分布式系统

原仓周

课程微信群

Definition of a Distributed System

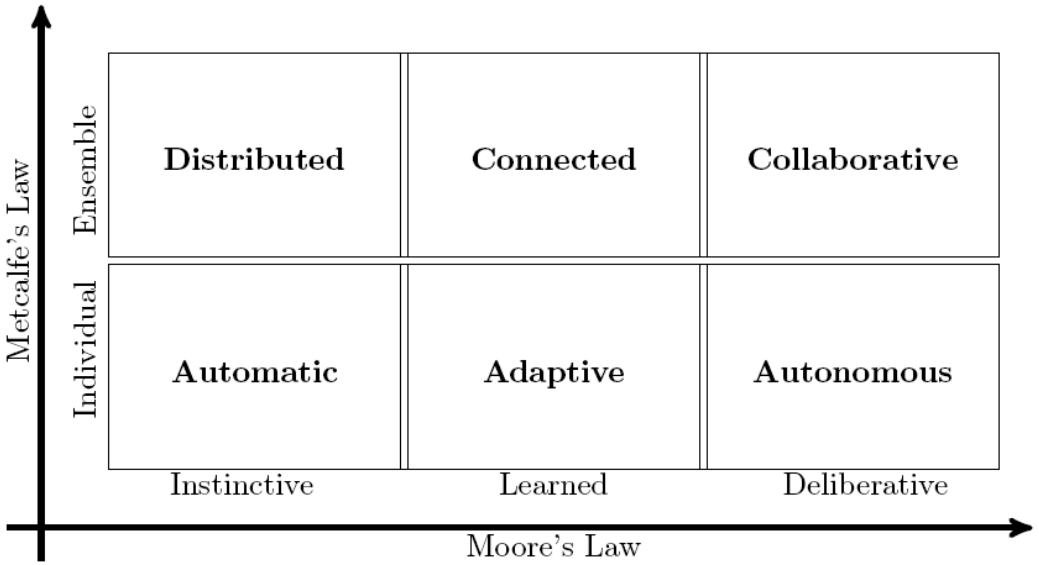
A distributed system is:

A collection of independent computers that appear to its users as a single coherent system (Tanenbaum book)

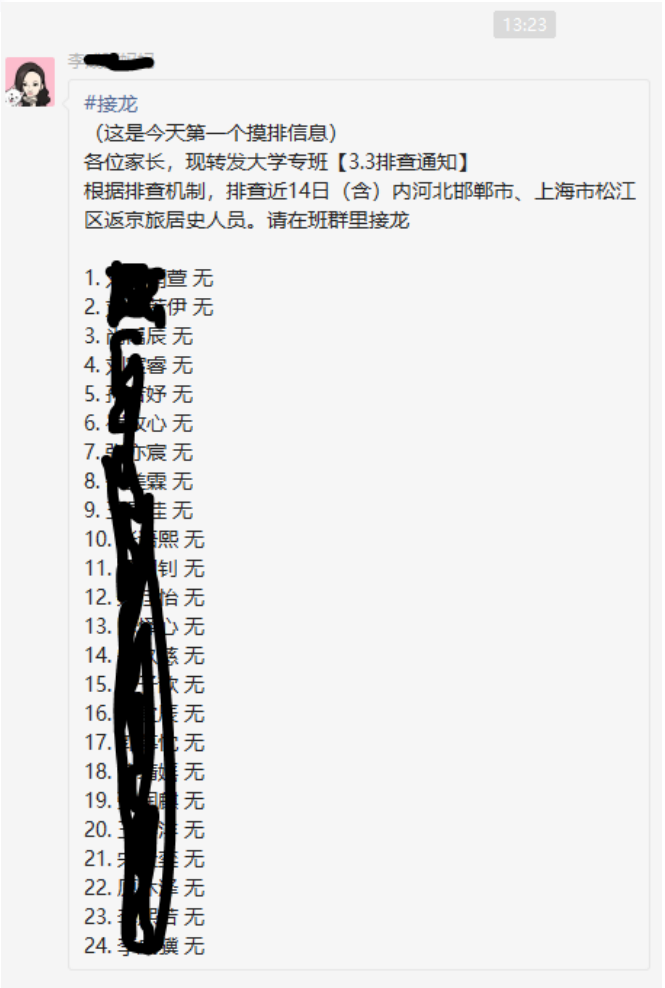
One in which components located at networked computers communicate and coordinate their actions only by passing messages (Coulouris book)

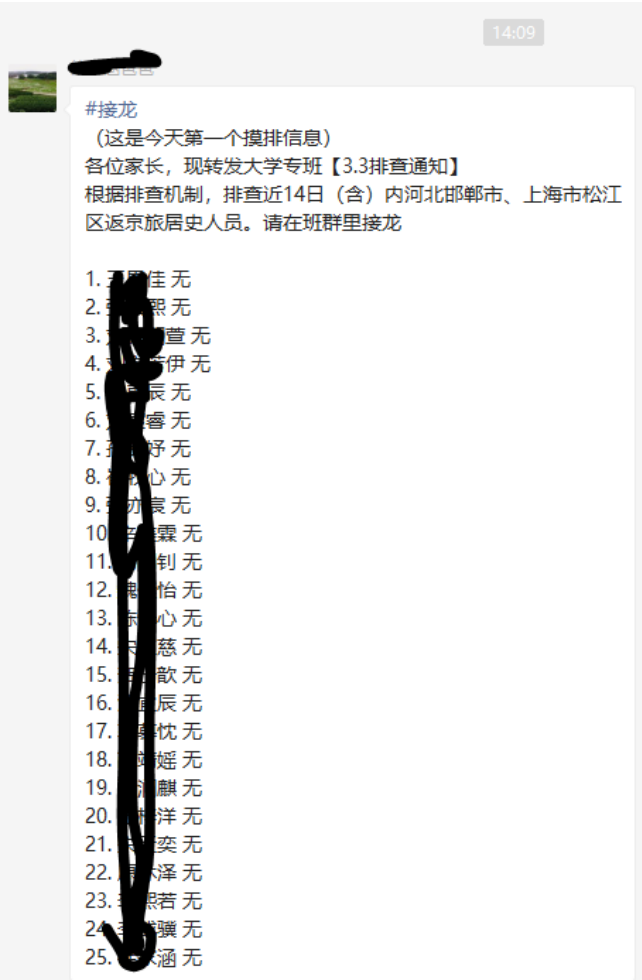
分布式系统 能力 分类

- 人的无意识反应分三层
  - Instinctive Reactions: Automatic and Distributed
  - Learned Reactions: Adaptive and Connected Systems
  - Deliberative Thinking: Autonomous and Collaborative Systems.



同一个微信接龙---不一样的顺序

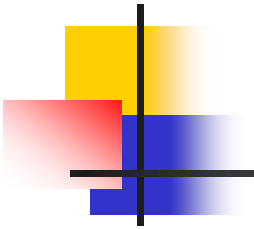




教材与参考书

- Distributed Systems – Concepts and Design
  分布式系统 概念与设计
  - 中文版，原书第5版
  - 机械工业出版社
  - 金蓓弘 曹冬磊 等译

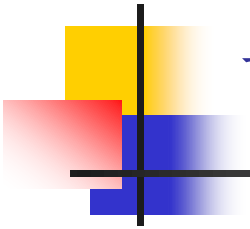
- Distributed systems principles and paradigms
  - 中文版，原书第2版
  - 清华大学出版社
  - 辛春生 陈宗斌 等译



Text Books

1. George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair, *Distributed Systems: Concepts and Design*, 5th E, Addison Wesley, 2011

2. Maarten Van Steen and Andrew S. Tannenbaum, *Distributed Systems*, 3nd E, Pearson, 2017.



课程内容

01-概述.ppt

02-系统架构.ppt

03-进程间通信.ppt

04-分布式对象和远程调用.ppt

05-命名系统.ppt

06-时间和全局状态.ppt

07-协调和协定.ppt

08-事务和并发控制.ppt

09-复制.ppt

11-分布式文件系统.ppt

Projects

Assignments

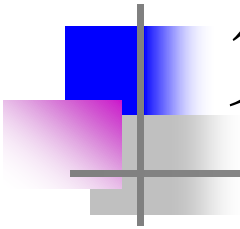
- 课后思考题
- Reading Assignments

Projects

- 设计实现一个基于网络时间协议（NTP）服务的客户端程序
- 设计实现一个基于事件的通知服务。
- 设计实现一个基于raft的选举服务。 （类似其他算法也可以）

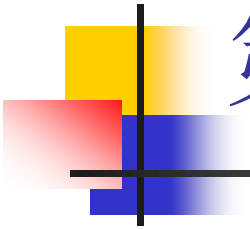
考核方法

Type	Weight
实验项目	50%
思考题和讨论	20%
课程总结	30%



第1章 概述

- 分布式系统的定义
- 分布式系统举例
- 为什么用分布式系统
- 分布式系统需要实现的需求
- 实现分布式系统的挑战
- 总结



第1章 概述

- 分布式系统的定义
- 分布式系统举例
- 为什么用分布式系统
- 分布式系统需要实现的需求
- 实现分布式系统的挑战
- 总结

Definition of a Distributed System

A distributed system is:

A collection of independent computers that appear to its users as a single coherent system (Tanenbaum book)

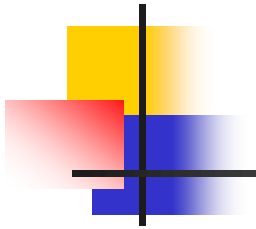
One in which components located at networked computers communicate and coordinate their actions only by passing messages (Coulouris book)

What Is A Distributed System?

"A collection of independent computers that appears to its users as a single coherent system."

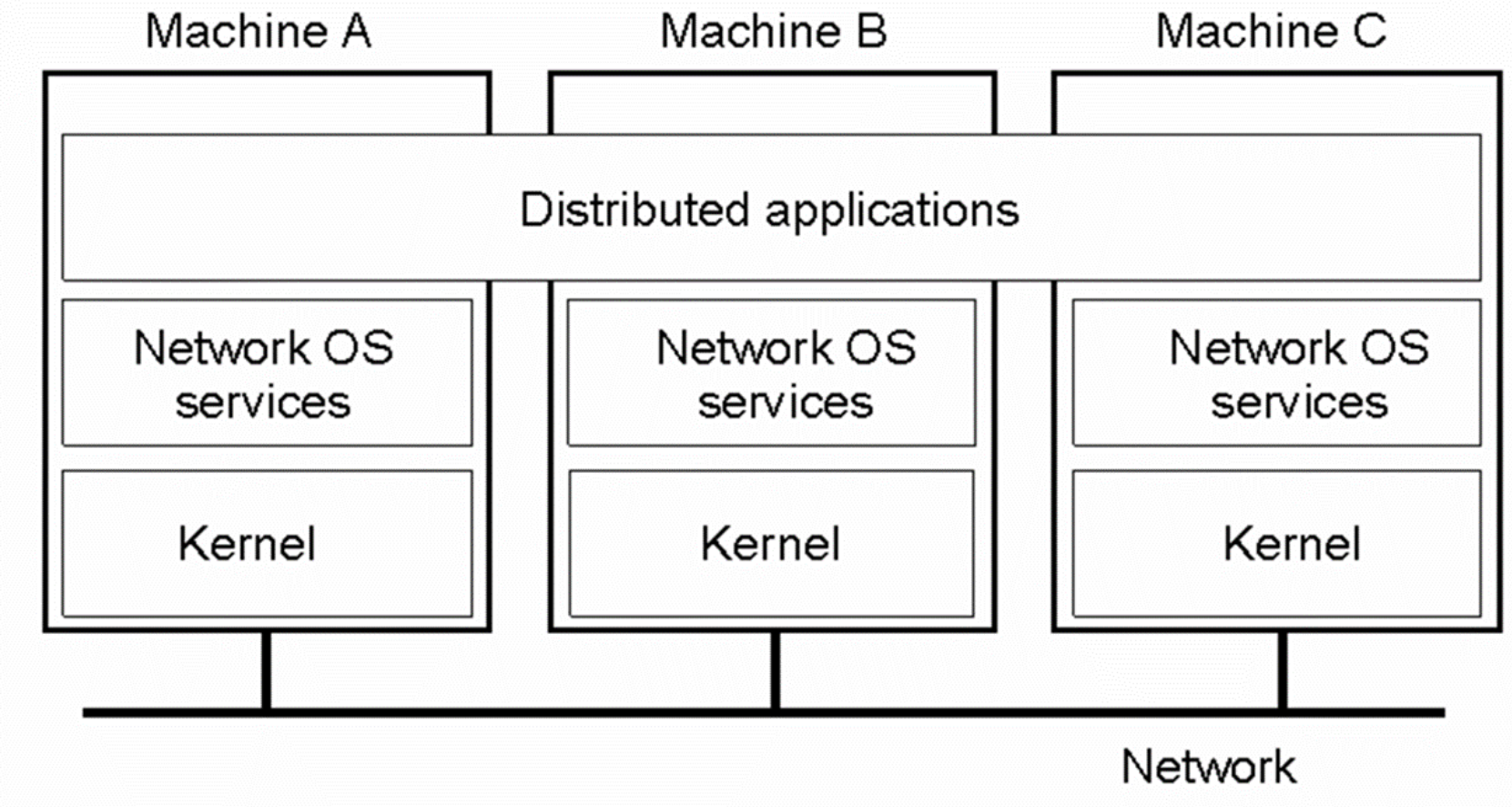
- Features:
  - No shared memory – message-based communication
  - Each runs its own local OS
  - Heterogeneity
- Ideal: to present a single-system image:
  - The distributed system "looks like" a single computer rather than a collection of separate computers.

