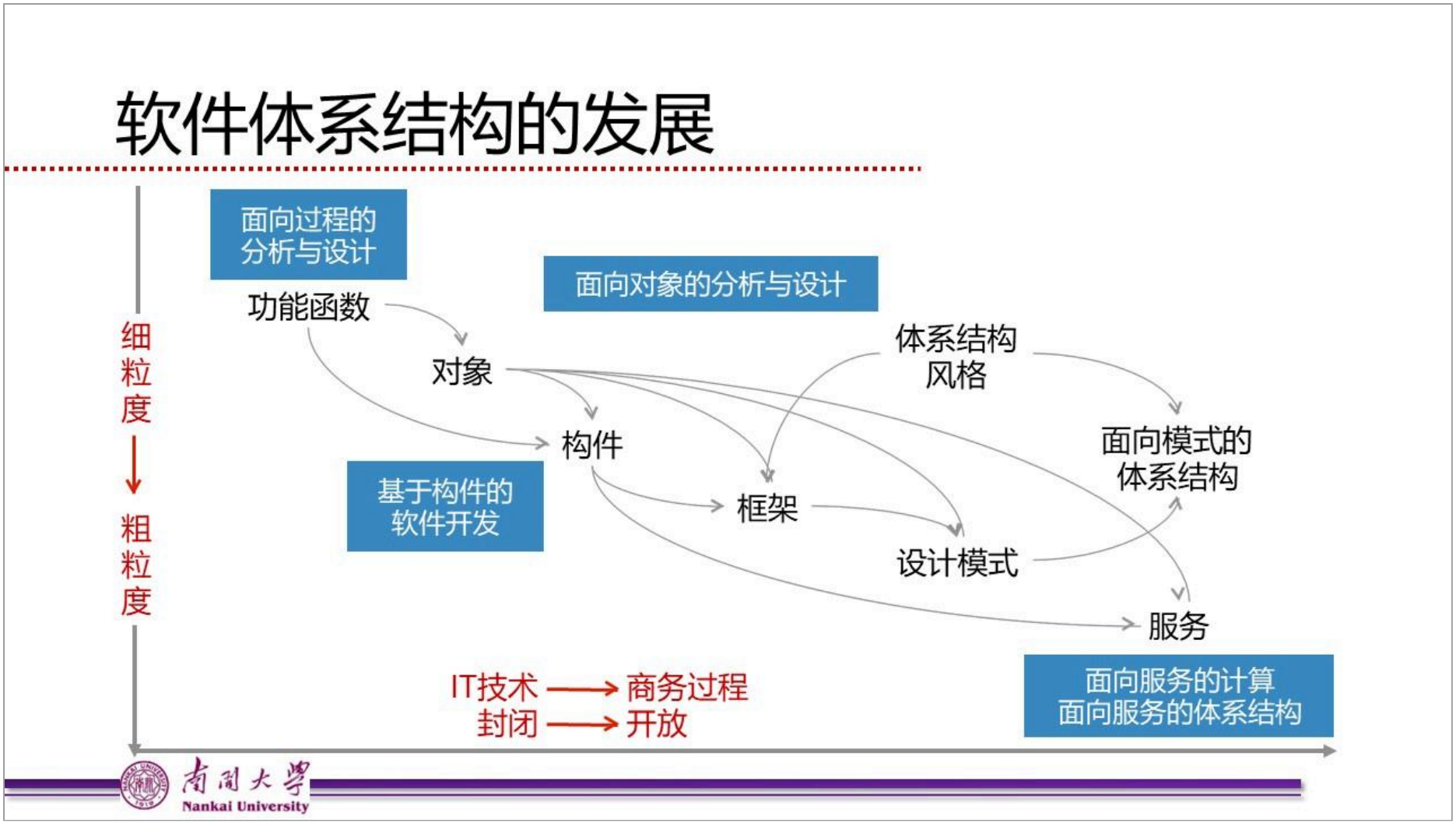
软件体系结构与设计原则

Xu Sihan （徐思涵）
College of Computer Science
Nankai University

Slides adapted from materials by Prof. Qiang Liu (Tsing Hua University) and Ivan Marsic (Rugers University)

软件体系结构的发展


面向过程的
分析与设计
功能函数
面向对象的分析与设计
对象
体系结构
风格
构件
面向模式的
体系结构
基于构件的
软件开发
框架
设计模式
服务
细
粒
度
粗
粒
度
IT技术 → 商务过程
封闭 → 开放
面向服务的计算
面向服务的体系结构
南開大學
Nankai University

风格、模式和框架

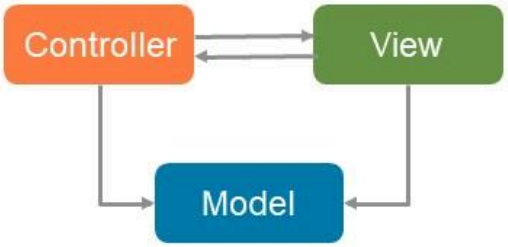
- 体系结构风格：用于描述某一特定应用领域中系统组织的惯用模式，反映了领域中众多系统所共有的结构和语义特性。
- 设计模式：描述了软件系统设计过程中常见问题的一些解决方案，通常是从大量的成功实践中总结出来的且被广泛公认的实践和知识。
- 软件框架：软件框架是由开发人员定制的应用系统的骨架，是整个或部分系统的可重用设计，由一组抽象构件和构件实例间的交互方式组成。

