中国科学院大学
University of Chinese Academy of Sciences

第11课
持久（下）
——超越时空

面向对象程序设计
Object-Oriented Programming

说明

- 仅供选修国科大“面向对象程序设计”课程的学生复习功课时参考，**不得用于其它目的的翻印制作。**
- 这里所提供的只是课堂讲授的部分内容，**万不可误认为课件以外的内容都不要求掌握。**

2025年秋季课程安排

- 第1周，导言
- 第2周，认识软件系统的复杂性、有组织的复杂系统
- 第3周，10.3国庆假期
- 第4周，程序语言、开发的发展
- 第5周，建模方法和建模语言
- 第6周，抽象
- 第7周，课堂研讨1：扩展点的需求分析与建模
- 第8周，课堂研讨2：扩展点的需求分析与建模
- 第9周，封装和模块化
- 第10周，复用与解耦1：层次结构（组合、继承、多态）
- 第11周，复用与解耦2：类型信息、抽象类、接口
- 第12周，持久化（时间+空间）
- 第13周，并发
- 第14周，课堂研讨3：设计模式的优化与重构
- 第15周，课堂研讨4：大作业汇报
- 第16周，课堂研讨5：大作业汇报
- 第17周，课堂研讨6：大作业汇报
- 第18周，课堂研讨7：大作业汇报
- 第19/20周，考试

第2周给出开源项目选题清单

第3周确定选题，需要具体到特定模块

第13周提交大作业报告（第二版）

第8周提交大作业报告（初版）

第18周提交大作业报告（第三版），以及扩展代码

中国科学院大学 University of Chinese Academy of Sciences

本学期课程大作业

1、目标代码（开源项目或其他系统）的阅读与分析

需求分析与建模：用例说明、用例图、类图等

核心流程分析：类图、类关系图、顺序图等

复杂设计意图分析：类图、类关系图的优化、设计模式的应用等

提交系统分析与设计报告

2、目标代码的扩展设计与实现

扩展点：功能扩展、非功能优化等，至少1项

围绕扩展点，完成需求分析与建模、核心流程设计、设计模式的应用

提交系统分析与设计报告（扩展版）

提交可运行的代码

本学期课程大作业：大家关心的问题（1/2）

1. 如何选题：二选一

① **推荐项目：**课程提供开源项目建议清单，具体模块的源码阅读分析、扩展实现

② **自荐项目：**学生自荐的开源项目模块或参与的研究所科研实践任务源码阅读分析、扩展实现（源码规模达到万级）

2. 是否组队：不组队，以个人为单位

3. 作业形式：

① 报告 - 系统分析与设计报告

② 代码 - 功能扩展、非功能优化等至少1项扩展点的源代码实现

③ 汇报 - 课堂研讨：按学号、每人2次（扩展点需求分析研讨、结题研讨）

根据国科大相关要求，《报告》应对大模型使用情况进行标注和说明，将作为大作业评分关键依据

4. 作业提交地址：提交至Gitee等源码托管平台的个人repo

5. 如何评分：

① 报告的质量

② 扩展点能否正常运行，是否有配套的测试代码，扩展后的实际效果（测试对比、社区贡献等）

③ 课堂研讨表现占比25%

本学期课程大作业：大家关心的问题（2/2）

6. 进度安排：

第2周，授课团队给出开源项目建议清单

第3周，学生确定选题，需要具体到开源项目或特定科研实践任务的具体模块

第7、8周，课堂研讨（扩展点需求分析），提交系统分析与设计报告（初版）

- 大模型辅助：利用大模型对目标项目的模块代码进行分析，从需求分析与建模、核心流程设计分析、高级设计意图分析依次展开，并给出扩展点的需求分析与建模
- 扩展点包括：功能扩展、非功能优化，至少1项
- 课堂研讨：主要讨论扩展点的“需求分析”进行PPT分享和研讨
- 参考报告

  雷正宇：https://842376130.gitbook.io/fastjson-learning-report/

  曾鸿斌：https://yuankong11.gitbook.io/fastjson/

第13周，修订并提交系统分析与设计报告（第二版）

- 修正大模型生成内容与实际情况的偏差
- 确认并完善扩展点的需求，完成扩展点的初步设计

第15-18周，课堂研讨（结题研讨），对目标项目模块以及扩展工作进行整体汇报

第18周，提交系统分析与设计报告（最终版），提交扩展后的源代码

助教有话说

过去的优秀作业：

- 源码阅读：
- 雷正宇：https://842376130.gitbook.io/fastjson-learning-report/
- 曾鸿斌：https://yuankong11.gitbook.io/fastjson/
- 代瀚堃：https://toscode.gitee.com/dhank/object-oriented-programming

- 设计实现：
- 王子曰：https://gitee.com/woopc/oopclass
- 高梓源：https://github.com/ETOgaosion/UCAS_OOP_course
- 肖书慧：https://gitee.com/xyaoerworkspace/OOP-works

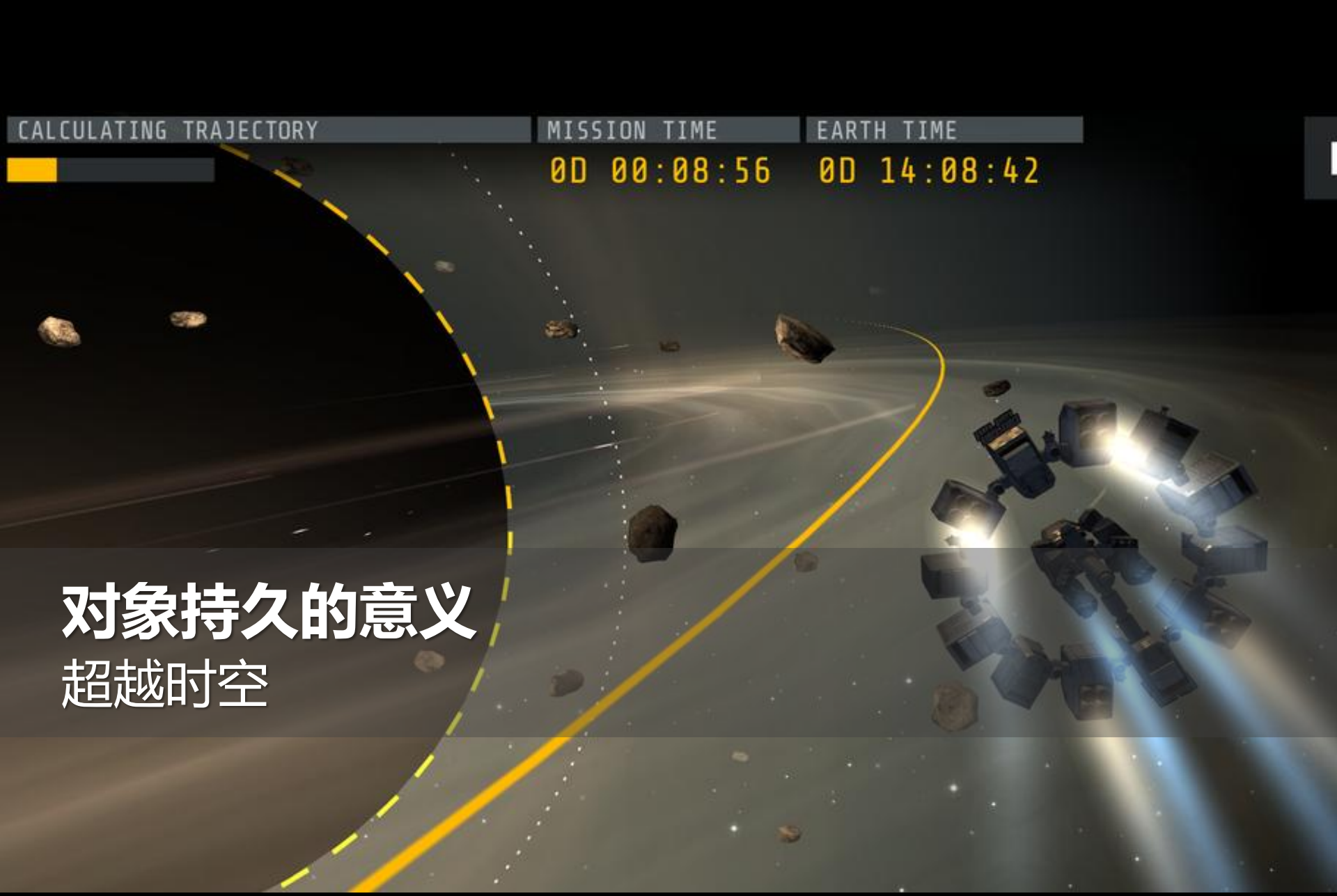
课堂研讨：设计模式优化与重构（加分）

- **研讨时间**：**2025.12.19**
- **研讨内容**：围绕大作业选定的项目内容，进行设计模式优化与重构，包括但不限于
  - 如何利用大模型对当前代码存在的设计不足、问题或缺陷进行检测和诊断
  - 需要采用哪些设计模式进行重构，并讲解这些设计模式的概念和特点
  - 如何利用大模型进行重构，并分析重构后的效果
- **研讨要求**：结合大作业报告（或PPT）、源代码等进行讲解，每人不超过15分钟
- **自愿报名**：最多7人
- **成绩奖励**：最高奖励+10%

https://en.wikipedia.org/wiki/Software_design_pattern

学习目标

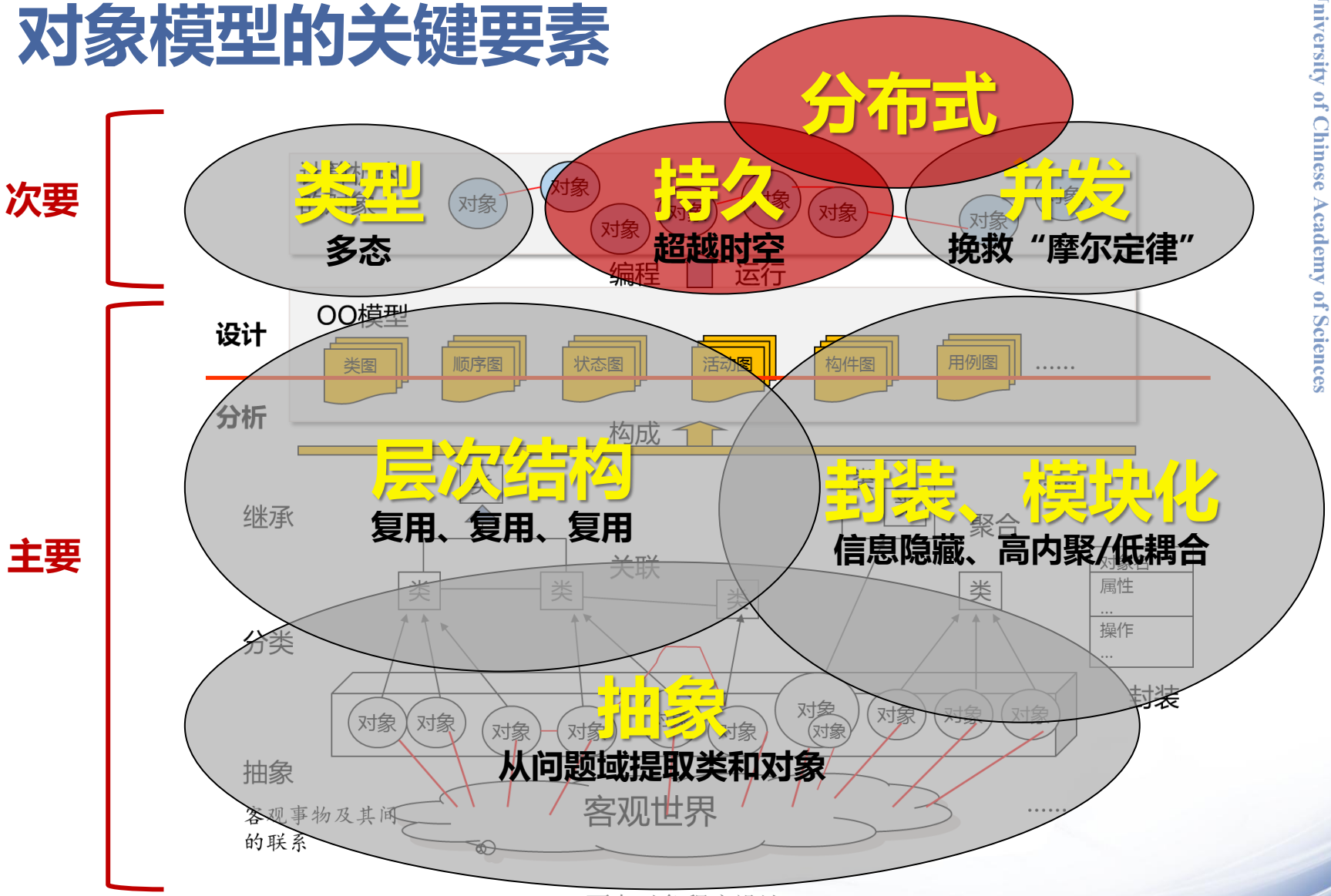
- 面向分布式系统的对象持久
  - # 分布式系统
  - # 分布式系统模型：物理模型、体系结构模型、架构模式
  - # 分布式系统的挑战



