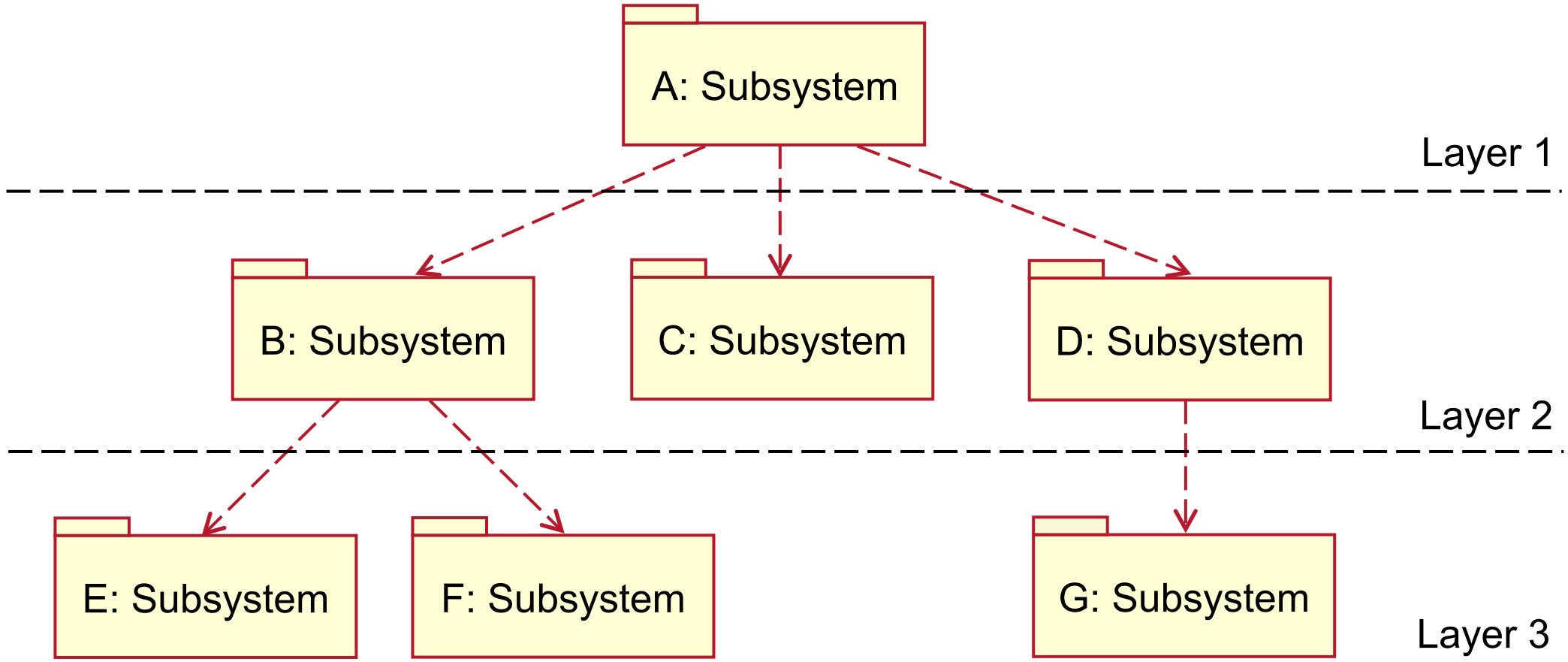
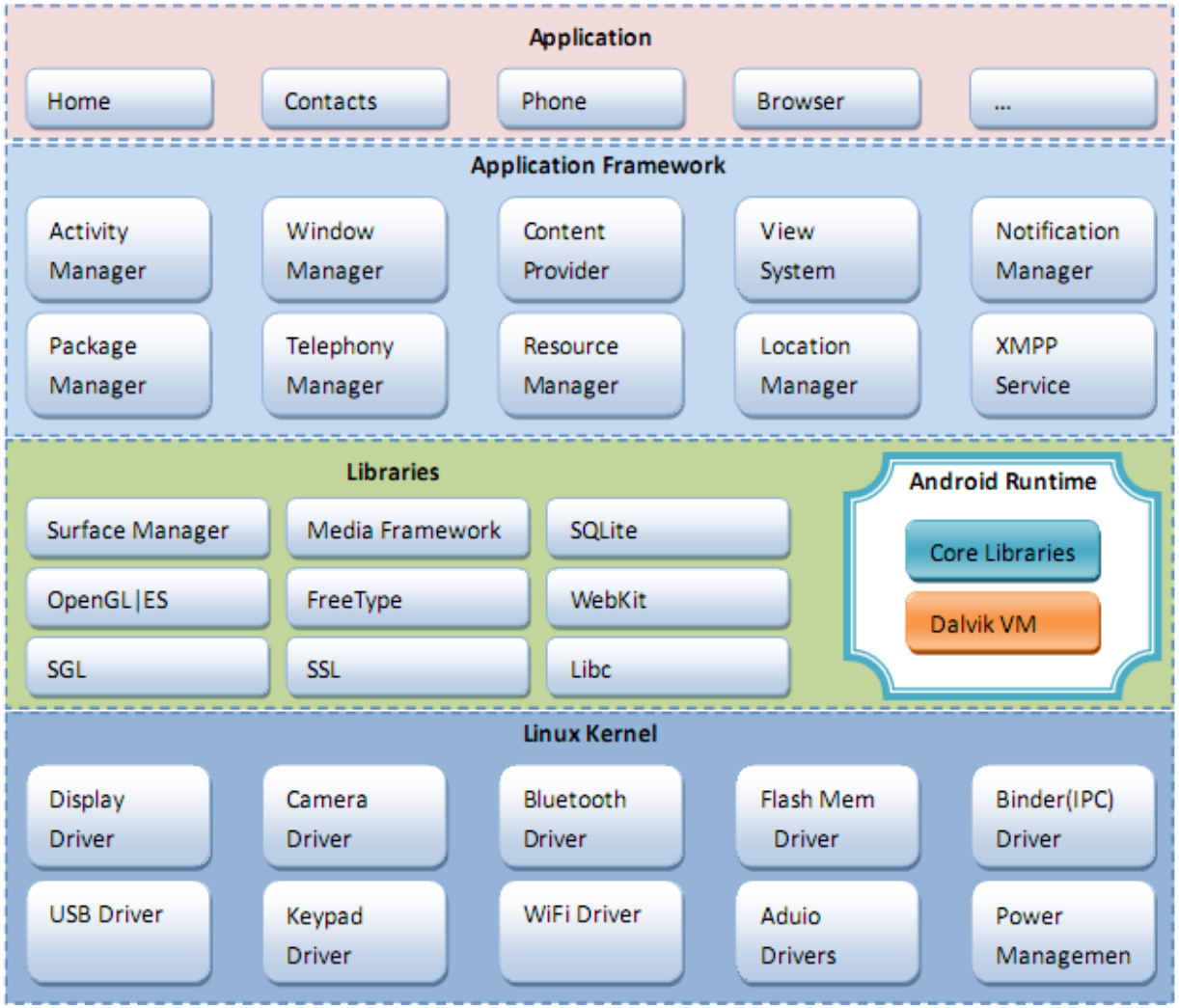
软件体系结构风格（二）