南開大學 Nankai University

风格、模式和框架

框架和体系结构的关系：

- 体系结构的呈现形式是一个设计规约，而框架则是“半成品”的软件；
- 体系结构的目的是指导软件系统的开发，而框架的目的是设计复用。



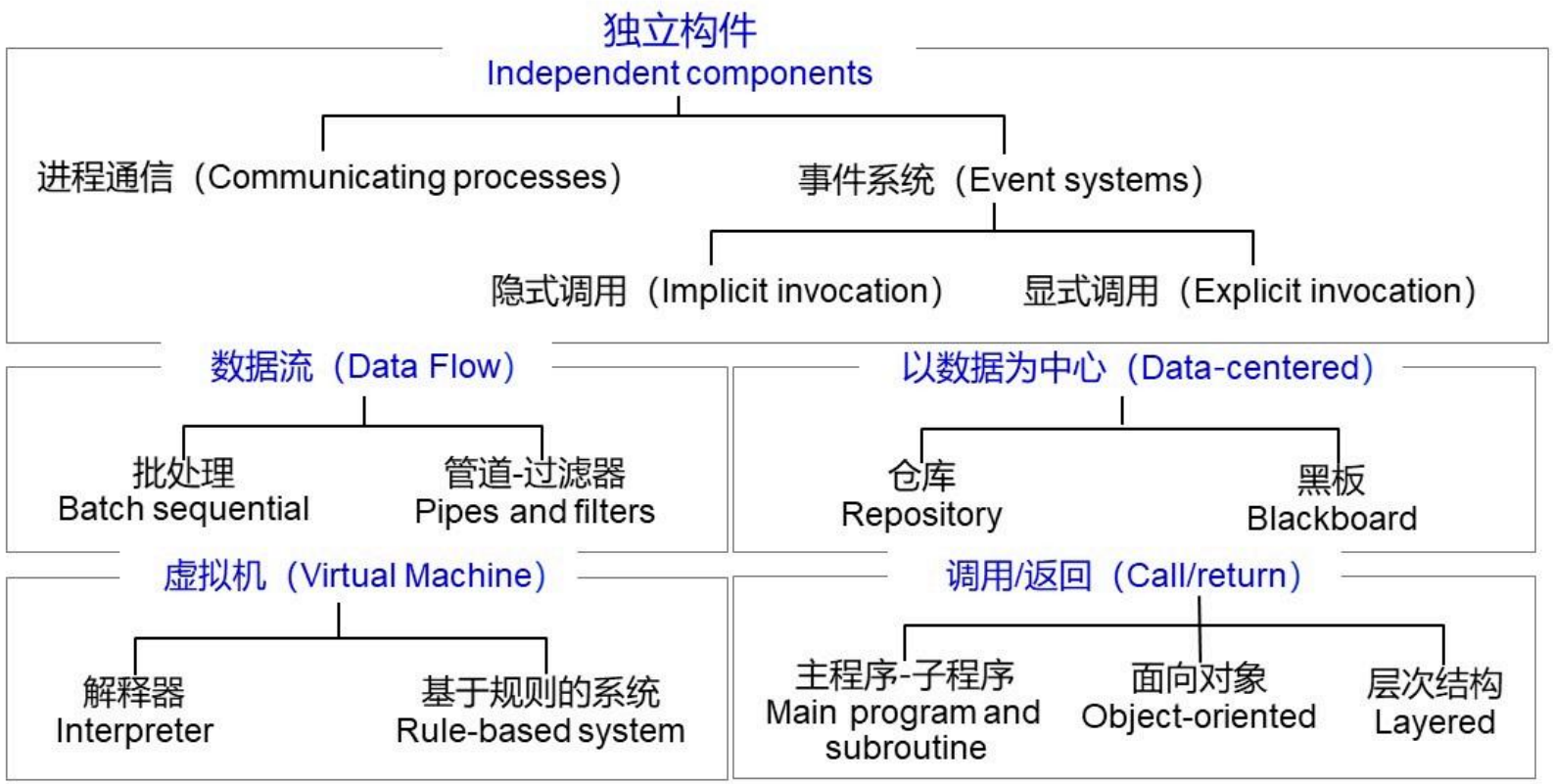
风格、模式和框架

框架和设计模式的关系：

- 框架给出的是整个应用的体系结构；而设计模式则给出了单一设计问题的解决方案，且可以在不同的应用程序或者框架中进行应用。
- *举例：*一个网络游戏可以基于网易的Pomelo框架开发，这是一个基于Node.js的高性能、分布式游戏服务器框架；在实现某个动画功能时，可能会使用观察者模式实现自动化的通知更新。
- 设计模式的目标是改善代码结构，提高程序的结构质量；框架强调的是设计的重用性和系统的可扩展性，以缩短开发周期，提高开发质量。

南開大學 Nankai University

常见的体系结构风格

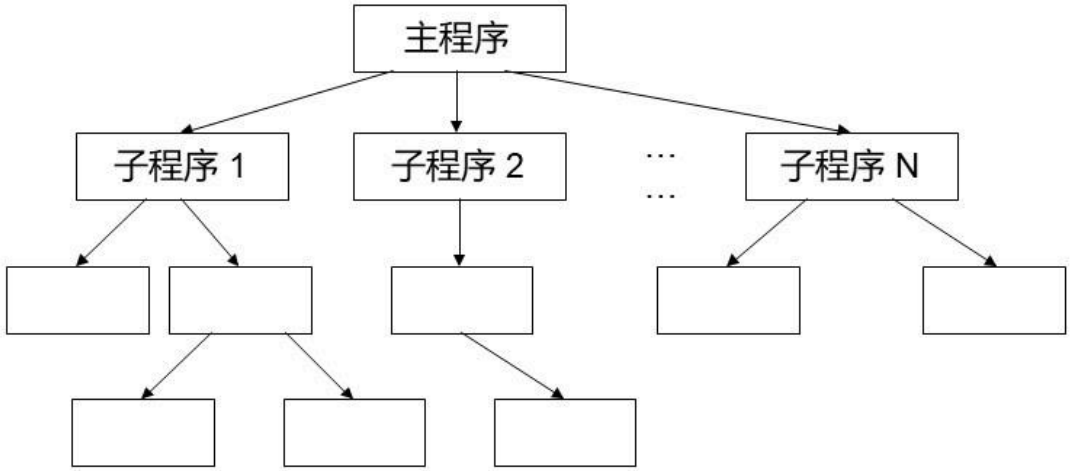


南開大學 Nankai University

6

主程序-子程序

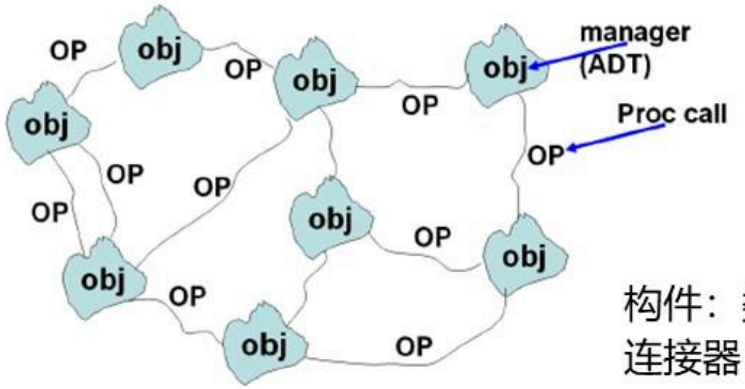
主程序-子程序风格是结构化程序设计的一种典型风格，从功能的观点设计系统，通过逐步分解和细化，形成整个系统的体系结构。



构件：主程序、子程序
连接器：调用-返回机制
拓扑结构：层次化结构

面向对象风格

- 系统被看作是对象的集合，每个对象都有一个它自己的功能集合；
- 数据及作用在数据上的操作被封装成抽象数据类型；
- 只通过接口与外界交互，内部的设计决策则被封装起来。