对象持久的意义

超越时空

对象模型的关键要素


次要
主要
分布式
类型
多态
持久
超越时空
并发
挽救“摩尔定律”
编程
运行
对象
OO模型
设计
类图
顺序图
状态图
活动图
构件图
用例图
.....
分析
构成
层次结构
复用、复用、复用
封装、模块化
信息隐藏、高内聚/低耦合
继承
聚合
关联
类
属性
...
操作
...
分类
抽象
封装
抽象
从问题域提取类和对象
客观事物及其间的联系
客观世界
.....

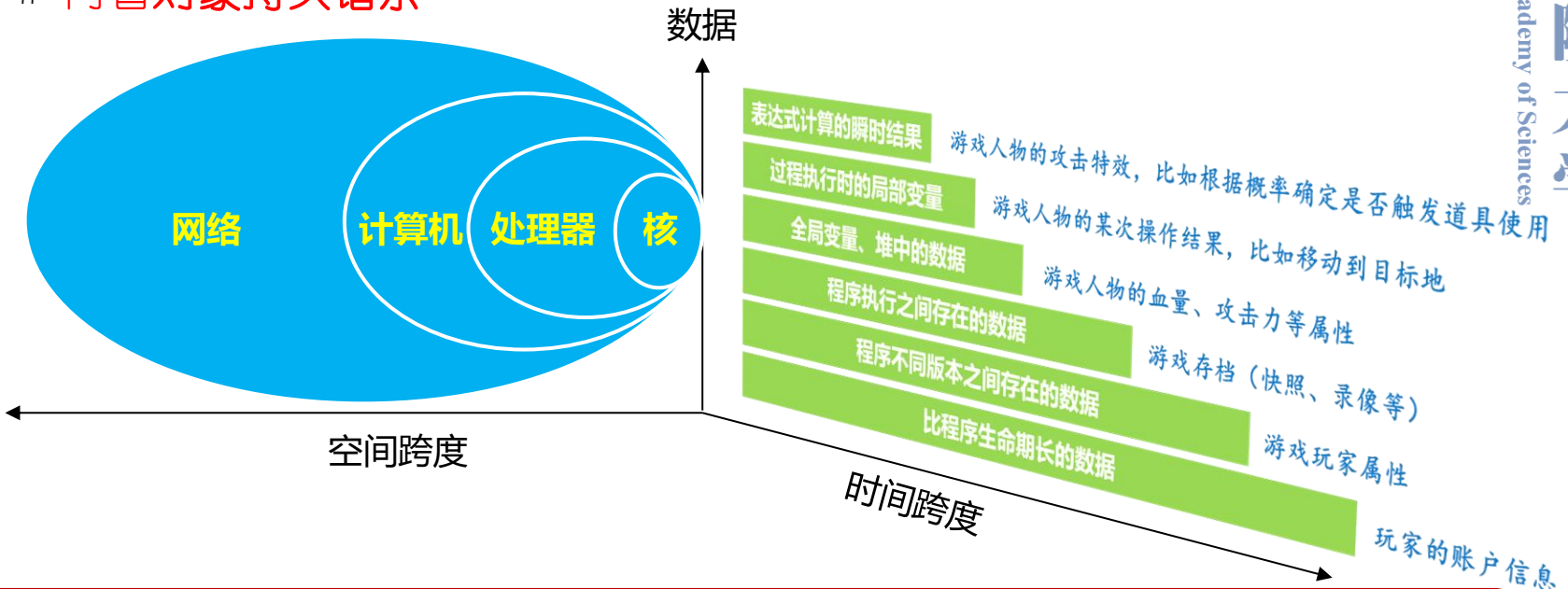
中国科学院大学
University of Chinese Academy of Sciences

面向分布式系统的对象持久
——分布式系统#初探#

从空间维度看对象持久

- 软件中的一个对象，会在空间分布的不同处理器核上传输、运行

  # 再看对象持久谱系



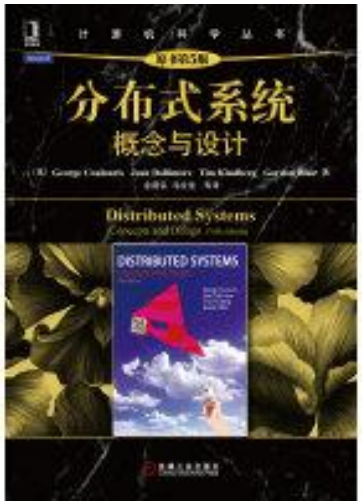
需要考虑谱系中对象数据的跨空间持久需求，选择合理的方案

是否真的需要远程处理？何种数据访问协议(RMI/SOAP/RESTful/…)？

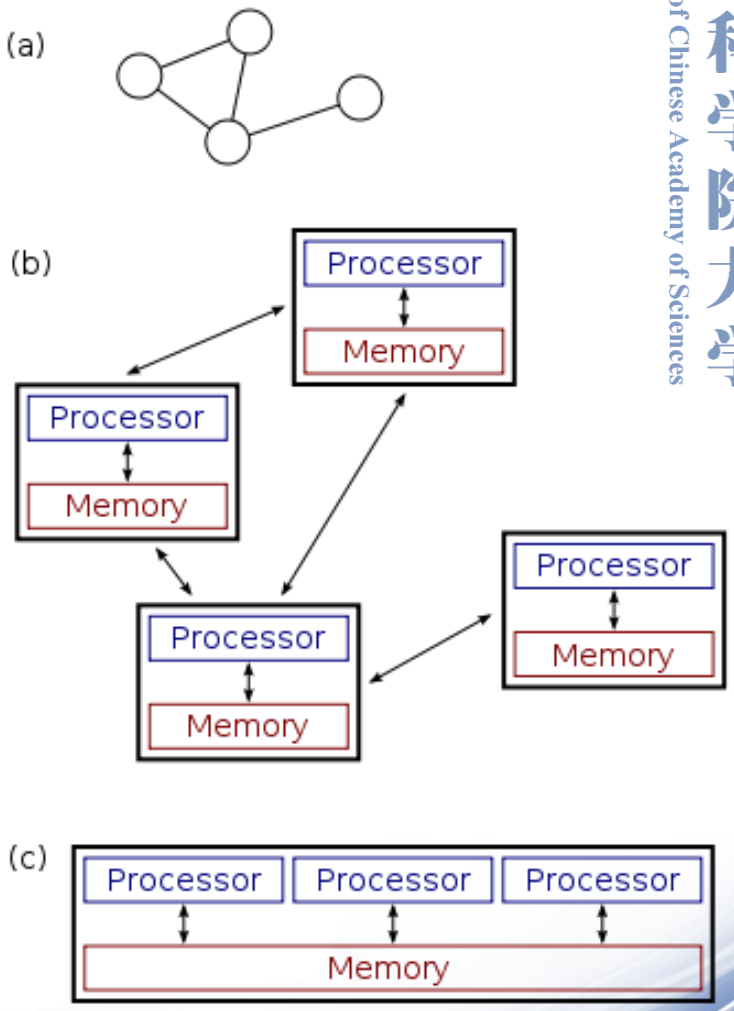
何种对象模型(有状态/无状态、同步/异步/并行、…)？等等

分布式系统(Distributed System)

A *distributed system* is a model in which components located on networked computers communicate and coordinate their actions by passing messages[1]



分布式系统：概念与设计（原书第5版）
[Distributed Systems：Concepts and Design,Fifth Edition]
深入理解互联网和其他分布式系统的体系结构、算法和设计的著作
George Coulouris，Jean Dollimore，Tim Kindberg，Gordon Blair 著



[1] Coulouris, George; Jean Dollimore; Tim Kindberg; Gordon Blair (2011). Distributed Systems: Concepts and Design (5th Edition). Boston: Addison-Wesley. ISBN 0-132-14301-1.

