)

- With mainframe software architectures all intelligence is within the central host computer. (所有的计算能力均属于中央宿主计算机)
- Users interact with the host through a terminal that captures keystrokes and sends that information to the host. (用户通过一台物理上与宿主机相连接的非智能终端来访问宿主机上的应用程序)
- User interaction can be done using PCs and UNIX workstations. (客户机可能为PC或工作站)

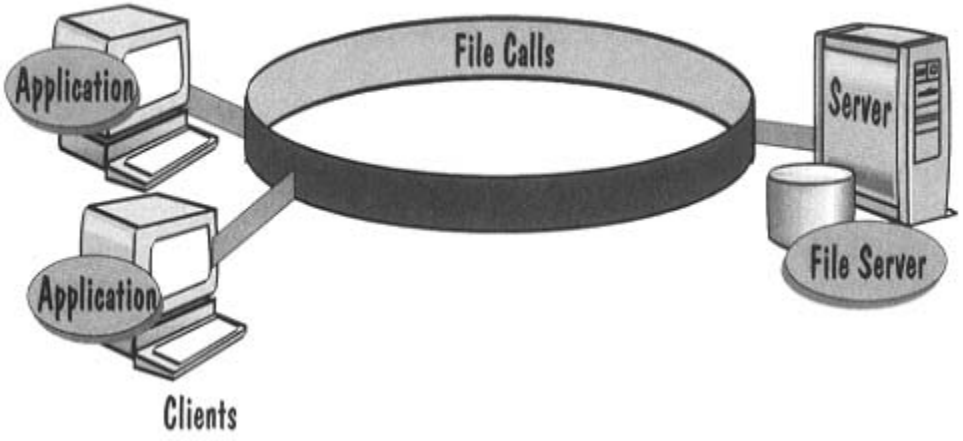
集中式计算模式
(Mainframe Computer)

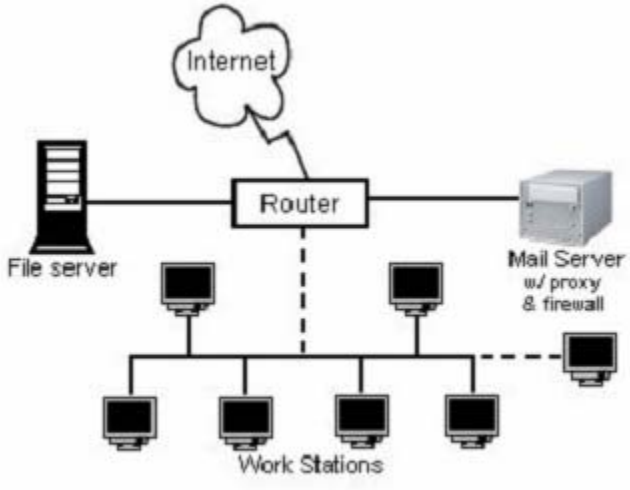


A 1990 *Honeywell-Bull DPS 7* mainframe CPU

传统文件共享体系结构
(File Sharing Architecture)

- 文件共享体系结构：文件存储在一个中央计算机或者共享服务器中，被网络上的多个计算机同时访问。





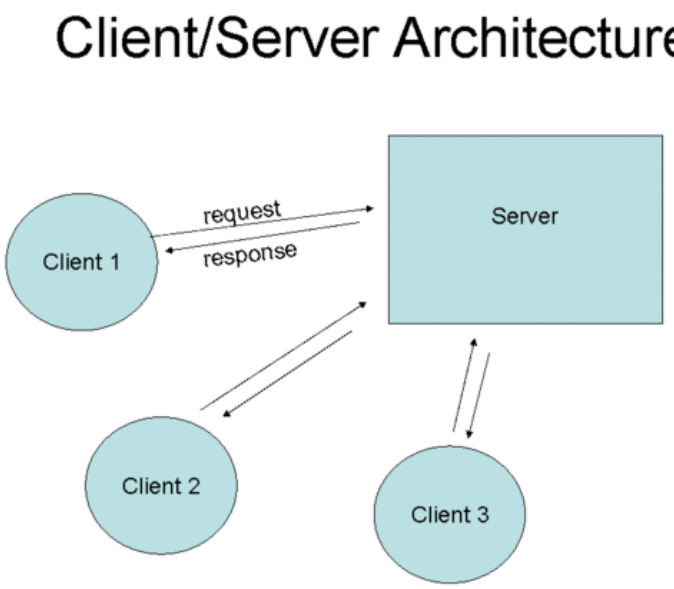
传统文件共享体系结构
(File Sharing Architecture)

- The original PC networks were based on file sharing architectures, where the server downloads files from the shared location to the desktop environment. (最初的PC网络就是基于此类结构，从共享服务器下载文件到客户机的桌面环境下)
- The requested user job is then run (including logic and data) in the desktop environment. (被请求的用户任务-包括业务逻辑和数据-在客户机环境下执行)
- File sharing architectures work if shared usage is low, update contention is low, and the volume of data to be transferred is low. (适合应用于传输数据量较低的场合)

传统文件共享体系结构
(File Sharing Architecture)

- 缺陷：
  - 客户端和服务器之间需要移动大量不必要的数据，降低了应用性能；
  - 客户端必须相当健壮，它要完成几乎所有的功能，同时必须有足够的磁盘空间来存储下载的文件和表；
  - 容易破坏数据完整性(多个用户共同访问同一个文件)；
  - 对环境变化及用户需求变更的适应性差，一旦发生变化，客户机与服务器的程序都要修改，增加了维护工作量。

3.7.2 客户机/服务器
Client-Server Architecture

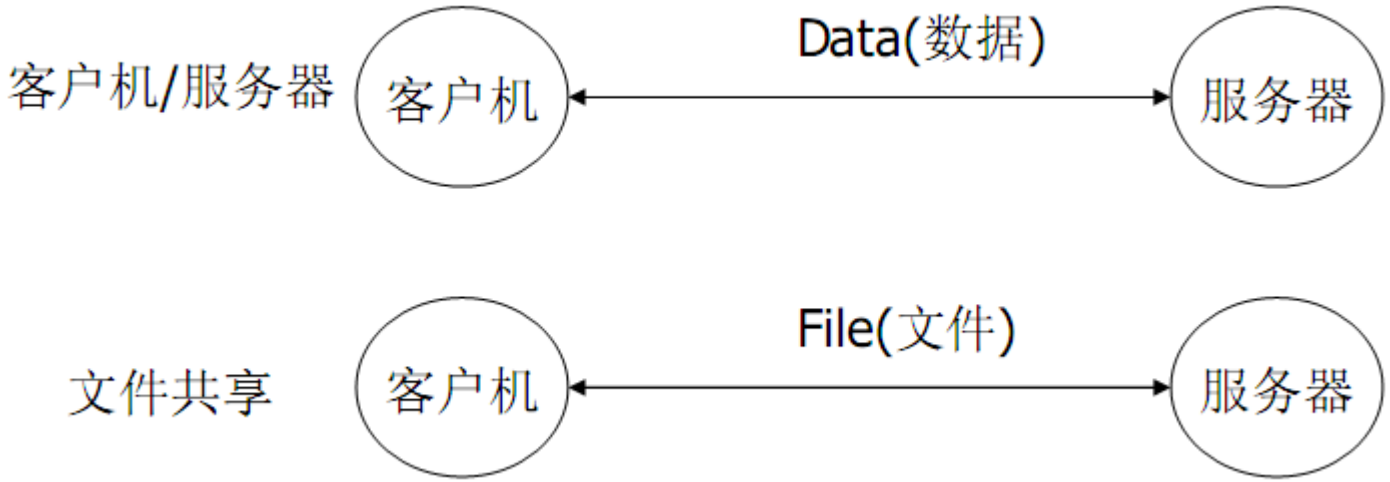


客户机/服务器
(Client-Server Architecture)

- As a result of the limitations of file sharing architectures, the client/server architecture emerged (文件共享结构的缺陷导致了C/S架构的出现)
  - This approach introduced a database server to replace the file server. (数据库服务器代替了文件服务器)
  - Using a relational DBMS, user queries could be answered directly. (服务器使用DBMS，快速应答用户请求)
  - RPCs or SQL statements are typically used to communicate between the client and server. (RPC或SQL是客户机和服务器之间的典型通讯模式)

客户机/服务器
(Client-Server Architecture)

- Advantages:
  - The client/server architecture reduced network traffic by providing a query response rather than total file transfer. (C/S降低了网络通讯量：提供请求/应答模式，而非文件传输)
  - It improves multi-user updating through a GUI front end to a shared database. (多用户通过GUI访问共享数据库)



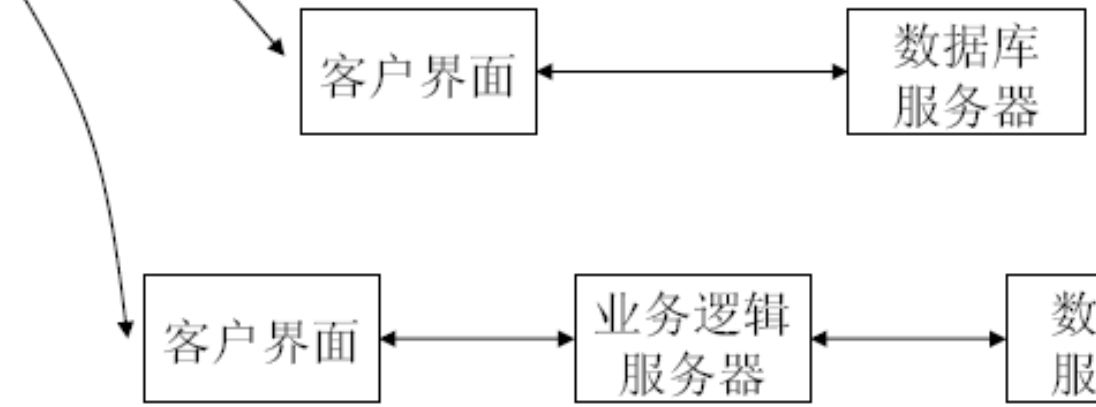
客户机/服务器
(Client-Server Architecture)

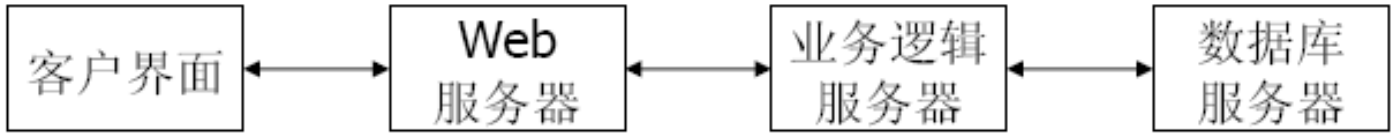
- 客户机/服务器：一个应用系统被分为两个逻辑上分离的部分，每一部分充当不同的角色、完成不同的功能，多台计算机共同完成统一的任务。
  - 客户机(前端，front-end)：业务逻辑、与服务器通讯的接口；
  - 服务器(后端：back-end)：与客户机通讯的接口、业务逻辑、数据管理。

- 一般的，
  - 客户机为完成特定的工作向服务器发出请求；
  - 服务器处理客户机的请求并返回结果。

客户机/服务器
(Client-Server Architecture)

- C/S结构的发展历程：
  - 两层C/S
  - 三层C/S
  - 多层C/S





客户机/服务器
(Client-Server Architecture)

- 在两层和多层C/S中，每两层之间形成C/S关系；
  - 例如，在三层C/S中，应用服务器既可以看作是客户机的“服务器”，同时也是数据库服务器的“客户机”；
  - 每个服务器充当不同的角色，完成不同的任务；
    - 终结服务器：不再是其他服务器的“客户端”；
    - 非终结服务器：需要得到其他服务器的支持；
- 服务器的分类：
  - 信息/数据库服务器：存储和处理数据；
  - 操作服务器：提供业务逻辑服务；
  - 管理类服务器：DNS服务器、路由服务器、消息服务器等。

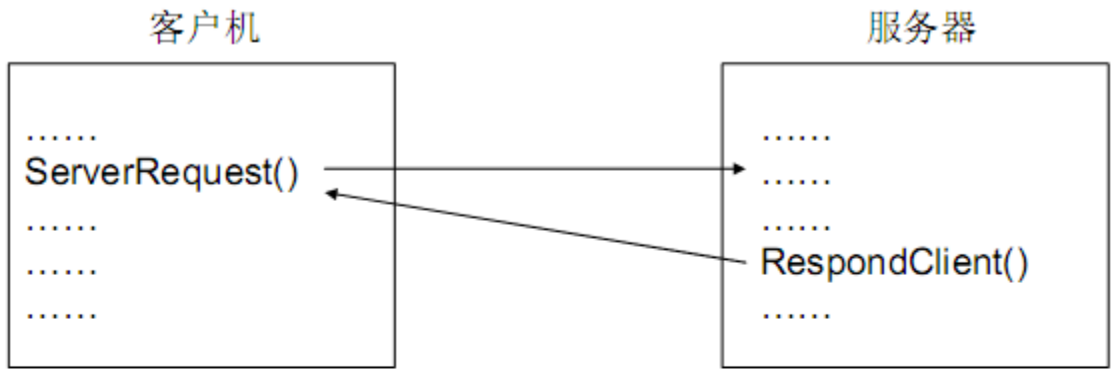
客户机/服务器的组成

- 客户机：
  - GUI；
  - 请求表达；
  - 业务逻辑；
  - 服务代理及通信；
- 服务器：
  - 服务器接口；
  - 调度管理；
  - 事务处理；
  - 业务逻辑；
  - 共享资源管理；
  - 通信；

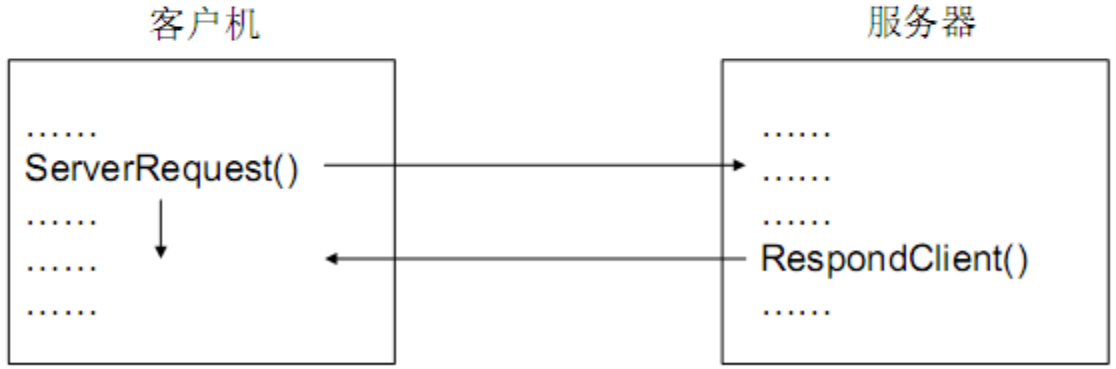
- 二者之间具有通信连接机制，遵循公共的通信协议：
  - 请求的表达；
  - 返回结果的表达；
  - 连接关系和状态的表达
- 例如：
  - RPC(远程过程调用)；
  - HTTP；
  - SOAP(简单对象访问协议)；