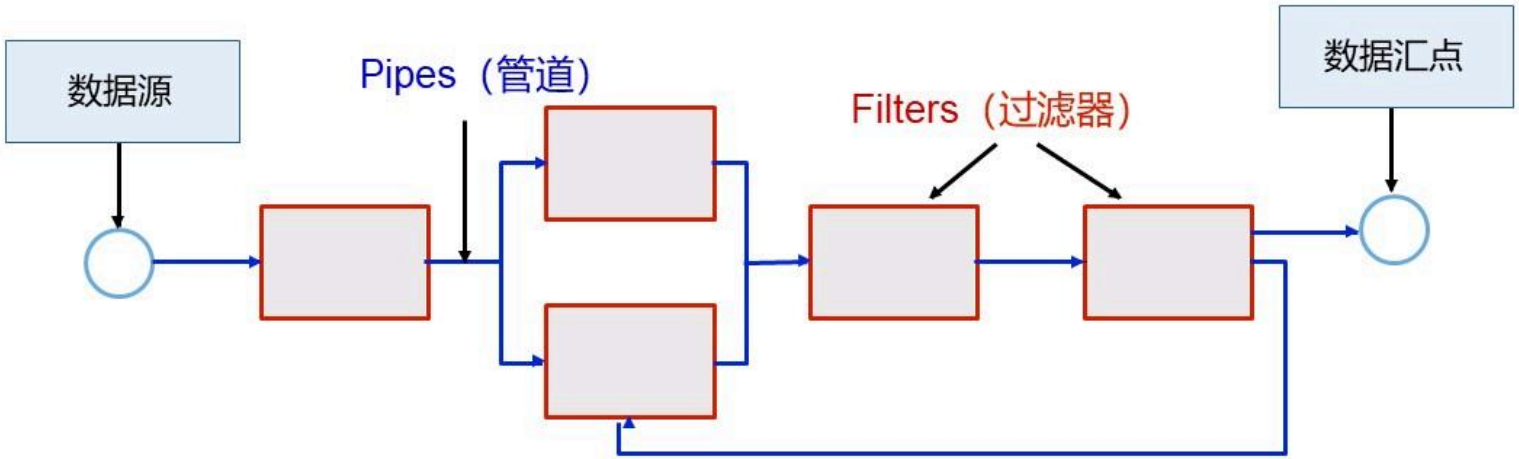


构件：类和对象

连接器：对象之间通过函数调用和消息传递实现交互

管道-过滤器风格

管道-过滤器风格把系统任务分成若干连续的处理步骤，这些步骤由通过系统的数据流连接，一个步骤的输出是下一个步骤的输入。



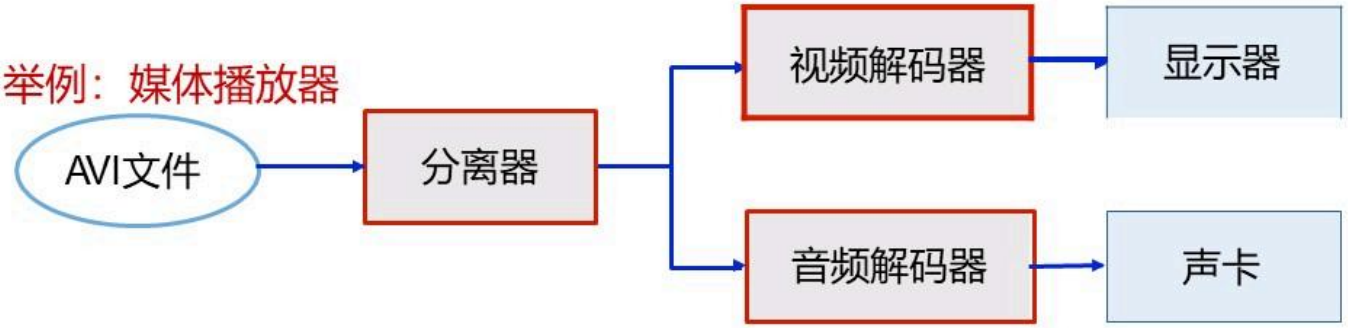
南開大學 Nankai University

9

管道-过滤器风格

- 管道-过滤器风格把系统任务分成若干连续的处理步骤，这些步骤由通过系统 的数据流连接，一个步骤的输出是下一个步骤的输入。

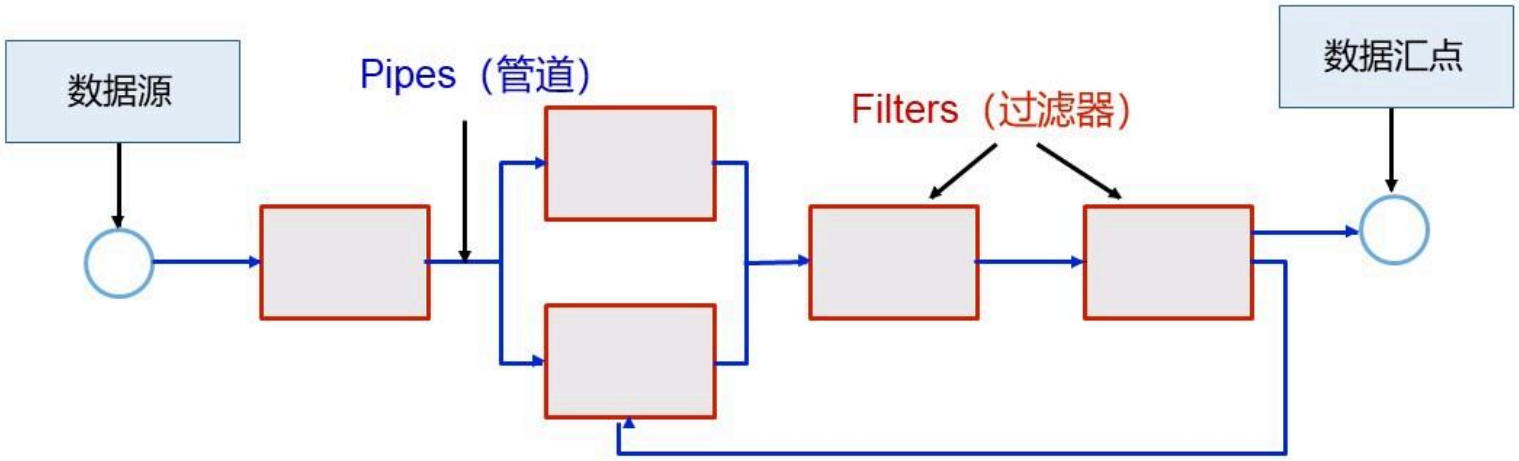
- 举例：媒体播放器



南開大學 Nankai University

管道-过滤器风格

管道-过滤器风格把系统任务分成若干连续的处理步骤，这些步骤由通过系统的数据流连接，一个步骤的输出是下一个步骤的输入。



这种结构不适合交互应用的情况，如果管道过长或者过滤器过于复杂的话，系统的性能就会大大降低

南開大學 Nankai University

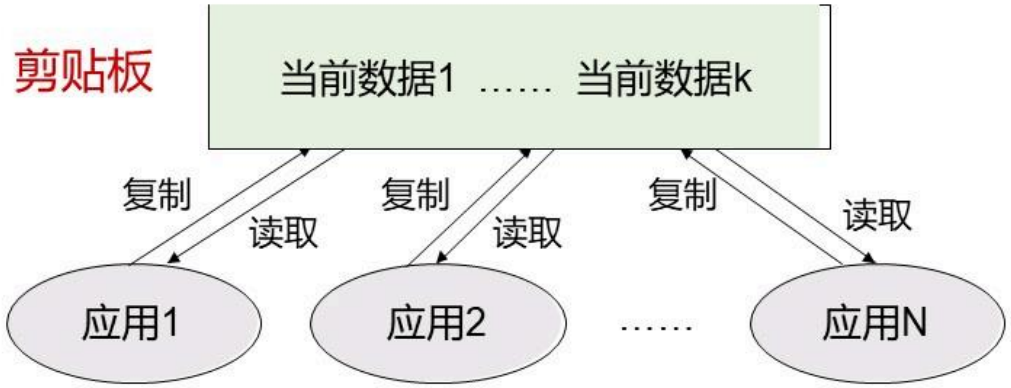