分布式系统模型——物理模型

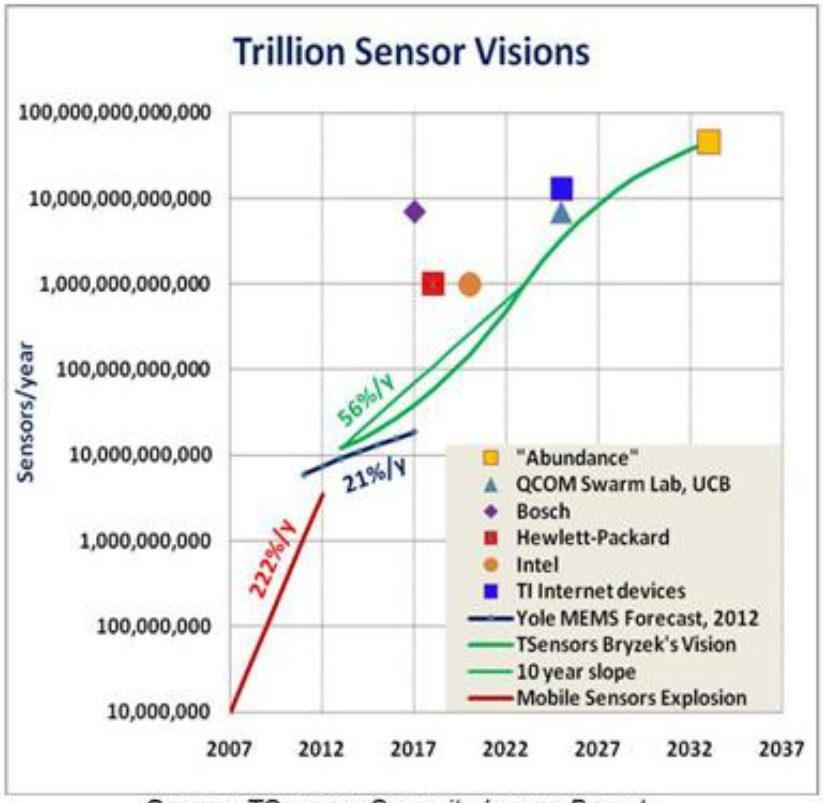
- 局域网 → 桌面互联网 → 移动互联网 → 物联网 → 万物互联网

 # Internet of Everything

Everything

= {People, Data, Processes, Things}

 # Distributed systems of systems



Source: TSensors Summit, Janusz Bryzek

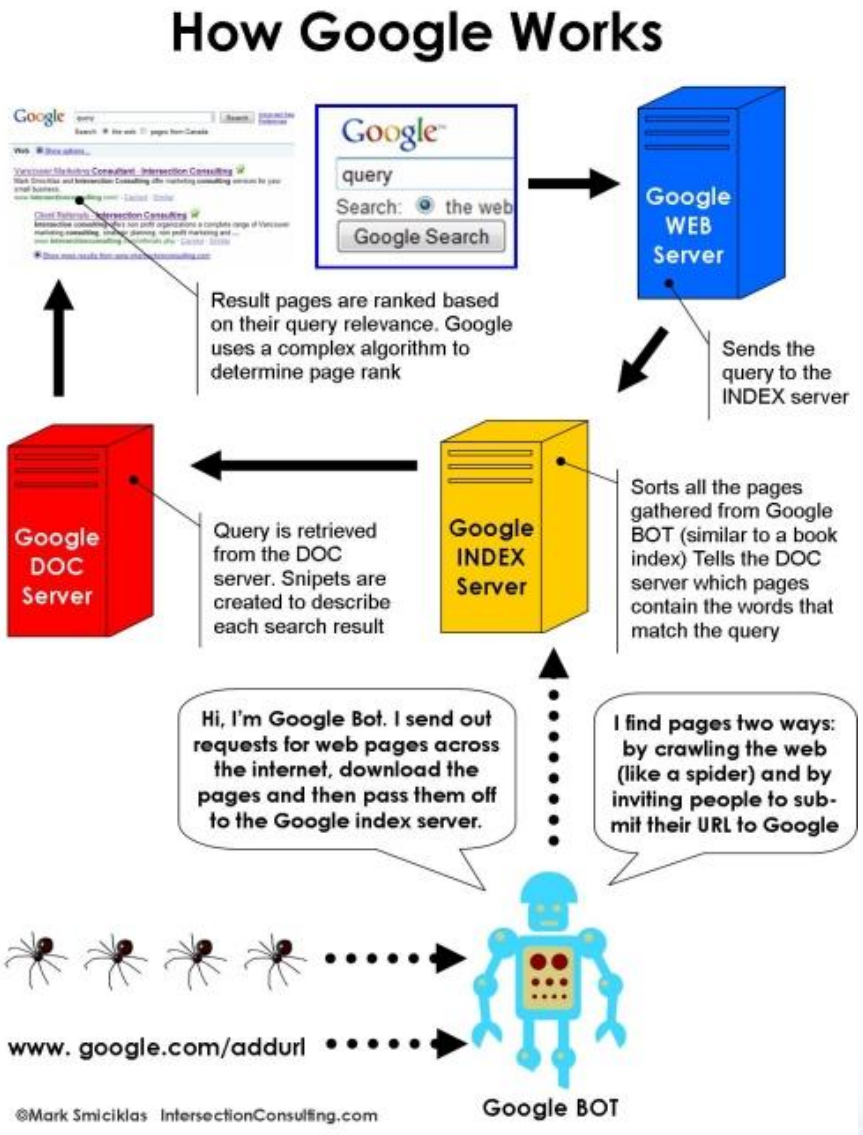
2030年，全球将有
千亿-万亿传感器
数百亿物端设备[4]

[4] http://www-bsac.eecs.berkeley.edu/frontpagefiles/BSACGrowingMEMS_Markets_%20SEMI.ORG.html

分布式系统的例子

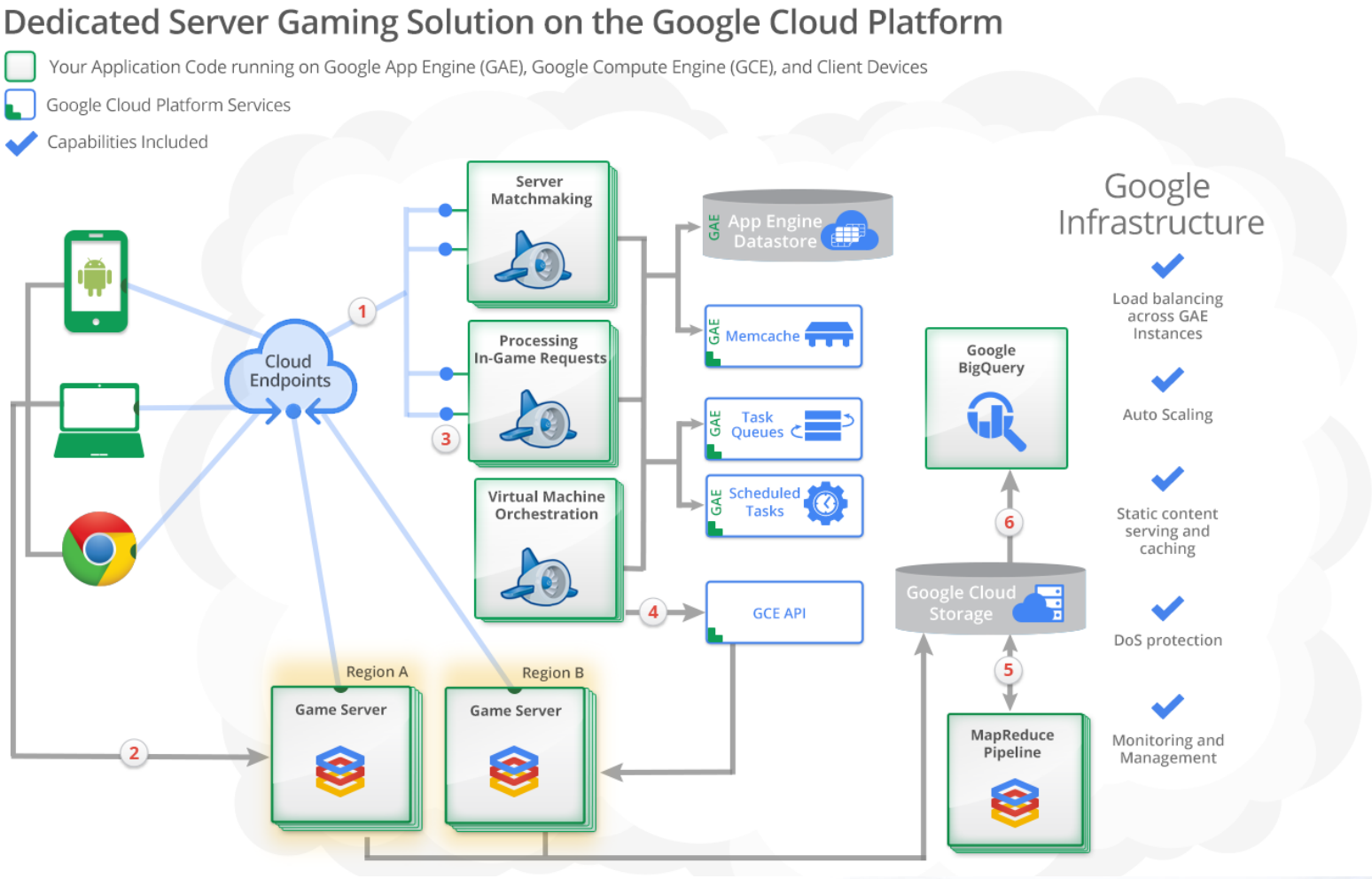
Web搜索

http://www.google.com/insidesearch/howsearchworks/thestory/index.html



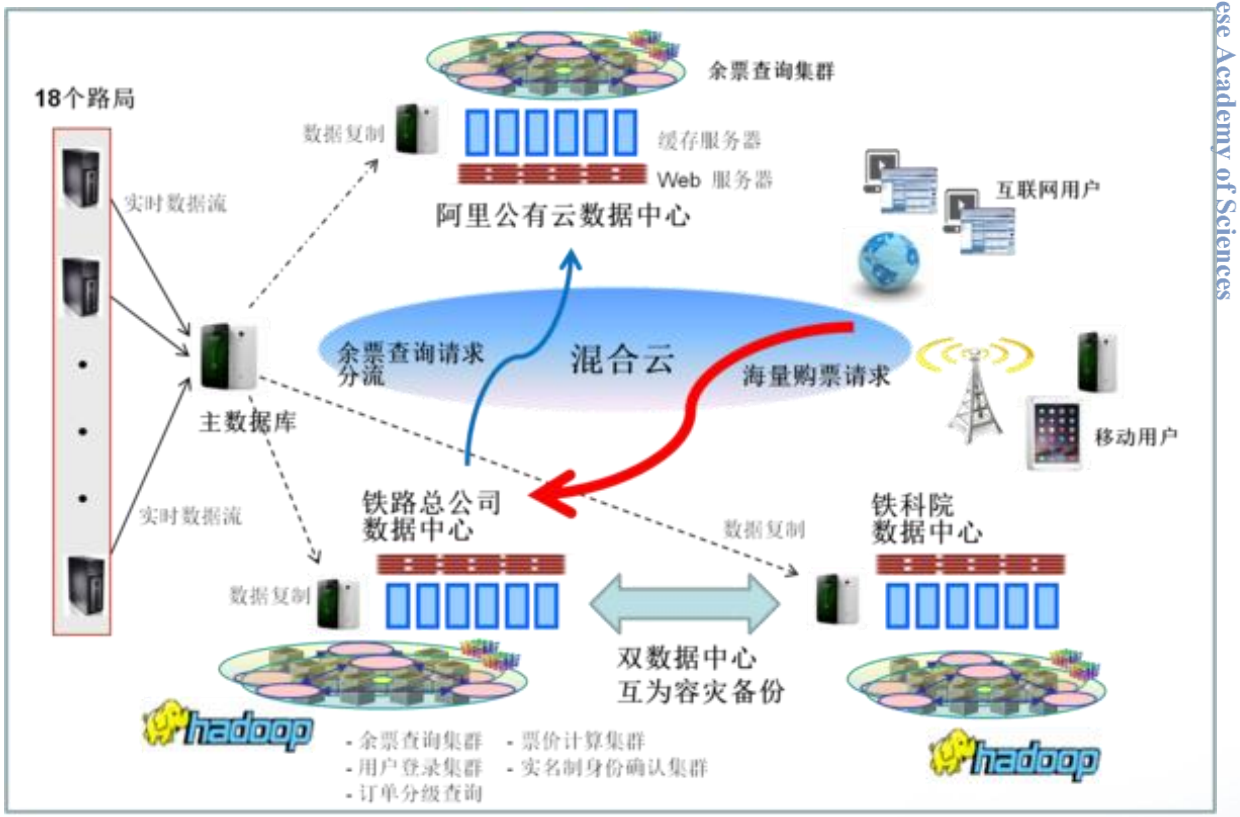
分布式系统的例子 Cont.