客户机/服务器的连接

- 连接的接口：
  - 基于过程：同步响应
  - 基于消息：异步响应



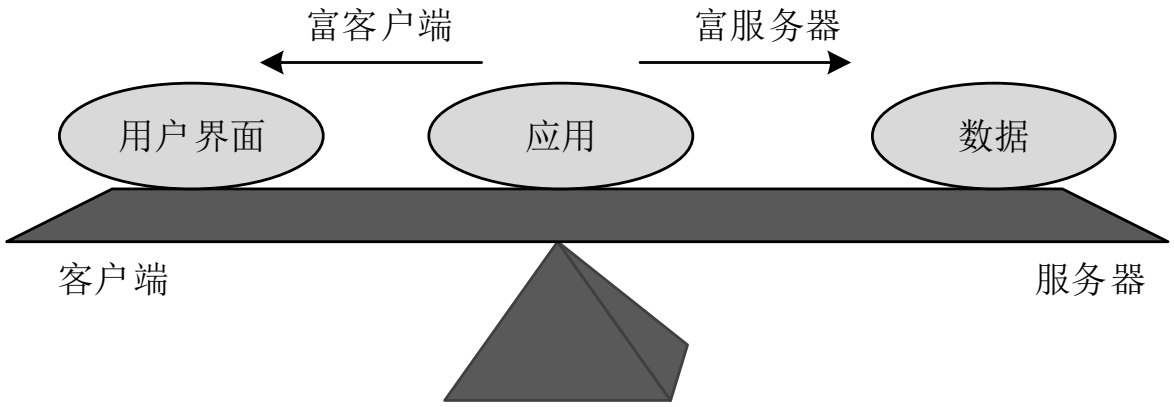
同步响应机制



异步响应机制

胖客户端 vs. 瘦客户端
（Fat client vs. Thin client）

- 业务逻辑的划分比重：在客户端多一些还是在服务器端多一些？
  - Fat Client: a client that performs the bulk of the data processing operations; (胖客户端：客户端执行大部分的数据处理操作)
  - Thin Client: has little or no application logic, so it has to depend primarily on the central server for processing activities. (瘦客户端：客户端具有很少或没有业务逻辑)



胖客户端特点

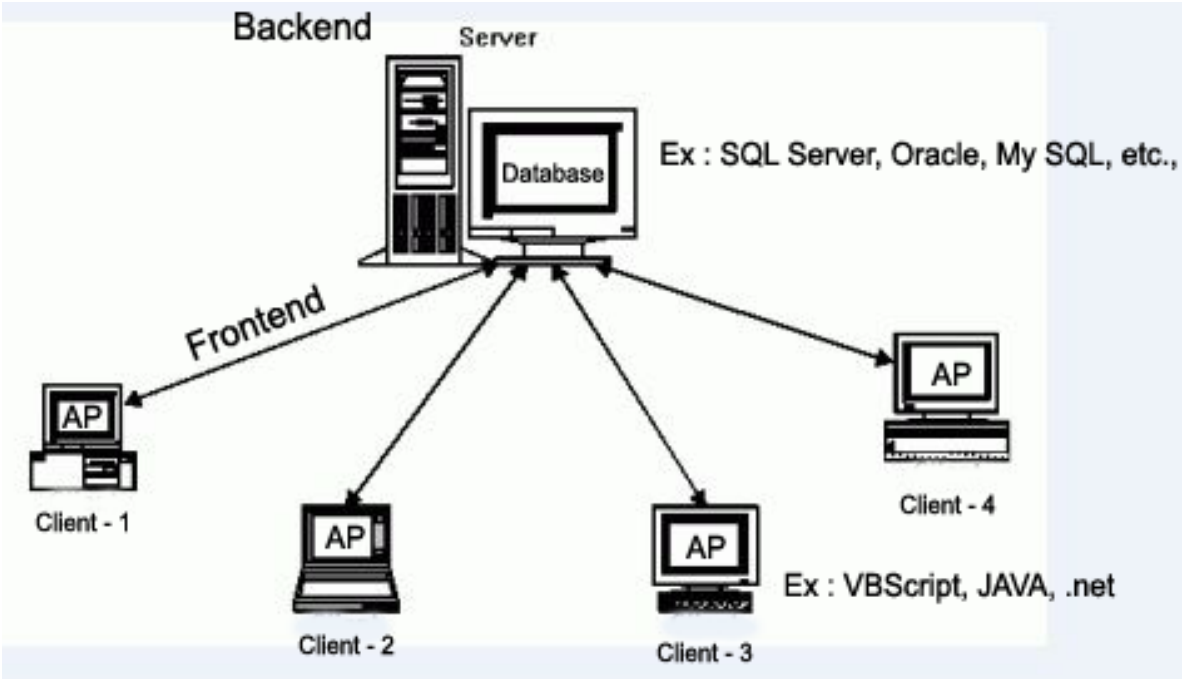
- 较低的服务器需求。与瘦客户端对应的服务器相比，胖客户端对应的服务器不需要达到同样高的性能（因为胖客户端自己就可以完成很多业务处理）。因此，可以使用非常便宜的服务器。
- 离线工作。胖客户端通常不需要和服务器之间维持长久的连接。
- 更好的多媒体性能。在网络带宽比较敏感的时候，胖客户端更擅长运行丰富的多媒体应用，例如，胖客户端非常适合于电子游戏。
- 更灵活。在某些个人电脑的操作系统上，软件产品都有它们自己的本地资源。在瘦客户端环境运行这类软件可能比较困难。
- 充分利用已有设备资源。现在很多人都拥有非常快速的个人电脑，他们已经不需要额外的花费就可以运行胖客户端软件了。
- 更高的服务器容量。客户端执行的工作越多，服务器所需要的工作就越少，这可以增加服务器支持的用户数量。

瘦客户端特点

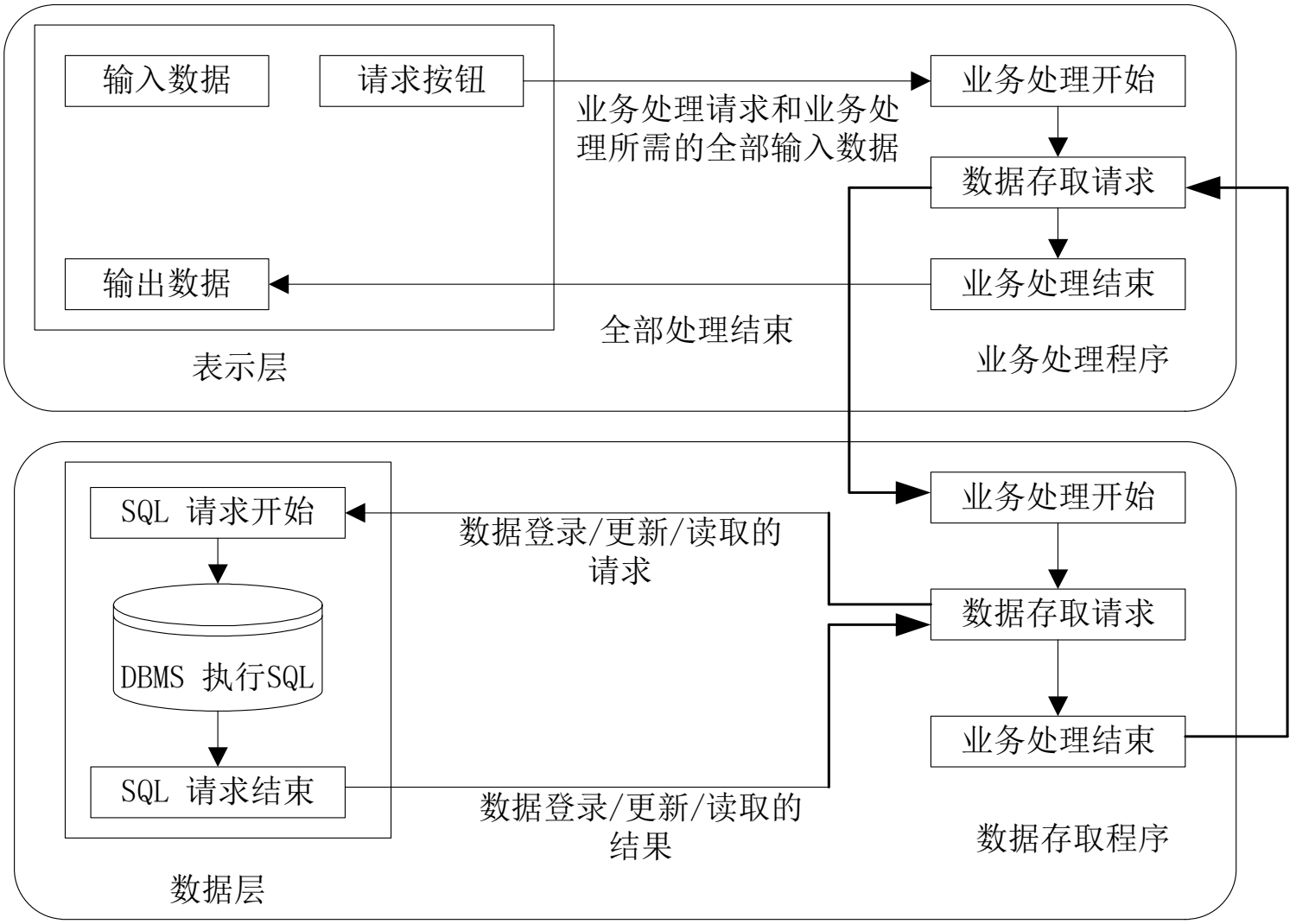
- 单点故障。服务器承担了大部分的业务处理，安全问题主要集中于服务器，易于维护，但服务器一旦崩溃，所有数据都会丢失。
- 廉价的客户端硬件。对客户端硬件的内存、硬盘容量以及CPU等要求不高，这也降低了客户端的功耗。
- 有限的显示性能。瘦客户端往往使用简单的直线、曲线和文字来优化显示性能，或通过缓存的位图数据来进行显示，很难支持复杂的交互式图形操作。

3.7.3 两层C/S结构
Two tier Client-Server Architecture

两层C/S结构
(2-Tier Client/Server)



两层C/S结构
(2-Tier Client/Server)



两层C/S结构
(2-Tier Client/Server)

- With two tier client/server architectures,
  - the user system interface is usually located in the user's desktop environment (用户界面处于客户机)
  - the database management services (usually stored procedures and triggers) are usually resides on a server that is a more powerful machine that services many clients (数据库管理服务处于服务器端，通常是存储过程/触发器的形式)
  - processing management (application or business logic) is split between the user system interface environment and the database management server environment. (业务处理过程—即业务逻辑—被分解为客户机与服务器两部分)

两层C/S结构
(2-Tier Client/Server)

- 基本构件：
  - 数据库服务器：存放数据的数据库、负责数据处理的业务逻辑；
  - 客户机应用程序：
    - GUI：用户界面
    - 业务逻辑：利用客户机上的应用程序对数据进行处理；
- 连接件：经由网络的调用-返回机制或事件机制。
  - 客户机←→服务器：客户机向服务器发送请求，并接收返回结果。

Limitations of 2-Tier Client/Server

- Scalability. When the number of users exceeds 100, performance begins to deteriorate. This limitation is a result of the server maintaining a connection via “keep-alive” messages with each client, even when no work is being done. (系统伸缩性差：当用户数超过100，性能急剧恶化)
  - 服务器成为系统的瓶颈。
- Interoperability. Implementation of processing management services using vendor proprietary database procedures restricts flexibility and choice of DBMS for applications. (互操作性差：使用DBMS所提供的私有的数据编程语言来开发业务逻辑，降低了DBMS选择的灵活性) ——导致：软件移植困难，新技术无法轻易使用
  - 例如：Oracle DB提供的若干存储过程函数，在SQL Server上就不能执行。

Limitations of 2-Tier Client/Server

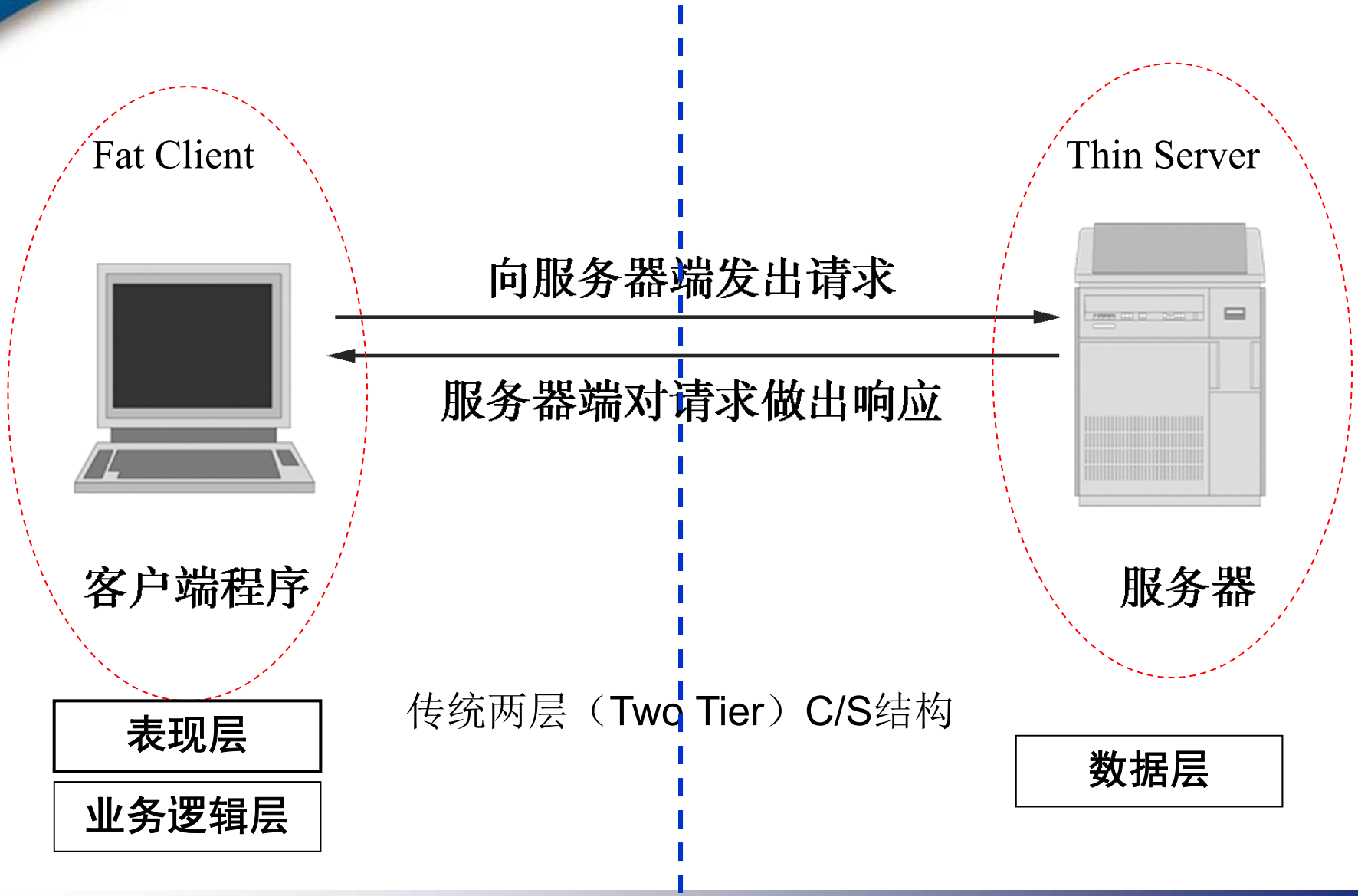
- **System administration and configuration.** Two tie architectures can be difficult to administer and maintain because when applications reside on the client, every upgrade must be delivered, installed, and tested on each client. (系统管理与配置成本高：当系统升级时，每个客户端都需要随之改变) ——导致：维护和升级困难

Usage Considerations

- Two tier C/S architectures are used extensively in non-time critical information processing where management and operations of the system are not complex. (两层C/S架构通常被用在那些管理与操作不太复杂的非实时的信息处理系统)
- This design is used frequently in decision support systems where the transaction load is light. (适合于轻量级事务) ——客户机对服务器的请求少，数据传输量少
- The two tier architecture works well in relatively homogeneous environments with processing rules (business rules) that do not change very often and when workgroup size is expected to be fewer than 100 users, such as in small businesses. (当业务逻辑较少变化以及用户数少于100时，两层C/S架构的性能较好)