11

以数据为中心的风格

举例：剪贴板是一个用来进行短时间的数据存储，并在文档/应用之间进行数据传递和交换的软件程序。

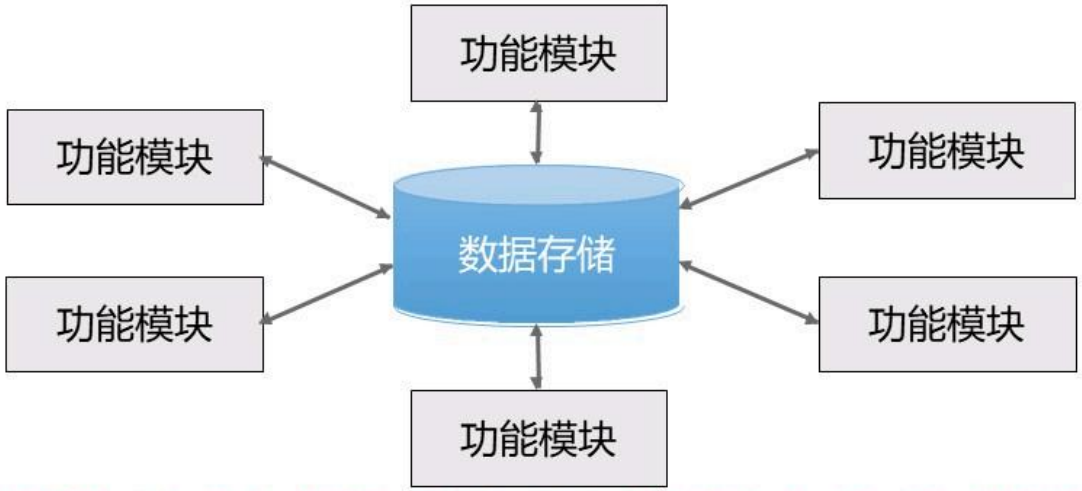
以数据为中心的风格

举例：剪贴板是一个用来进行短时间的数据存储，并在文档/应用之间进行数据传递和交换的软件程序。



以数据为中心的风格

仓库体系结构（Repository Architecture）是一种以数据为中心的体系结构，适合于数据由一个模块产生而由其他模块使用的情形。

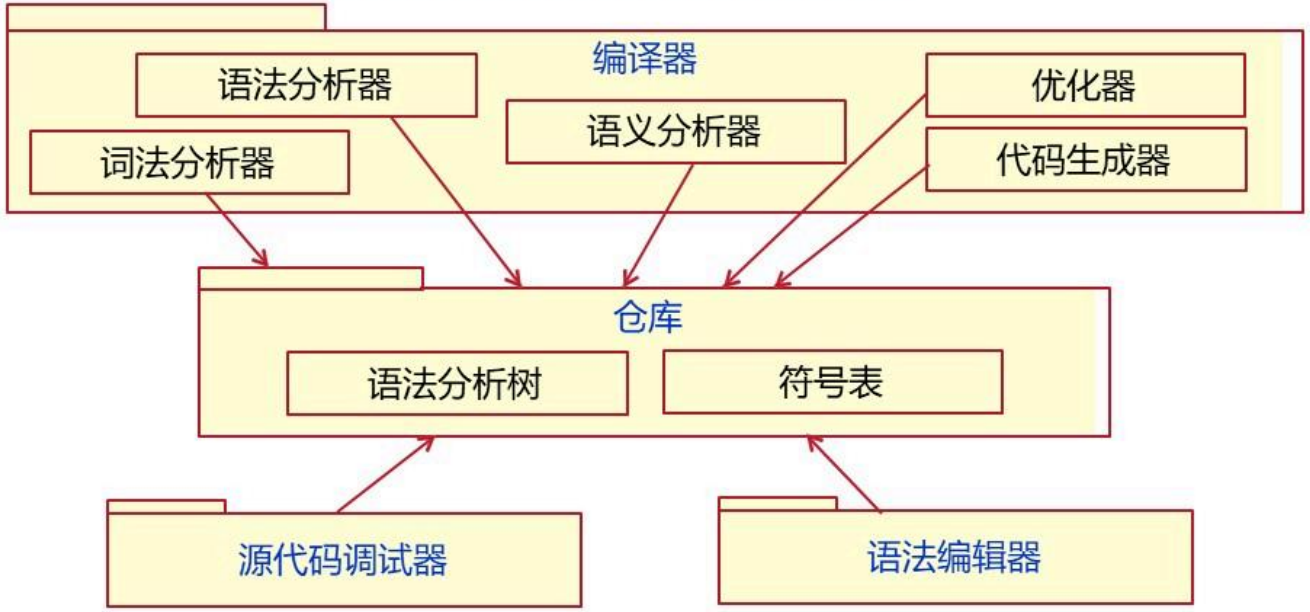


每个功能模块和仓库之间的耦合非常高，集中式的仓库很有可能成为系统性能的瓶颈

南開大學 Nankai University

14

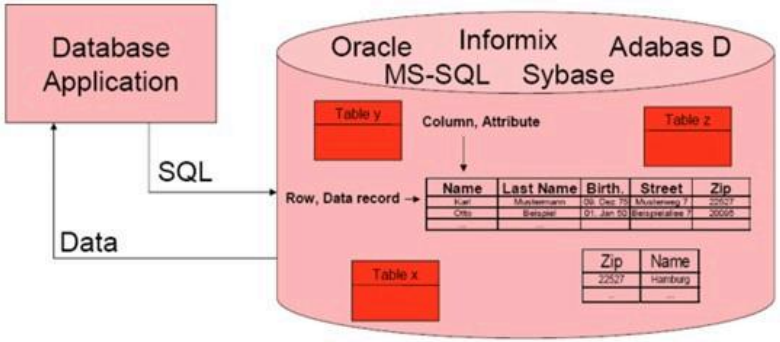
示例1：程序设计语言编译器

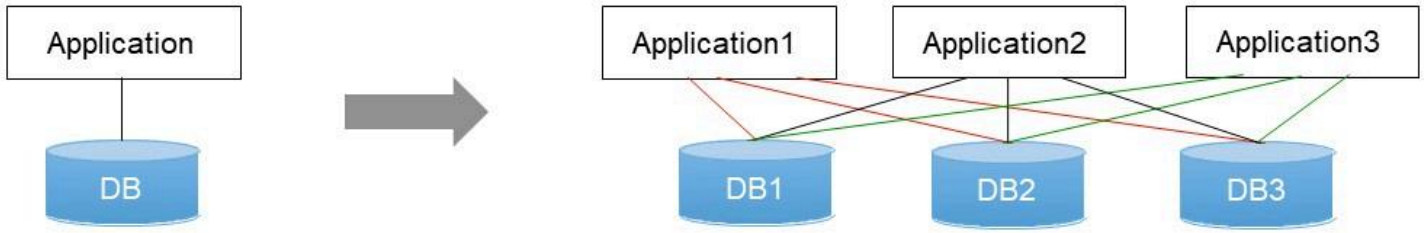


南開大學 Nankai University

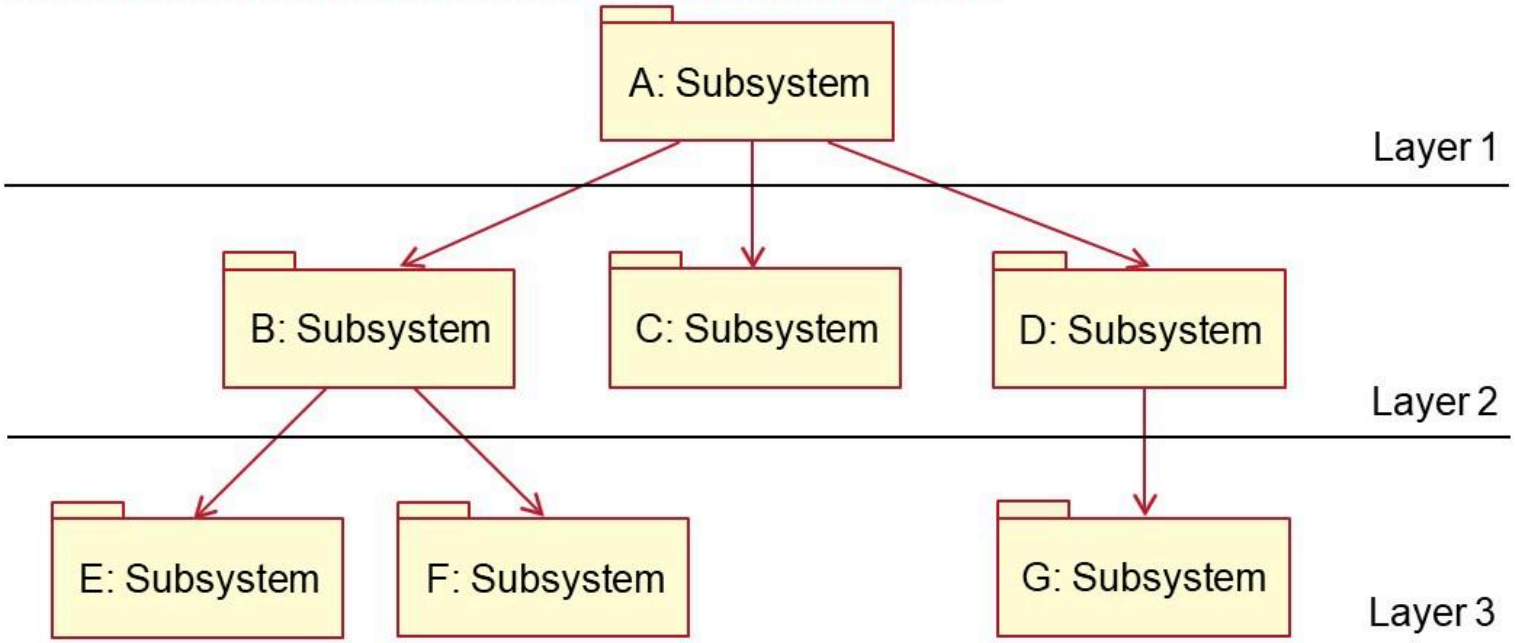
15