大型多人在线游戏

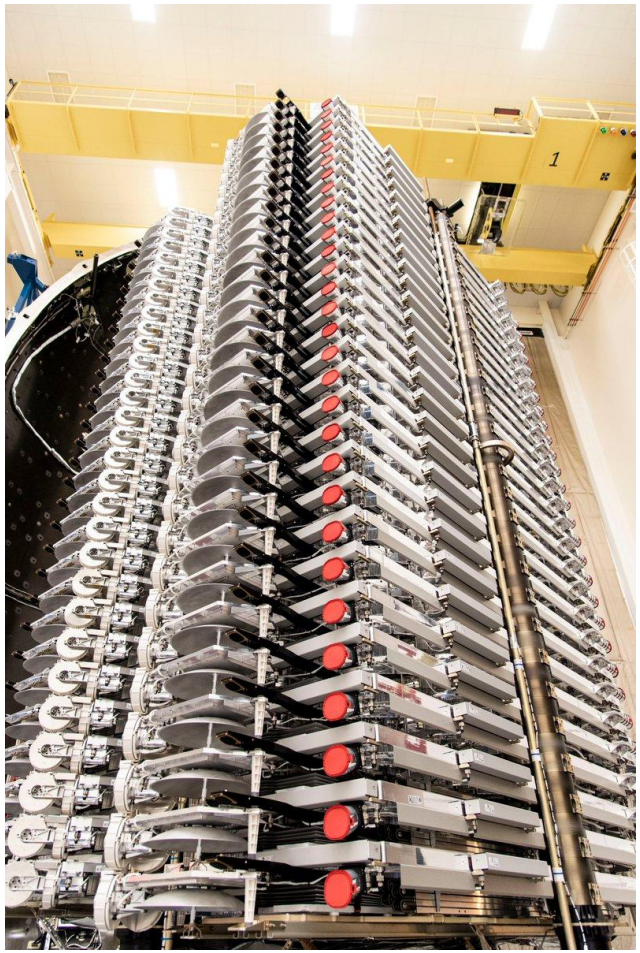


分布式系统的例子 Cont.

在线售票系统



分布式系统的例子 Cont.



分布式系统模型——体系结构模型

Q1：分布式系统中有哪些实体在通信?

Q2：这些实体如何通信?

Q3：这些实体在系统中的角色和责任是什么?

Q4：这些实体在物理环境中如何部署?

分布式系统模型——体系结构模型

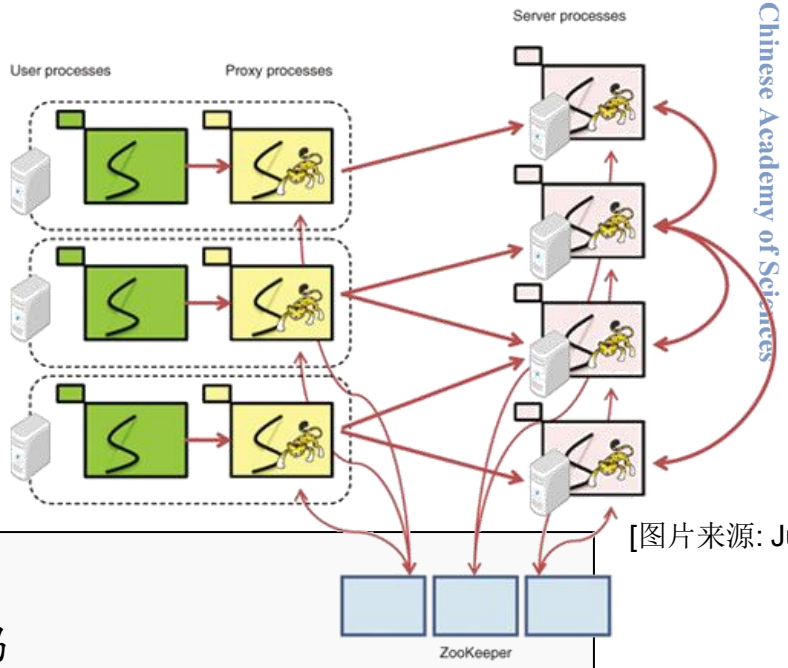
Q1：分布式系统中有哪些实体在通信？

- **从操作系统的角度来看**

  # 实体 = {进程，线程}

- **从应用程序的角度来看**

  # 实体 = {对象，组件，服务}



[图片来源: Jubatus]

分布式对象

对分布式应用程序问题域的最基本解耦

分布式组件

组件是一组对象，本身是内聚的，对外提供某种有意义的功能集

Web服务

利用跨平台技术来实现实体通信

分布式系统模型——体系结构模型

分类	代表技术	特点
分布式对象	Java RMI等	面向对象
分布式组件	EJB, CORBA, DCOM等	组件级复用，提供事务、持久存储、安全等支持
Web服务	SOAP, RESTful等	跨平台访问

参考资料

- Java RMI(Remote Method Invocation)，Java远程对象访问，一种分布式Java对象模型[5]
- EJB(Enterprise JavaBeans)，Java EE提供的服务器端组件模型[6]
- CORBA(Common Object Request Broker Architecture)，OMG组织制订的面向对象应用程序体系规范[7]
- DCOM(Distributed Component Object Model)，微软提供的分布式组件对象模型[8]
- SOAP(Simple Object Access Protocol)，一种基于XML的对象访问协议，是早期Web服务的基础[9]
- RESTful(Representational state transfer)，一种轻量级的Web服务交互风格[10]

[5] Wollrath, Ann, R. Riggs, and J. Waldo. A distributed object model for the java TM, system. Conference on USENIX Conference on Object-Oriented Technologies USENIX Association, 1996:265-290.
[6] http://www.oracle.com/technetwork/java/javaee/ejb/index.html
[7] http://www.corba.org
[8] https://en.wikipedia.org/wiki/Distributed_Component_Object_Model
[9] https://en.wikipedia.org/wiki/SOAP
[10] https://en.wikipedia.org/wiki/Representational_state_transfer

分布式系统模型——体系结构模型

Q2：这些实体如何通信?

- **进程间通信(Inter-process Communication)**

  # 相对底层的通信支撑机制

  # 进程间通信，通常是一个进程的一个Socket与另一个进程的一个Socket之间通过通信协议（UDP、TCP）来传递消息

  # 应用层的数据（对象、或类型值）需要转换（编码，*Marshalling*）为字节序列，然后进行传递，并在另一端进行恢复（解码，*Unmarshalling*）

Java的对象序列化机制（Object Serialization）

将一个对象或一组有关联的对象转换为适合消息传送或存储的字节序列

分布式系统模型——体系结构模型

Q2：这些实体如何通信?

- **远程调用(Remote Invocation)**

  # 分布式系统中最常用的通信机制

  请求-应答协议，例如HTTP协议

  RPC(Remote Procedure Call)，远程过程调用

  RMI(Remote Method Invocation)，分布式对象之间的方法调用

- **间接通信(Indirect Communication)**

  # 消息发送方和接收方，在时间（相互不需要同时存在）或空间（相互不知道是否存在）上解耦组通信

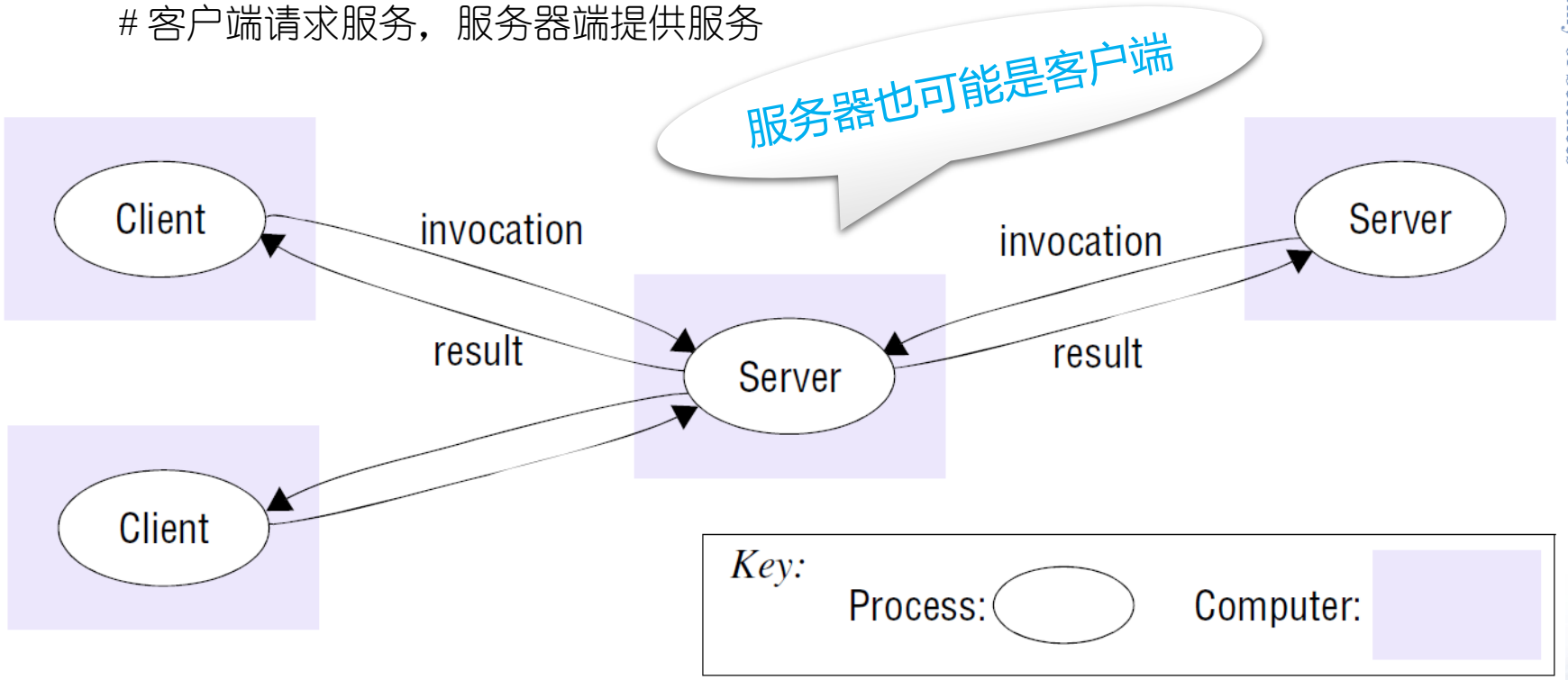
  组通信、发布-订阅、消息队列、分布式共享内存...