Characteristics of a DS

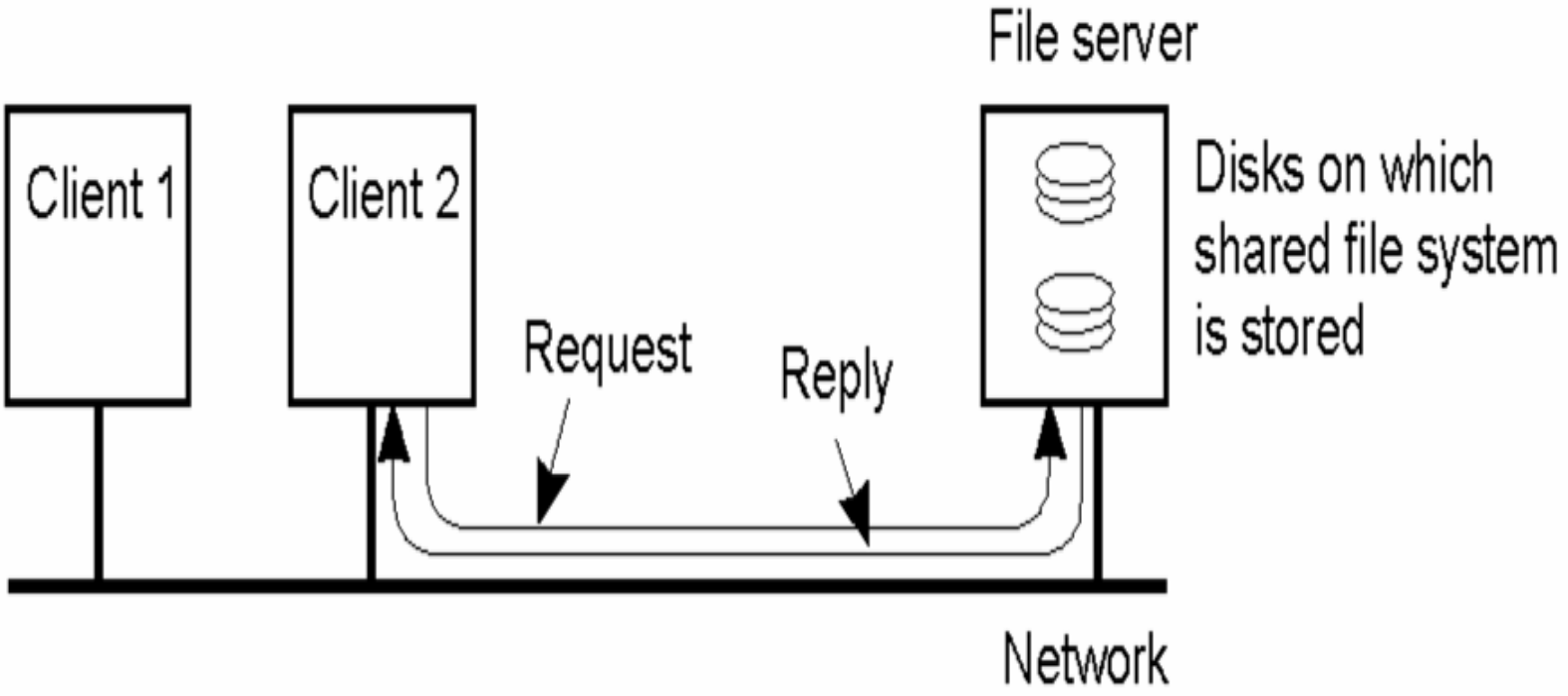


- Present a single-system image
  - Hide internal organization, communication details
  - Provide uniform interface
- Easily expandable
  - Adding new servers is hidden from users
- Continuous availability
  - Failures in one component can be covered by other components
- Supported by middleware

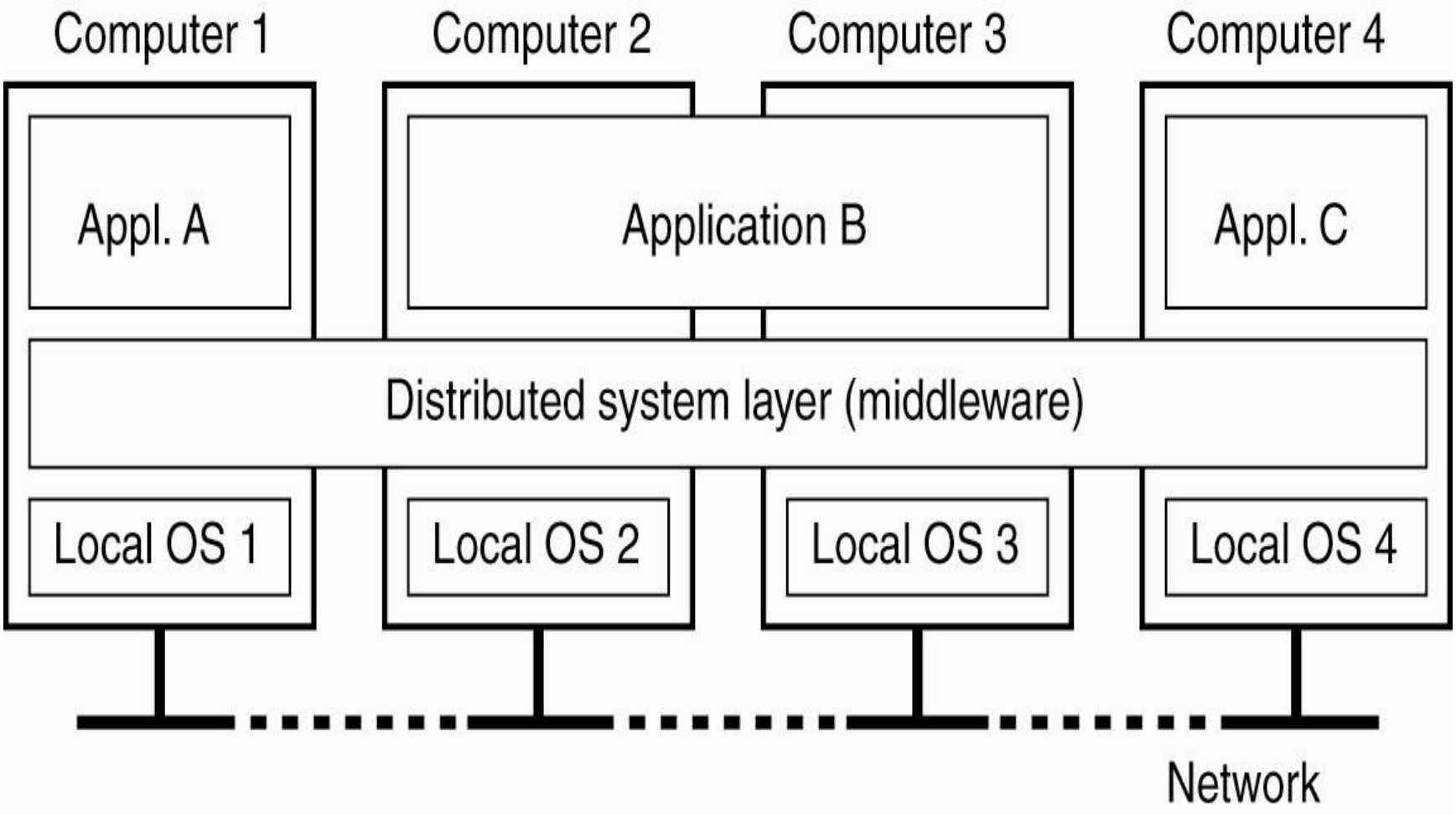
网络操作系统（NOS）


Machine A
Machine B
Machine C
Distributed applications
Network OS services
Network OS services
Network OS services
Kernel
Kernel
Kernel
Network

客户/服务器结构

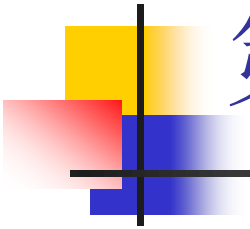

File server
Client 1
Client 2
Request
Reply
Disks on which shared file system is stored
Network

分布式系统通常表现为中间件形式



分布式系统的功能

- 单一的、全局的进程间通信机制
  - 协议、位置
- 全局进程管理
  - 创建、启动、挂起、撤销
- 全局文件系统
  - 文件名、目录、操作
- 统一的系统调用接口



第1章 概述

- 分布式系统的定义
- 分布式系统举例
- 为什么用分布式系统
- 分布式系统需要实现的需求
- 实现分布式系统的挑战
- 总结

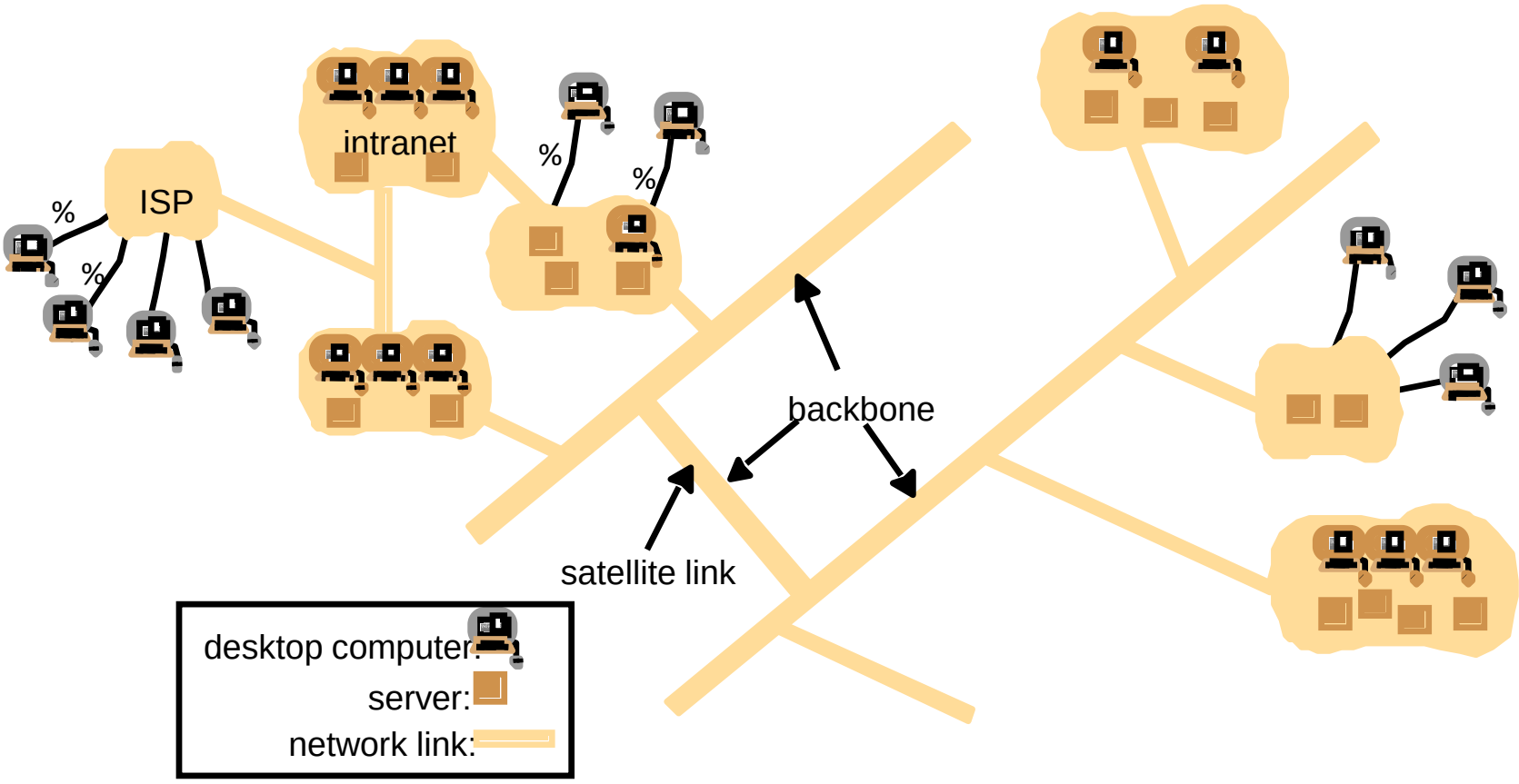
分布式系统举例

Application Domain	Associated Networked Application
Finance and commerce	eCommerce e.g. Amazon and eBay, PayPal, online banking and trading
The information society	Web information and search engines, ebooks, Wikipedia; social networking: Facebook and MySpace.
Creative industries and entertainment	online gaming, music and film in the home, user-generated content, e.g. YouTube, Flickr
Healthcare	health informatics, on online patient records, monitoring patients
Education	e-learning, virtual learning environments; distance learning
Transport and logistics	GPS in route finding systems, map services: Google Maps, Google Earth
Science	The Grid as an enabling technology for collaboration between scientists
Environmental management	sensor technology to monitor earthquakes, floods or tsunamis