3.7.4 三层C/S结构
Three tier Client-Server Architecture

传统客户服务器风格

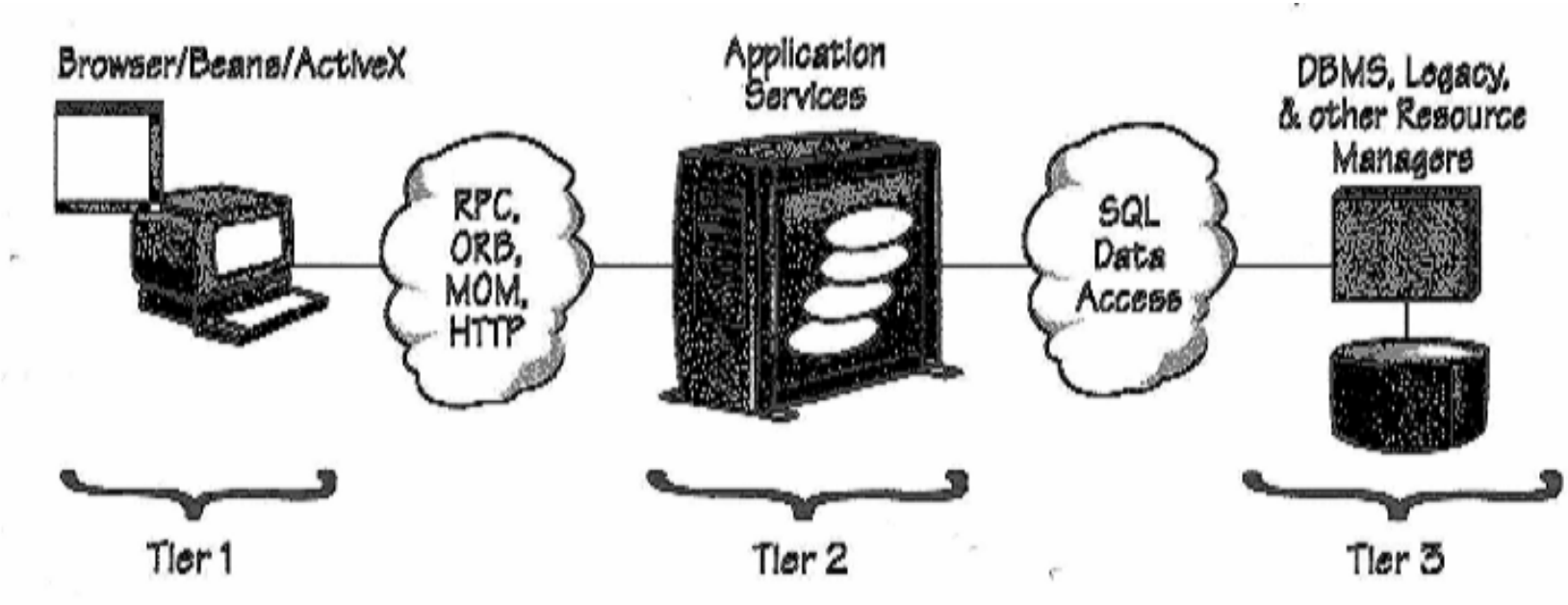

Fat Client
Thin Server
向服务器端发出请求
服务器端对请求做出响应
客户端程序
服务器
表现层
业务逻辑层
传统两层（Two Tier）C/S结构
数据层

三层C/S结构
(3-Tier C/S Architecture)

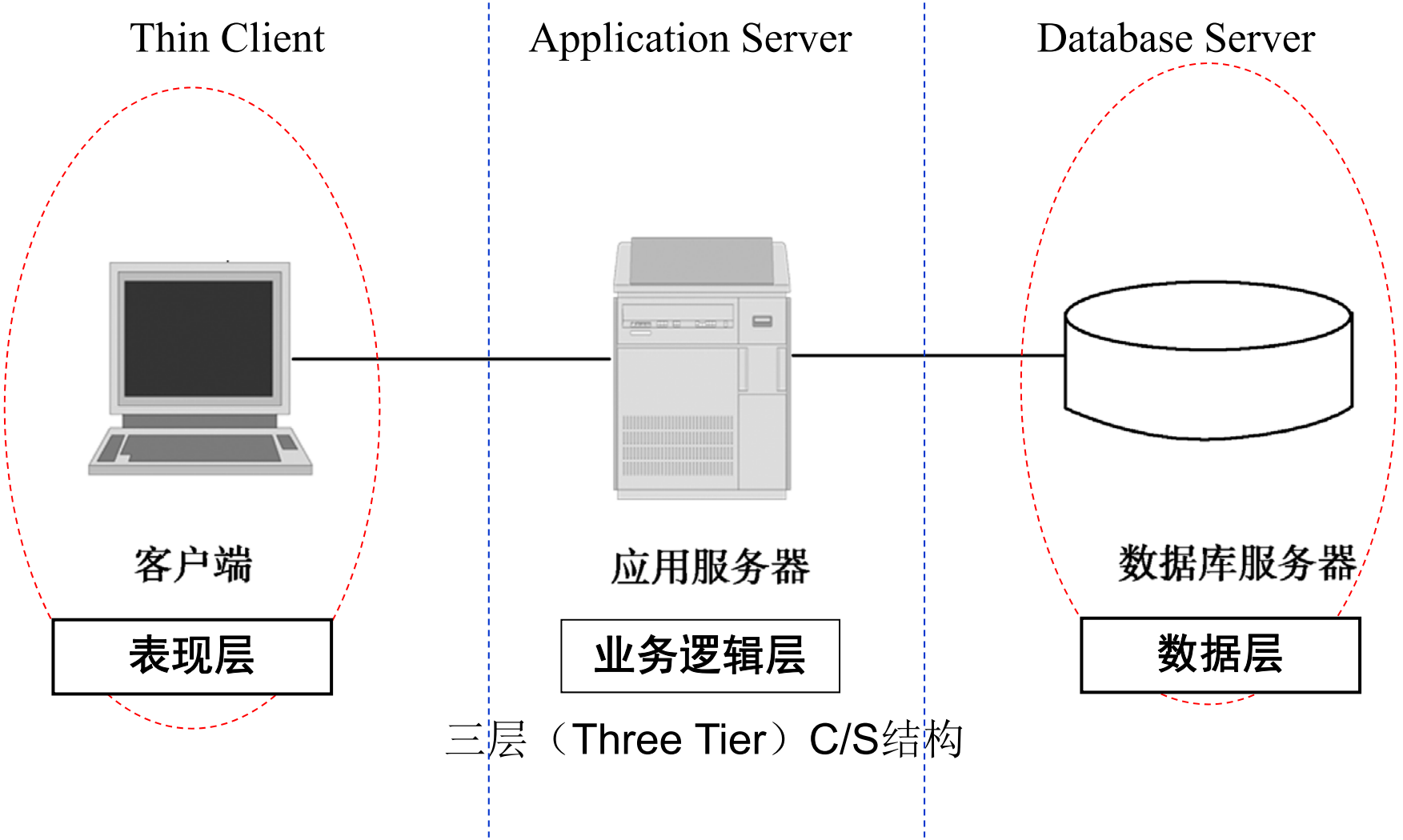
- The three tier architecture emerged to overcome the limitations of the two tier architecture.(三层C/S体系结构的出现克服了两层C/S的缺陷)
- In the three tier architecture, a middle tier was added between the user system interface client environment and the database management server environment. (在客户端与数据库服务器之间增加了一个中间层)
  - There are a variety of ways of implementing this middle tier, such as transaction processing monitors, message servers, or application servers. (中间层可能为：事务处理服务器、消息服务器、应用服务器等)
  - The middle tier can perform queuing, application execution, and database staging. (中间层负责调度、业务逻辑执行、数据传输等功能)

三层C/S结构
(3-Tier C/S Architecture)

- Tier 1: GUI; (第一层：用户界面—表示层)
- Tier 2: Business Logic; (第二层：业务逻辑—功能层)
- Tier 3: Data; (第三层：数据库—数据层)

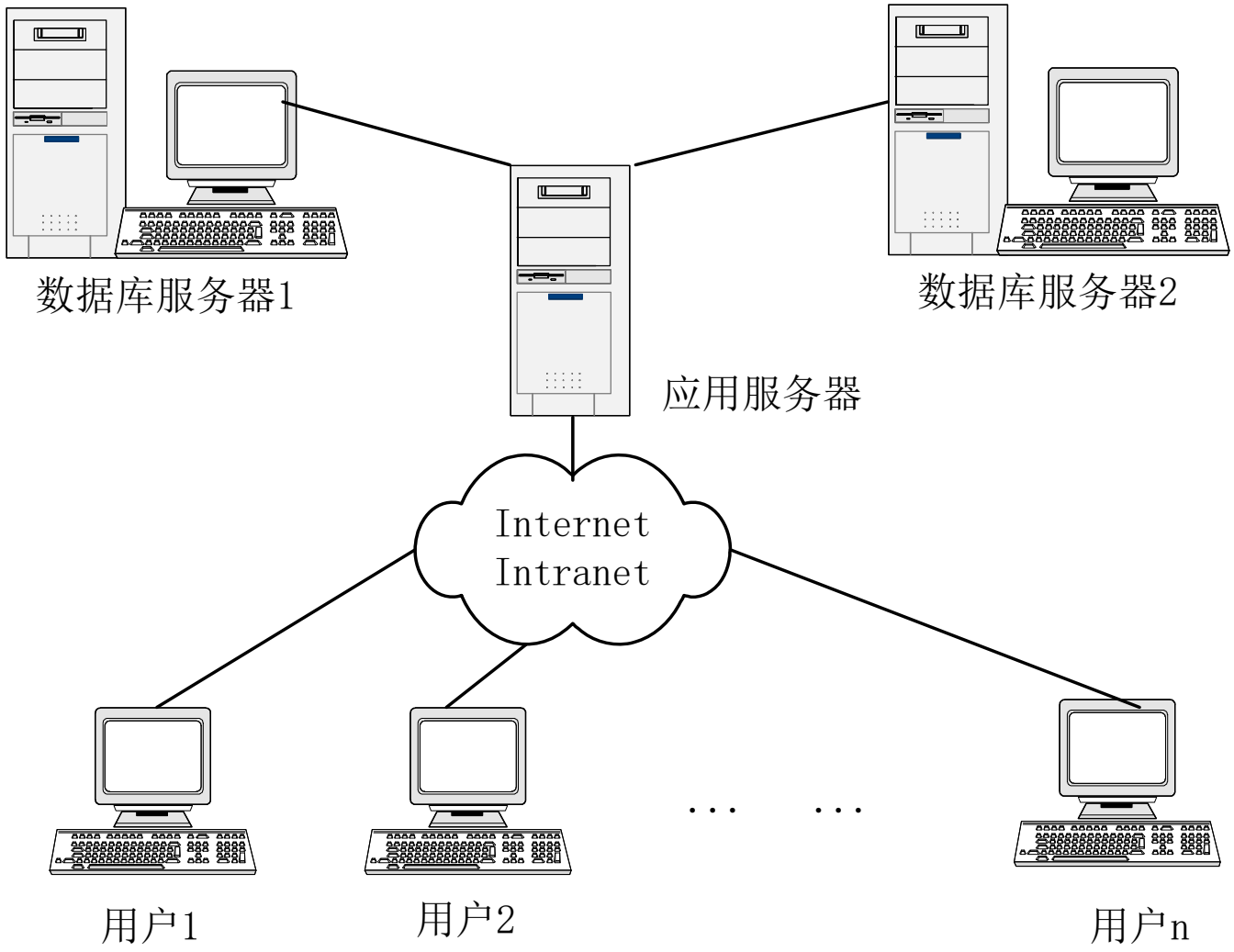


三层C/S结构
(3-Tier C/S Architecture)

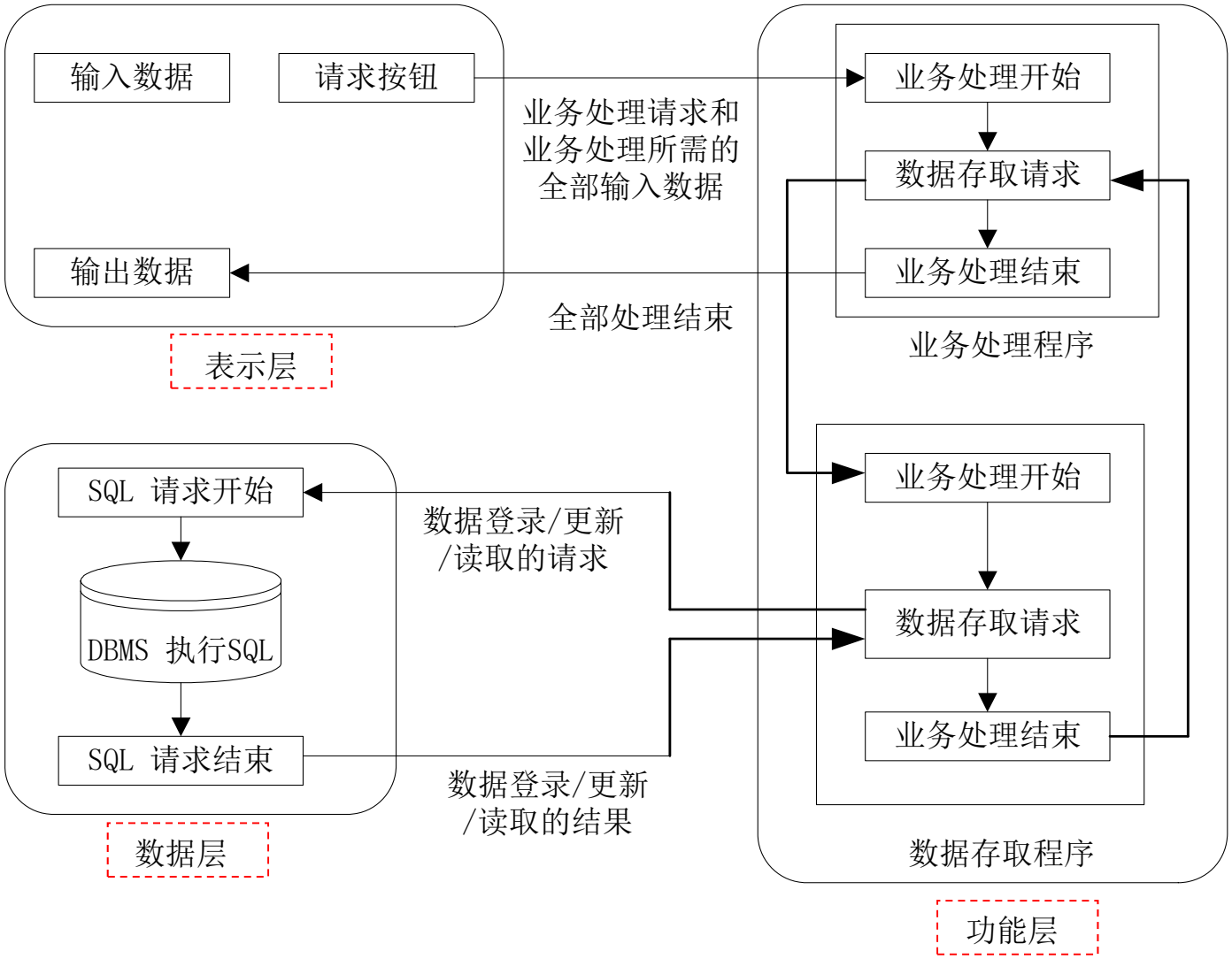


三层（Three Tier）C/S结构

三层C/S结构
(3-Tier C/S Architecture)