分布式系统模型——体系结构模型

Q3：这些实体在系统中的角色和责任是什么？

- **客户端-服务器(Client-Server)**

  # 客户端请求服务，服务器端提供服务



分布式系统模型——体系结构模型

Q3：这些实体在系统中的角色和责任是什么？

- **P2P(Peer to Peer)**

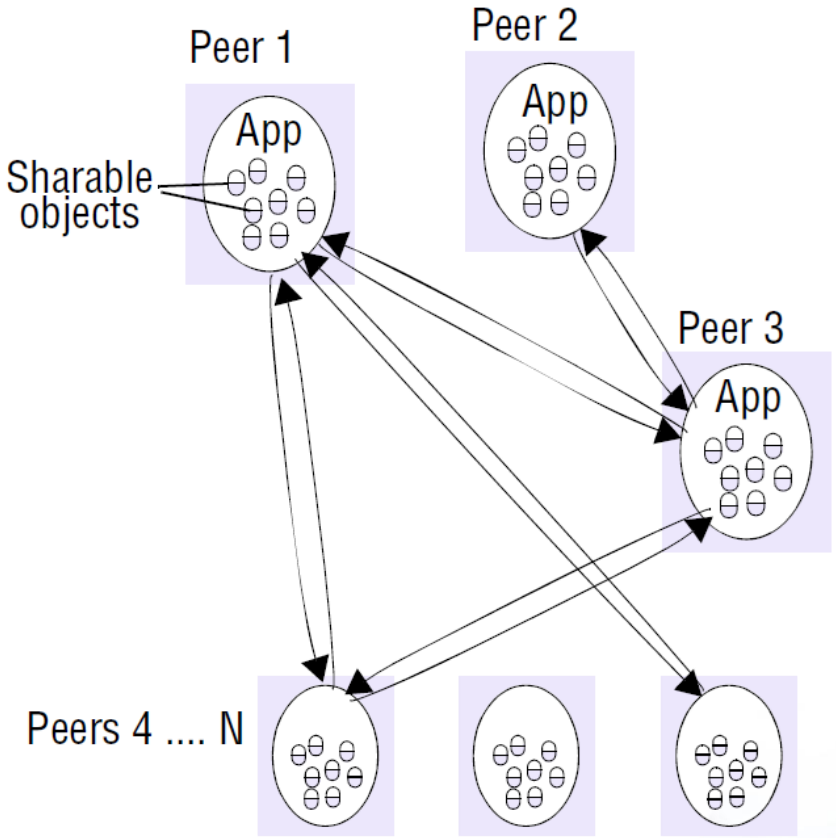
  # 角色对等、职责对等

  # 应用场景广泛，例如

  文件共享

  区块链

  …



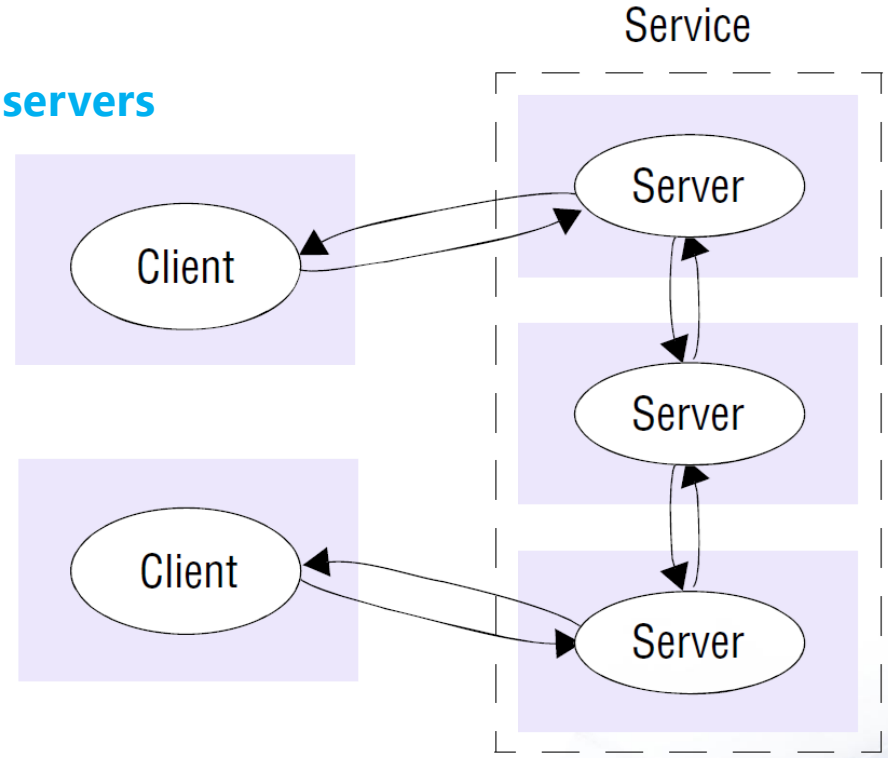
分布式系统模型——体系结构模型

Q4：这些实体在物理环境中如何部署？

- 分布式系统的部署问题是一个复杂问题

Mapping of services to multiple servers

服务器承载了对象的副本或分区



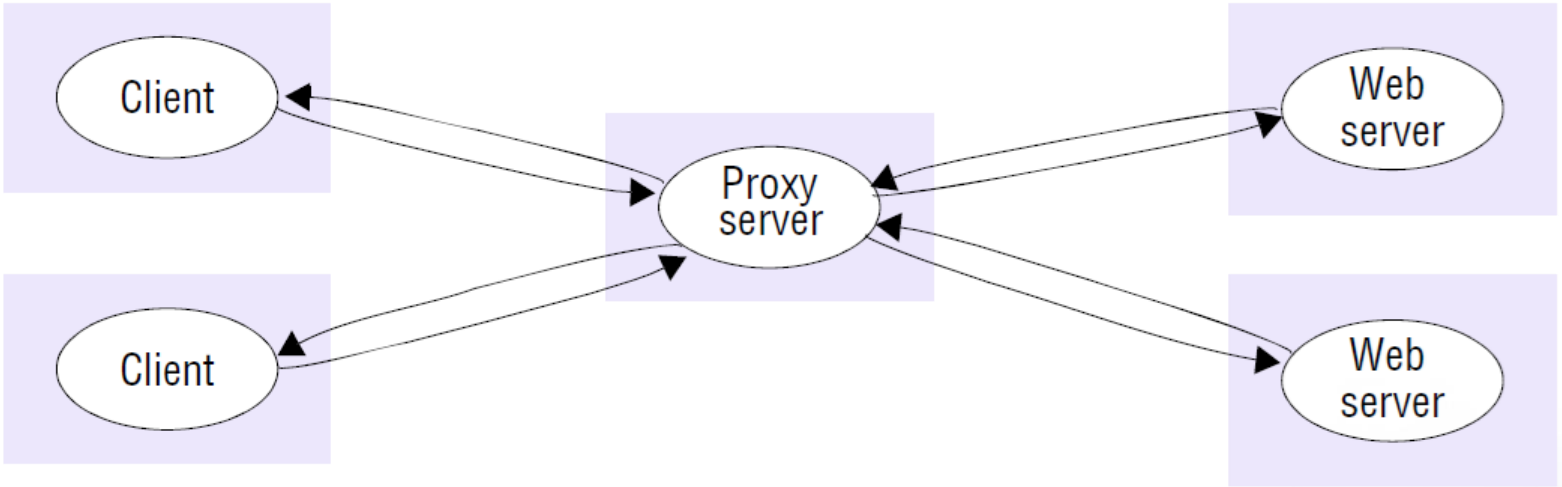
分布式系统模型——体系结构模型

Q4：这些实体在物理环境中如何部署？

- 分布式系统的部署问题是一个复杂问题

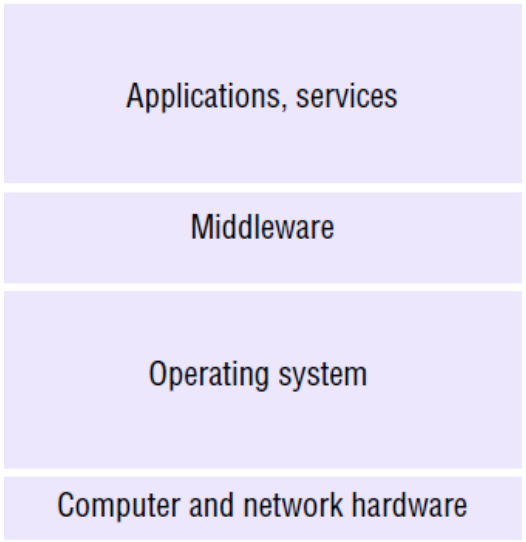
Caching

将数据对象放置在距离客户端更近的地方

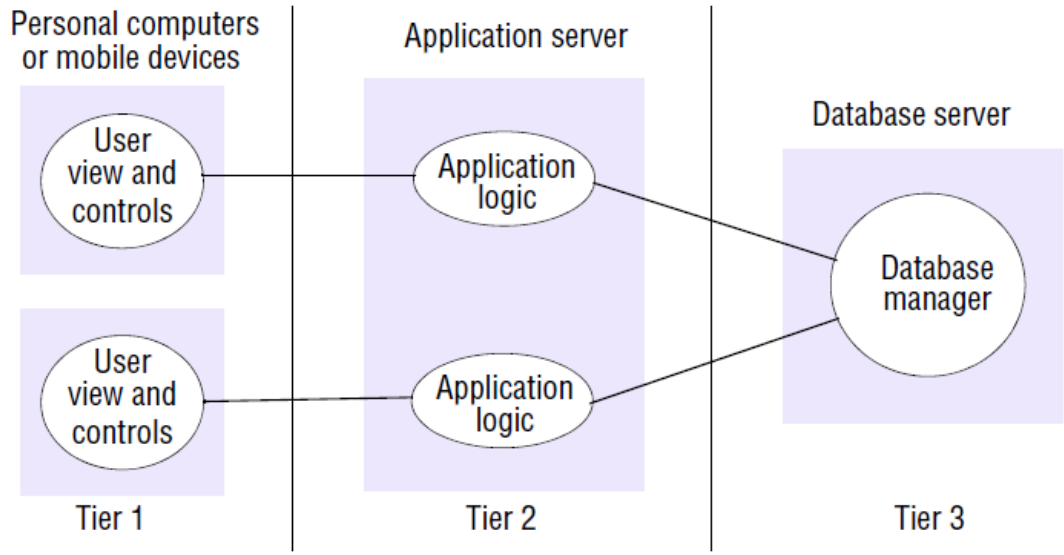


分布式系统模型——架构模式

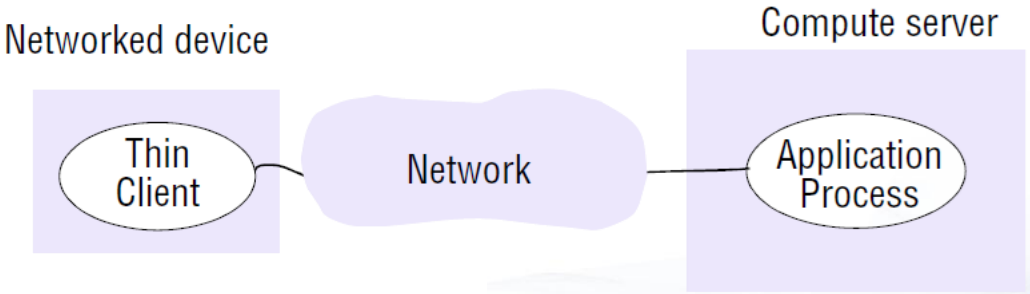
Layering



Tiered architecture