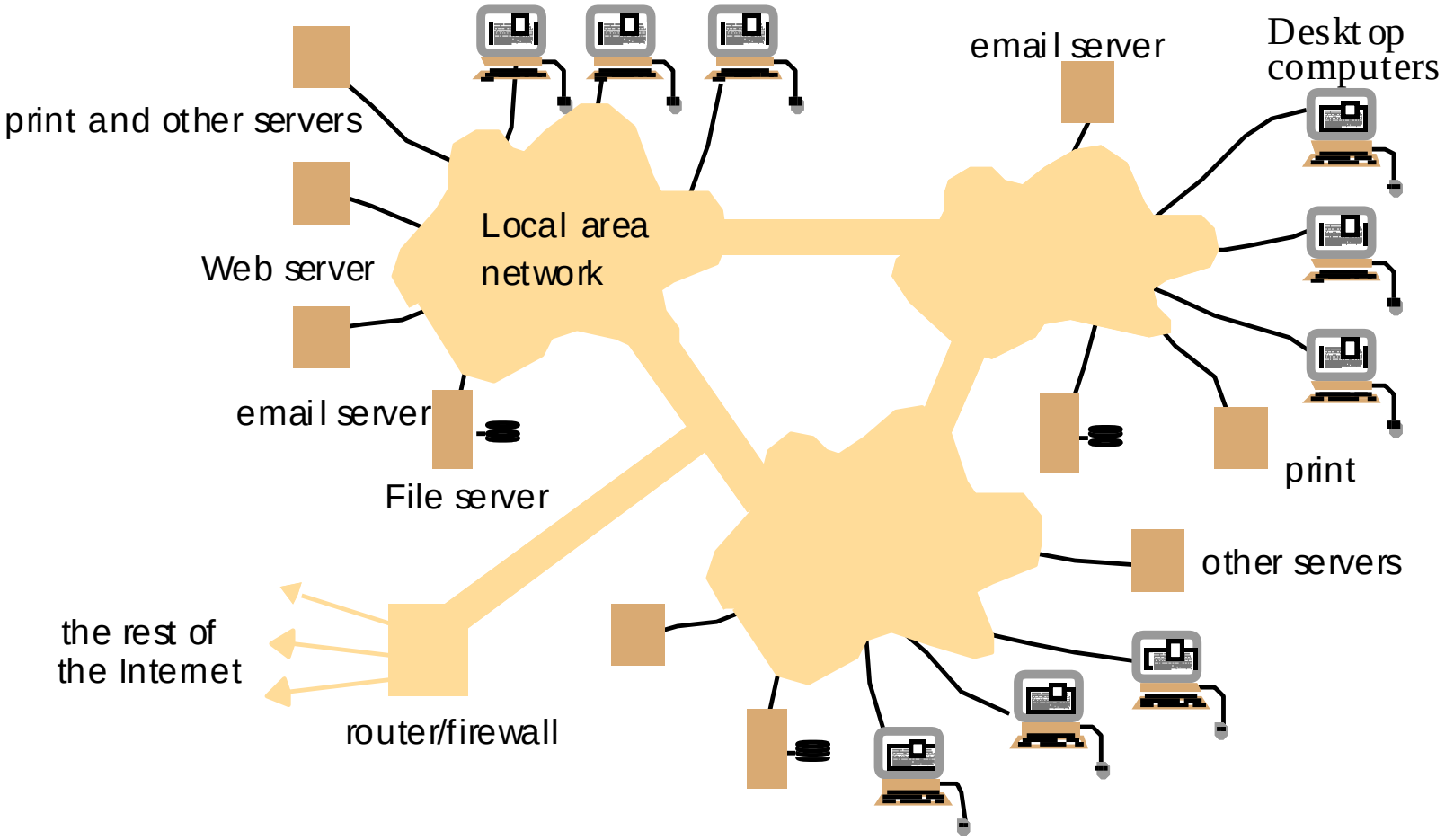
分布式系统举例

- 典型的分布式系统
  - The Internet
  - Intranet
  - 移动计算和普适计算
  - 人 + 信息 + 物理 融合
- 分布式系统的新应用
  - P2P computing 第一个10年
  - Cloud + Mobile computing第二个10年
  - CPS（Cyber-physical system）+IOT 未来10年？

分布式系统举例


intranet
ISP
backbone
satellite link
desktop computer:
server:
network link:

分布式系统举例


email server
Desktop computers
print and other servers
Local area network
Web server
email server
File server
print
other servers
the rest of the Internet
router/firewall

分布式系统举例

- Internet & Intranet
  - 难点：
    - 可扩展性（DNS, IP）
    - 资源的定位
    - 异构
  - 成就：
    - TCP/IP协议是因特网最重要的技术成果

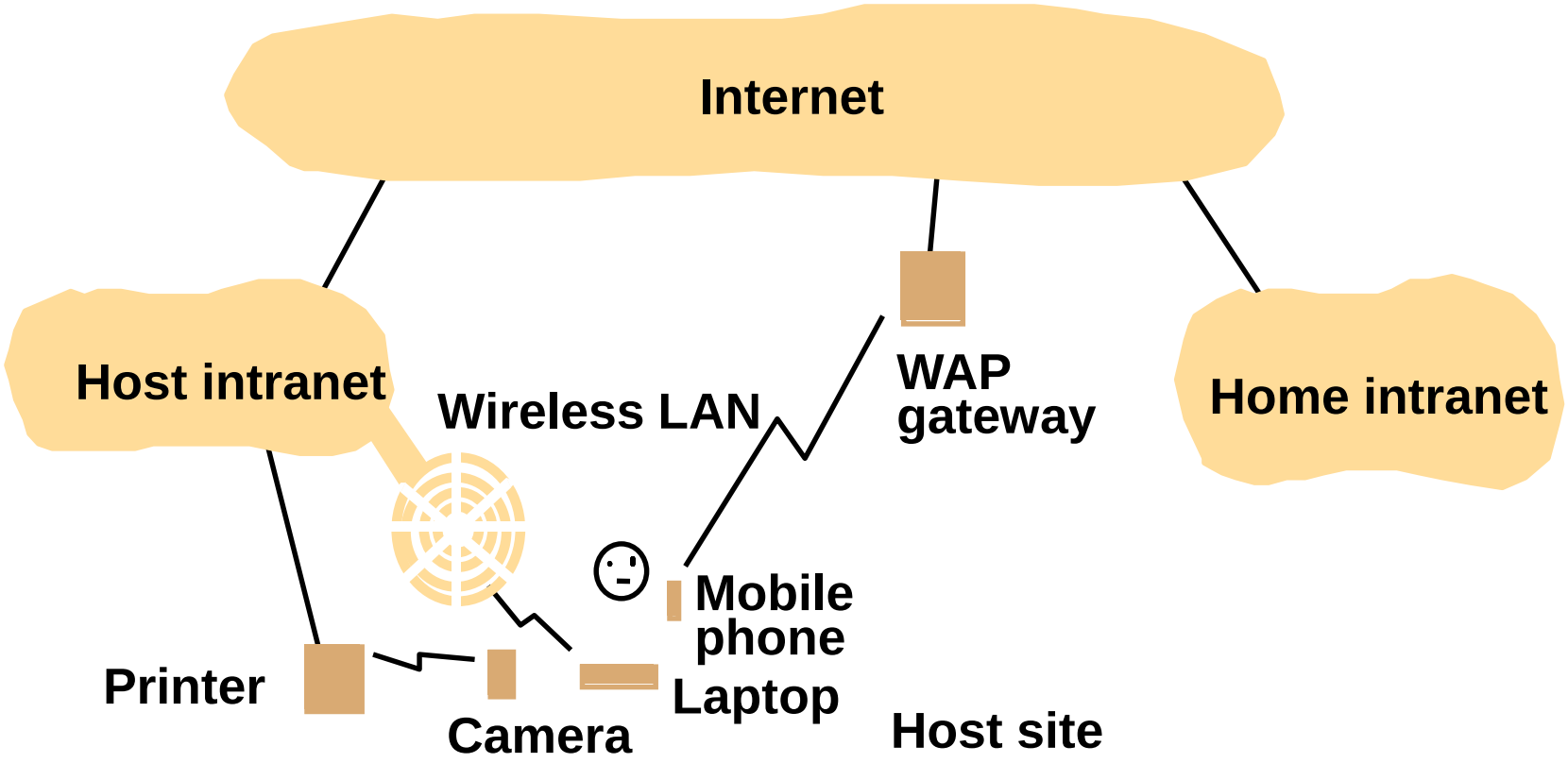
分布式系统举例

- 移动计算（mobile computing）
  - 移动设备
    - 笔记本电脑
    - 手持设备
      - PDA, 手机, 摄像机,数码照相机
    - 可穿戴设备
      - 计算机手表,数字眼镜
    - 家电设备

分布式系统举例

- 移动计算
  - 在移动中执行计算任务的能力，访问因特网的资源
  - 位置感知的计算（location–aware computing)：
    - 在移动环境中，能够发现并附近的资源
  - 自组网络(Ad Hoc)
- 要解决的问题
  - 避免由于移动而需要重新配置的问题（DHCP）
  - 无线带宽有限，需要考虑QoS
  - 私秘和安全问题的解决
  - Ad Hoc网络的路由问题