三层C/S结构
(3-Tier C/S Architecture)



表示层

- 用户接口部分，担负着用户与应用之间的对话功能；
- 检查用户的输入，显示应用的输出；
- 通常使用GUI；
- 在变更时，只需要改写显示控制和数据检查程序，而不影响其他层；
- 不包含或包含一部分业务逻辑。

功能层

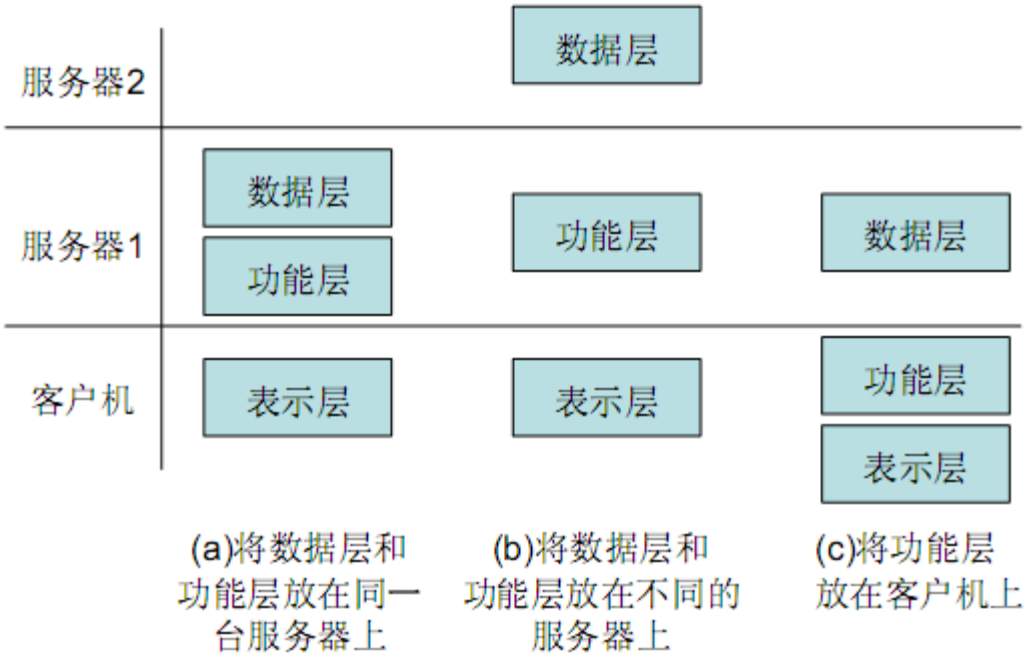
- 应用系统的主体，包括大部分业务处理逻辑 (通常以业务构件的形式存在，如DLL/JavaBean/EJB/COM等)；
- 从表示层获取用户的输入数据并加以处理；
- 处理过程中需要从数据层获取数据或向数据层更新数据；
- 处理结果返回给表示层加以输出给用户。

数据层

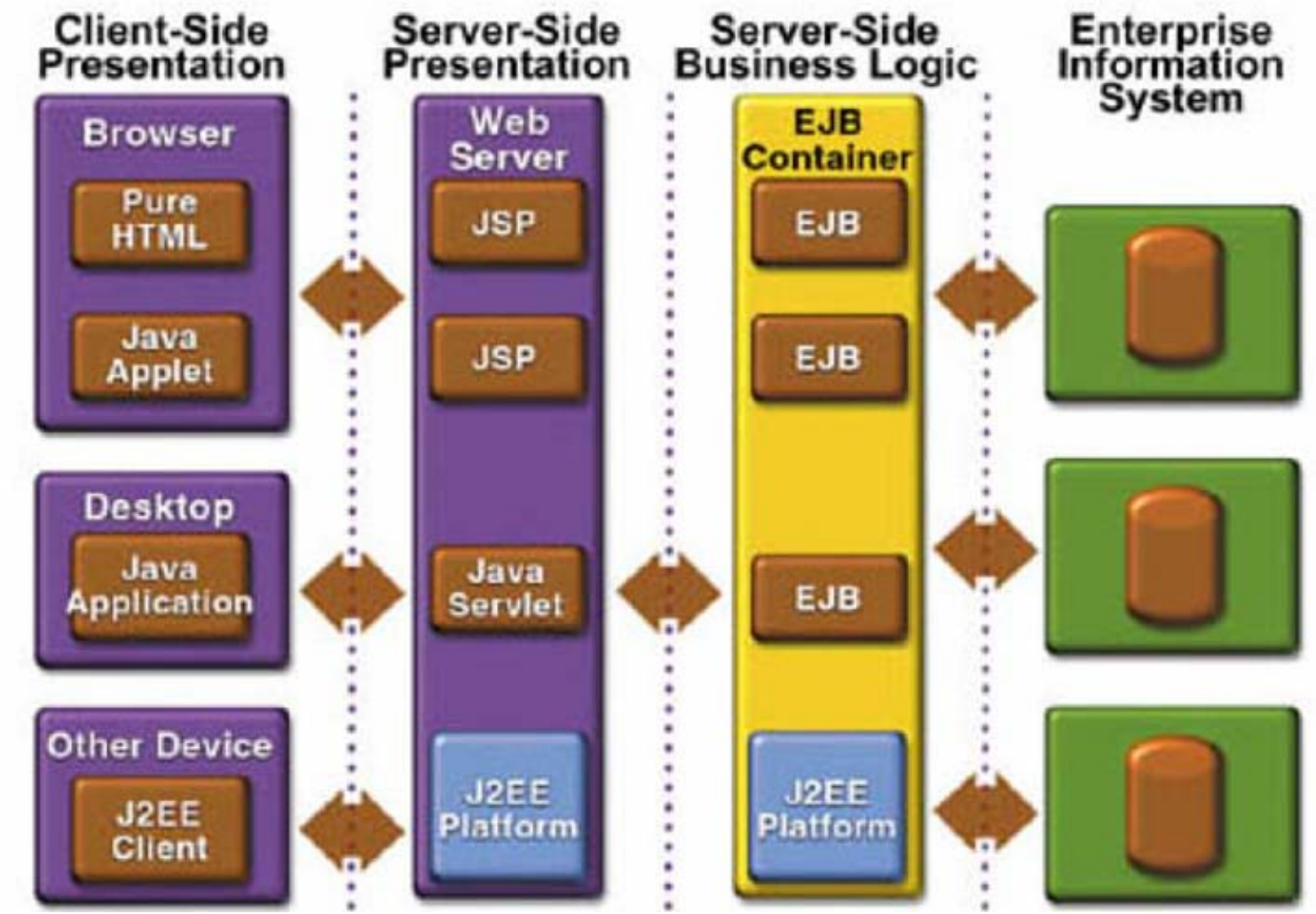
- DMBS；
- 接受功能层的数据查询请求，执行请求，并将查询结果返回给功能层；
- 从功能层接受数据存取请求，并将数据写入数据库；
- 请求的执行结果也要返回给功能层。

三层C/S结构的物理结构

- 两层C/S已经将数据层分离出来，三层C/S则要将表示层与功能层分离开来，形成独立的程序，并使二者之间的接口简洁明了。
- 问题：这三个层次在物理上是如何分布的？



多层体系结构例子—J2EE
(Java 2 platform,Enterprise Edition)



三层C/S结构的优点（1）
(Advantages of 3-Tier C/S Architecture)

- 在用户数目较多的情况下，三层C/S结构将极大改善性能与灵活性(通常可支持数百个甚至更多用户)；
- 允许合理地划分三层结构的功能，使之在逻辑上保持相对独立性，能提高系统和软件的可维护性和可扩展性；——UI、BL、DB可以分别加以复用
- 允许更灵活有效地选用相应的平台和硬件系统，并且这些平台和各个组成部分可以具有良好的可升级性和开放性。

三层C/S结构的优点（2）
(Advantages of 3-Tier C/S Architecture)

- 应用的各层可以并行开发，可以选择各自最适合的开发平台和开发语言。
- 利用功能层有效地隔离开表示层与数据层，未授权的用户难以绕过功能层而非法的访问数据层，为严格的安全管理奠定了坚实的基础。
- 将遗留系统移植到三层C/S下将非常容易；

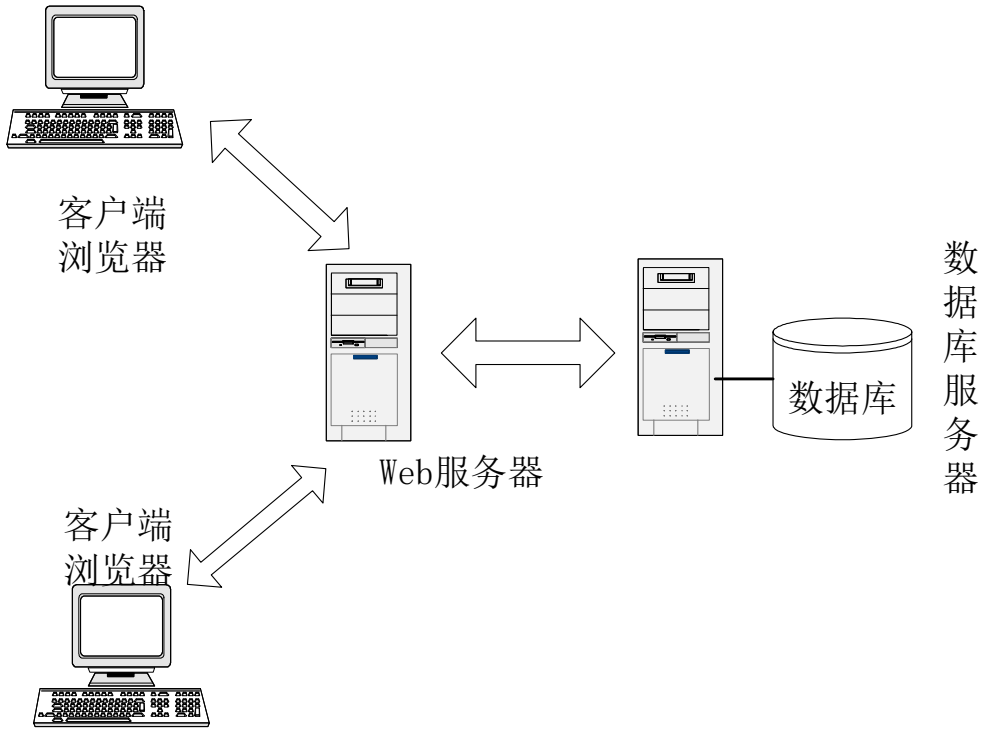
三层C/S结构的缺点
(Disadvantages of 3-Tier C/S Architecture)

- 三层C/S结构各层间的通信效率若不高，即使分配给各层的硬件能力很强，其作为整体来说也达不到所要求的性能。
- 设计时必须慎重考虑三层间的通信方法、通信频度及数据量，这和提高各层的独立性一样是三层C/S结构的关键问题。——分层风格的固有缺点

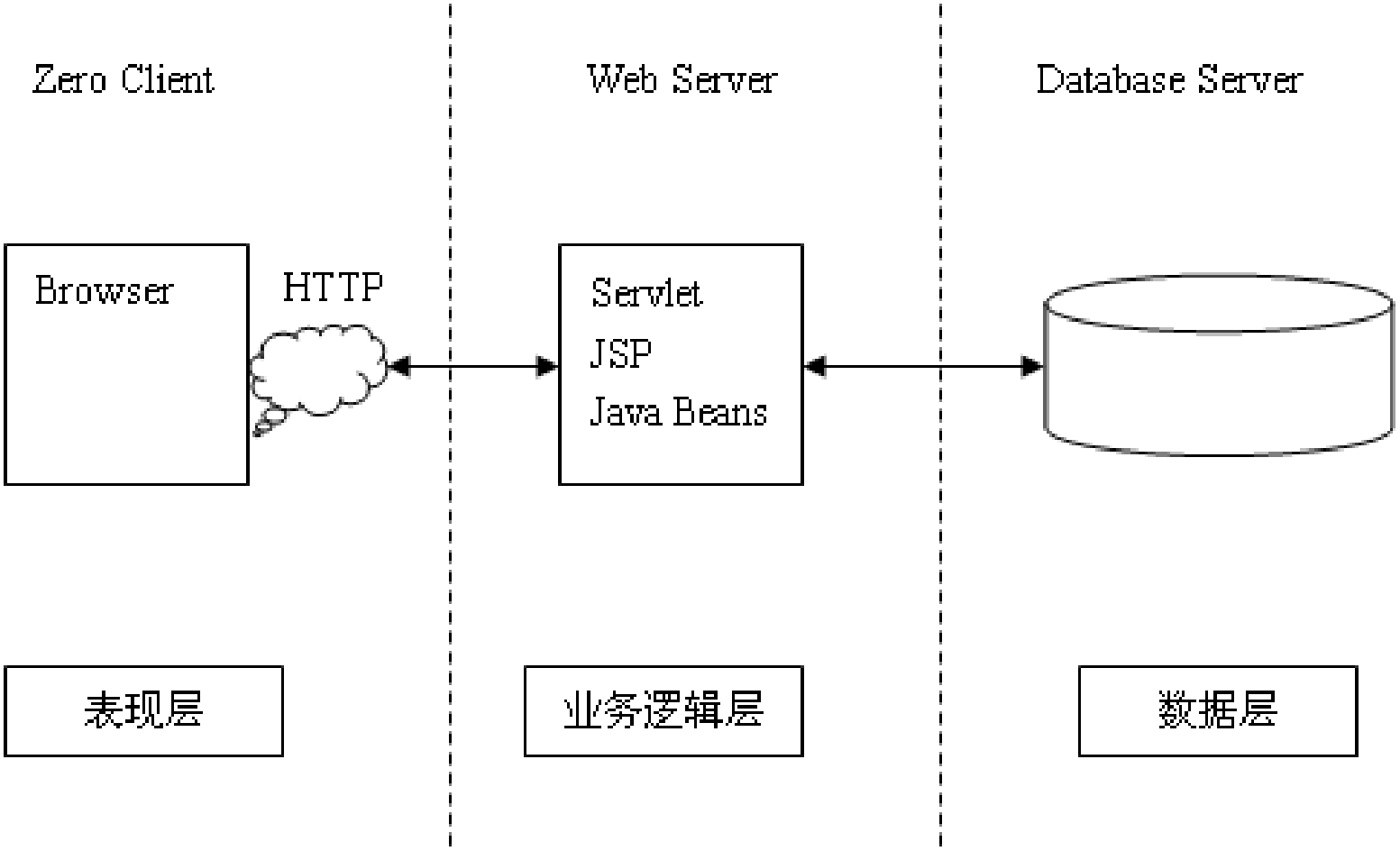
3.7.5 浏览器/服务器 (Browser/Server Architecture)

浏览器/服务器
(Browser/Server Architecture)