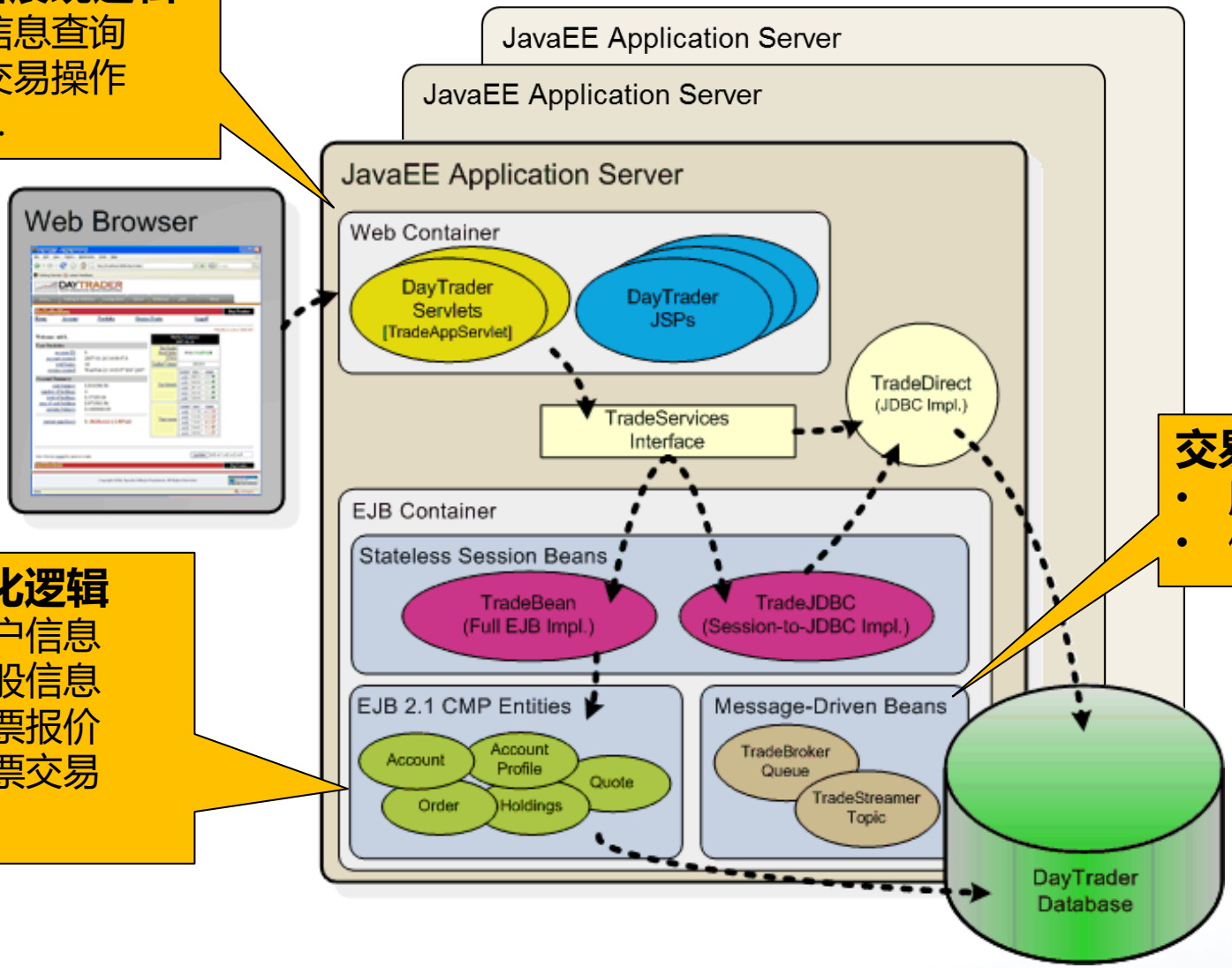


Thin clients



一个简单的证券交易系统示例

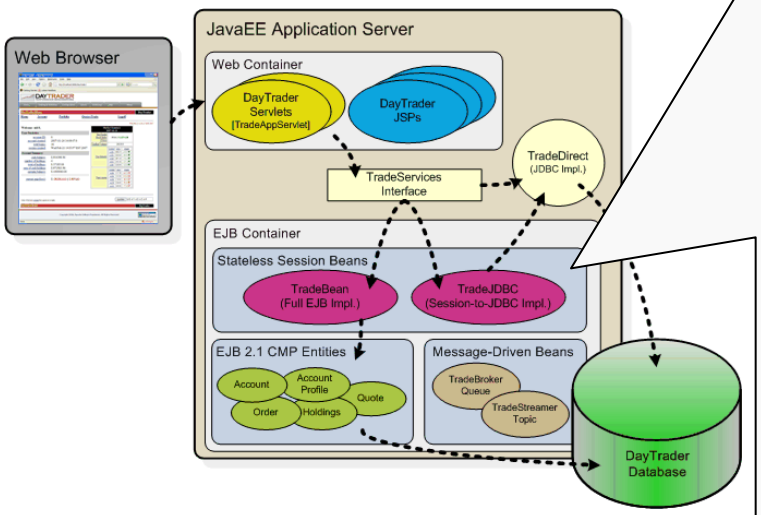
页面展现逻辑
- 信息查询
- 交易操作
- …



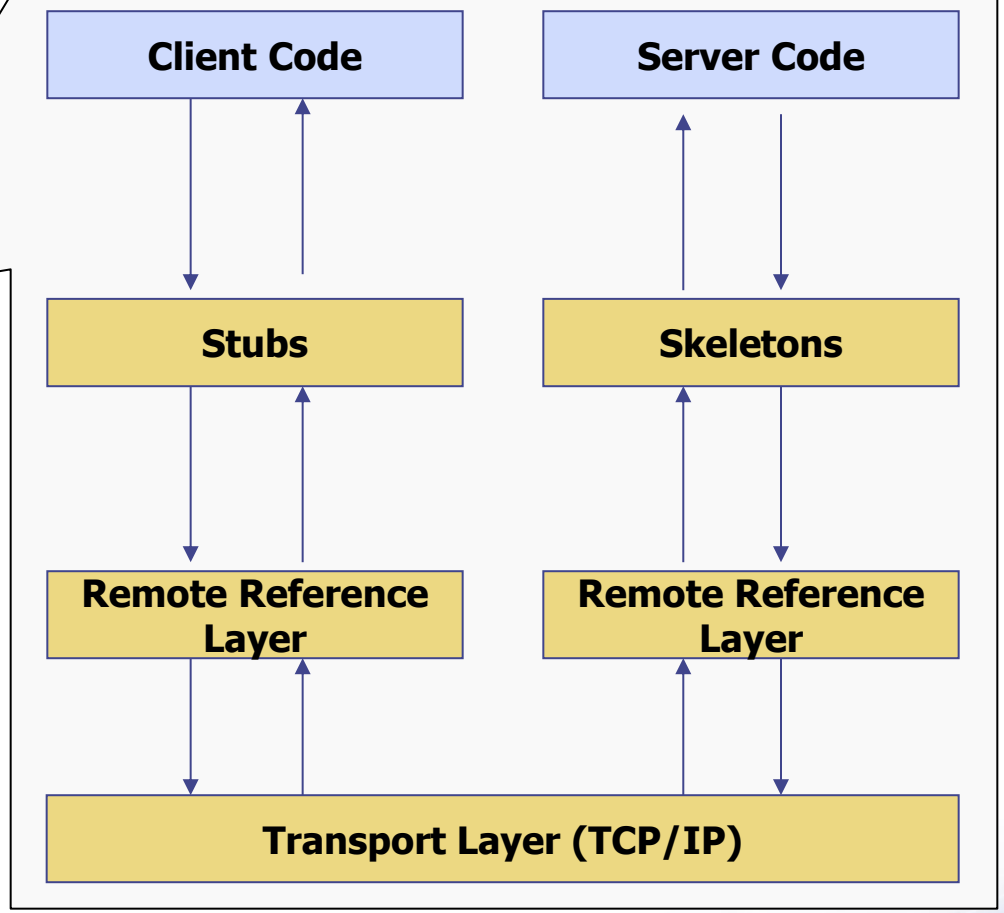
持久化逻辑
- 账户信息
- 持股信息
- 股票报价
- 股票交易
- …

交易消息队列
- 股票交易队列
- 价格变动提醒

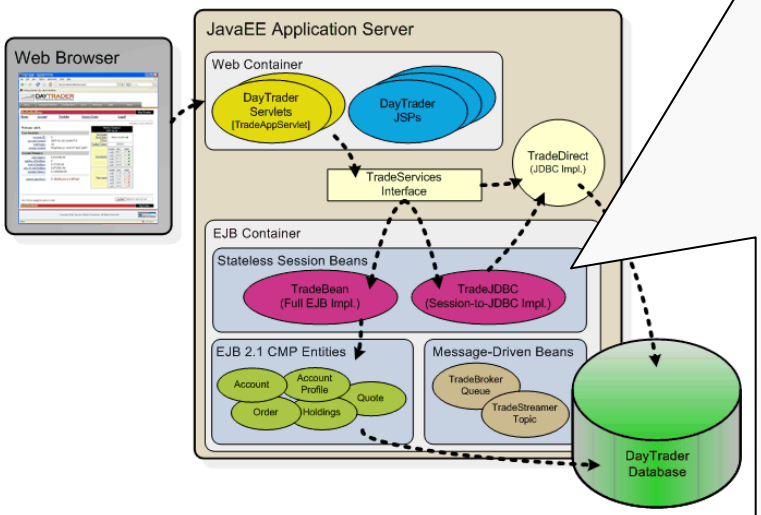
一个简单的证券交易系统示例



服务器端的各类组件、对象之间的交互调用，主要是基于RMI机制

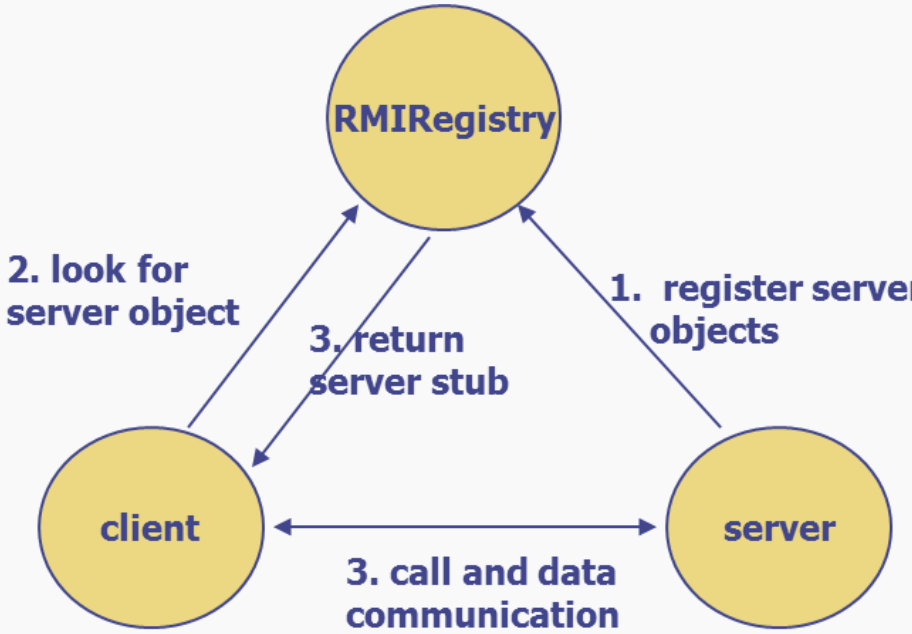


一个简单的证券交易系统示例



示例1

服务器端的各类组件、对象之间的交互调用，主要是基于RMI机制



分布式系统的挑战

- **异构性(Heterogeneity)**
  # 网络、计算机硬件、操作系统、编程语言…
- **开放性(Openness)**
  # 计算机系统的开放性：能否扩充以不同方式的重新实现，例如OS微内核
  # 分布式系统的开放性：能否扩充新的被大量客户端访问的共享资源服务，可能是硬件（节点、集群、云…），也可能是软件（系统软件、应用服务…）
- **安全性(Security)**
  # 机密性（防止未授权的访问）
  # 完整性（防止数据篡改）
  # 可用性（防止对所提供服务的干扰）