清华大学软件学院 刘强

层次结构


A: Subsystem
Layer 1
B: Subsystem
C: Subsystem
D: Subsystem
Layer 2
E: Subsystem
F: Subsystem
G: Subsystem
Layer 3

示例1：安卓操作系统层次结构



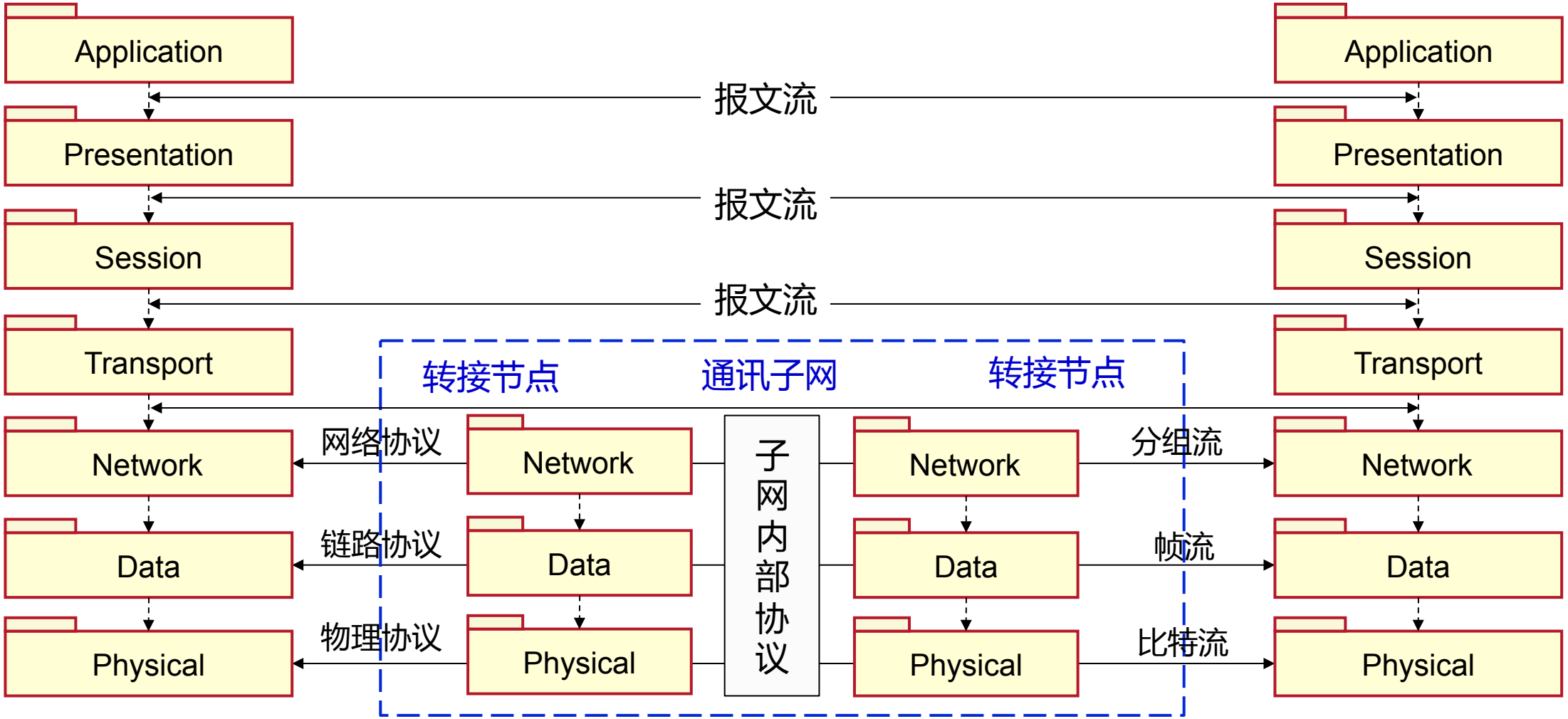
应用层：运行在虚拟机上的Java应用程序。

应用框架层：支持第三方开发者之间的交互，使其能够通过抽象方式访问所开发的应用程序需要的关键资源。

系统运行库层：为开发者和类似终端设备拥有者提供需要的核心功能。

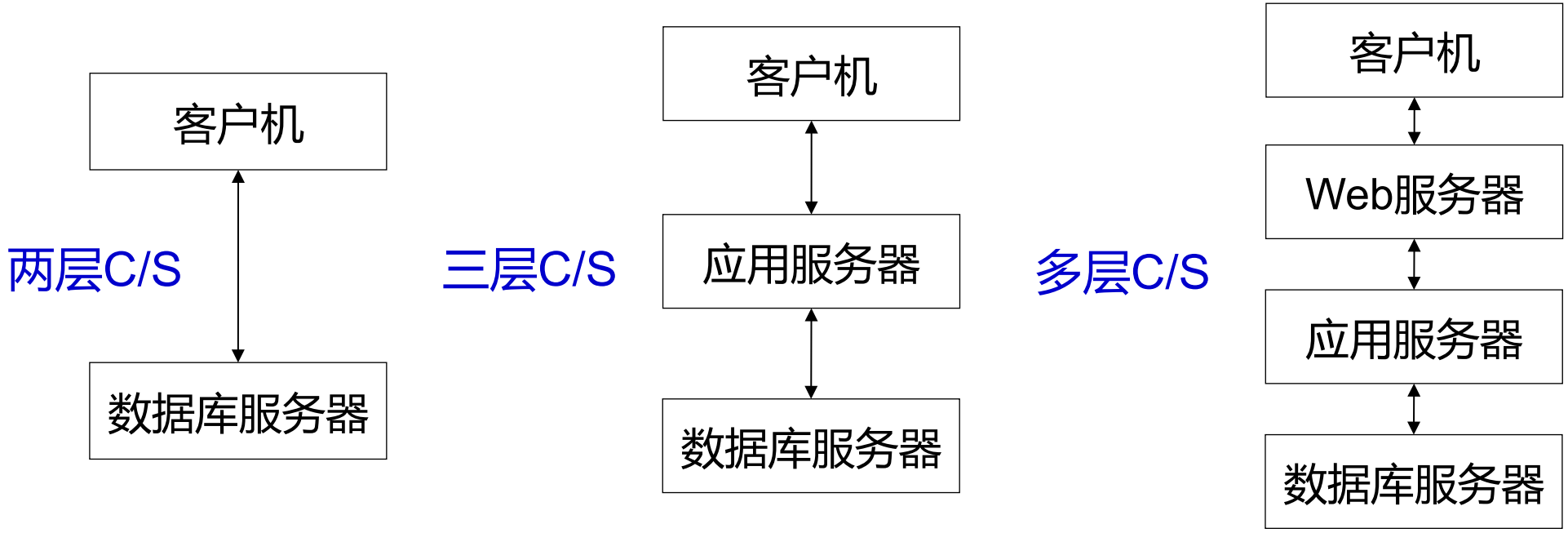
Linux内核层：提供启动和管理硬件以及Android应用程序的最基本的软件。