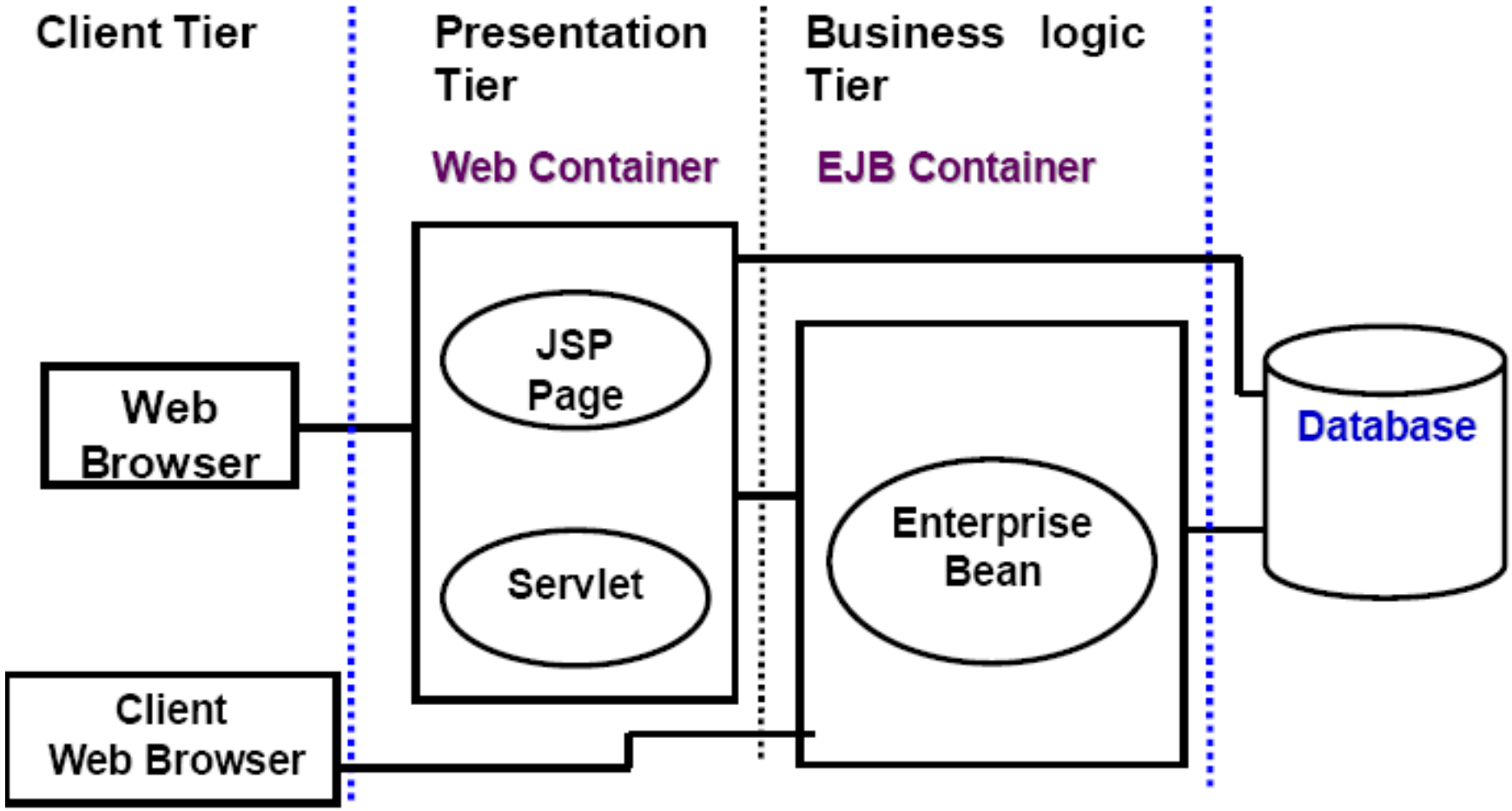
- 浏览器/服务器(B/S)是三层C/S风格的一种实现方式。
  - 表现层：浏览器
  - 逻辑层：
    - Web服务器
    - 应用服务器
  - 数据层：数据库服务器



浏览器/服务器
(Browser/Server Architecture)

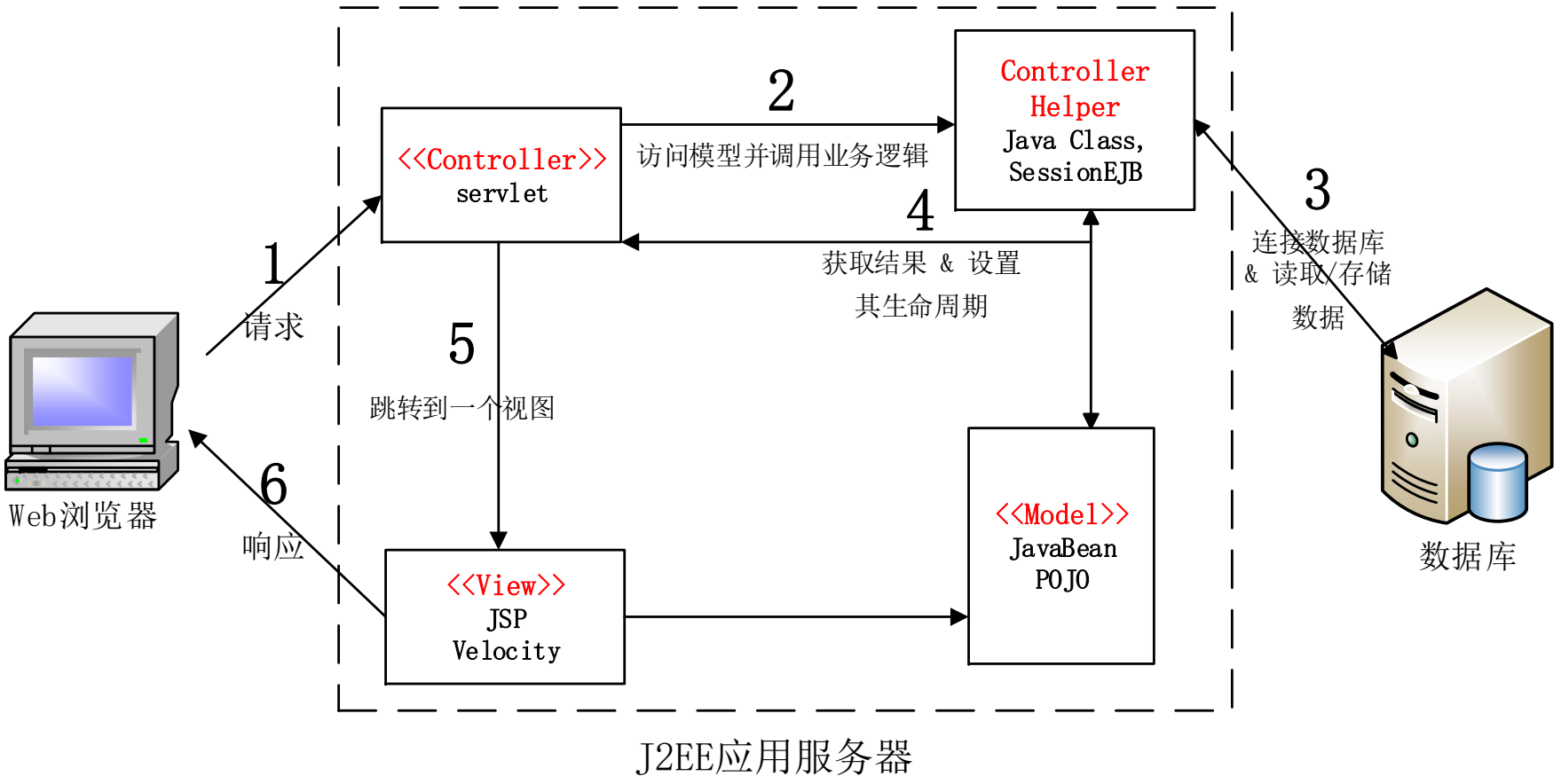


浏览器/服务器
(Browser/Server Architecture)



J2EE平台典型B/S结构的实现方式

Java Model 2架构



B/S架构的优点（1）

- 基于B/S体系结构的软件，系统安装、修改和维护全在服务器端解决，系统维护成本低：
  - 客户端无任何业务逻辑，用户在使用系统时，仅仅需要一个浏览器就可运行全部的模块，真正达到了“零客户端”的功能，很容易在运行时自动升级。
  - 良好的灵活性和可扩展性：对于环境和应用条件经常变动的情况，只要对业务逻辑层实施相应的改变，就能够达到目的。

B/S架构的优点（2）

- 较好的安全性：在这种结构中，客户应用程序不能直接访问数据，应用服务器不仅可控制哪些数据被改变和被访问，而且还可控制数据的改变和访问方式 。
- 三层模式成为真正意义上的“瘦客户端”，从而具备了很高的稳定性、延展性和执行效率。
- 三层模式可以将服务集中在一起管理，统一服务于客户端，从而具备了良好的容错能力和负载平衡能力。

B/S架构的缺点

- 客户端浏览器一般情况下以同步的请求/响应模式交换数据，每请求一次服务器就要刷新一次页面；
- 受HTTP协议“基于文本的数据交换”的限制，在数据查询等响应速度上，要远远低于C/S体系结构；
- 数据提交一般以页面为单位，数据的动态交互性不强，不利于在线事务处理(OLTP)应用；
- 受限于HTML的表达能力，难以支持复杂GUI (如报表等)。

AJAX(Asynchronous JavaScript and XML)

RIA技术

来源：
http://baike.baidu.com/link?url=IvNoTtaHgV_Qnk7Zep4xoICH6DQYmsVLqasNnWpUjLZanSWJf5Vx2zURbufuNgi9bdVVvKTKikX0Jze4lg2fT

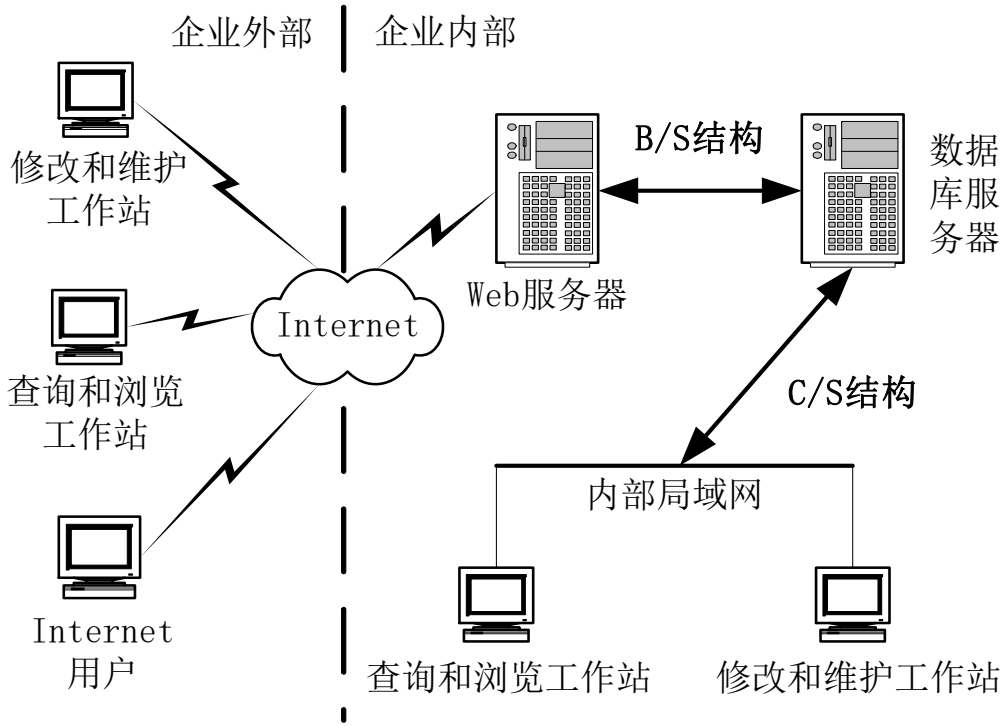
3.7.6 C/S+B/S混合体系结构

C/S+B/S混合体系结构

- 为了克服C/S与B/S各自的缺点，发挥各自的优点，在实际应用中，通常将二者结合起来

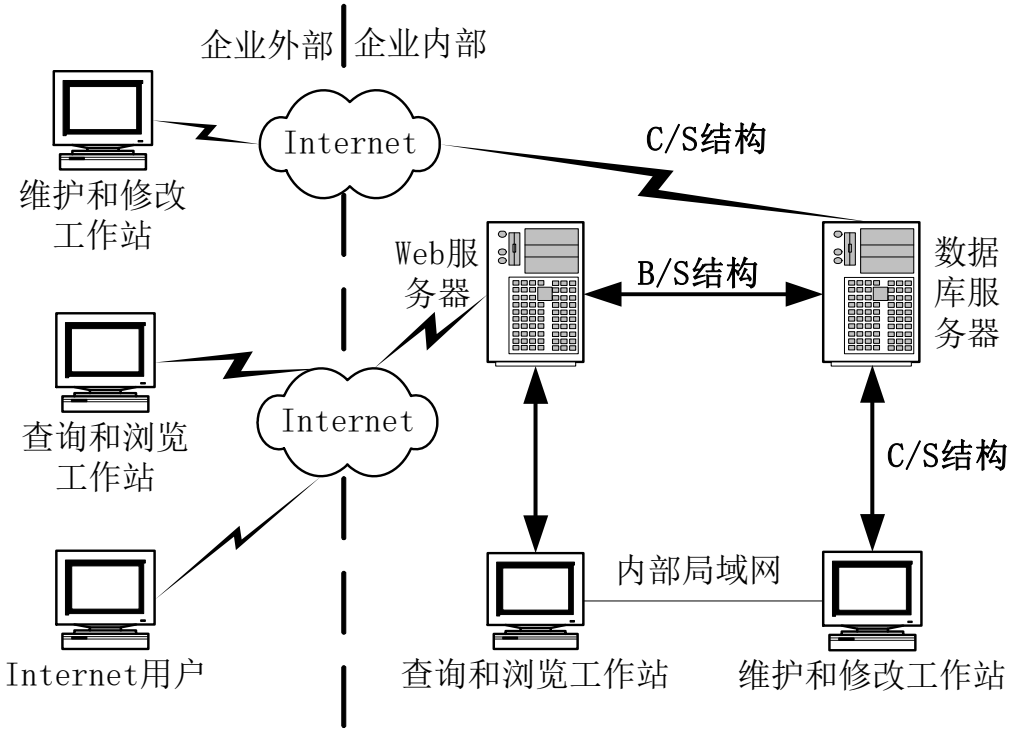
C/S+B/S混合体系结构

- 遵循“内外有别”的原则：
  - 企业内部用户通过局域网直接访问数据库服务器
    - C/S结构；
    - 交互性增强；
    - 数据查询与修改的响应速度高。
  - 企业外部用户通过Internet访问Web服务器/应用服务器
    - B/S结构；
    - 用户不直接访问数据，数据安全。