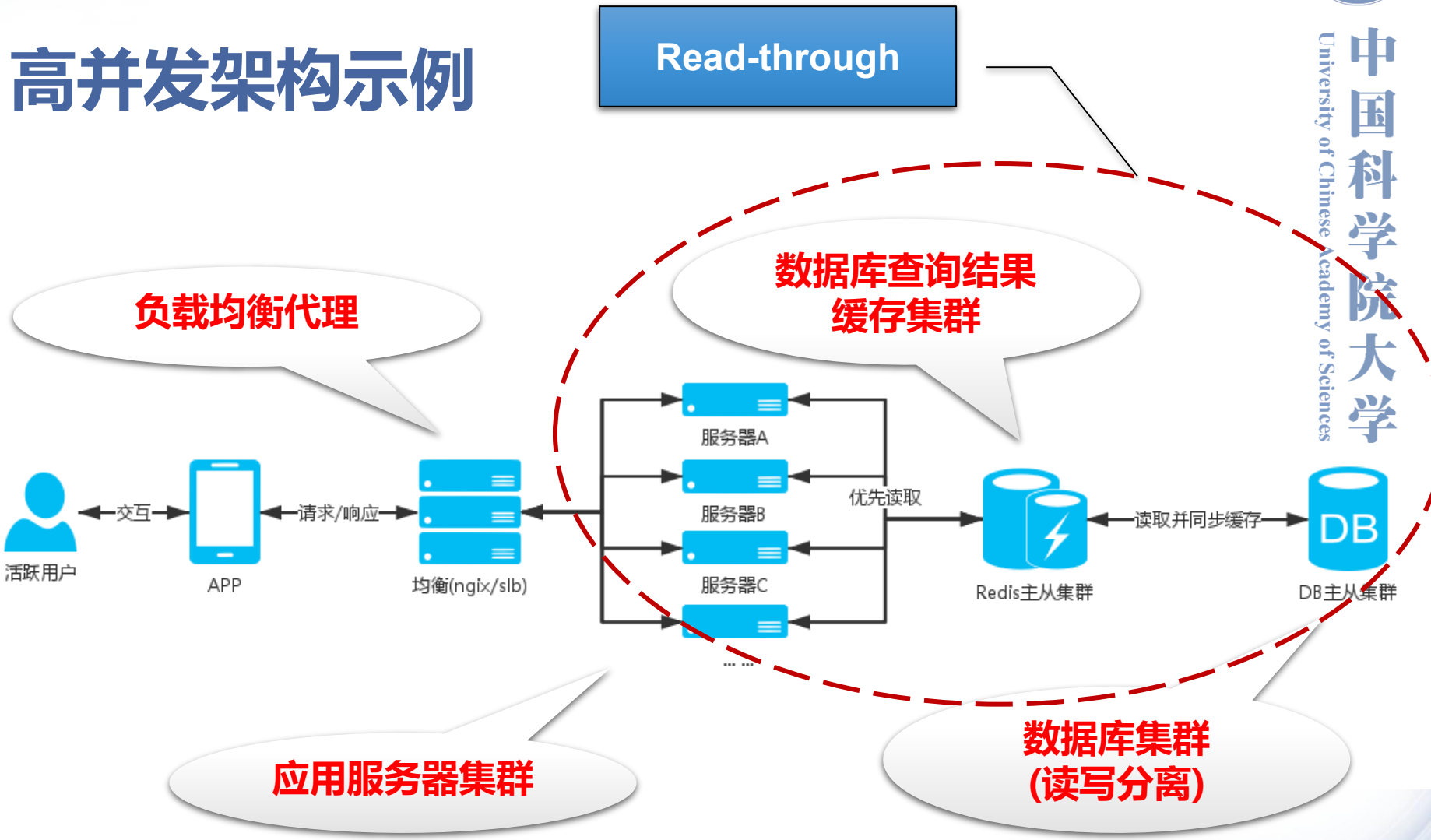
分布式系统的挑战 Cont.

- **可伸缩性(Scalability)**

  # 如果系统用户、资源的规模增加后，系统仍能保持有效，则称该系统是可伸缩的

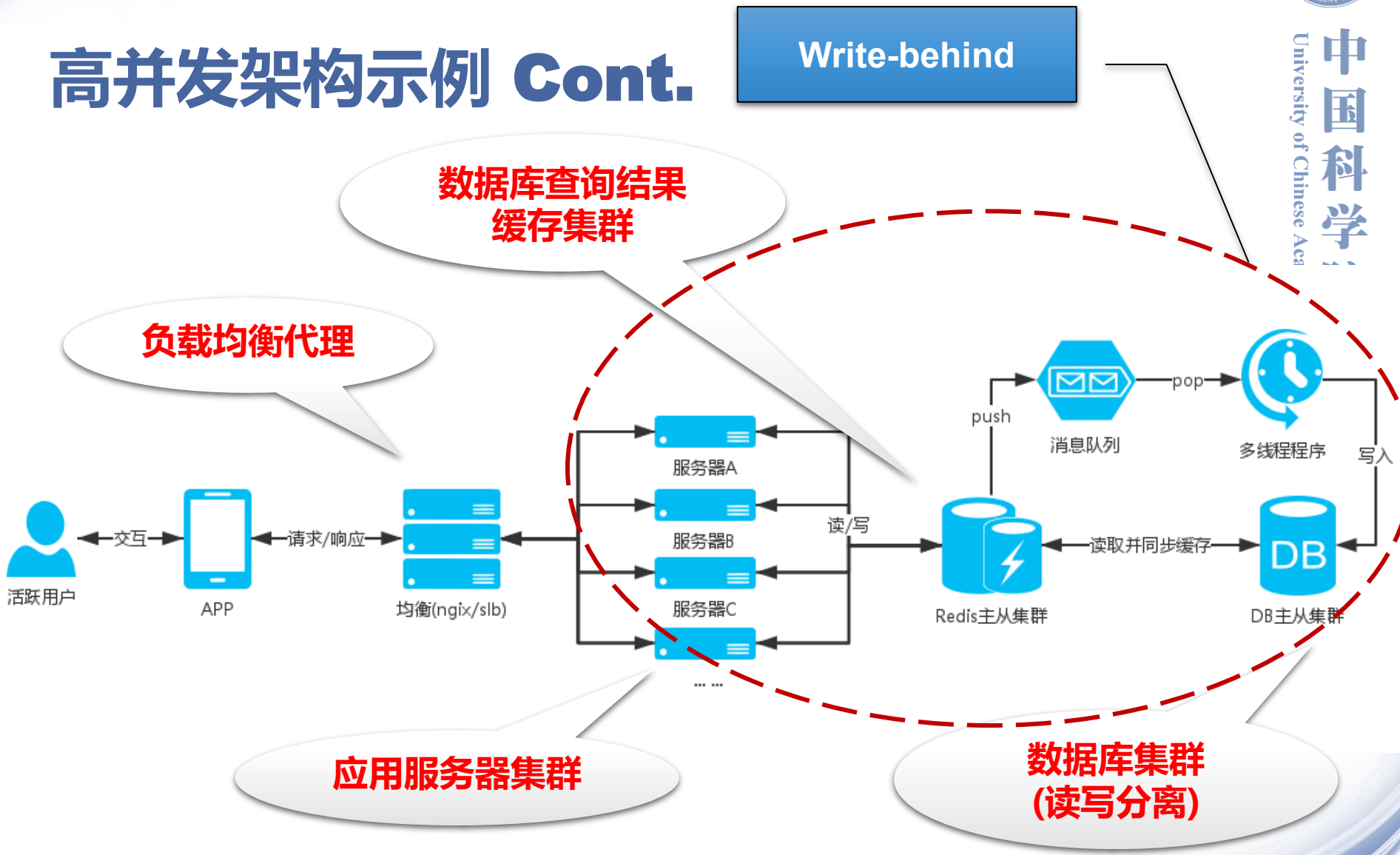
  例如，用户规模或数据增加，系统性能是否不会下降？为了保障系统性能而增加的物理资源是否非线性暴涨？

高并发架构示例



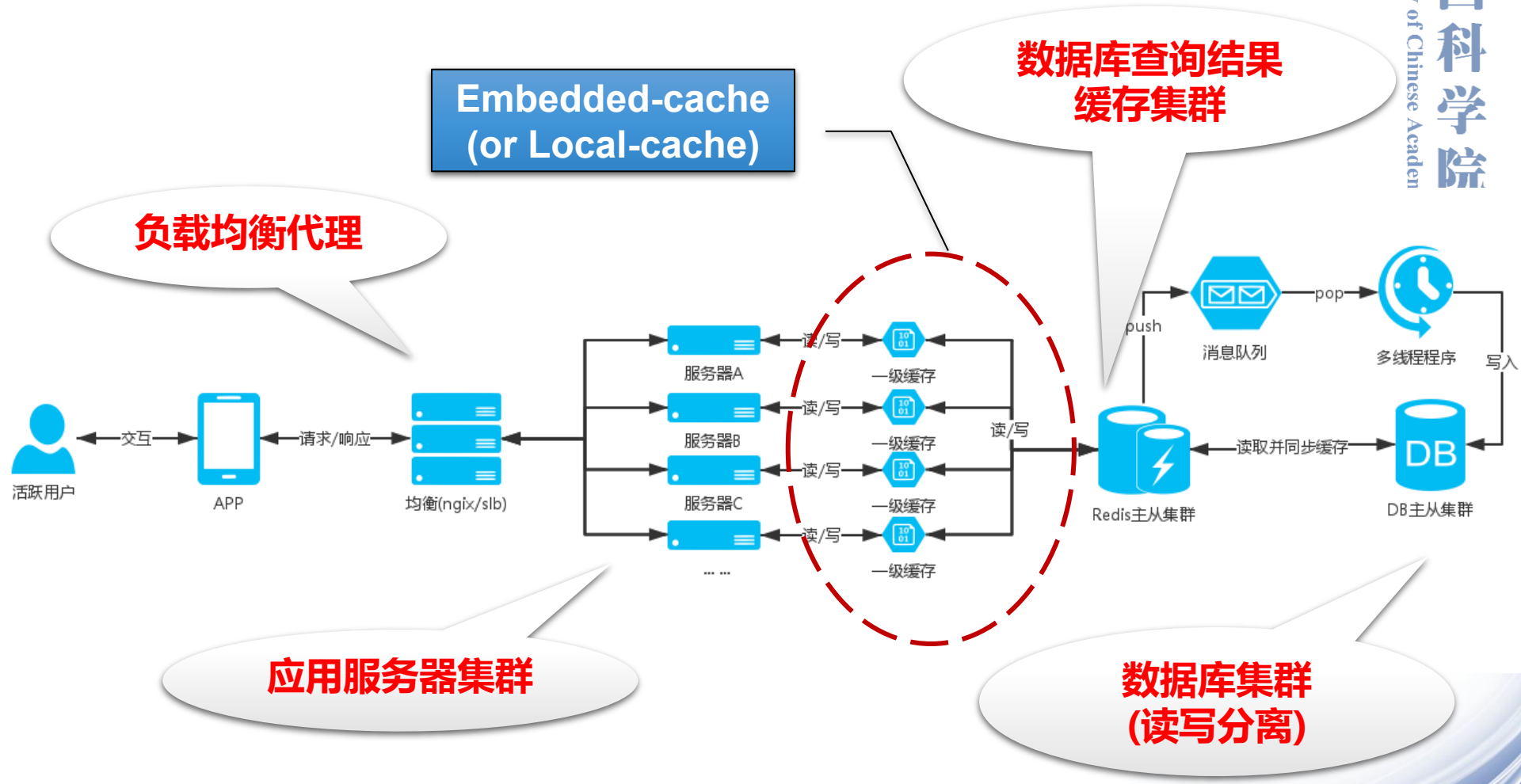
[图片来源：http://www.importnew.com/22546.html]

高并发架构示例 Cont.