示例2：基于数据库的系统结构





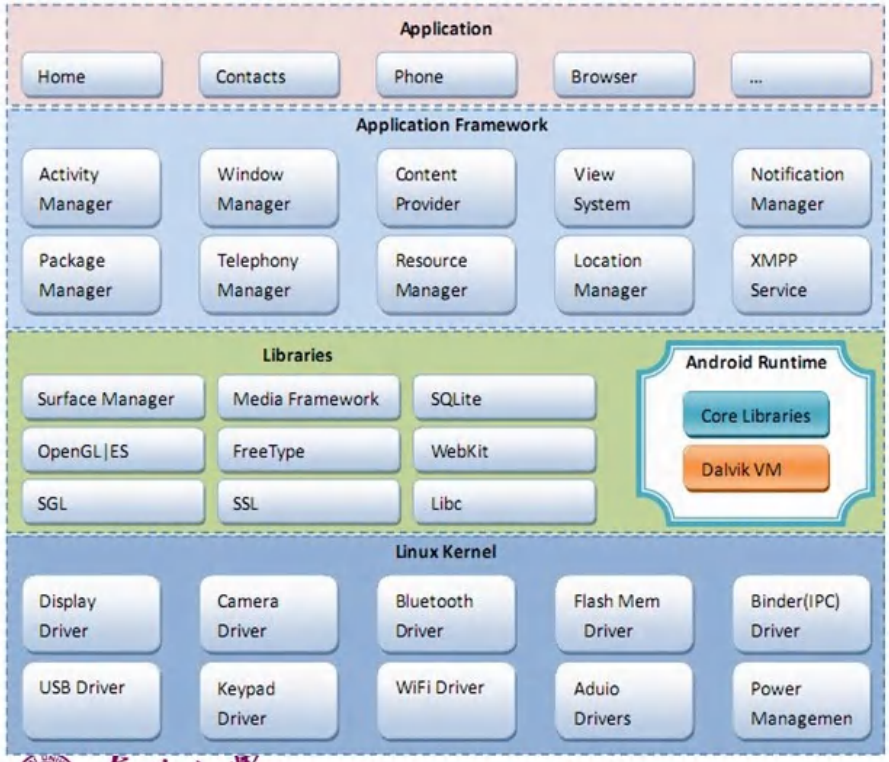
层次结构



南開大學 Nankai University

17

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。

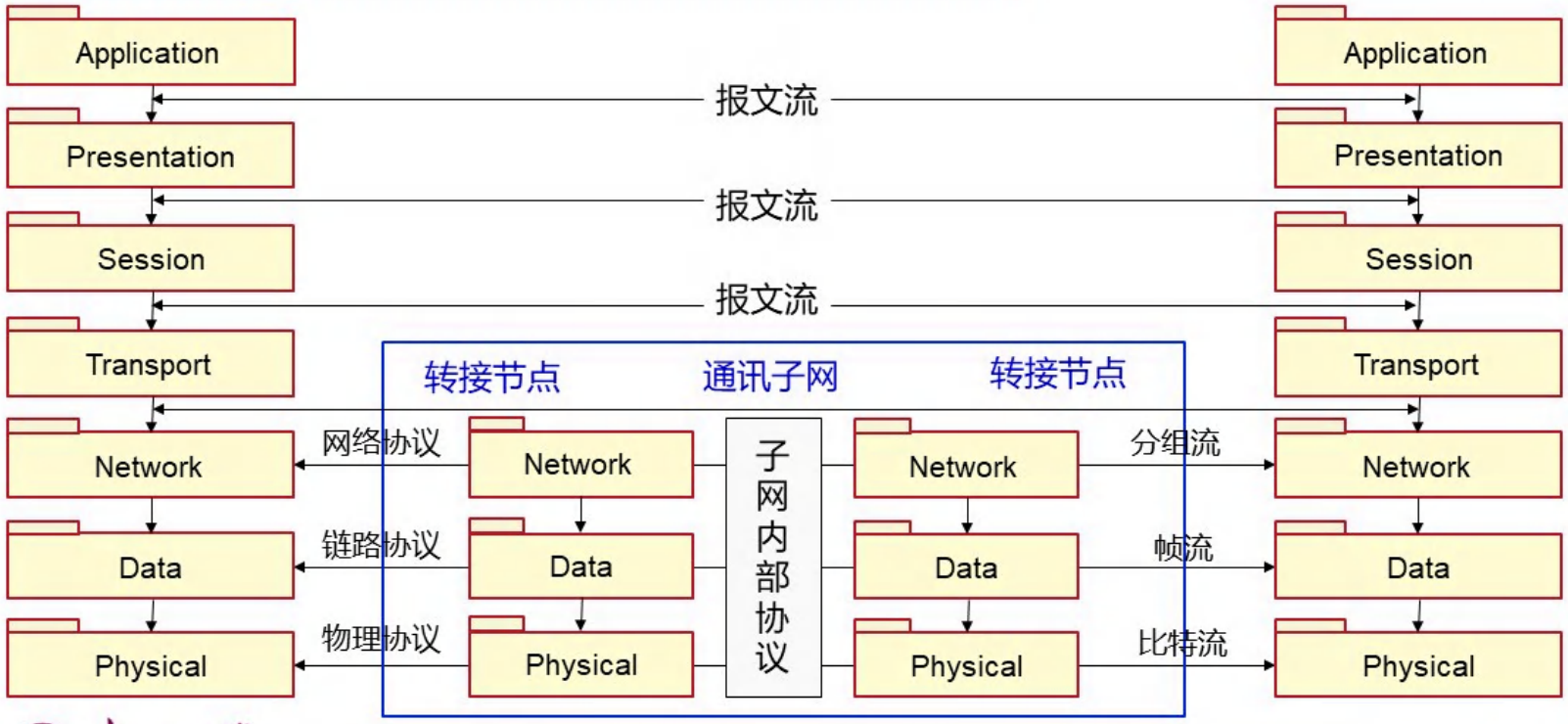
应用框架层：支持第三方开发者之间的交互，使其能够通过抽象方式访问所开发的应用程序需要的关键资源。

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。

南開大學 Nankai University

示例2：网络分层模型

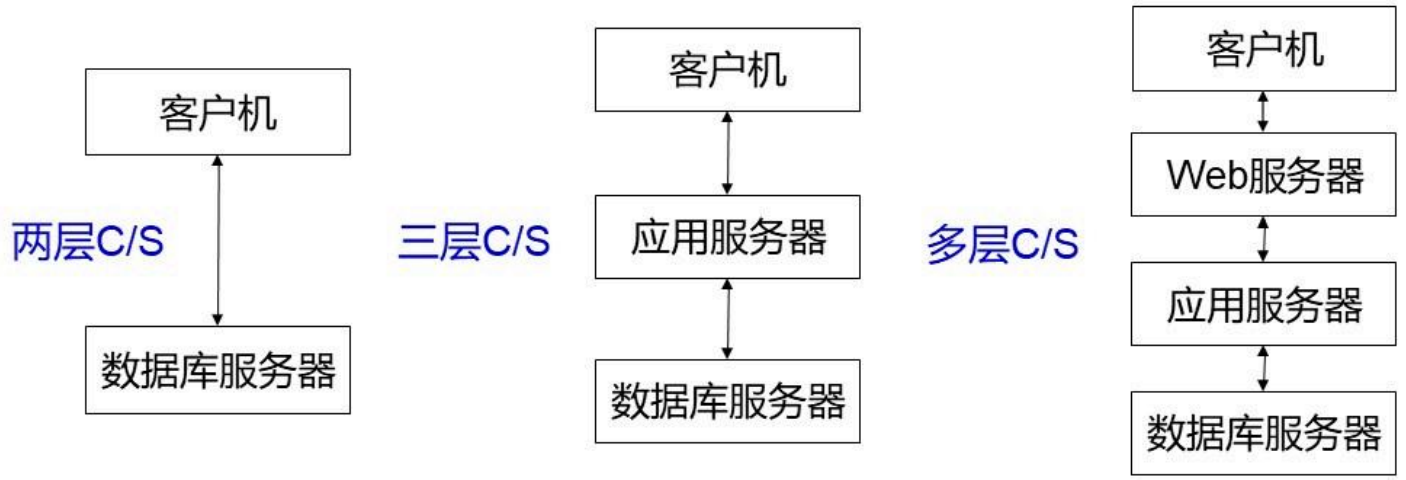


南開大學 Nankai University

19

客户机 / 服务器结构

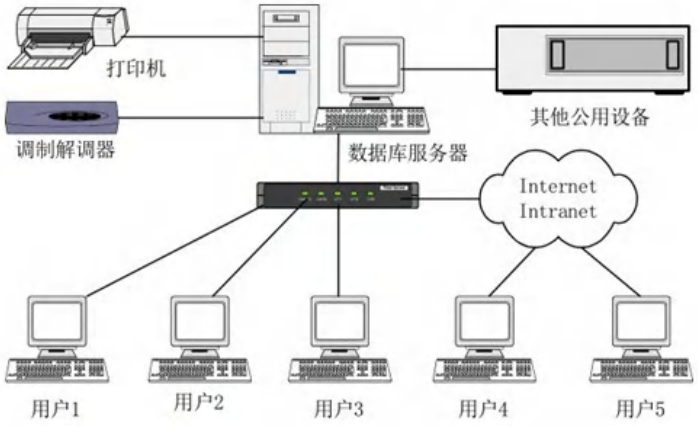
客户机 / 服务器体系结构（Client/Server）是一种分布式系统模型，作为服务器的子系统为其他客户机的子系统提供服务，作为客户机的子系统负责与用户的交互。