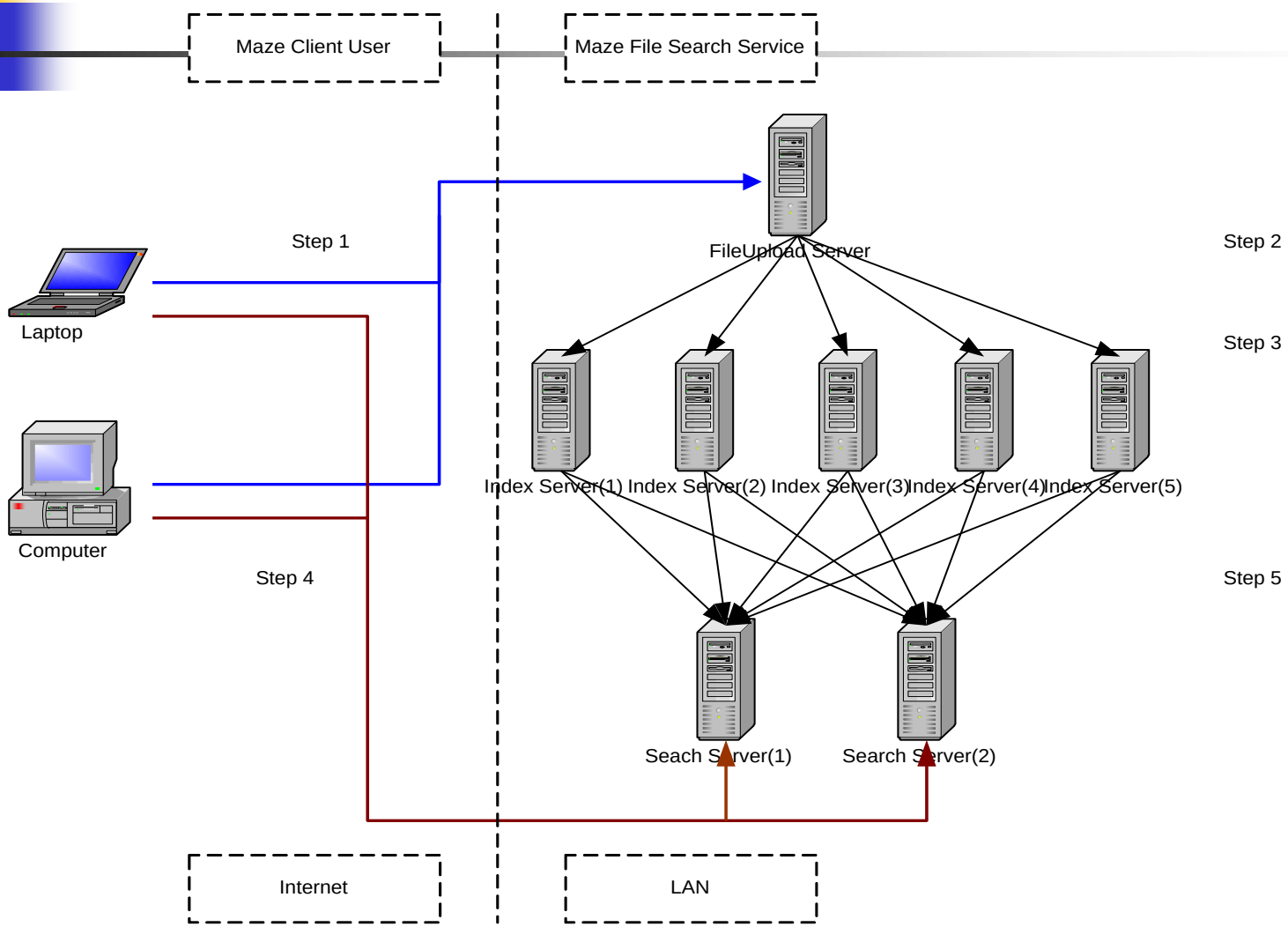
分布式系统举例


Internet
Host intranet
Wireless LAN
WAP
gateway
Home intranet
Mobile
phone
Printer
Laptop
Camera
Host site

分布式系统举例

- 普适计算（Ubiquitous computing，pervasive泛在 computing)
  - 无处不在是指各种小型计算设备最终将普及到现在的日常物品中，不被注意。
  - IP,IP,…,IPv6
- 移动计算和普适计算的区别
  - 有些技术可能通用，例如，无处不在的计算环境可能是无线的。但是，两种计算的应用目标是完全不同的。

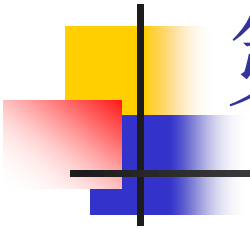
分布式系统举例


Maze Client User
Maze File Search Service
Step 1
Step 2
Step 3
Step 4
Step 5
Laptop
Computer
FileUpload Server
Index Server(1)
Index Server(2)
Index Server(3)
Index Server(4)
Index Server(5)
Seach Server(1)
Search Server(2)
Internet
LAN

星链

智能网联汽车

智慧农业



第1章 概述

- 分布式系统的定义
- 分布式系统举例
- 为什么用分布式系统
- 分布式系统需要实现的需求
- 实现分布式系统的挑战
- 总结

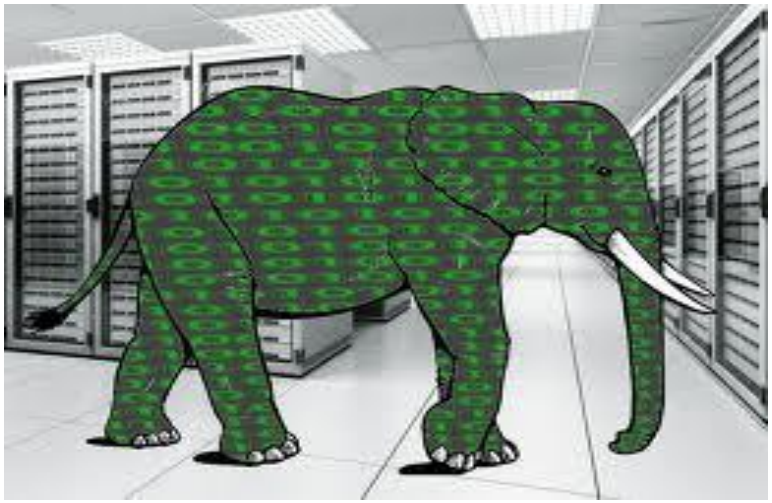
Why Distributed Systems?

- Scale规模伸缩
  - Processing
  - Data
- Diversity in Application Domains应用领域
- Collaboration协作
- Cost

Why Distributed Systems?

A. *Big data* continues to grow:
   - In mid-2020, the information universe carried 64.2 zettabytes and 2025 predictions expect nearly 175 zettabytes coming each year.

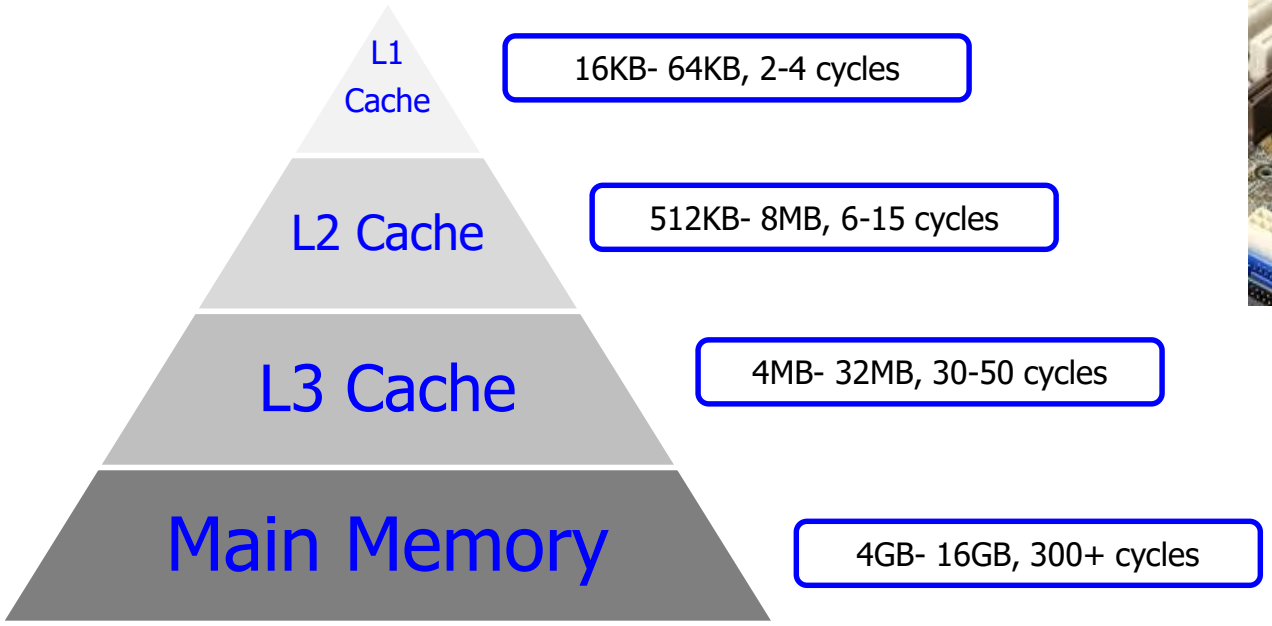
B. Applications are becoming *data-intensive*.

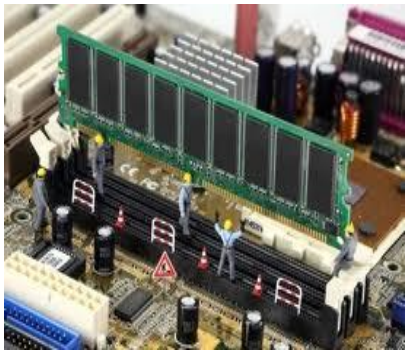


Why Distributed Systems?

C. Individual computers have limited resources compared to scale of current day problems & application domains:

1. Caches and Memory:

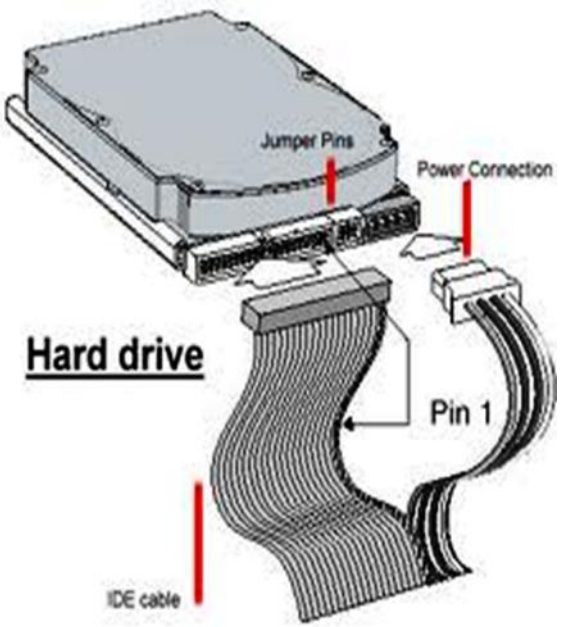




Why Distributed Systems?

2. Hard Disk Drive:

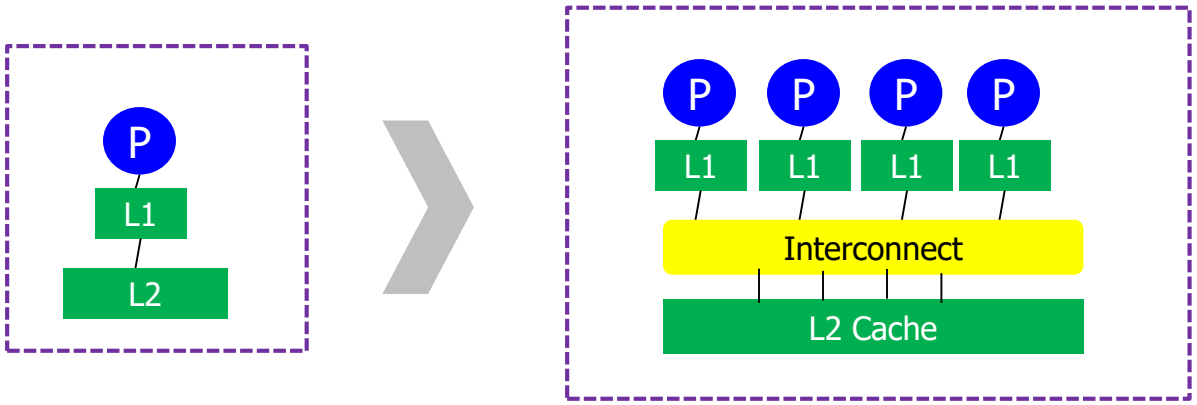
- Limited capacity
- Limited number of channels
- Limited bandwidth



Why Distributed Systems?