[图片来源：http://www.importnew.com/22546.html]

高并发架构示例 Cont.



[图片来源：http://www.importnew.com/22546.html]

分布式系统的挑战 Cont.

- **故障处理(Failure handling)**

  # 分布式系统的故障往往是局部的，故障的处理难度更高

  故障检测：有些故障无法检测，只能推测，例如服务器是否还“活着”？“脑裂”？

  故障容忍：客户端需要被设计成能够容忍远端出现的错误

  故障恢复：恢复后，状态是否一致？数据是否完整？恢复效率能否满足要求？

  …

分布式系统的挑战 Cont.

- **透明性**

  # 对用户、应用开发人员屏蔽分布式系统的组件分布性

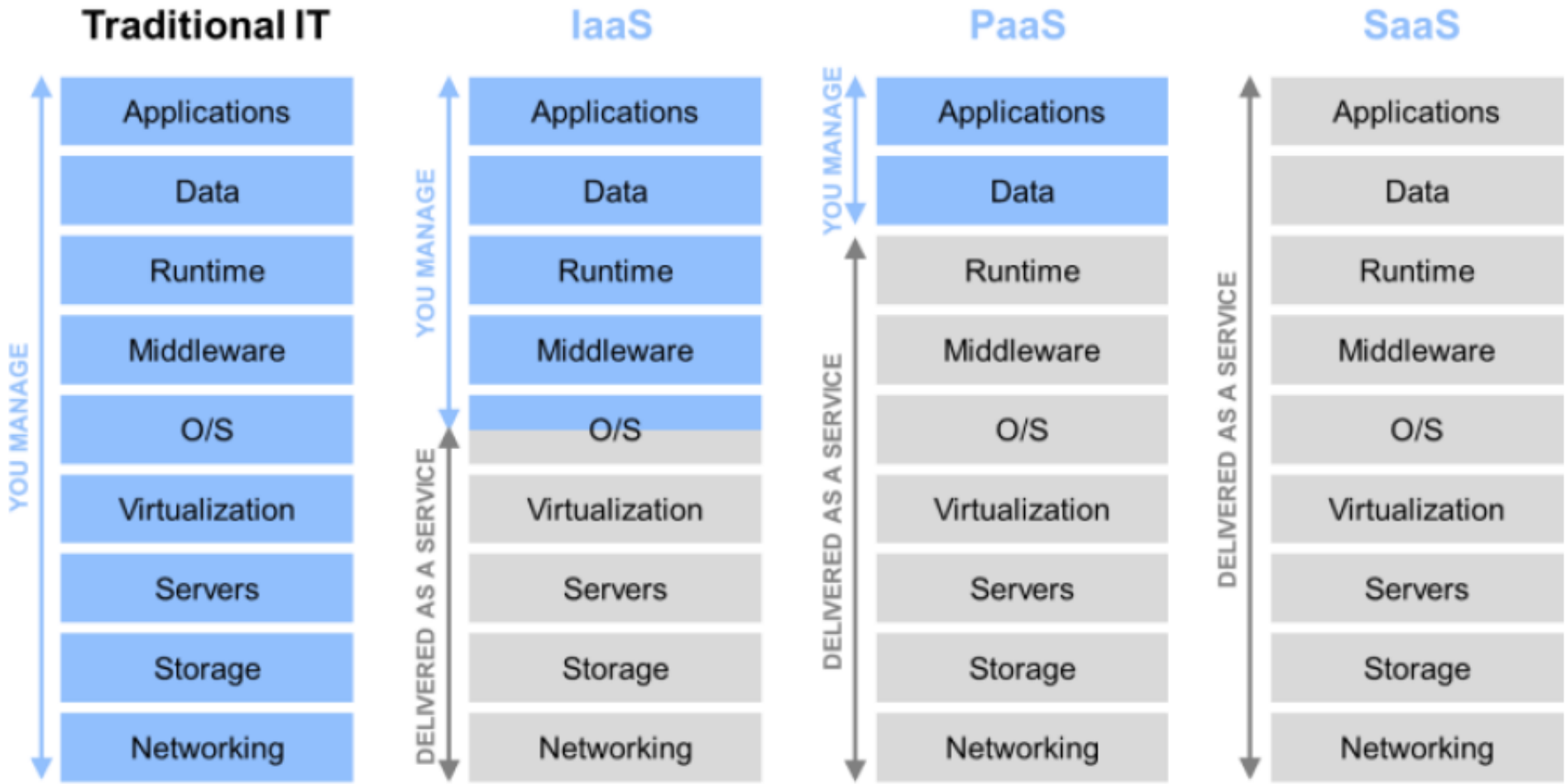
  国际标准化组（ISO）给出的8种透明性[2，3]

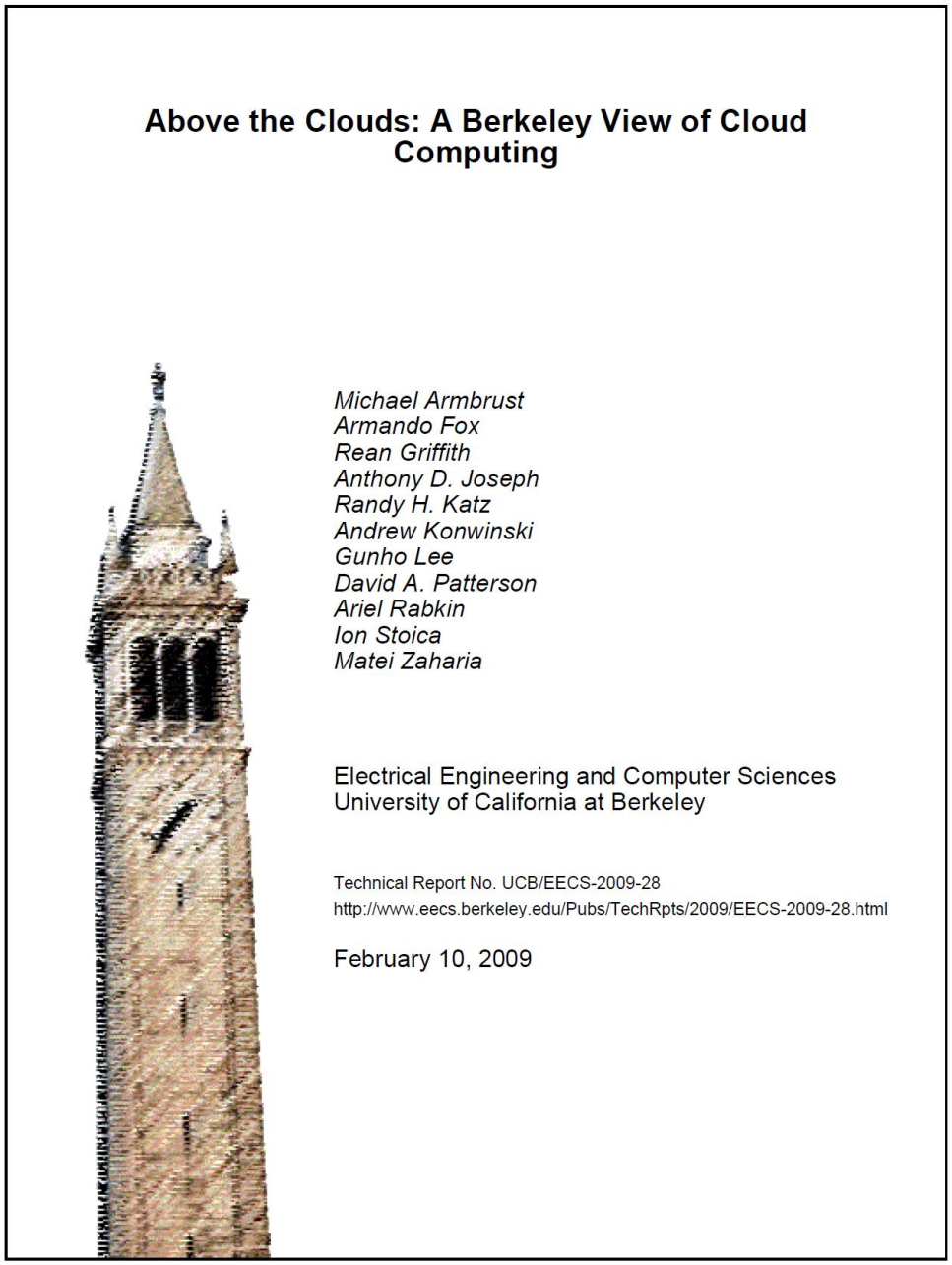
访问透明：	本地操作与远程操作一样
位置透明：	不需要知道远程位置信息就能访问
并发透明：	并发控制对用户和应用开发人员是透明的
复制透明：	增加副本实例来提升性能和可靠性，用户并不感知副本的存在
故障透明：	软硬件故障不影响用户或应用程序完成它们的工作
迁移透明：	允许系统资源、组件的迁移而不影响系统运行和使用
性能透明：	负载发生变化时，允许系统重构以提高性能
伸缩透明：	在不改变应用程序结构和算法的前提下，允许系统扩展

[2] The Advanced Network Systems Architecture (ANSA) Reference Manual. Castle Hill, Cambridge, England: Architecture Project Management.
[3] Basic Reference Model of Open Distributed Processing, Part 1: Overview and Guide to Use. ISO/IEC JTC1/SC212/WG7 CD 10746-1, International Organization for Standardization.

云计算提供了新的资源交付模式
IaaS/PaaS/SaaS

“烟囱式”专有软硬件设施 → 多租户共享软硬件设施





Above the Clouds: A Berkeley View of Cloud Computing

Michael Armbrust
Armando Fox
Rean Griffith
Anthony D. Joseph
Randy H. Katz
Andrew Konwinski
Gunho Lee
David A. Patterson
Ariel Rabkin
Ion Stoica
Matei Zaharia

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-28
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

February 10, 2009

https://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

中国科学院大学
University of Chinese Academy of Sciences

中国科学院大学
University of Chinese Academy of Sciences

Thank You!

- 面向对象程序设计
- Object-Oriented Programming

编程实训（选做，不计入期末成绩）

www.educoder.net

面向对象程序设计
2025秋季班级

邀请码：W28OA