示例2：网络分层模型


Application
Presentation
Session
Transport
Network
Data
Physical
报文流
报文流
报文流
转接节点
通讯子网
转接节点
网络协议
链路协议
物理协议
Network
Data
Physical
子网内部协议
Network
Data
Physical
分组流
帧流
比特流
Application
Presentation
Session
Transport
Network
Data
Physical

客户机 / 服务器结构

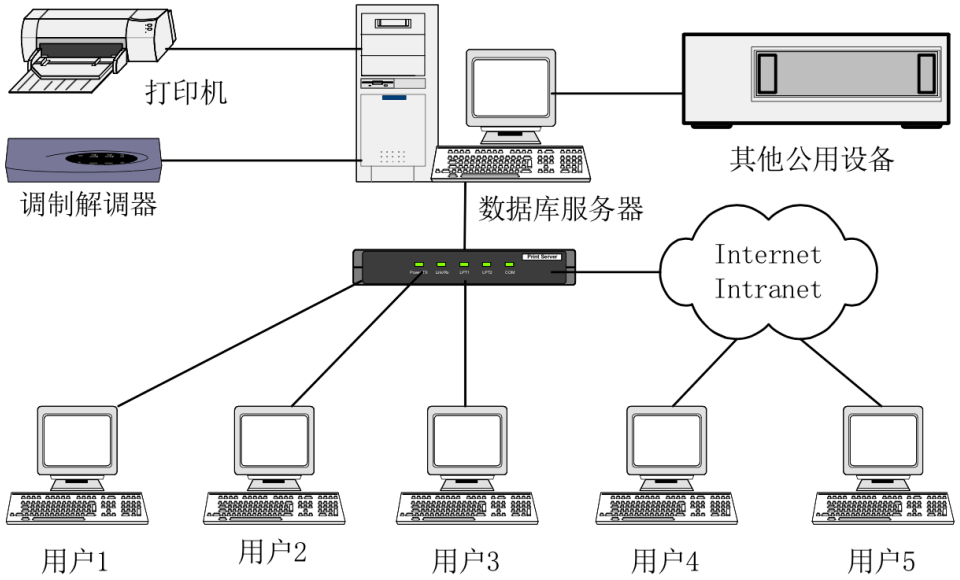
客户机 / 服务器体系结构（Client/Server）是一种分布式系统模型，作为服务器的子系统为其他客户机的子系统提供服务，作为客户机的子系统负责与用户的交互。