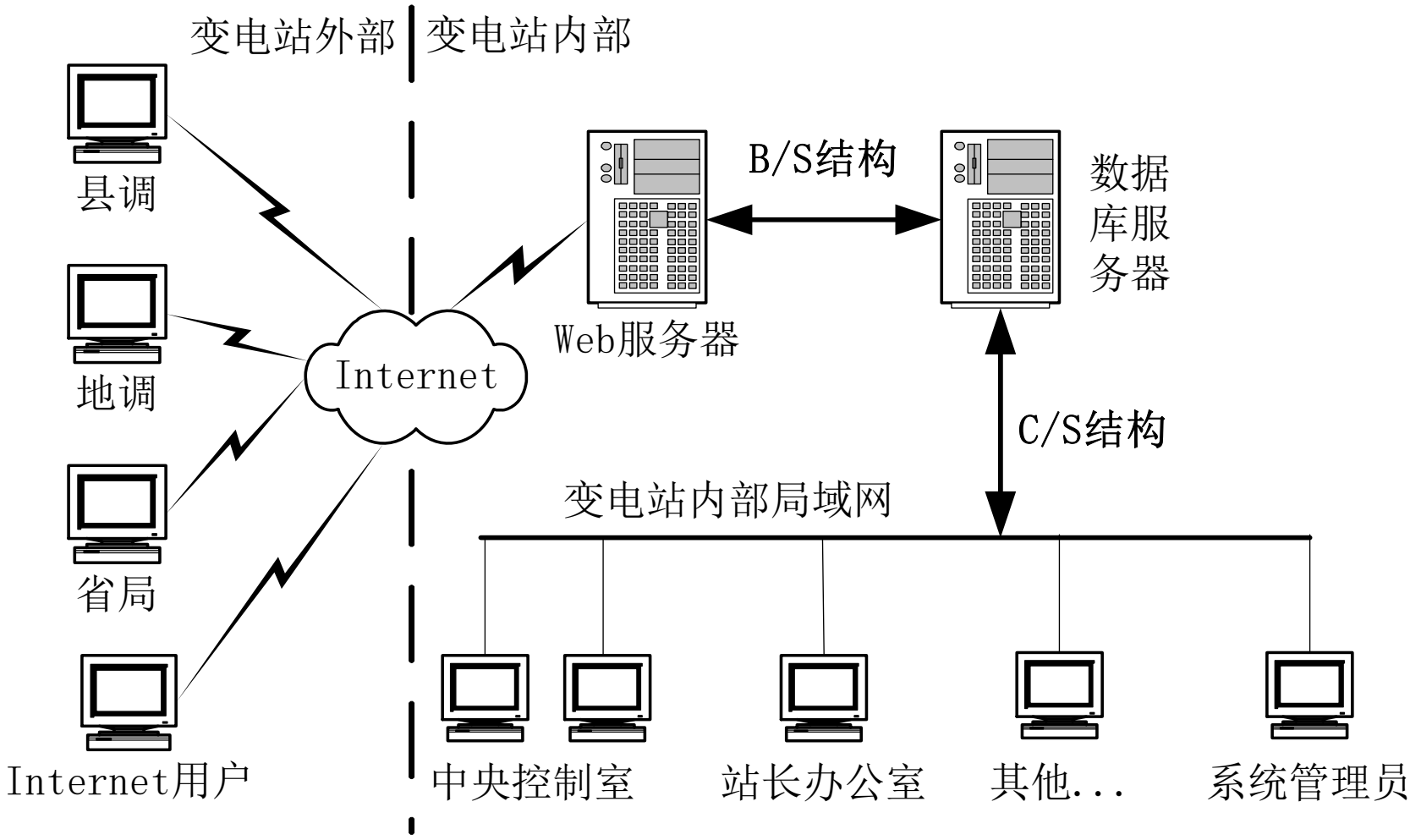


C/S+B/S混合体系结构

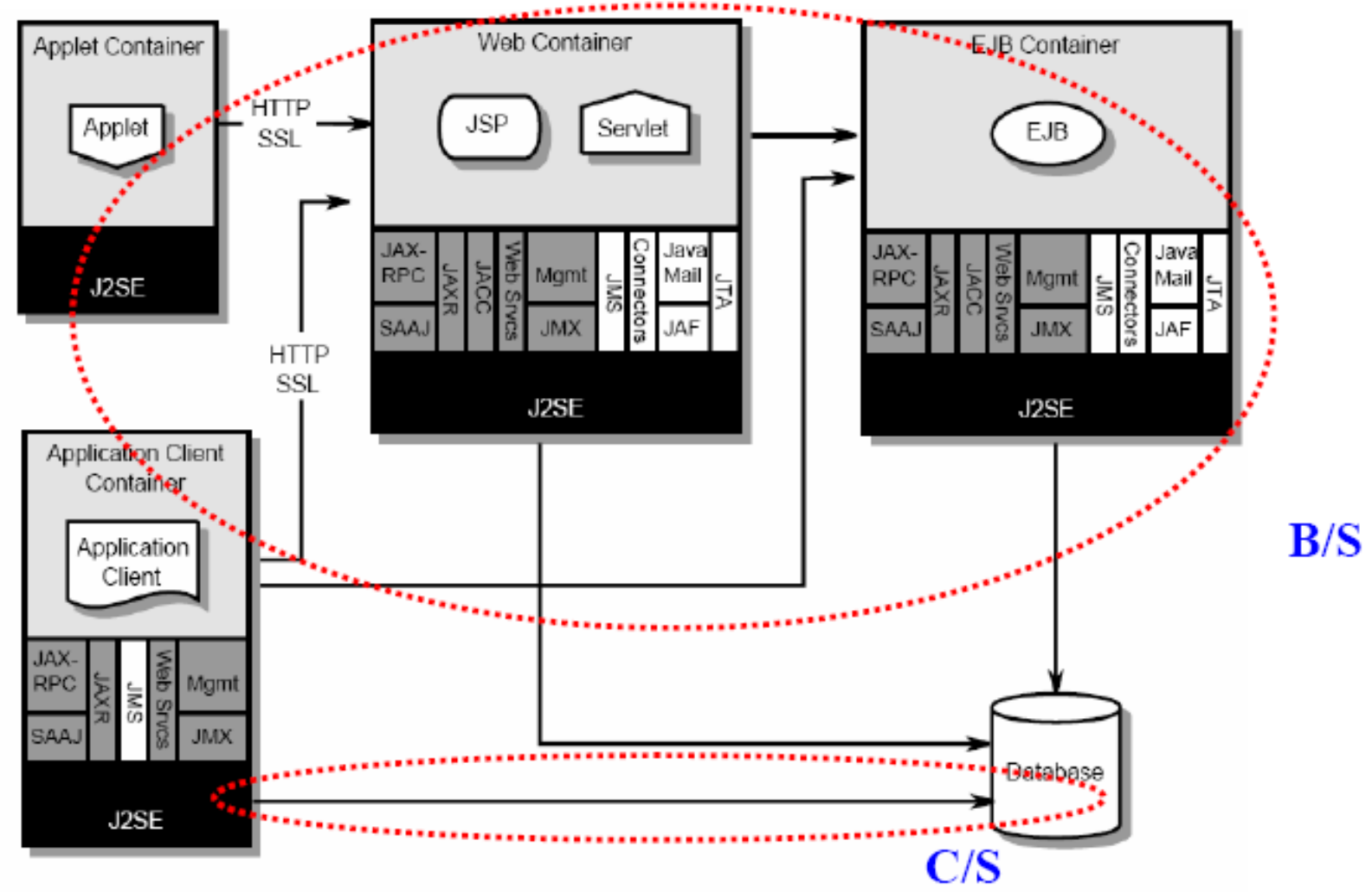
- 遵循“查改有别”的原则：
  - 不管用户处于企业内外什么位置(局域网或Internet)，凡是需要对数据进行更新操作的(Add, Delete, Update)，都需要使用**C/S结构**；
  - 如果只是执行一般的查询与浏览操作(Read/Query)，则使用B/S结构。