3. Processor:

- The number of transistors that can be integrated on a single die has continued to grow at Moore's pace.
- Chip Multiprocessors (*CMPs*) are now available



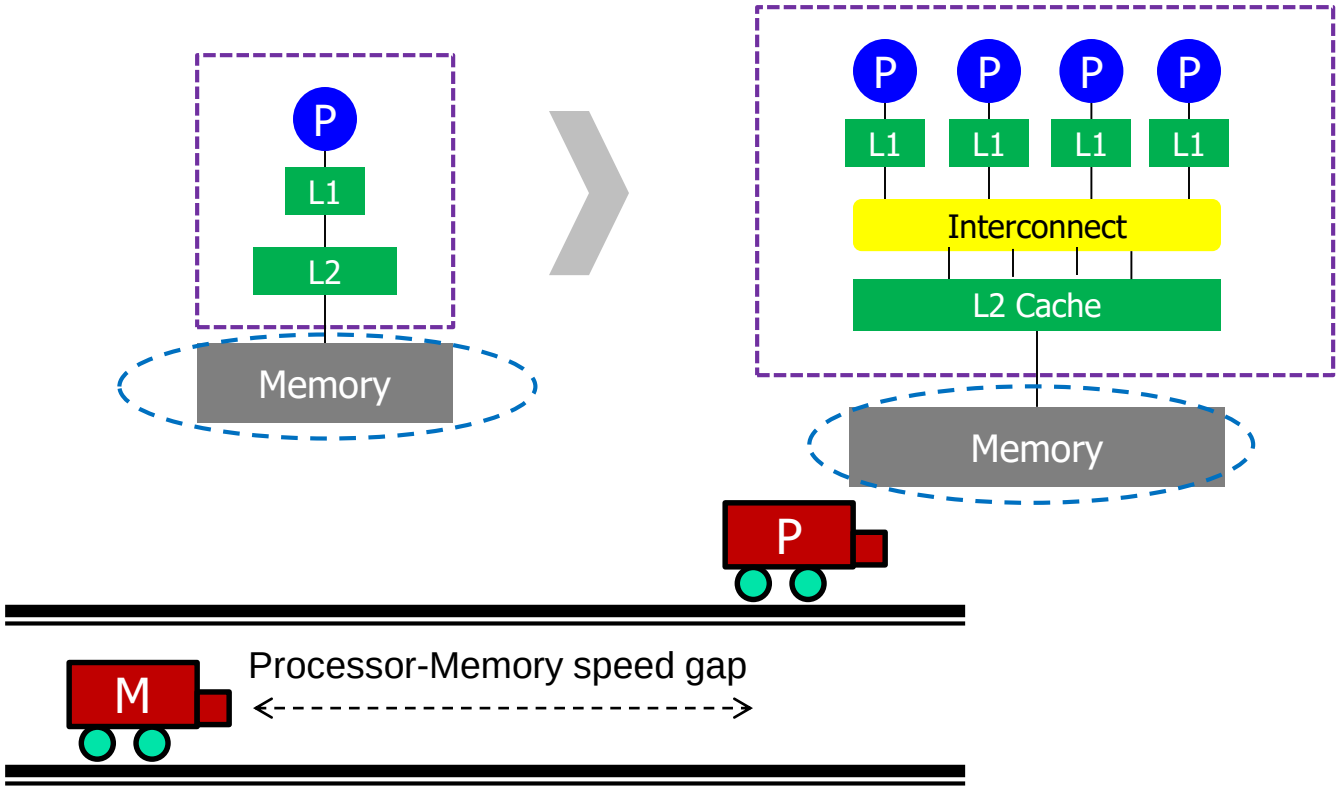
A single Processor Chip

A CMP

Why Distributed Systems?

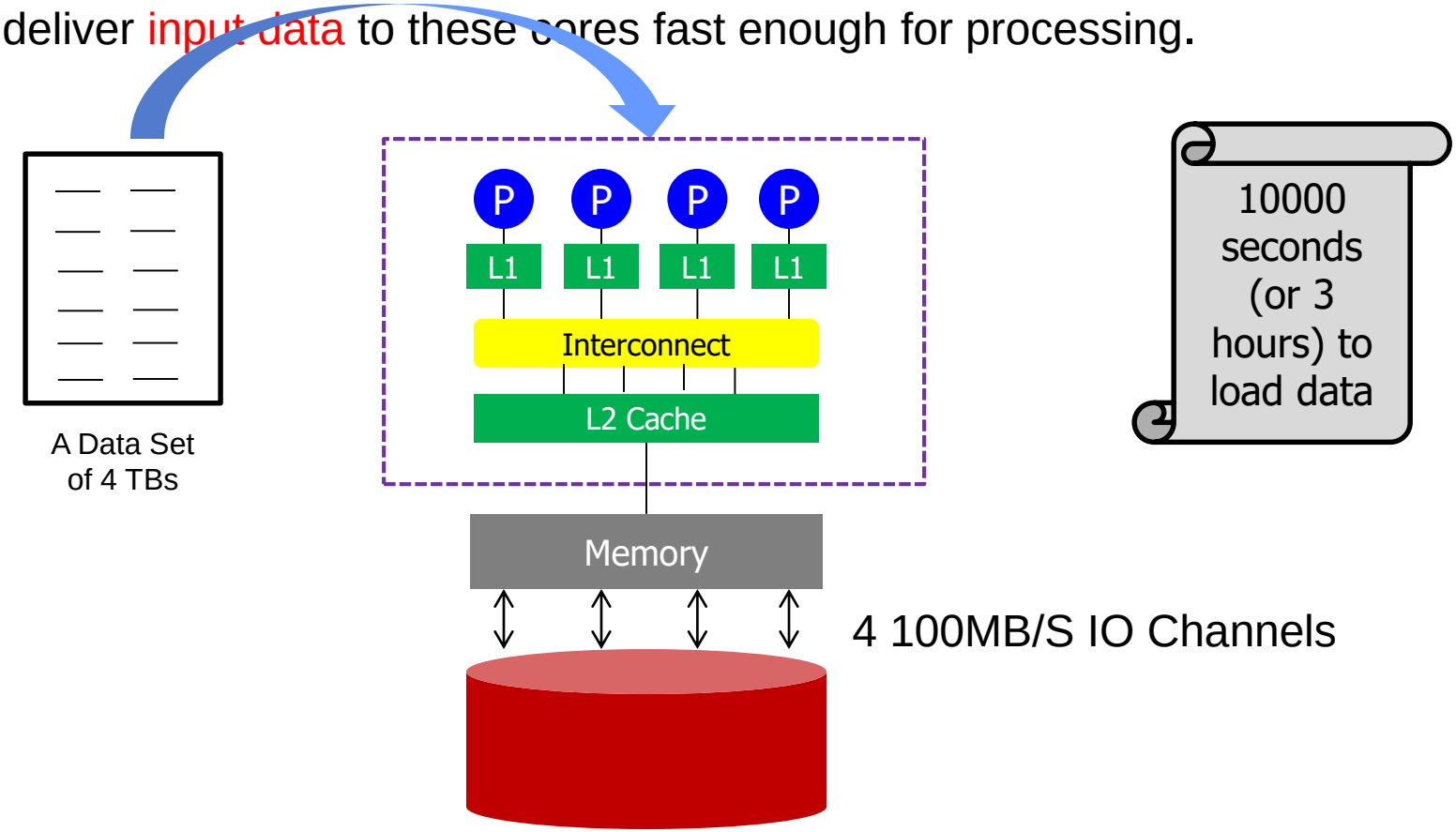
3. Processor (cont'd):

- Up until a few years ago, CPU speed grew at the rate of 55% annually, while the memory speed grew at the rate of only 7% [H & P].

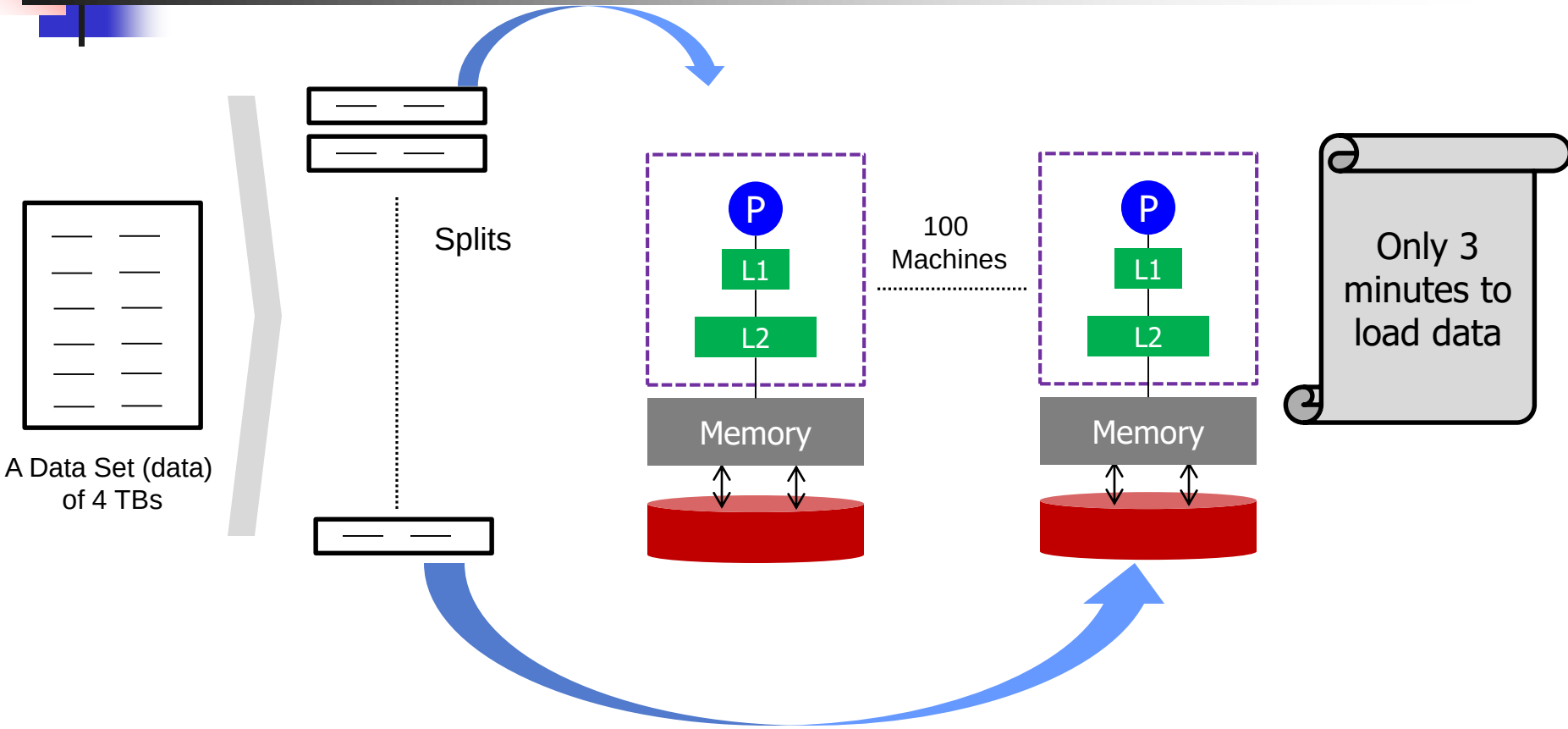


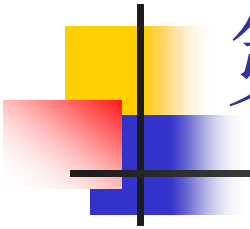
Why Distributed Systems?

- Even if 100s or 1000s of cores are placed on a CMP, it is a challenge to deliver input data to these cores fast enough for processing.



Why Distributed Systems?





第1章 概述

- 分布式系统的定义
- 分布式系统举例
- 为什么用分布式系统
- 分布式系统需要实现的需求
- 实现分布式系统的挑战
- 总结

分布式系统的三个特点

- 并发性（concurrence)
  - 多个程序（进程，线程）并发执行，共享资源
- 没有全局时钟(global clock)
  - 每个机器的有各自的时间，没有办法做到统一，程序间的协调靠交换消息
- 故障独立性(independent failure)
  - 一些进程出现故障，并不能保证其它进程都能知道

我们通常是这么假设的

- 如果网络能保证信息不丢失
- 如果所有发出去的消息都能在预期的时间内到达
- 如果每个机器上的时钟都是精准的
- 如果机器不会宕机，进程不会出故障
- 一个没有实践经验的开发者，通常会在上述假设，很理想化一个系统的实现。只注重应用需求，而忽略实际中会出现什么问题。
- 这样的系统，在实际中根本不能用。