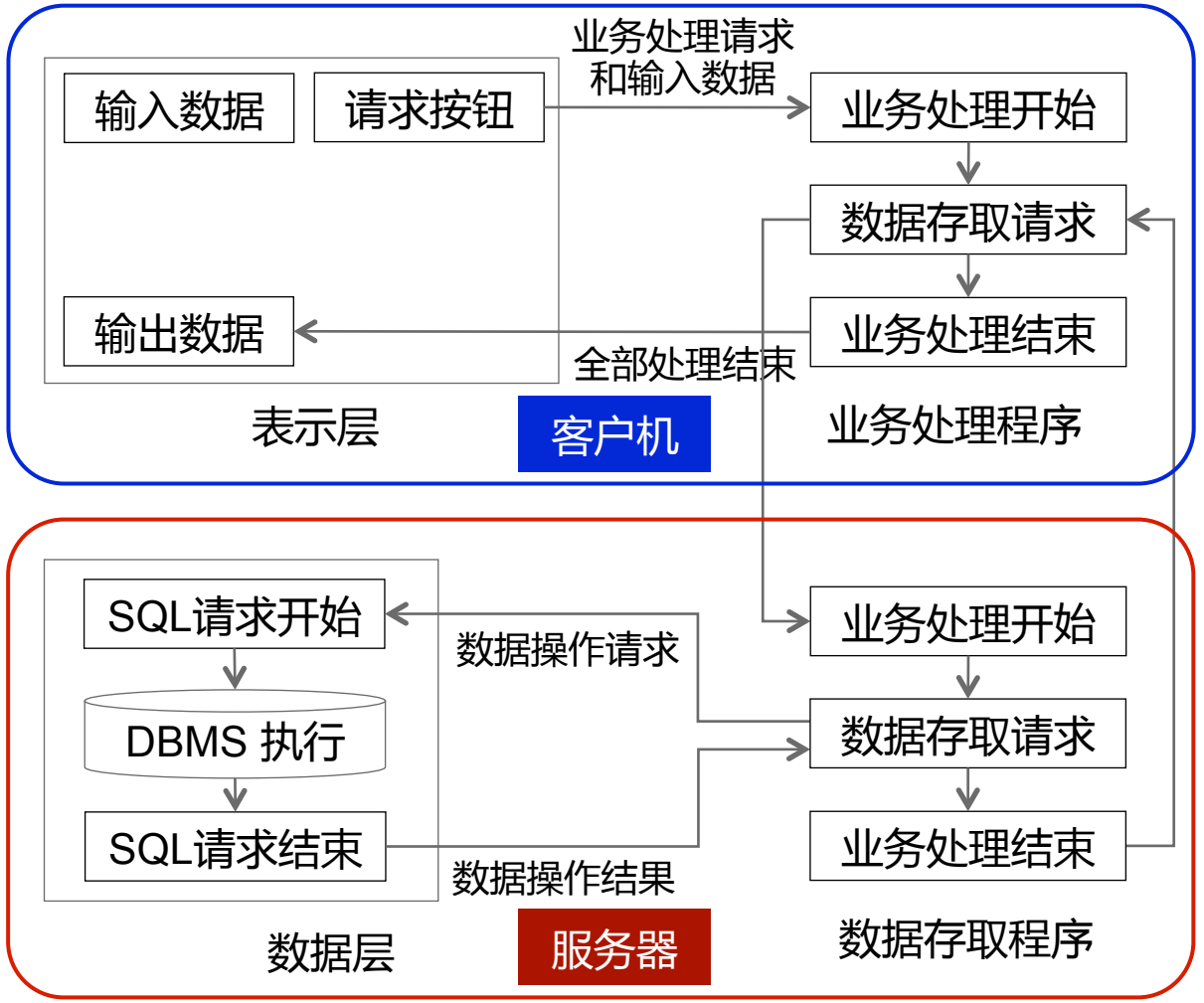


两层C/S结构

胖客户端模型：

- 服务器只负责数据的管理
- 客户机实现应用逻辑和用户的交互

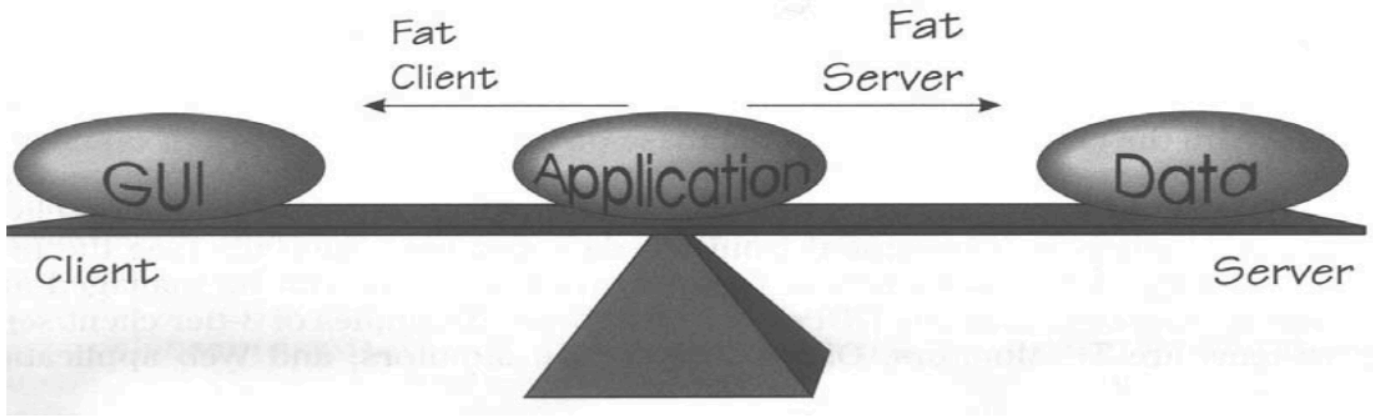




胖客户端与瘦客户端

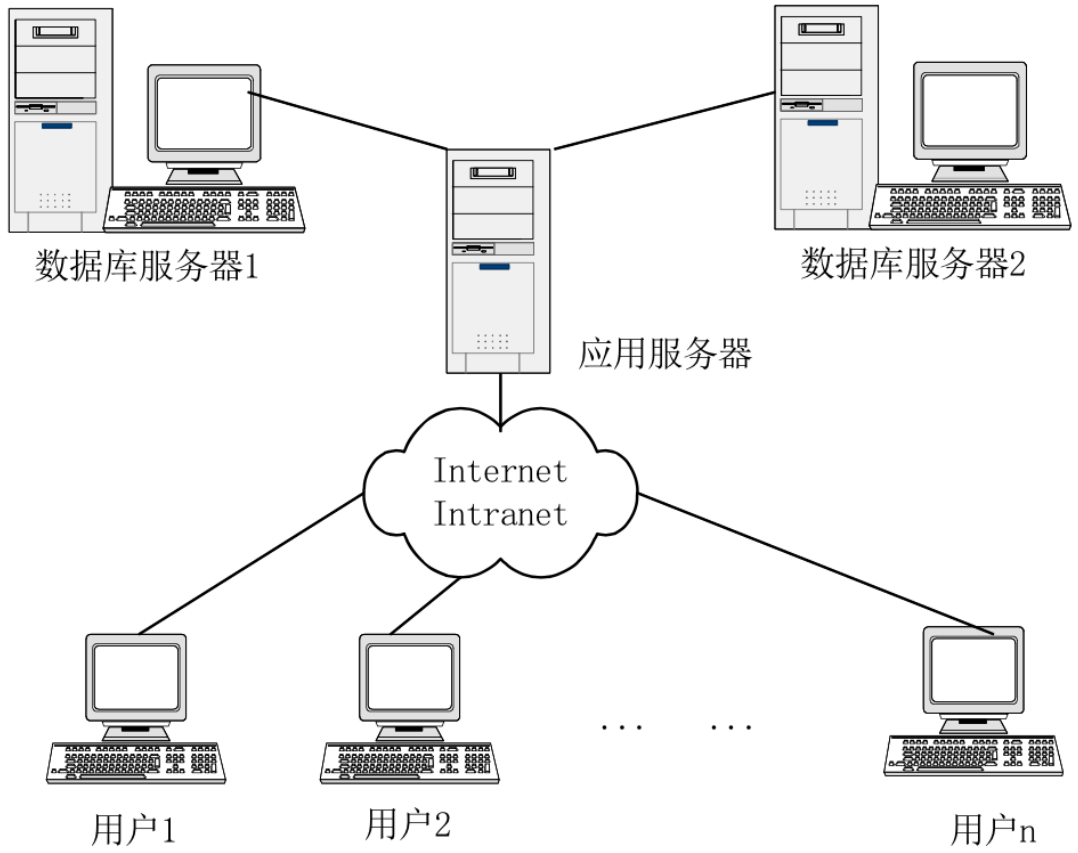
业务逻辑的划分比重：在客户端多一些还是在服务器端多一些?