C/S+B/S混合体系结构实例



J2EE B/S+C/S Architecture

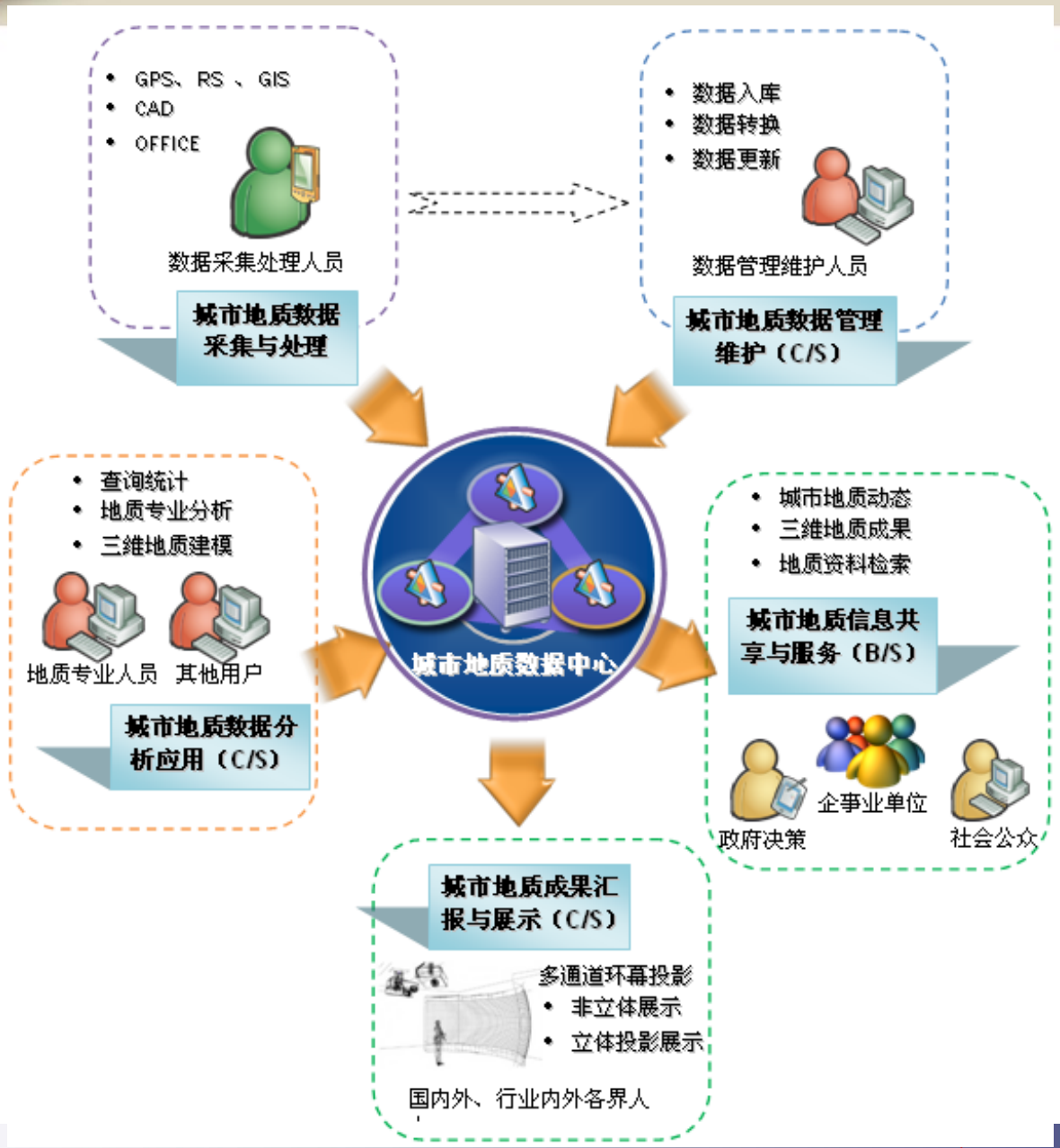


异构风格案例一
上海城市地质基础信息管理与服务系统

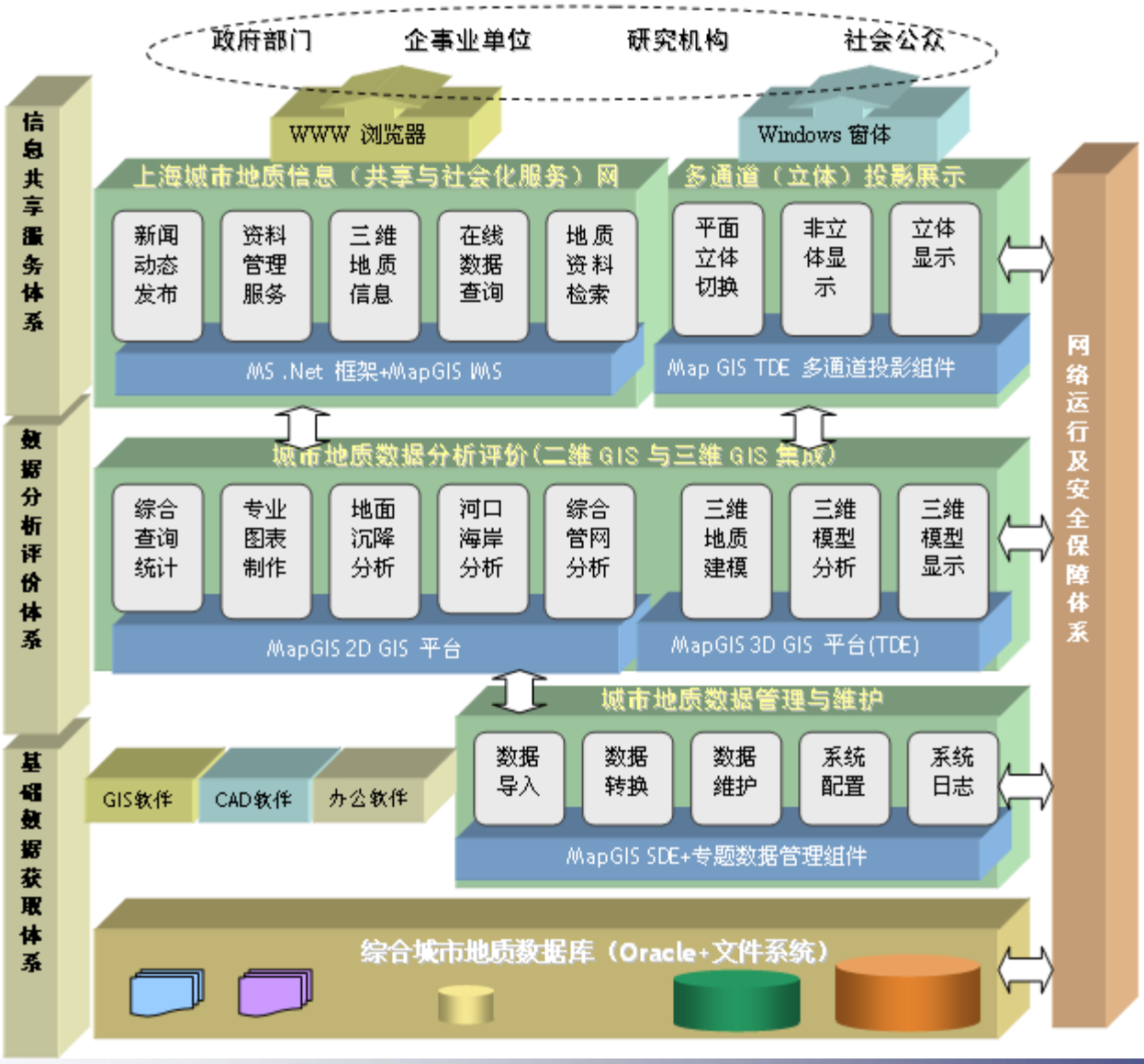
上海城市地质基础信息管理与服务系统
——SHGeo3DSys

- C/S+B/S混合架构
- 两层C/S（Two-tiers）
- 三层B/S（Three-tiers）
- 异构网络环境

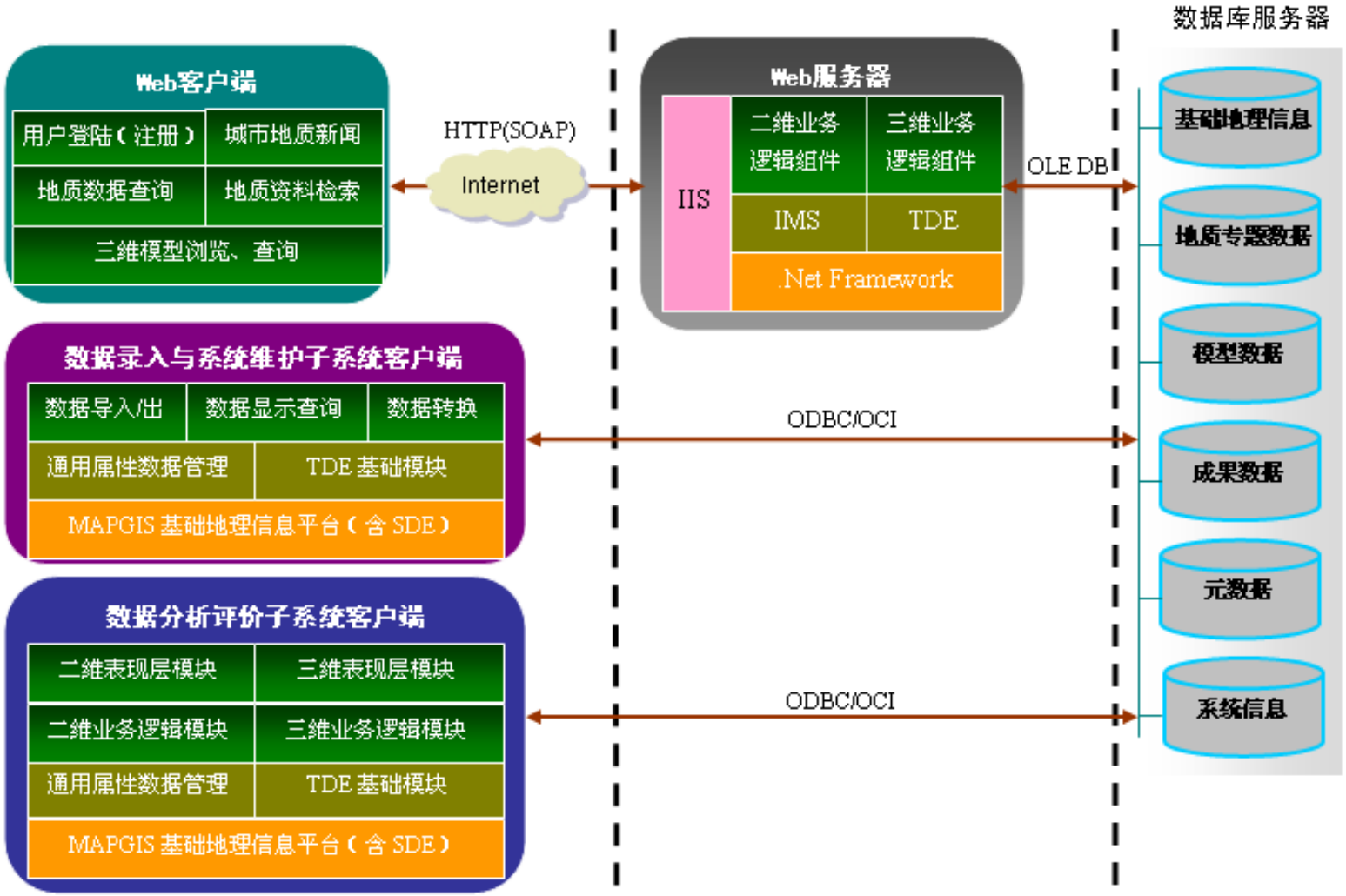
多层次服务用户需求


GPS、RS 、GIS
CAD
OFFICE
数据采集处理人员
城市地质数据采集与处理
数据入库
数据转换
数据更新
数据管理维护人员
城市地质数据管理维护（C/S）
查询统计
地质专业分析
三维地质建模
地质专业人员
其他用户
城市地质数据分析应用（C/S）
城市地质数据中心
城市地质动态
三维地质成果
地质资料检索
城市地质信息共享与服务（B/S）
政府决策
企事业单位
社会公众
城市地质成果汇报与展示（C/S）
多通道环幕投影
非立体展示
立体投影展示
国内外、行业内外各界人

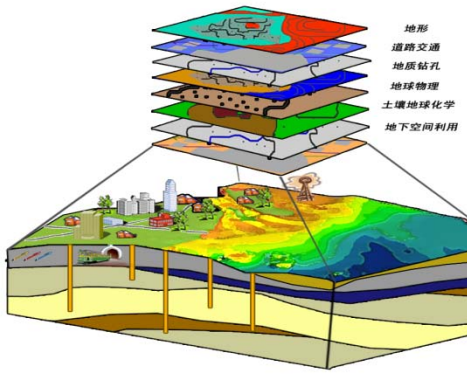
Multi-Layers Architecture

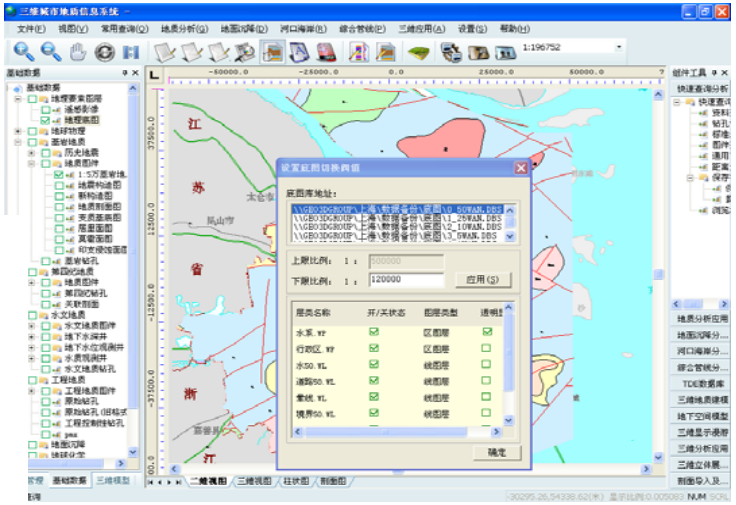


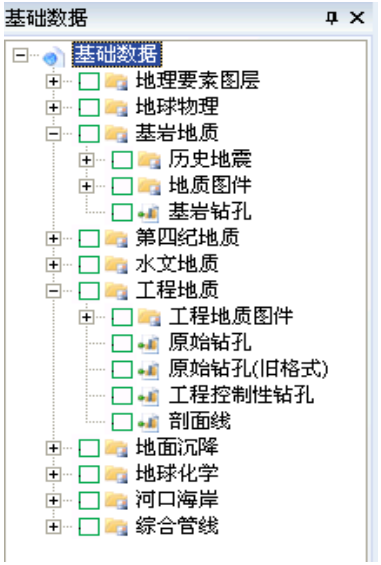
Multi-tiers Architecture

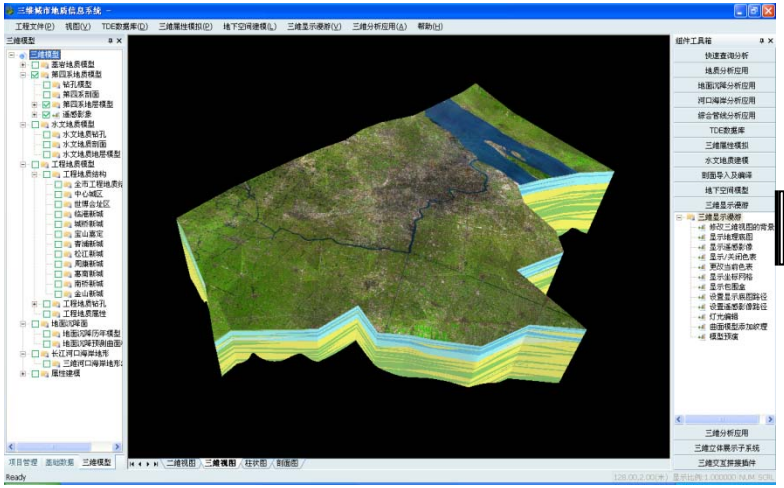


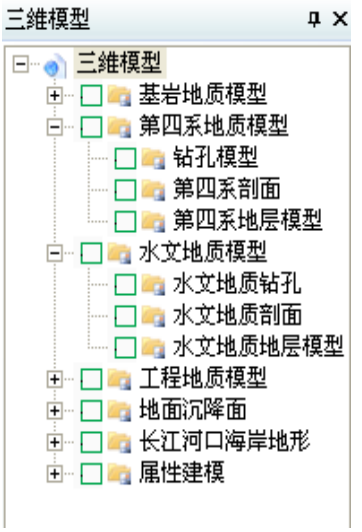
C/S系统


地形
道路交通
地质钻孔
地球物理
土壤地球化学
地下空间利用




基础数据
基础数据
地理要素图层
地球物理
基岩地质
历史地震
地质图件
基岩钻孔
第四纪地质
水文地质
工程地质
工程地质图件
原始钻孔
原始钻孔(旧格式)
工程控制性钻孔
剖面线
地面沉降
地球化学
河口海岸
综合管线

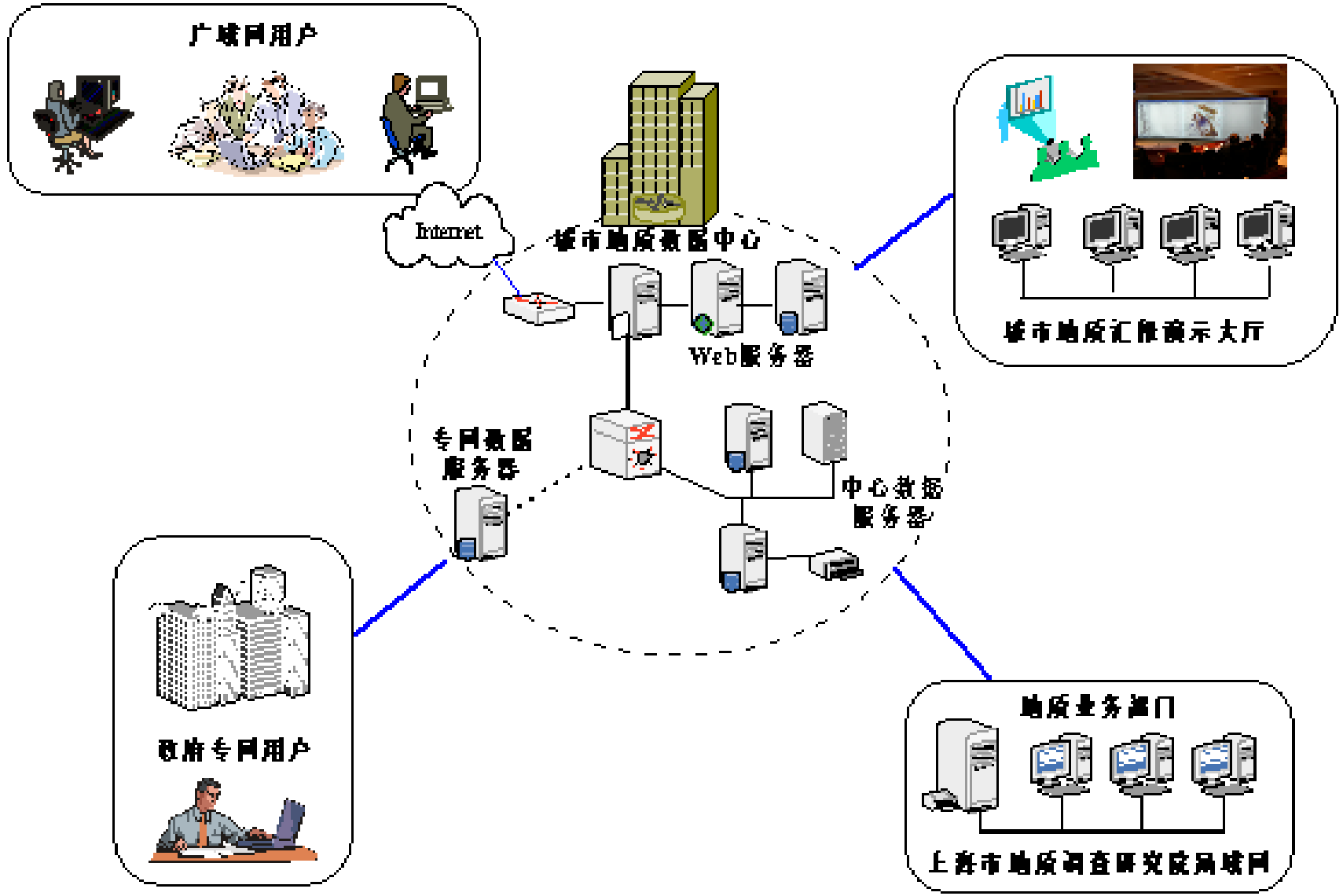



三维模型
三维模型
基岩地质模型
第四系地质模型
钻孔模型
第四系剖面
第四系地层模型
水文地质模型
水文地质钻孔
水文地质剖面
水文地质地层模型
工程地质模型
地面沉降面
长江河口海岸地形
属性建模