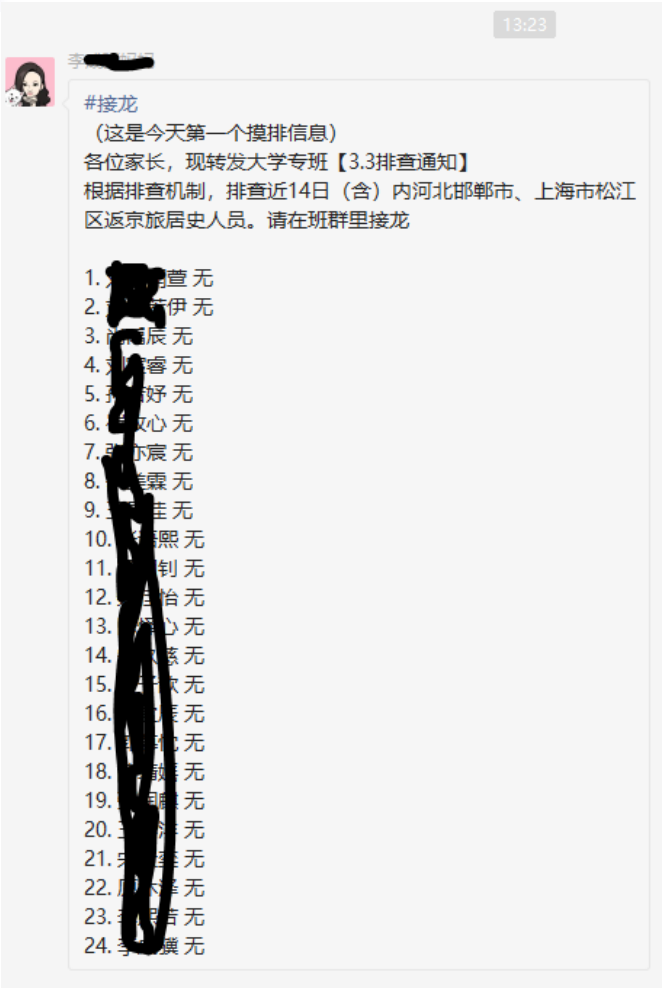
实际上需要面对的问题（1/2）

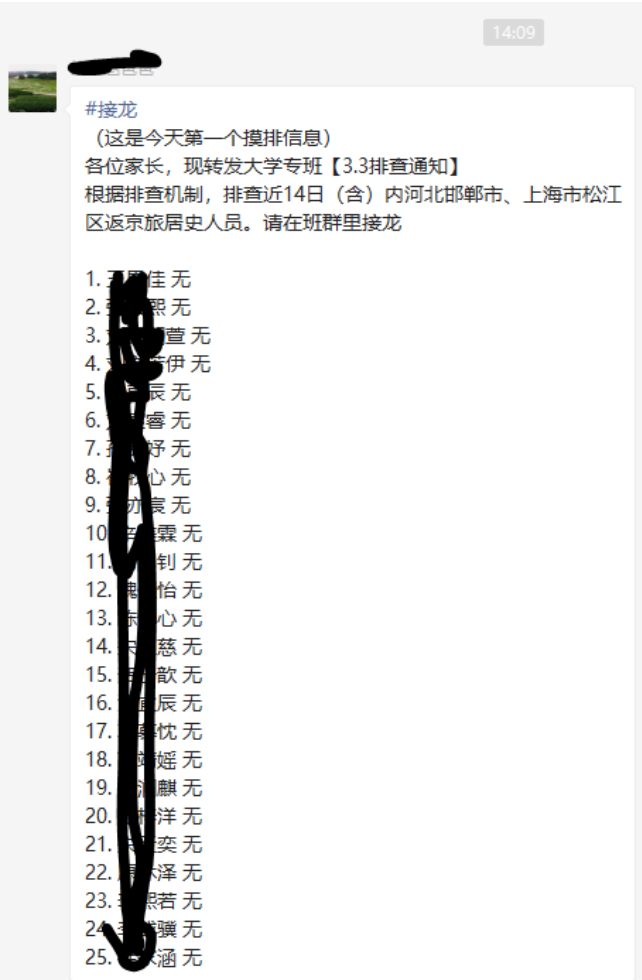
- 什么样的架构合适？
  - 模型
- 两个机器要进行消息传递，我怎么知道对方是否收到了我的信息？我在等待一个回复，可是它迟迟不到，怎么办？
  - 进程间的通信
- 网络上异构的机器怎样进行互操作？
  - 分布对象（中间件）
- 我的文件能不能被别人访问了
  - 分布式系统中的安全

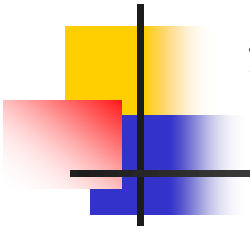
实际上需要面对的问题（2/2）

- 能像用资源管理器一样管理远程文件吗？
  - 文件服务器
- 如何分布资源，如何找到它们？在海量资源的情况下，服务器不堪重负怎么办？
  - 名字服务
- 通知其它机器我上传新的资源了，我怎么知道其它节点是不是收到了？我收到了一个信息，我怎么知道是真是假？
  - 协调和商定
- 上传或下载文件操作执行到中途，不想做了，能删除影响吗？
  - 分布式事务与一直性维护

同一个微信接龙---不一样的顺序

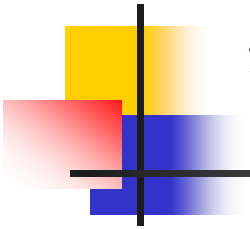






需要实现的需求

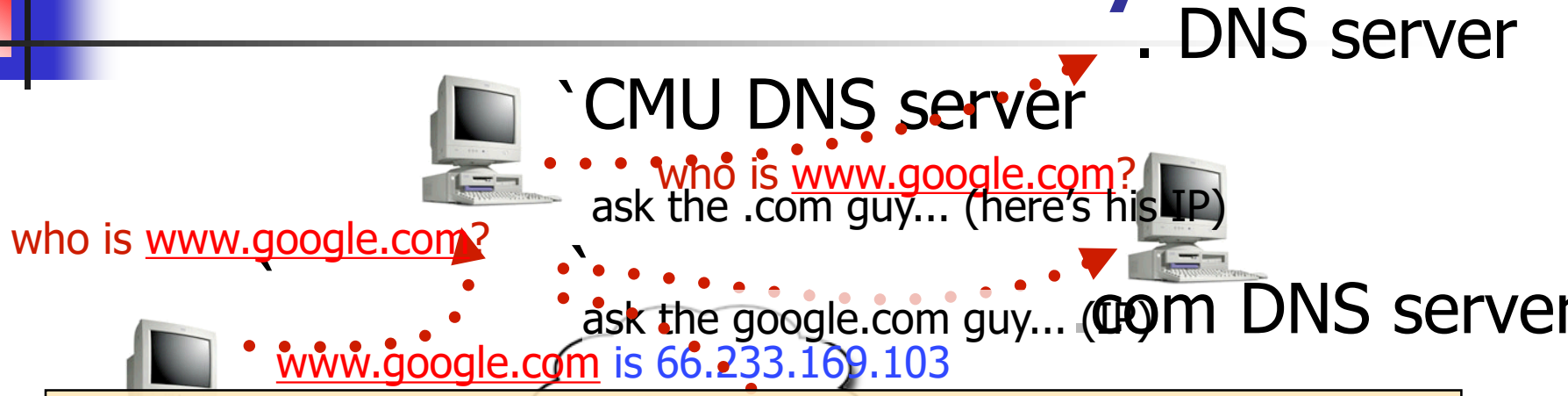
- But this requires:
  - A way to express the problem as parallel processes and execute them on different machines (Programming Models and Concurrency).
  - A way for processes on different machines to exchange information (Communication).
  - A way for processes to cooperate, synchronize with one another and agree on shared values (Synchronization).
  - A way to enhance reliability and improve performance (Consistency and Replication).



需要实现的需求

- But this requires (*Cont.*):
  - A way to recover from partial failures (Fault Tolerance).
  - A way to secure communication and ensure that a process gets only those access rights it is entitled to (Security).
  - A way to extend interfaces so as to mimic the behavior of another system, reduce diversity of platforms, and provide a high degree of portability and flexibility (Virtualization)

Domain Name System

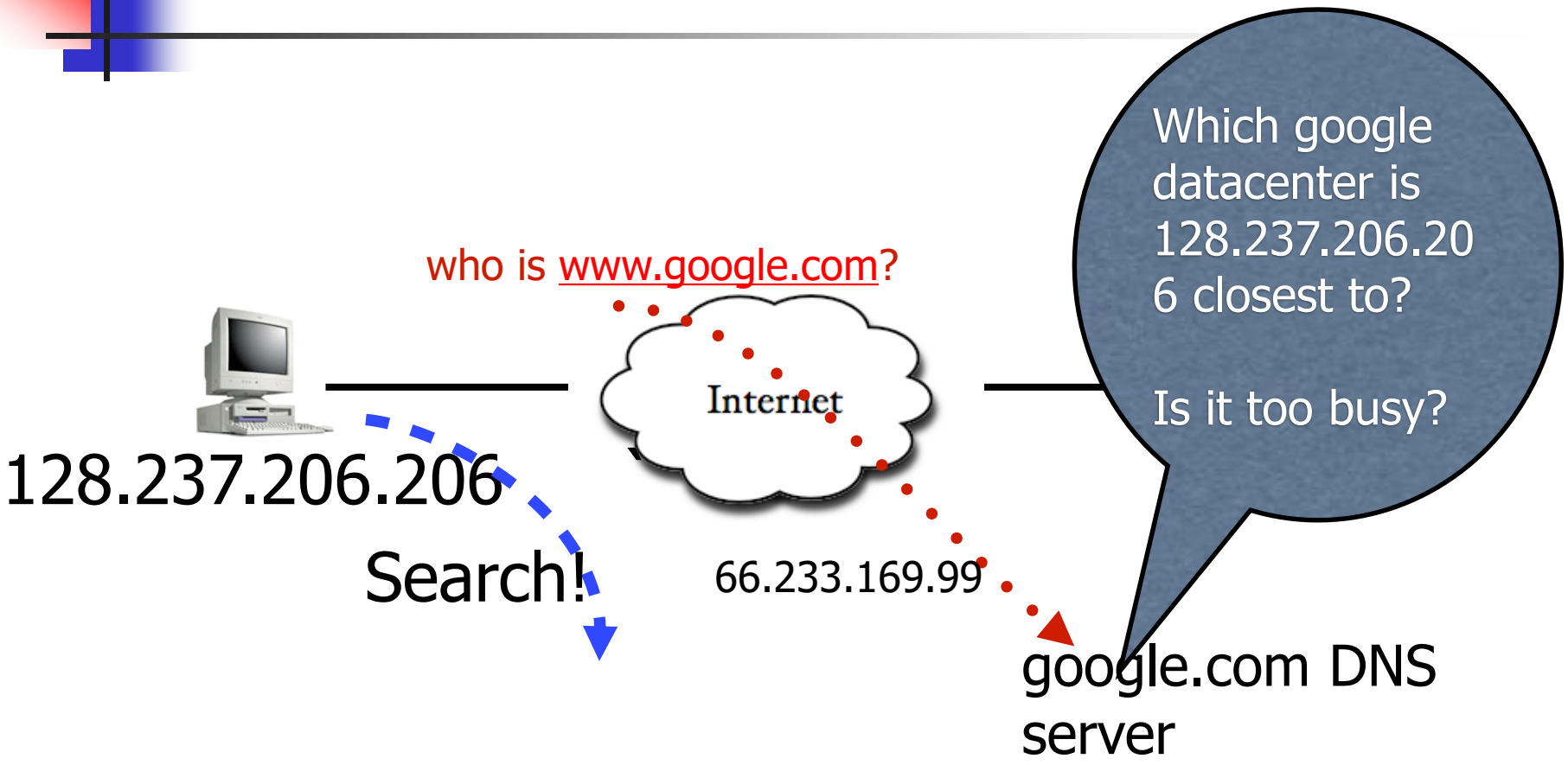


Decentralized - admins update own domains without coordinating with other domains

Scalable - used for hundreds of millions of domains

Robust - handles load and failures well

But there's more...