- 胖客户端：客户端执行大部分的数据处理操作
- 瘦客户端：客户端具有很少或没有业务逻辑



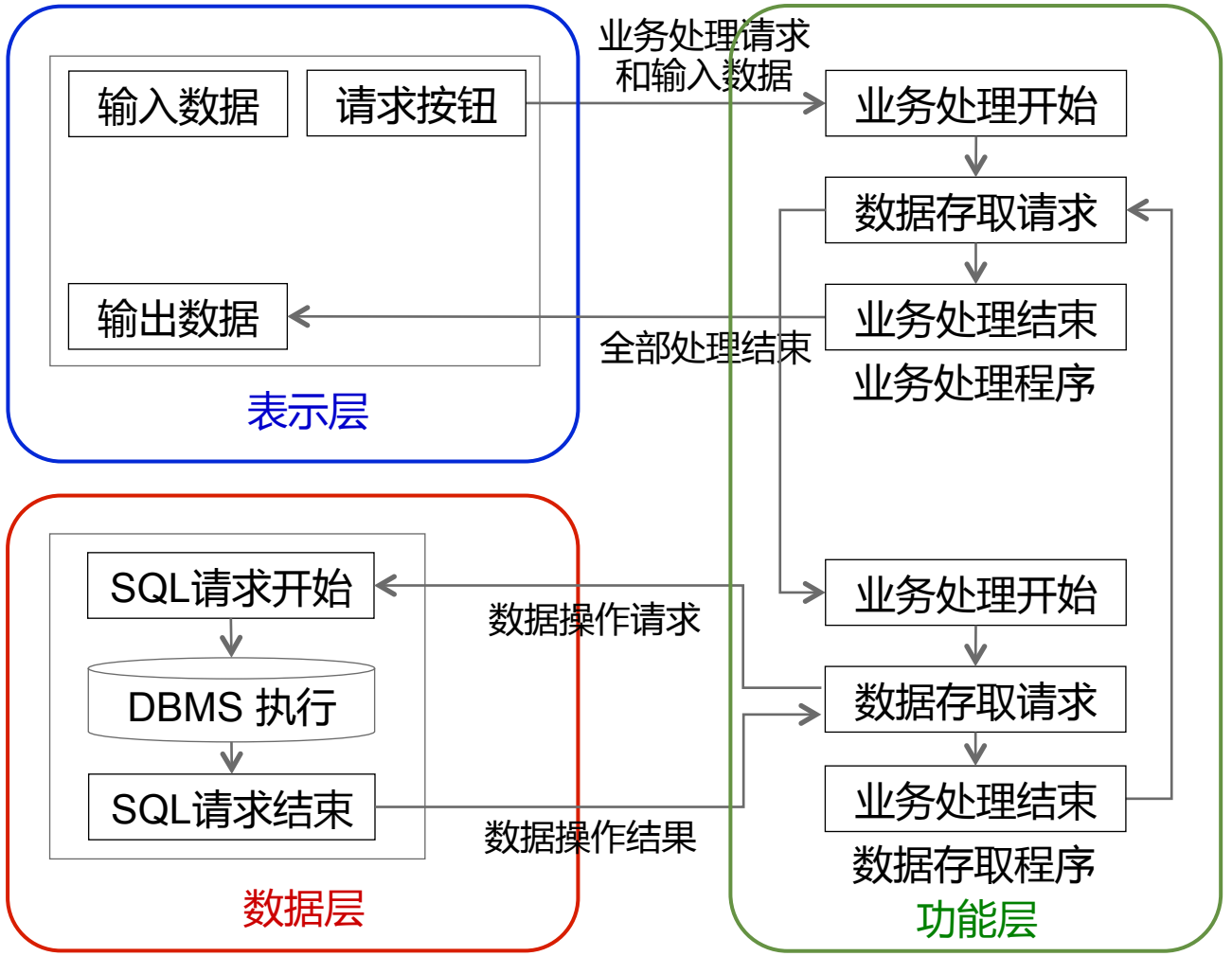
三层C/S结构

- 表示层：包括所有与客户机交互的边界对象，如窗口、表单、网页等。
- 功能层（业务逻辑层）：包括所有的控制和实体对象，实现应用程序的处理逻辑和规则。
- 数据层：实现对数据库的存储、查询和更新。