B/S系统

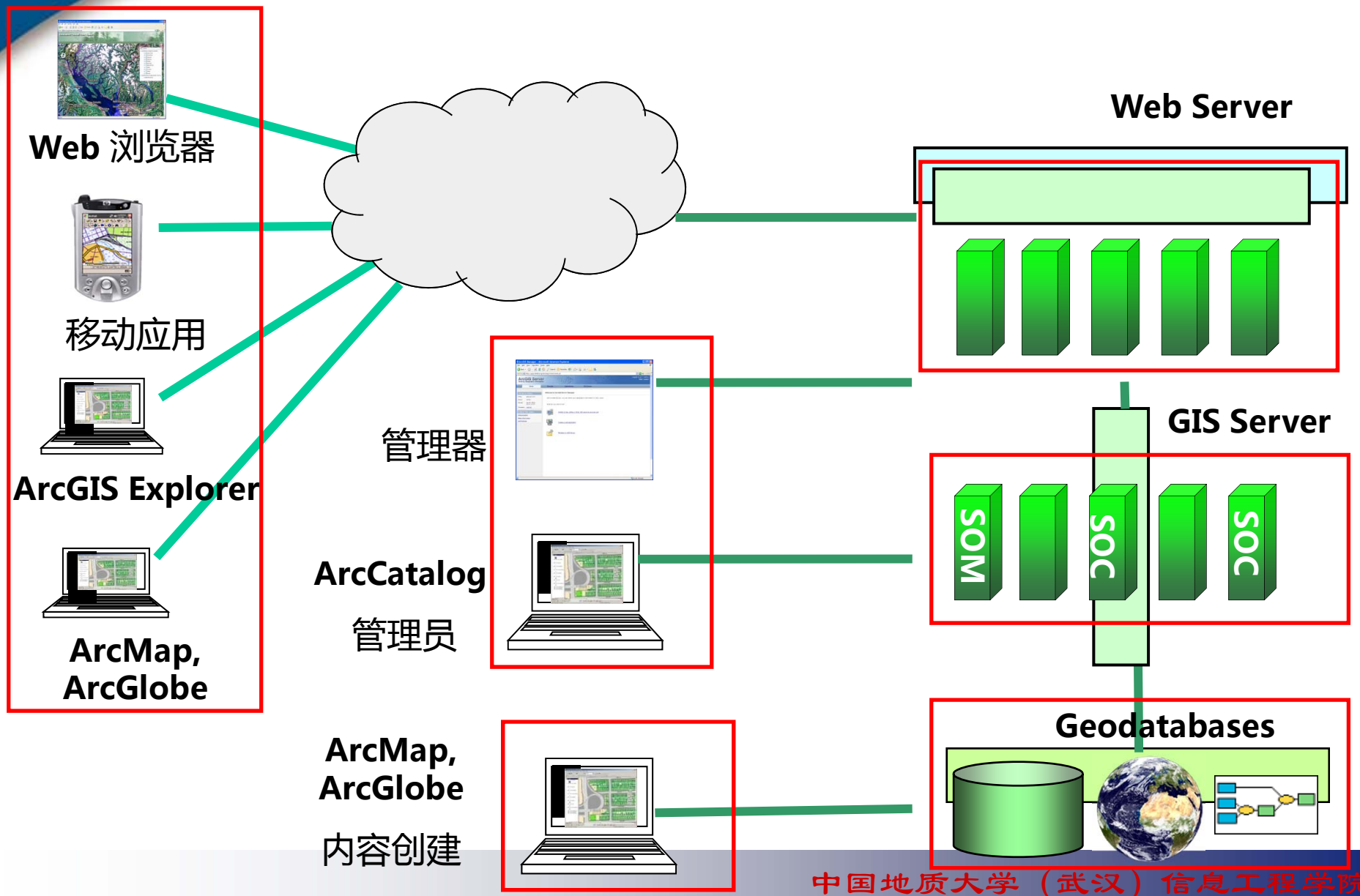

上海城市地质信息网络服务系统 - Microsoft Internet Explorer
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 收藏夹
地址(D) http://www.sigs.com.cn/
转到 链接 SnagIt
SIGS 上海城市地质信息
系统首页 地质资料管理与服务 三维地质结构 地质数据查询 地质资料检索 地质调查机构 帮助
您当前的位置:服务系统首页
游客，欢迎您！ 2008-7-4 16:07:17
用户登录
用户名：
密 码：
注 册 登 录
友情链接
中国地质调查局 China Geological Survey
上海市地质资源 www.shfdz.gov.cn
中国地质环境信息网 www.cigem.gov.cn
全国地质资料馆 www.ngac.cn
全国地质资料管理与服务
欢迎来到上海城市地质信息网！
《上海城市地质信息》由上海市房屋土地资源管理局直属的上海市地质资料馆主办，上海市地质调查研究院提供技术支持。本网站面向上海城市规划、建设和管理部门、科研机构、企事业单位及社会公众，提供上海城市地质信息的发布和检索。 在上海市房地资源管理局支持下，上海市地质资料馆整理了馆藏地质成果，向社会全面公布了地质资料档案目录、各专业地质钻孔空间分布与基本属性，收集有关的行业规范、国内外城市地质信息动态，为城市规划、建设与管理部门提供技术支持与服务。 在上海市建设和交通委、上海市房地资源管理局的大力推动下，认真落实国务院《地质资料管理条例》（自2002年7月1日起施行），自2006年7月1日起试行地质资料汇交机制。上海市地质资料馆严格按照《地质资料管理条例》要求，认真完成地质资料汇交、日常管理与社会服务的职责。
上海市地质资料馆位于上海市灵石路930号地质大厦10层，联系电话：56615718。
最新动态 · 新 闻
《上海市地质资料管理信息动态》专辑1(2008) 2008-4-2
《上海市地质资料管理信息动态》专辑10 2008-1-4
《上海市地质资料管理信息动态》专辑9 2008-1-4
《上海市地质资料管理信息动态》专辑8 2007-10-27
《上海市地质资料管理信息动态》专辑7 2007-9-14
《上海市地质资料管理信息动态》专辑6 2007-9-14
《上海市地质资料管理信息动态》专辑4 2007-8-14
《上海市地质资料管理信息动态》专辑3 2007-8-14
更多....
网站导航
上海地质期刊
地质资料目录检索
公开出版物
公开地质成果
完毕
Internet

局域网+Internet+专用网的异构网络环境



异构风格实例二
服务架构下的ArcGIS应用体系结构

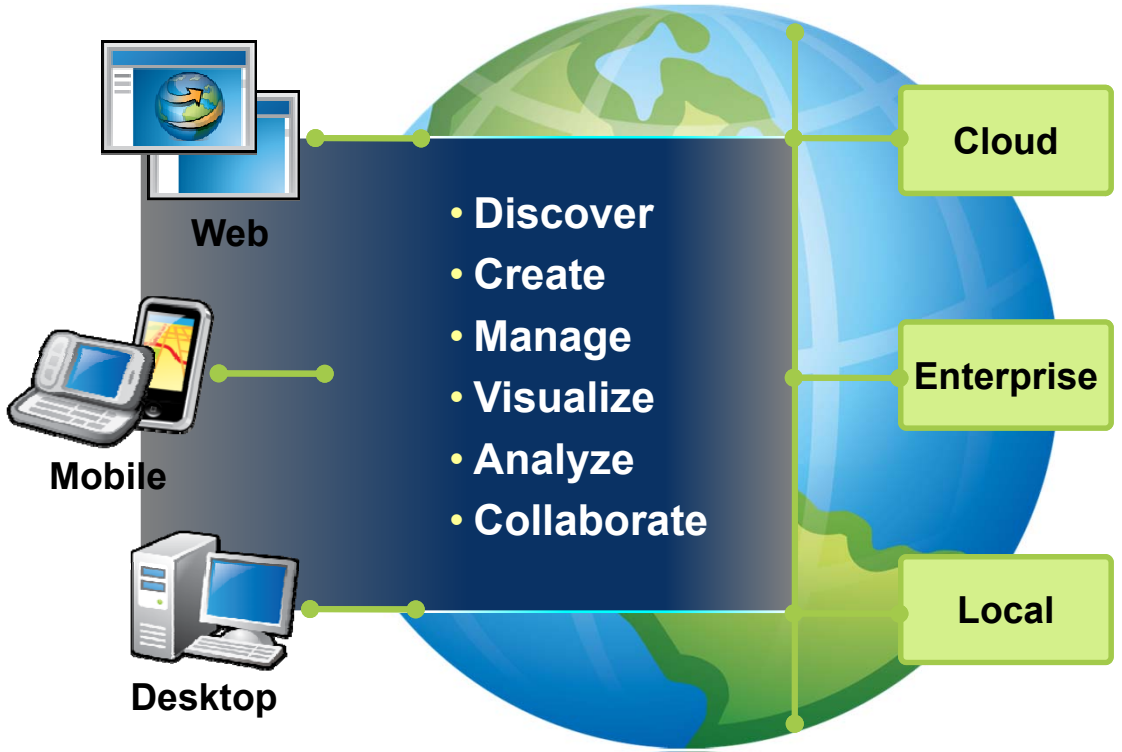
ArcGIS Server部署模式



ArcGIS 10 Vision

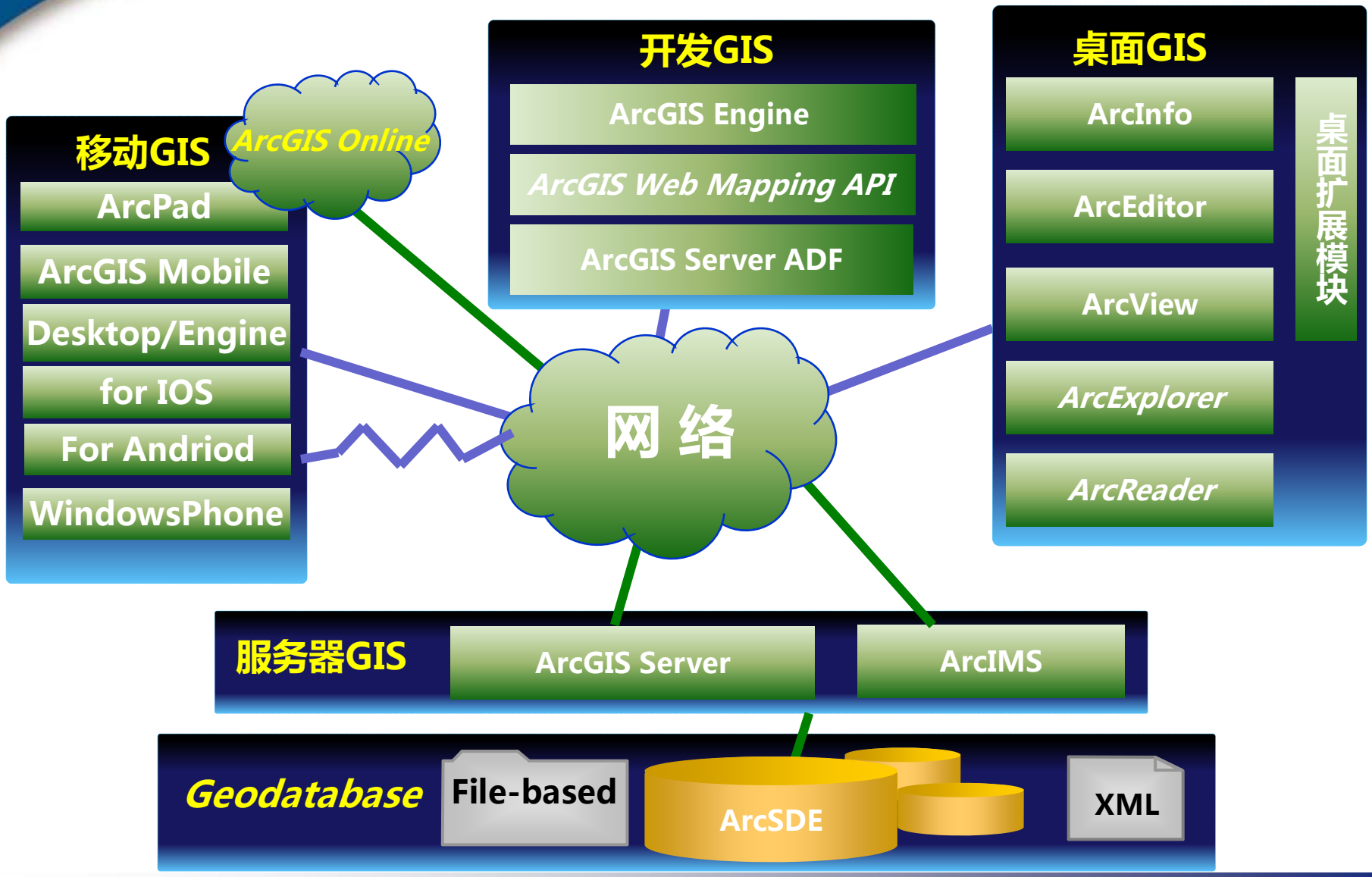
一个提供完整空间地理信息服务解决方案的平台

- 简单
- 强大
- 随时随地



为专业GIS与大众GIS之间创造一个桥梁

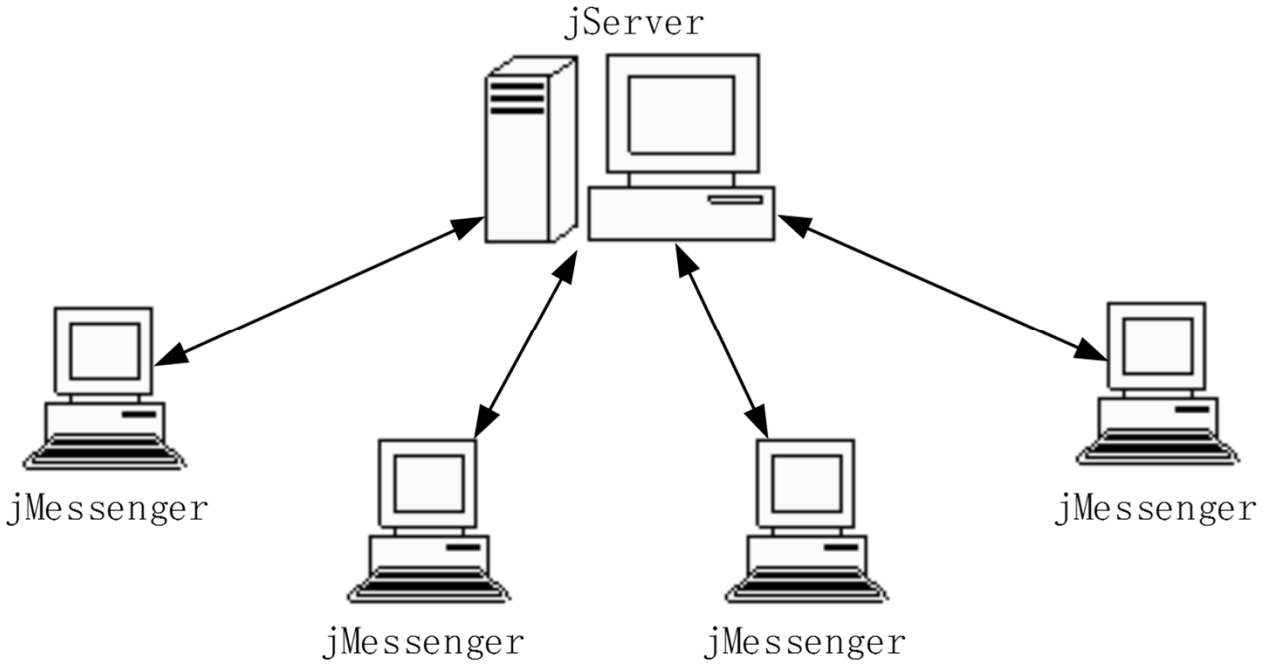
ArcGIS – 完整基于服务架构GIS解决方案



3.7.7 案例分析

Java Chat Application（1）
（C/S架构网络聊天软件）

- 功能：用户注册登录、聊天、发送公共消息和私人消息、文件传输等



来源：http://www.codeproject.com/Articles/524120/A-Java-Chat-Application

Java Chat Application（2）
（C/S架构网络聊天软件）

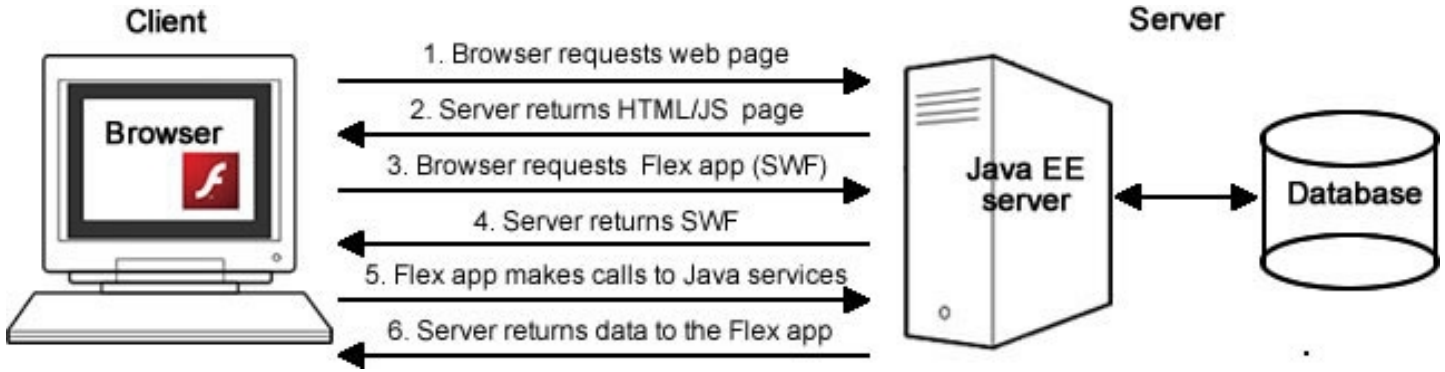
- jServer主要有两个类分别处理连接和消息：
  - 在启动时**SocketServer**单独运行在一个线程中，它会监听是否有客户端连接服务器，一旦发现客户端jServer就会创建一个单独的线程去运行ServerThread。
  - **ServerThread**创建后就会持续的监听来自客户端的消息，并将消息交由SocketServer处理。同样，它支持将来自客户端的消息转发到其他客户端。jServer将客户端用户名和密码保存在data.xml中，以支持用户注册功能。

Java Chat Application（3）
（C/S架构网络聊天软件）

- jMessenger通过指定的ip地址和端口号连接到jServer。然后，到达的消息以及它们的发送者会显示在留言板上。需要说明的是，在传输文件时，文件并不通过服务器，而是客户端之间启用单独线程直接传输，这样可以同时进行聊天和文件传输。jMessenger将消息记录保存在History.xml中，可以查看聊天历史记录。

作 业

- 1.传统的B/S架构有什么缺点？根本原因是什么？为改进这些缺点出现了哪些RIA(富因特网应用程序)技术？下图所采用的是哪种RIA技术？并说明与传统B/S架构相比，这一技术有哪些改进之处。



- 2.上机调试Java Chat Application程序代码，并试着和你的同学/朋友使用这款软件进行聊天，你认为还有哪些功能需要修改完善？请给出你的实现方案。