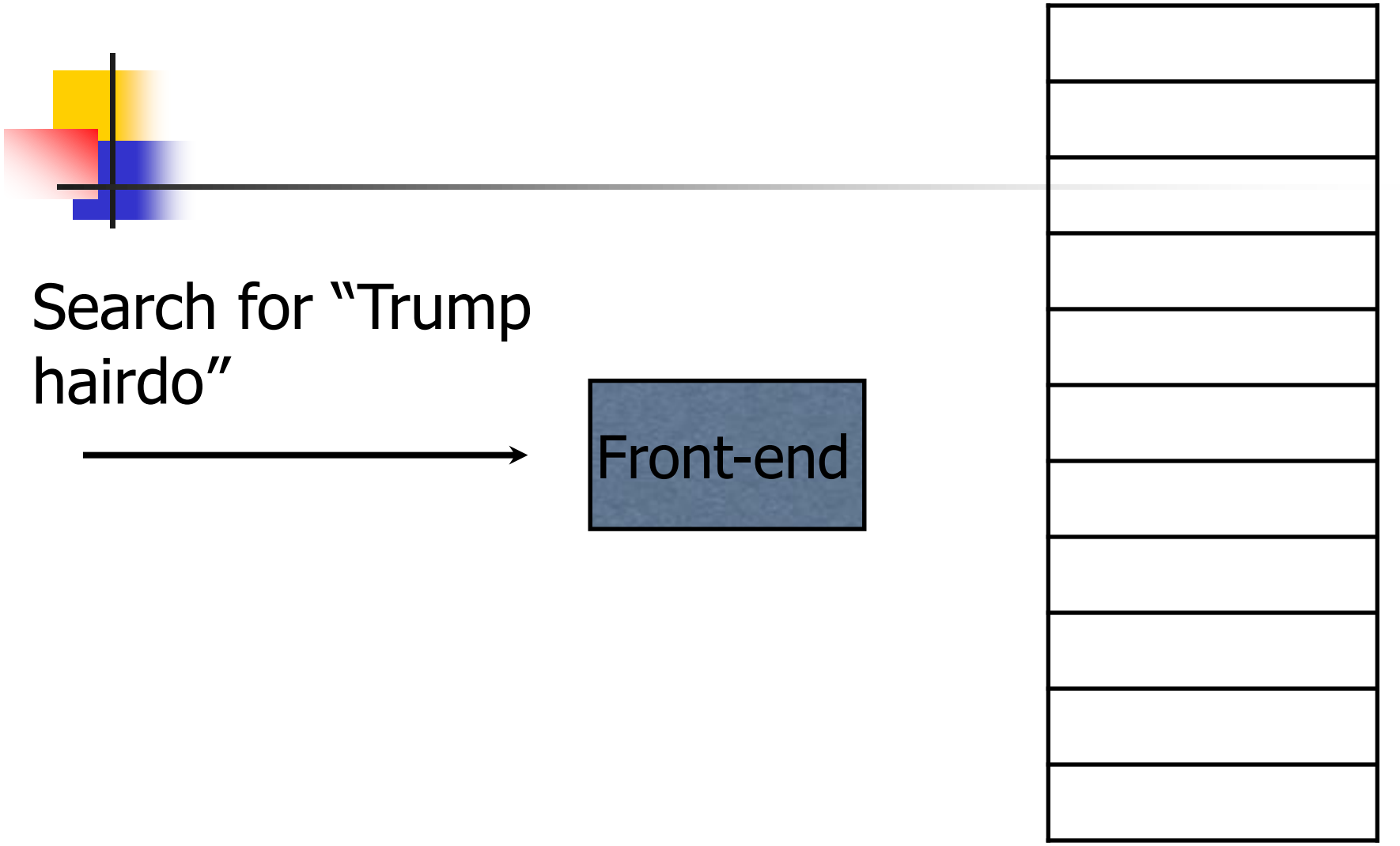


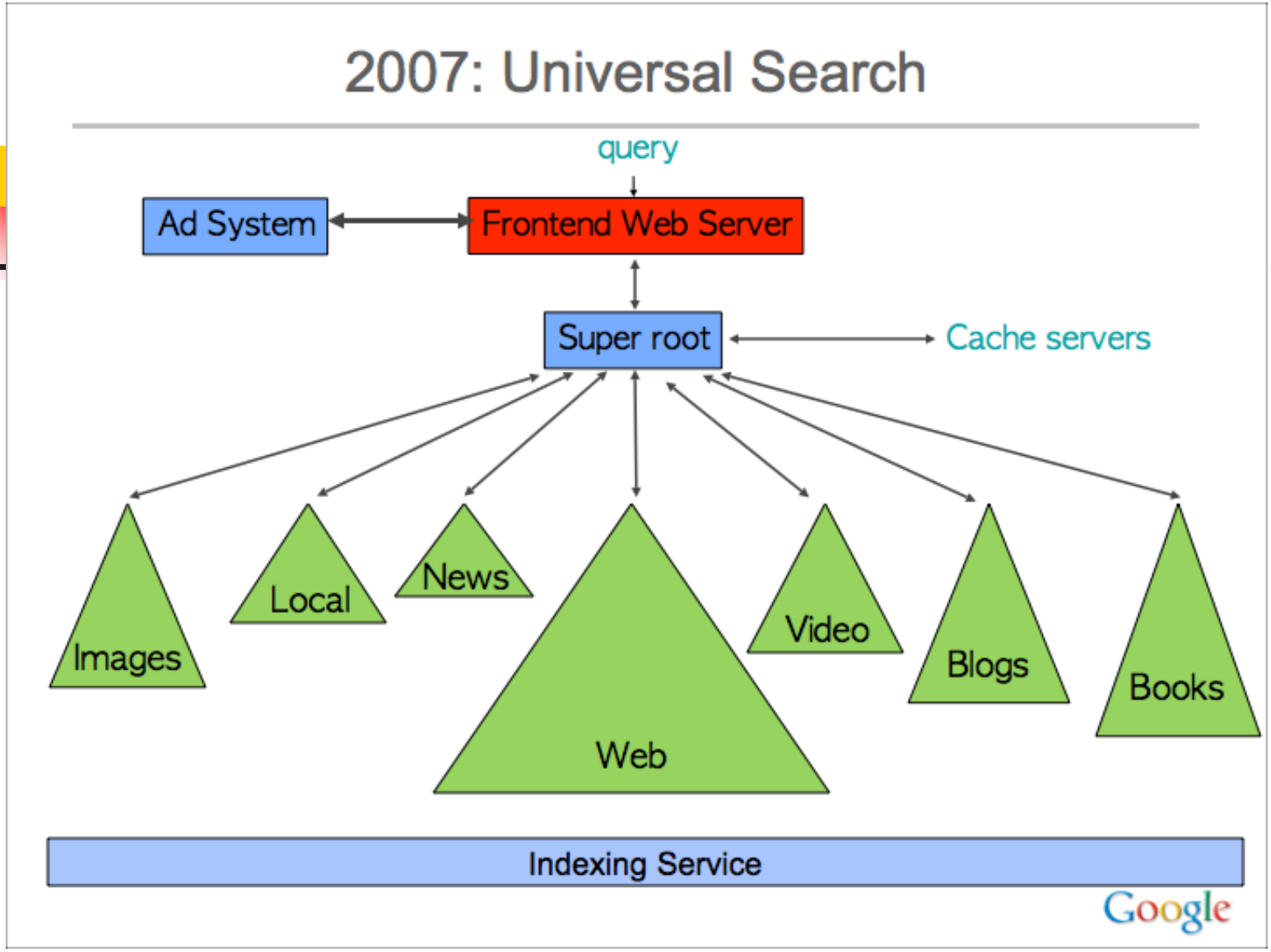
A Google Datacenter

How big? Perhaps one million+ machines

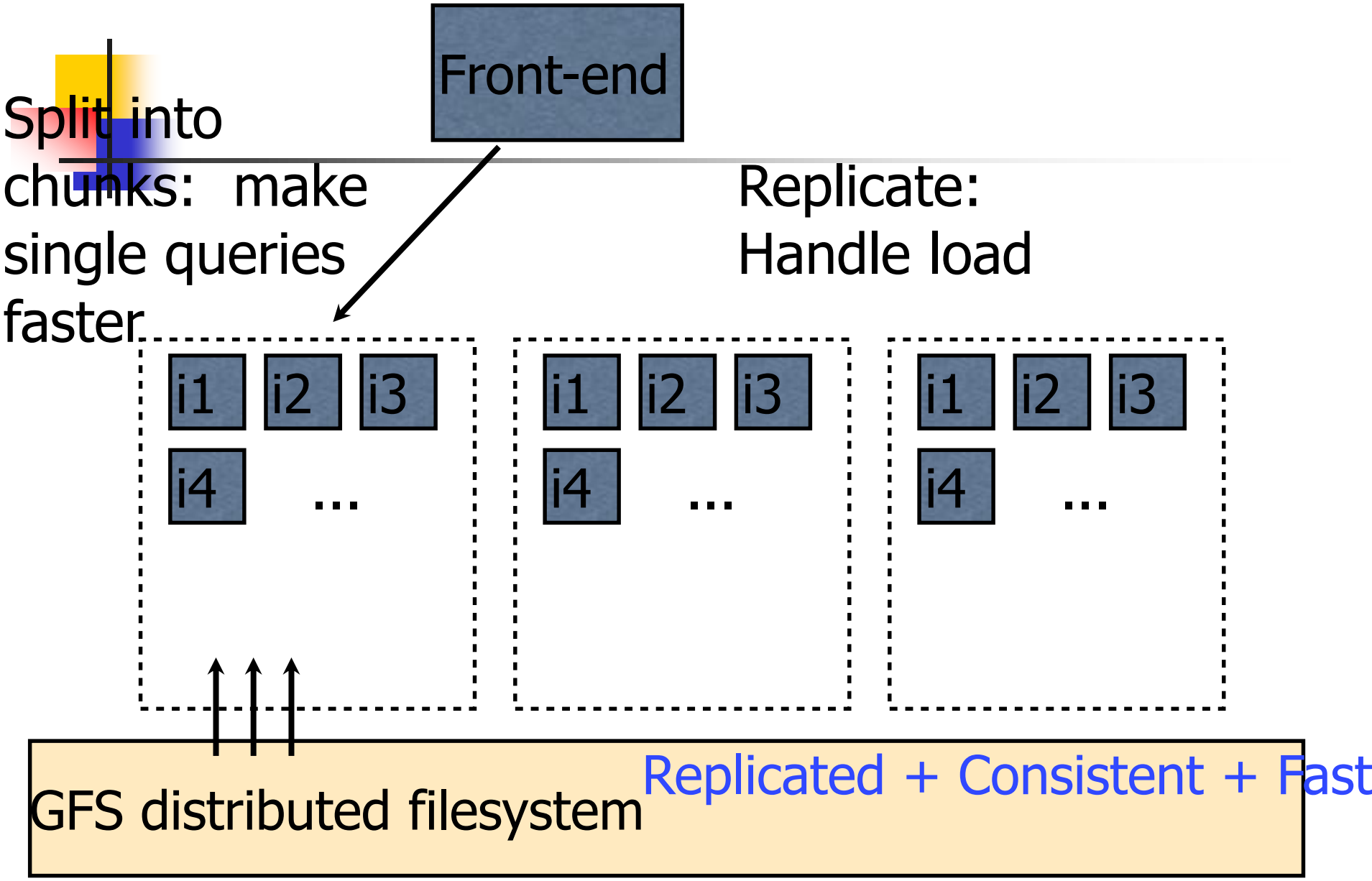
but it's not that bad...

- usually don't use more than 20,000 machines to accomplish a single task. [2009, probably out of date]

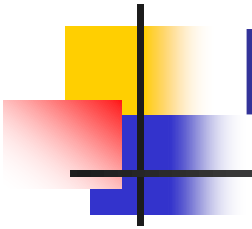

Search for “Trump hairdo”
Front-end



slide from Jeff Dean, Google

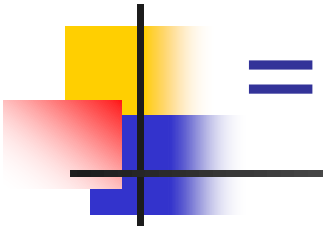

Front-end
Split into chunks: make single queries faster
Replicate: Handle load
i1
i2
i3
i4
...
i1
i2
i3
i4
...
i1
i2
i3
i4
...
GFS distributed filesystem
Replicated + Consistent + Fast

How do you index the web?



- Get a copy of the web.
- Build an index.
- Profit.

There are over 1 trillion unique URLs
Billions of unique web pages
Hundreds of millions of websites
30?? terabytes of text



=

- Crawling -- download those web pages
- Indexing -- harness 10s of thousands of machines to do it
- Profiting -- we leave that to you.