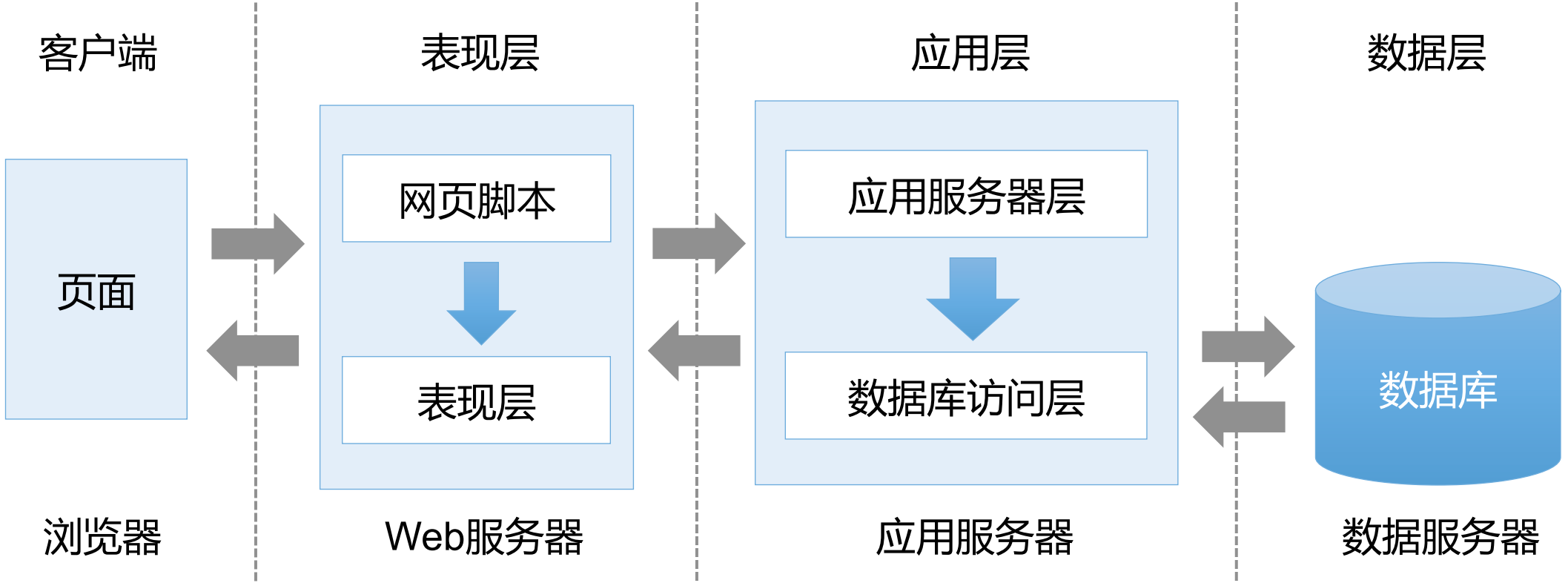
三层C/S结构

- 表示层：包括所有与客户机交互的边界对象，如窗口、表单、网页等。
- 功能层（业务逻辑层）：包括所有的控制和实体对象，实现应用程序的处理逻辑和规则。
- 数据层：实现对数据库的存储、查询和更新。

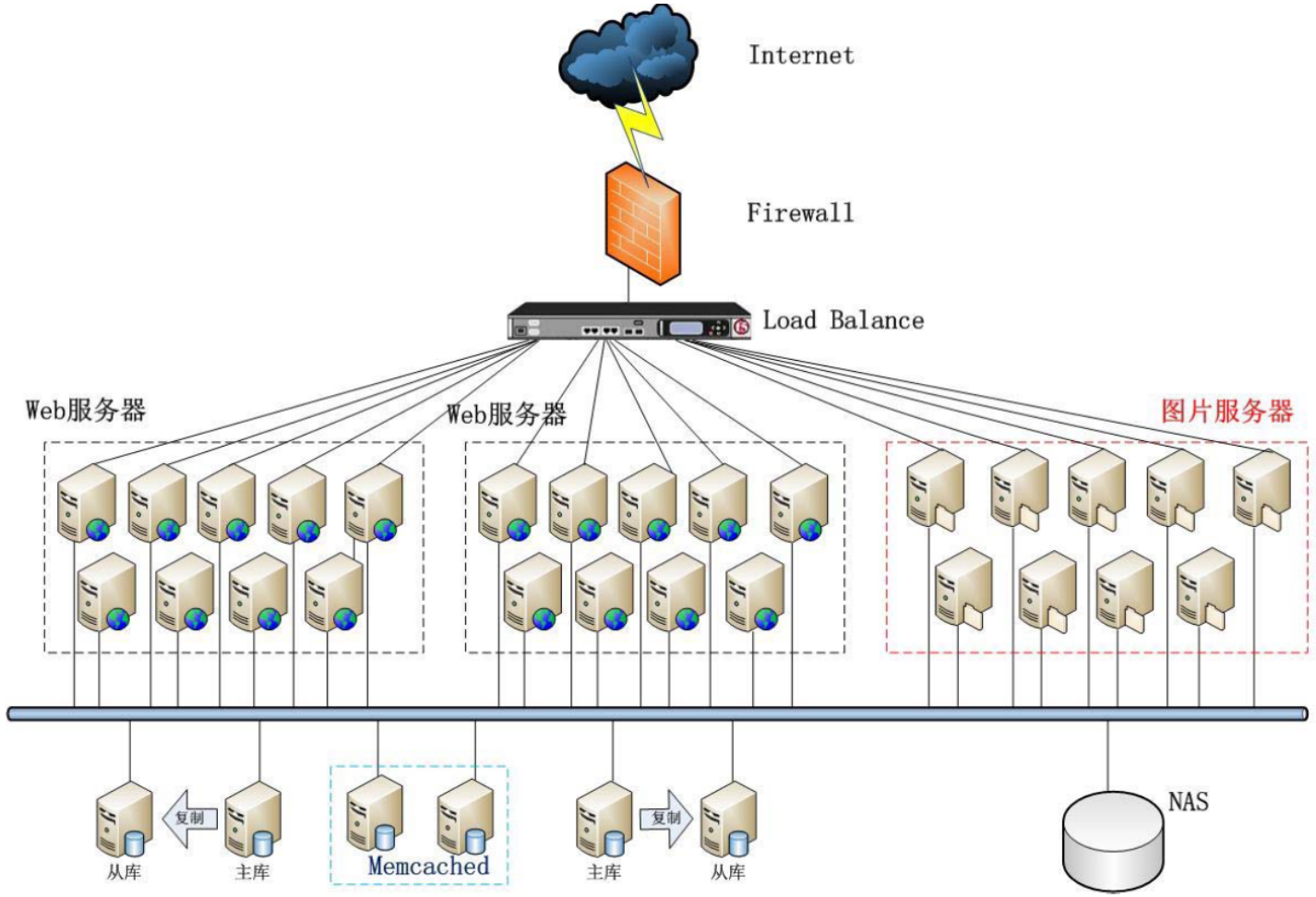


B/S结构

浏览器/服务器（Browser/Server）结构是三层**C/S**风格的一种实现方式。



集群结构



集群内各服务器上的内容保持一致
（通过冗余提高可靠性与可用性）

○ = ○ = ○ = ○ = 系统

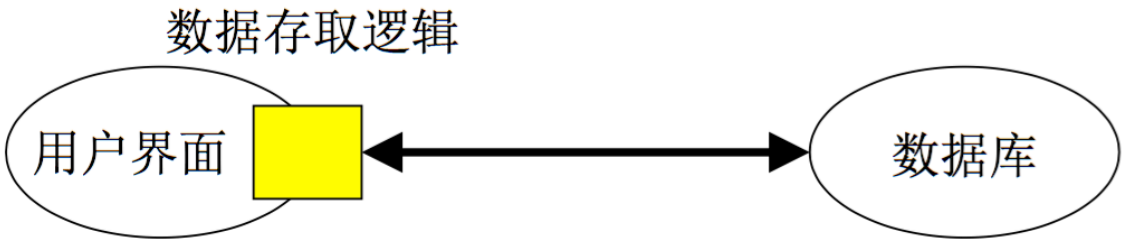
集群内各服务器上的内容之和构成
系统完整的功能/数据
（通过分布式提高速度与并发性）