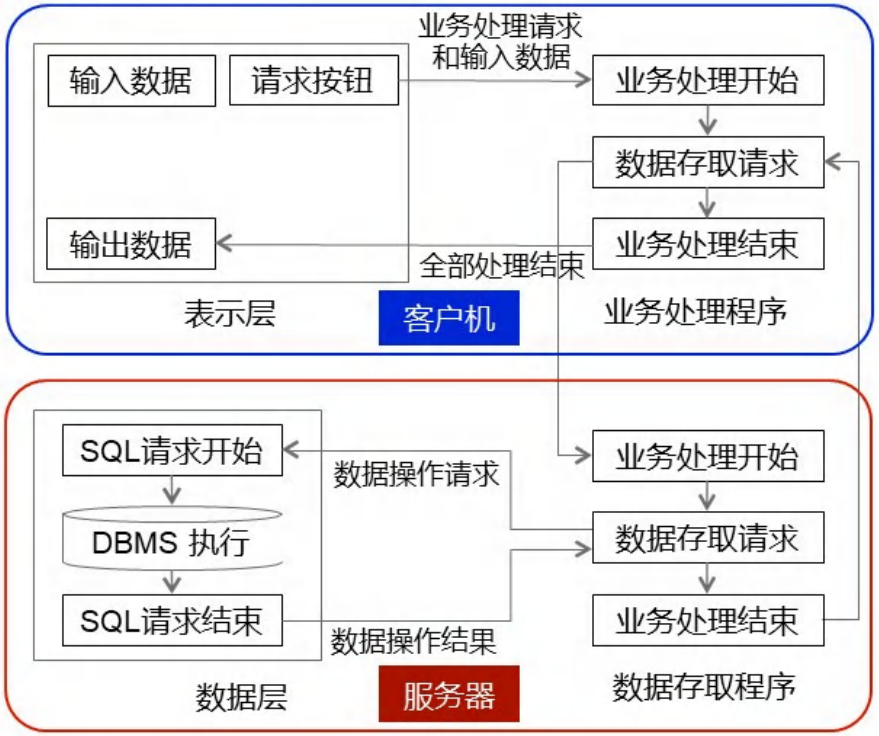


两层C/S结构

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



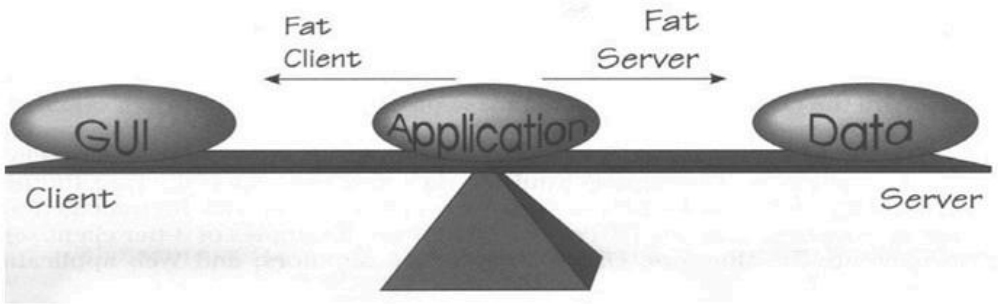


南开大学 Nankai University

胖客户端与瘦客户端

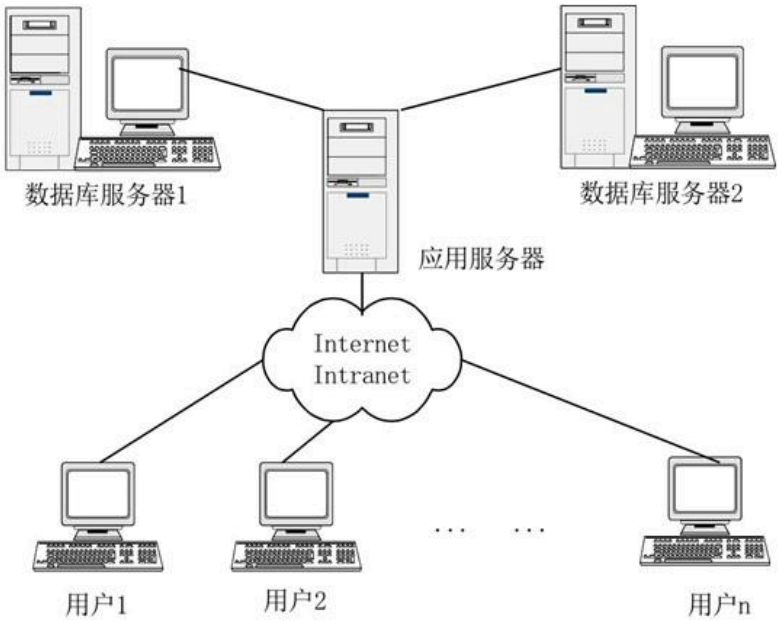
业务逻辑的划分比重：在客户端多一些还是在服务器端多一些?

- 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

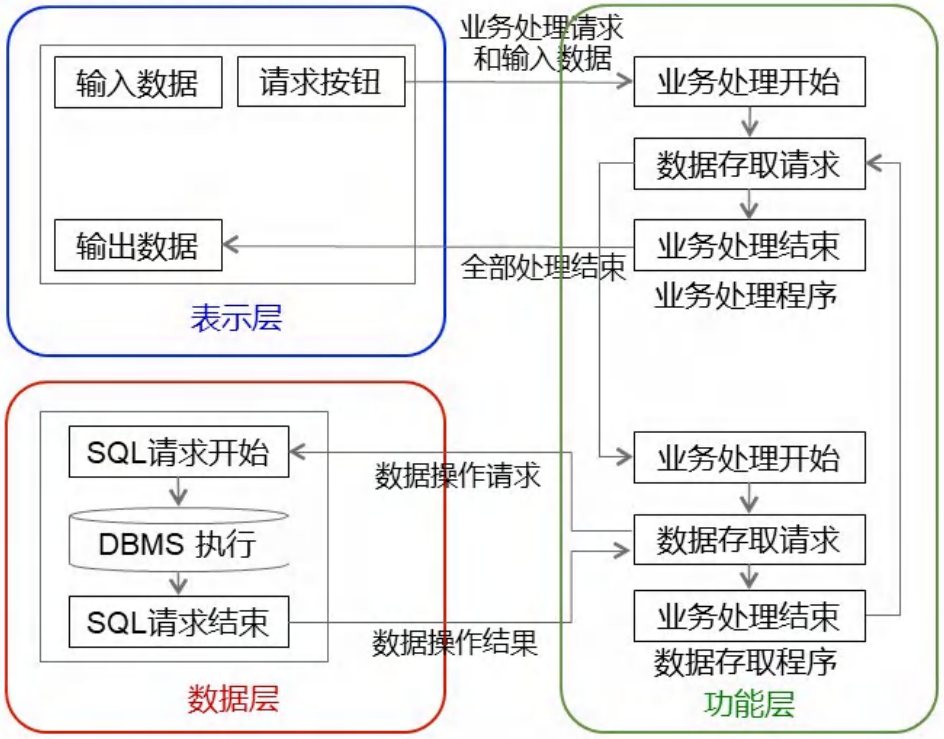
- 表示层：包括所有与客户机交互的边界对象，如窗口、表单、网页等。
- 功能层（业务逻辑层）：包括所有的控制和实体对象，实现应用程序的处理逻辑和规则。
- 数据层：实现对数据库的存储、查询和更新。



三层C/S结构

在增加新的业务处理时，可以相应地增加装有功能层的服务器，系统的灵活性和伸缩性变得很强。

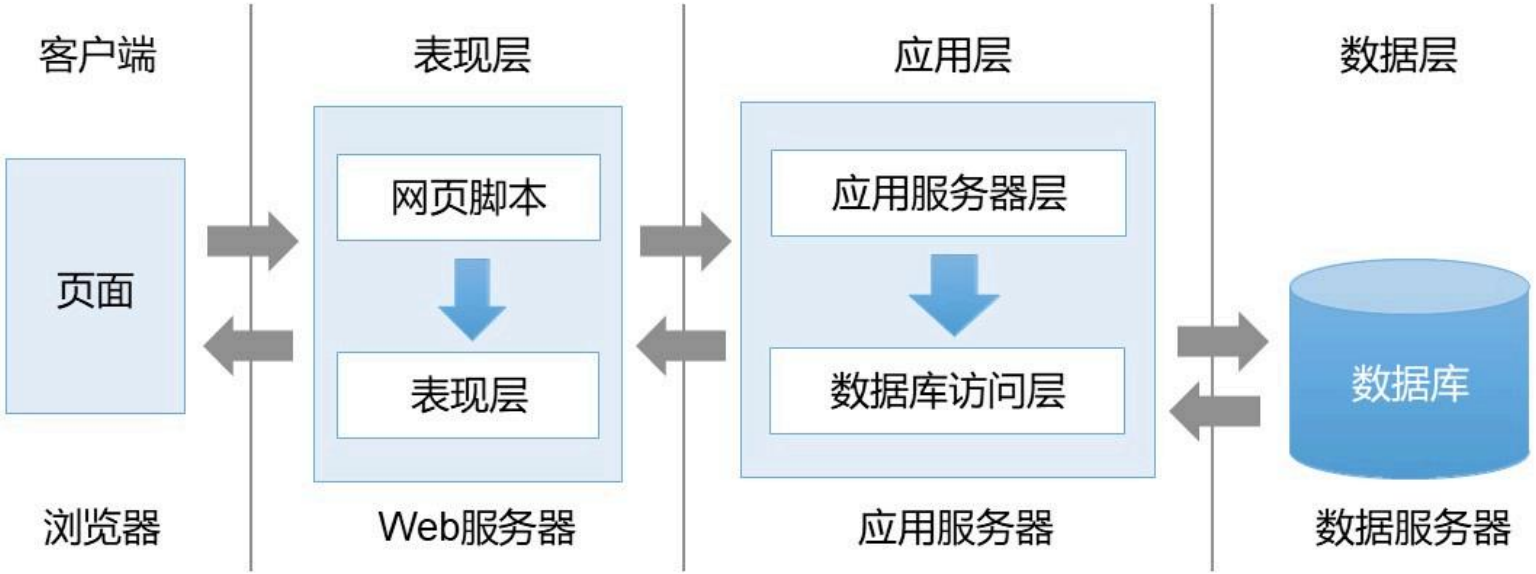
- 表示层：包括所有与客户机交互的边界对象，如窗口、表单、网页等。
- 功能层（业务逻辑层）：包括所有的控制和实体对象，实现应用程序的处理逻辑和规则。
- 数据层：实现对数据库的存储、查询和更新。