- “Data-Intensive Computing”

MapReduce / Hadoop

Why? Hiding details of programming

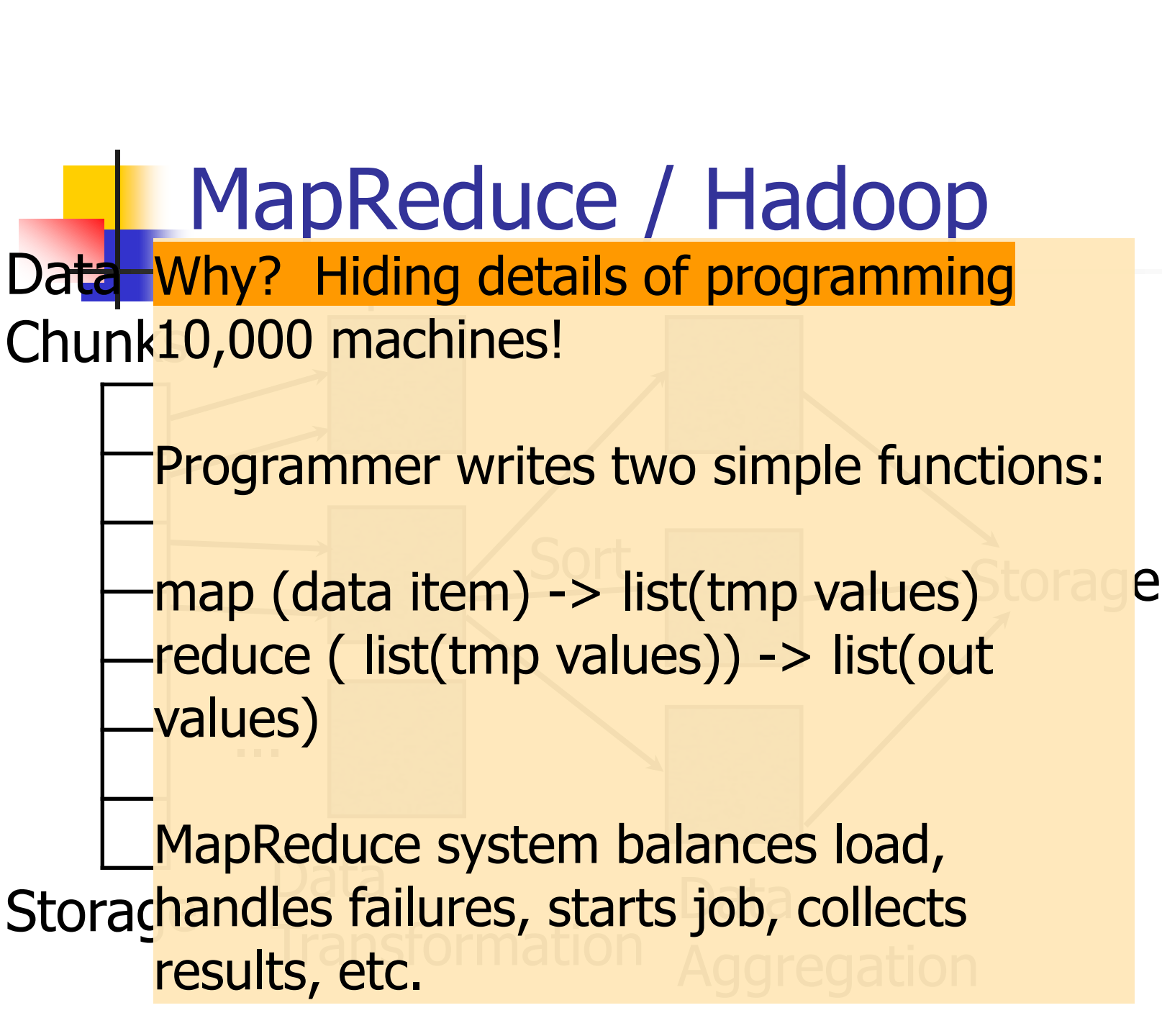
10,000 machines!

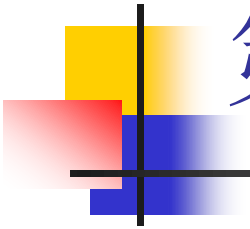
Programmer writes two simple functions:

map (data item) -> list(tmp values)

reduce (list(tmp values)) -> list(out values)

MapReduce system balances load, handles failures, starts job, collects results, etc.



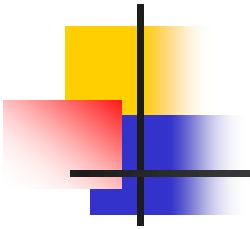


第1章 概述

- 分布式系统的定义
- 分布式系统举例
- 为什么用分布式系统
- 分布式系统需要实现的需求
- 实现分布式系统的挑战
- 总结

挑战---异构性(Heterogeneity)

- 网络协议
  - Ethernet, token ring, etc
- 硬件
  - big endian / little endian
- 操作系统
  - different API of Unix and Windows
- 编程语言
  - different representations for data structures
- 开发者实现方式的不同
  - no application standards



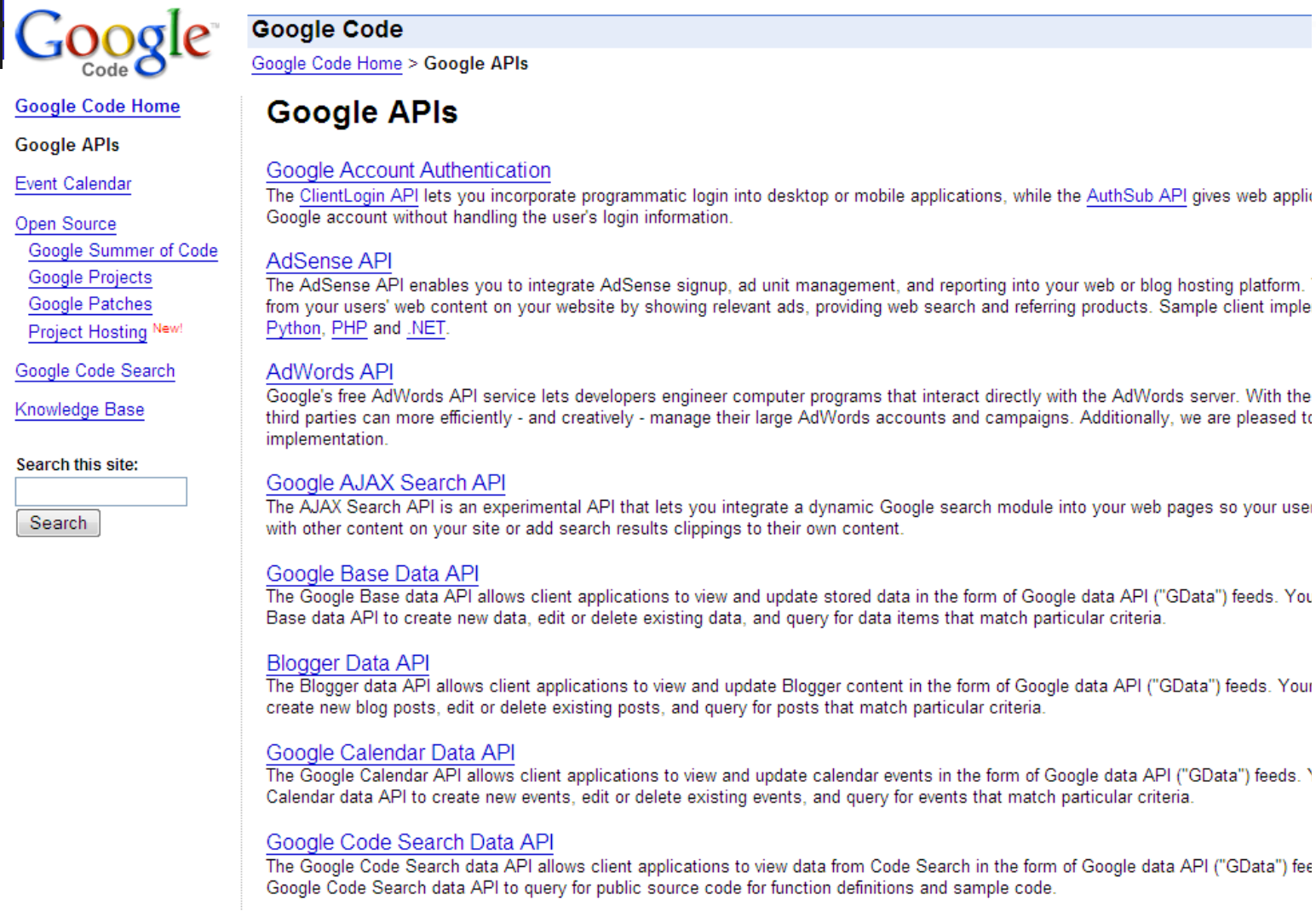
挑战---异构性(Heterogeneity)

- 中间件（Middleware）
  - 应用到软件层，用来屏蔽底层的异构性。例如Java的RMI, 提供远程调用的接口，可以在任何操作系统上运行。
- 移动代码（Mobile code）
  - 移动代码需要在不同的机器间移动并执行，必须解决异构的问题。
  - 虚拟机运行在不同的机器或系统上，代码在虚拟机上运行。

挑战—开放性（Openness)

- 计算机系统的开放性
  - 一个系统是否可以扩充以不同的方式重新实现。例如：UNIX
- 分布式系统的开放性
  - 在多大程度上新的资源共享服务可以加到系统中来。例如：WEB, plug-in
  - RFC

挑战—开放性（Openness)



挑战—开放性（Openness)

挑战—安全性（Security）

- 机密性（Confidentiality）
  - 防止未经授权的个人访问资源
  - e.g. ACL in Unix File System
- 完整性(Integrity)
  - 防止数据被篡改和破坏
  - e.g. checksum
- 可用性（Availability）
  - 防止对所提供服务的干扰
  - e.g. Denial of service

挑战—可扩展性（Scalability）

- 即使系统规模有一定规模的扩展，无论是资源还是用户，系统的性能保持在一定的水平
  - E.g. the Internet
- 设计上的挑战
  - 控制物理资源的代价，e.g.,随着用户数的增长，服务器的增长代价不能超过 O(n)
  - 控制性能损失, e.g., DNS no worse than O(logn)
  - 控制软件资源被耗尽，e.g.，IP address
  - 防止性能瓶颈, e.g., partitioning name table of DNS, cache and replication

挑战—可扩展性（Scalability）

挑战—故障处理（Failure handling ）

- 检测故障
  - e.g. 用校验和检测数据
  - 但是在分布式系统中确切的知道远程服务器是否出现故障是很难做到的。
- 屏蔽故障
  - e.g. 重发没有收到的消息, 备份服务器等
- 故障容错
  - e.g. 无法做到屏蔽故障，至少让用户知道出现了问题，让用户自由选择是否继续请求服务。
- 故障恢复
  - e.g. 操作日志，恢复。
- 冗余策略
  - e.g. IP route, replicated name table of DNS

挑战—并发（Concurrency）

- 正确性
  - 多个进程并发访问共享资源，要保证被访问数据的正确性，不能出现不一致。
- Performance
  - 多个并发操作保证性能
  - E.g. Maze 索引服务器，按UID，将任务分布给10个逻辑进程来处理。

挑战—透明性（Transparency）

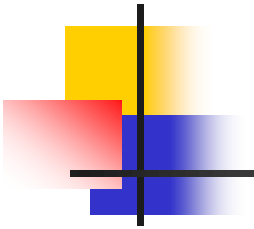
- 访问透明（Access transparency）
  - 使用同样的操作去访问本地资源和远程资源。E.g. NFS / Windows File Sharing
- 位置透明（Location transparency）
  - 访问资源的时候，不需要知道资源的位置。E.g. URL
- 并发透明（Concurrency transparency）
  - 几个进程同时访问资源，互不干扰

挑战—透明性（Transparency）

- 复制透明（Replication transparency）
  - 使用多个资源的副本来提高可靠性和性能，用户或者应用程序开发者并不需要了解副本技术。
- 故障透明（Failure transparency）
  - 在存在故障的情况下，用户和应用仍可完成他们的任务, e.g., email

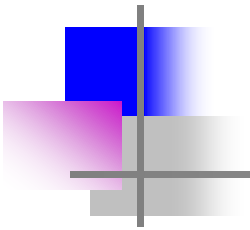
挑战—透明性（Transparency）

- 移动透明（Mobility transparency）
  - 资源或者客户端的移动不影响用户及程序的操作。
    E.g. mobile phone
- 性能透明（Performance transparency）
  - 允许系统重新配置改善性能，例如改变负载。
- 扩展透明（Scaling transparency）
  - 允许系统和应用扩大规模无需改变系统的结构和用算法。



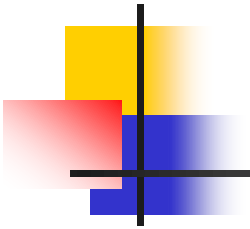
第1章 概述

- 分布式系统的定义
- 分布式系统举例
- 为什么用分布式系统
- 分布式系统需要实现的需求
- 实现分布式系统的挑战
- 总结



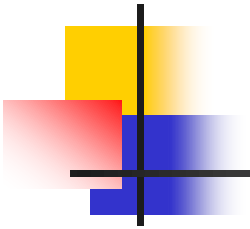
总结

- 分布式系统无处不在（Pervasive 泛在）
- 构造分布式系统的主要动机是资源共享和协同计算
- 分布式系统的特点
  - 并发性
  - 没有全局时钟
  - 故障独立性



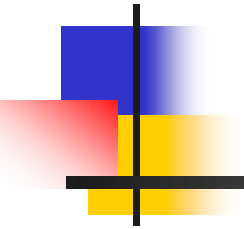
总结

- 构造分布式系统面临的挑战
  - 异构性（Heterogeneity）
  - 开放性（Openness）
  - 安全性（Security）
  - 扩展性（Scalability）
  - 故障处理（Failure handling）
  - 并发行（Concurrency）
  - 透明性（Transparency）



思考题

- 中间件在分布式系统中扮演什么角色？
- 解释（分布）透明性的含义，并且给出各种类型透明性的例子。
- 在分布式系统中，为什么有时难以隐藏故障的发生以及故障恢复过程？
- 请对可扩展系统的含义做出准确描述。
- 为什么有时候要求最大程度地实现透明性并不好？



END