○ + ○ + ○ + ○ = 系统

MVC结构

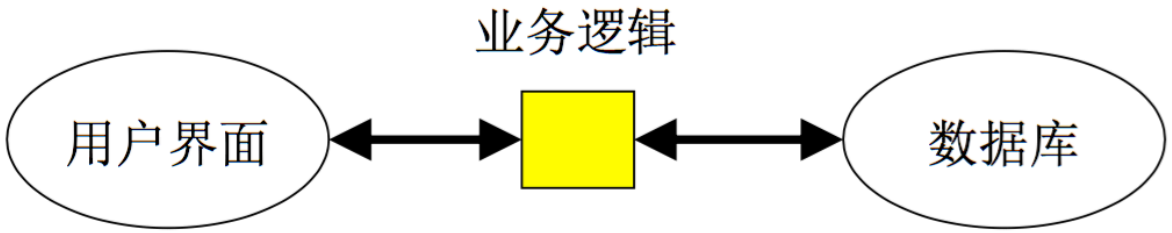
客户机—服务器结构：

- 许多应用系统的用途都是从数据库中检索数据，并将其显示给用户。
- 在用户更改数据之后，系统再将更新内容存储到数据存储中。
- 因为关键的信息流发生在数据存储和用户界面之间，所以一般倾向于将这两部分捆绑在一起，以减少编码量并提高应用程序性能。



MVC结构

思考：纯粹的B/S结构会不会解决这个问题？



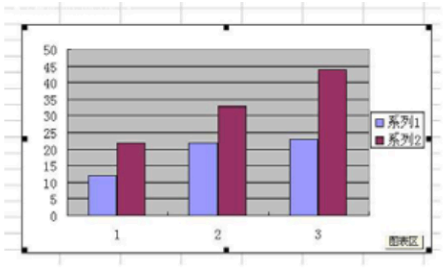
如何实现模块化，以便可以轻松地单独修改各个部分而不影响其他部分？

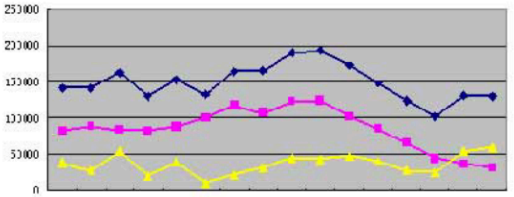
- 用户界面仍然需要显式地调用功能层的业务逻辑
- 仍然难以避免“用户界面的修改→业务逻辑的修改”的问题

MVC结构

影响因素：

- 在基于Web的应用程序中，用户界面逻辑的更改往往比业务逻辑频繁。
- 如果将UI代码和业务逻辑组合一起并放在UI中，则每次更改界面都可能引起对业务逻辑的修改。
- 在某些情况下，应用程序以不同的方式显示同一数据。