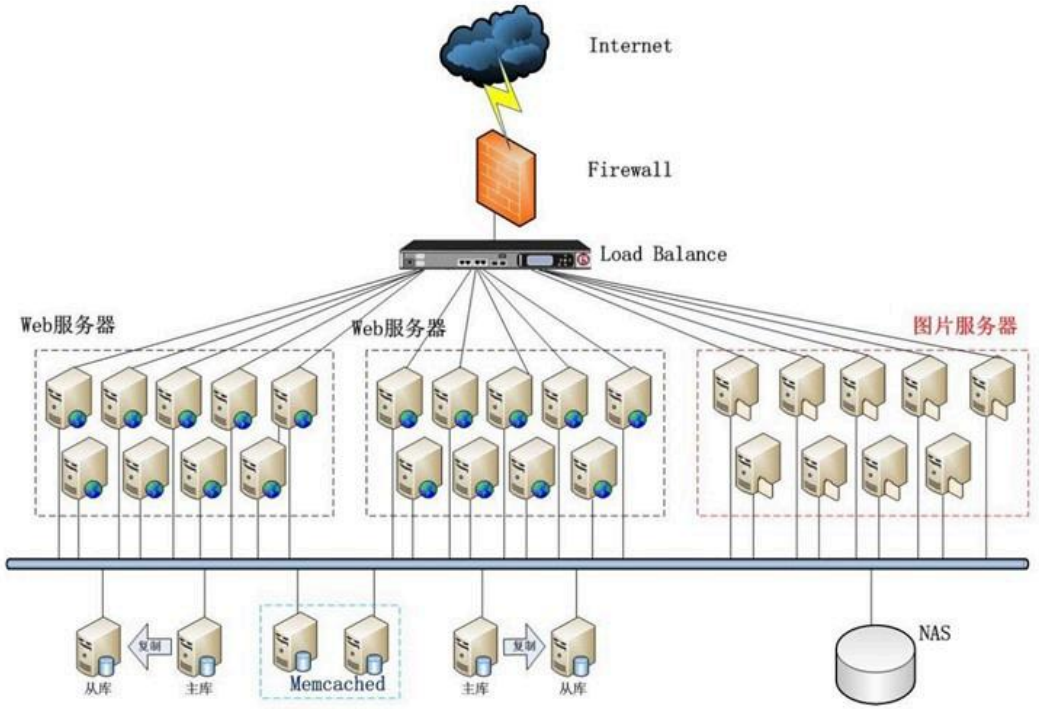
南開大學 Nankai University

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



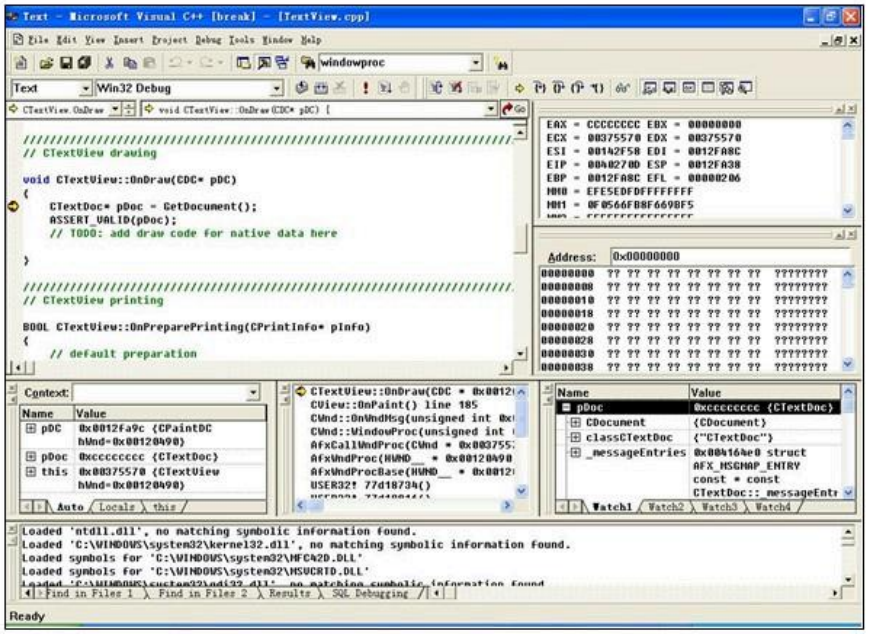
集群内各服务器上的内容保持一致
（通过冗余提高可靠性与可用性）

○ = ○ = ○ = ○ = 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）

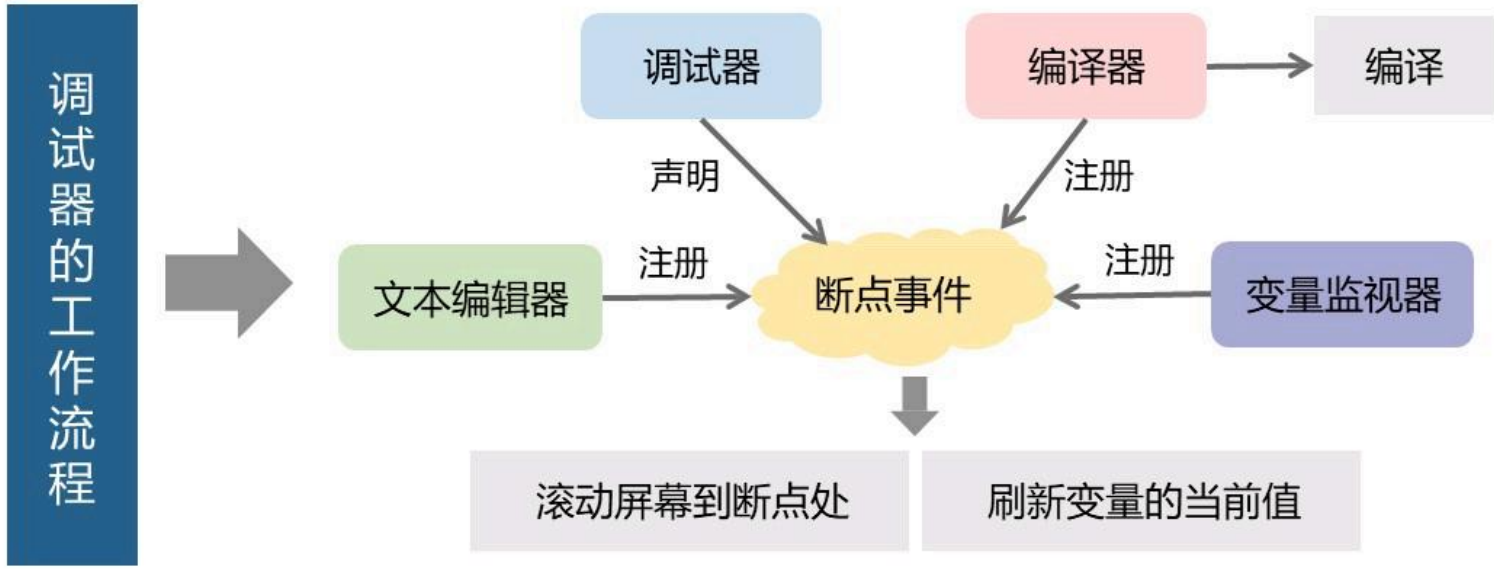
○ + ○ + ○ + ○ = 系统

南開大學 Nankai University


Text - Microsoft Visual C++ [break] - [TextView.cpp]
File Edit View Insert Project Debug Tools Window Help
windowproc
Text
Win32 Debug
// CTextView drawing
void CTextView::OnDraw(CDC* pDC)
{
CTextDoc* pDoc = GetDocument();
ASSERT_VALID(pDoc);
// TODO: add draw code for native data here
}
// CTextView printing
BOOL CTextView::OnPreparePrinting(CPrintInfo* pInfo)
{
// default preparation
EAX = CCCCCCCC EBX = 00000000
ECX = 00375570 EDX = 00375570
ESI = 0014 2F58 EDI = 0012FA8C
EIP = 00A0270D ESP = 0012FA38
EBP = 0012FA8C EFL = 00000206
Address: 0x00000000
Context:
Name Value
pDC 0x0012fa9c {CPaintDC hWnd=0x00120498}
pDoc 0xcccccccc {CTextDoc}
this 0x00375570 {CTextView hWnd=0x00120498}
Auto Locals this
CTextView::OnDraw(CDC * 0x0012
CView::OnPaint() line 185
CWnd::OnWndMsg(unsigned int 0x
CWnd::WindowProc(unsigned int
AfxCallWndProc(CWnd * 0x003755
AfxWndProc(HWND__ * 0x00120498
AfxWndProcBase(HWND__ * 0x0012
USER32! 77d1873b()