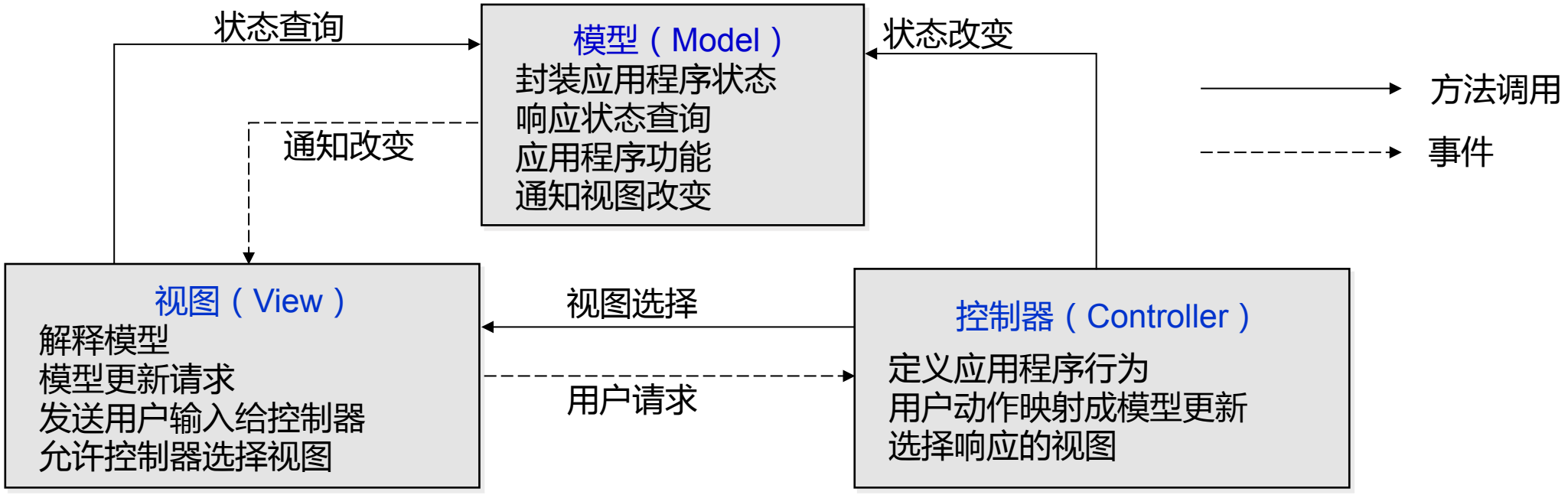




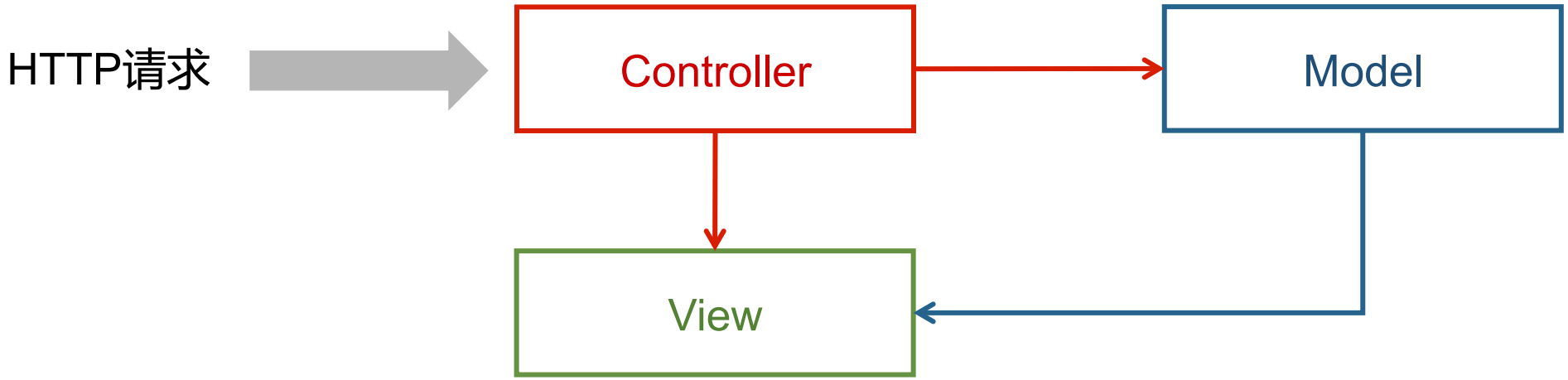
- 与业务逻辑相比，用户界面代码对设备的依赖性往往更大。
- 设计美观而有效的用户界面与开发复杂业务逻辑需要不同的编程技能。
- 通常，为用户界面创建自动测试比为业务逻辑更难、更耗时。

MVC结构

模型-视图-控制器（MVC）结构将应用程序的数据模型、业务逻辑和用户界面分别放在独立构件中，这样对用户界面的修改不会对数据模型/业务逻辑造成太大影响。

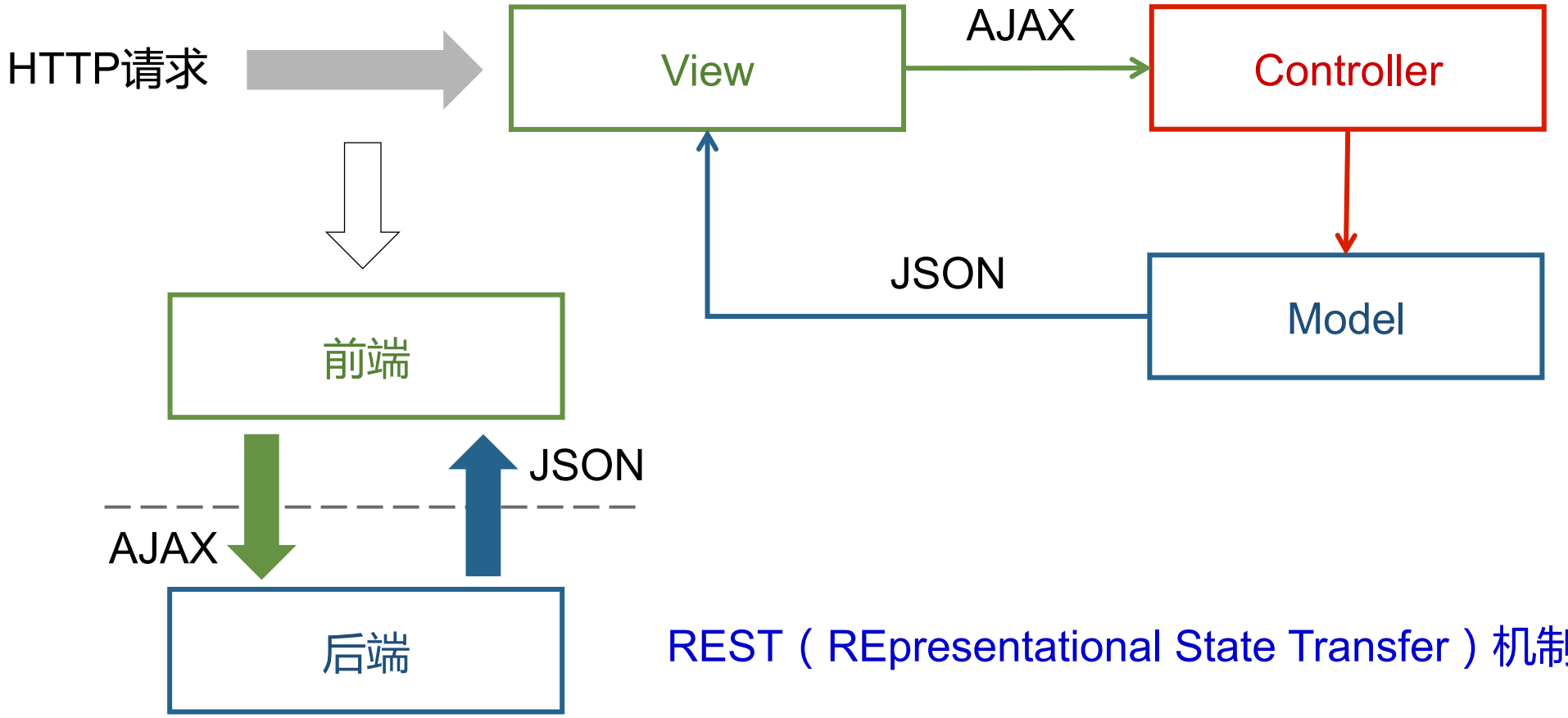


MVC结构



- 每次请求必须经过“控制器->模型->视图”过程，才能看到最终展现的界面
- 视图是依赖于模型的
- 渲染视图在服务端完成，呈现给浏览器的是带有模型的视图页面，性能难优化

MVC结构



REST（REpresentational State Transfer）机制

谢谢大家！

THANKS