Name Value
pDoc 0xcccccccc {CTextDoc}
CDocument {CDocument}
classCTextDoc {"CTextDoc"}
_messageEntries 0x00416ae0 struct AFX_MSGMAP_ENTRY const * const CTextDoc:: messageEntr
Watch1 Watch2 Watch3 Watch4
Loaded 'ntdll.dll', no matching symbolic information found.
Loaded 'C:\WINDOWS\system32\kernel32.dll', no matching symbolic information found.
Loaded symbols for 'C:\WINDOWS\system32\MFC42D.DLL'
Loaded symbols for 'C:\WINDOWS\system32\MSVCRTD.DLL'
Find in Files 1 Find in Files 2 Results SQL Debugging
Ready

程序调试器的体系结构

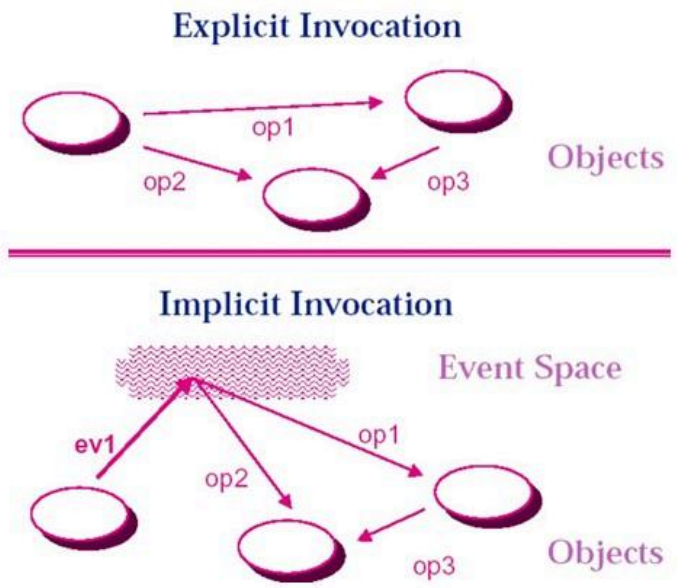
事件风格



南開大學 Nankai University

28

事件风格



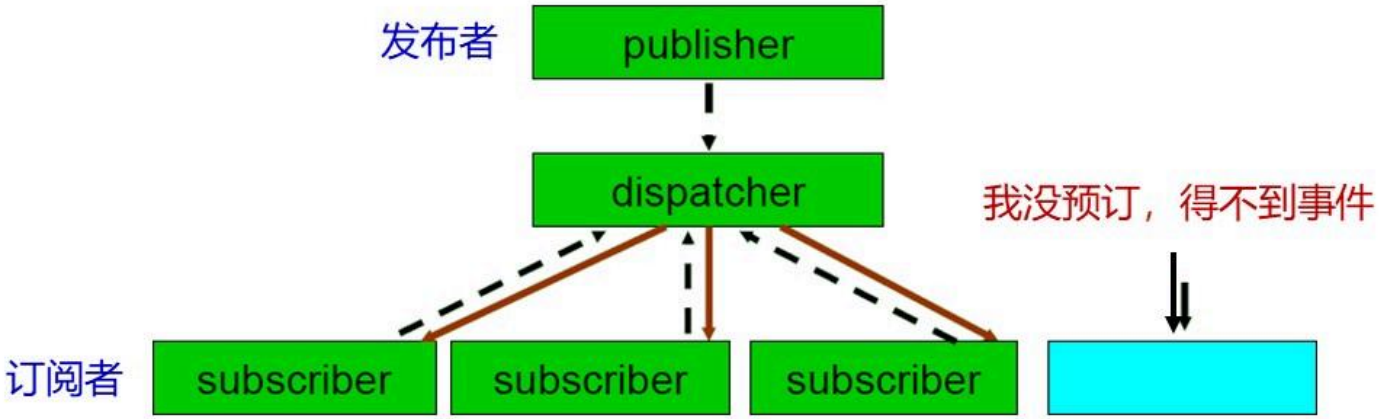
显式调用：

- 各个构件之间的互动是由显性调用函数或程序完成的
- 调用过程与次序是固定的、预先设定的

隐式调用：

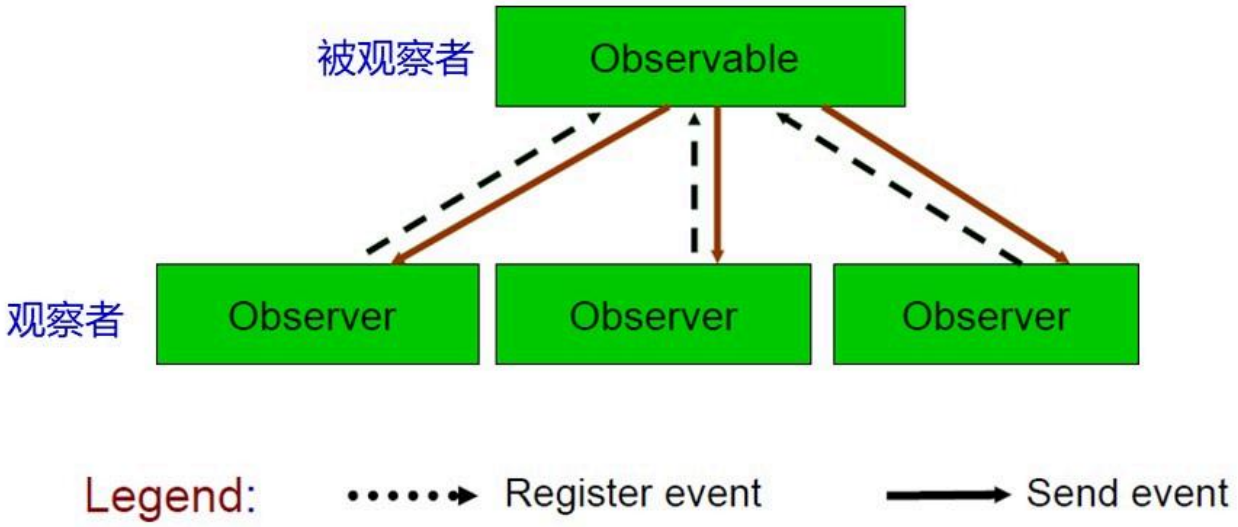
- 调用过程与次序不是固定的、预先未知
- 各构件之间通过事件的方式进行交互

事件风格的实现策略之一：选择广播式



说明：这种方式是有目的广播，只发送给那些已经注册过的订阅者。

事件风格的实现策略之二：观察者模式



软件体系结构风格的选择

简单地判断某一个具体的应用应该采取何种体系结构是非常困难的，需要借助于丰富的经验。

绝大多数实际运行的系统都是几种体系结构的复合：

- 在系统的某些部分采用一种体系结构而在其他部分采用另外的体系，故而需要将复合几种基本的体系结构组合起来形成复合体系结构。
- 在实际的系统分析和设计中，首先将整个系统作为一个功能体进行分析和权衡，得到适宜的和最上层的体系结构；如果该体系结构中的元素较为复杂，可以继续进行分解，得到某一部分的局部体系结构。

将焦点集中在系统总体结构的考虑上，避免较多地考虑使用的语言、具体的技术等实现细节上。

单选题 1分

与C/S架构的信息系统相比，B/S架构的信息系统的优势是（）。

A 具备更高的安全性

B 更容易部署和升级维护

C 具备更强的事务处理能力，易于实现复杂的业务流程

D 用户界面友好，具有更快的响应速度

南开大学 Nankai University