分布式数据管理系统

分布式系统课程安排

- **分布式系统概念和模型**
  - 基本概念
  - 实践：
    - 定义和描述分布式系统
    - 开发、调试、部署分布式系统

- **1.分布式系统概念**
- **2.系统模型（抽象）**
  - 抽象模型的意义
  - 形式化方法（以 TLA+ 为例）
- **3.系统模型（实现）**
  - 抽象模型 → 具体平台实现；
  - 编程范式与实现技巧

分布式系统课程安排

- **分布式数据管理系统具体技术：**
  - 例，高可用…
    - 读论文
      - 复制，Raft，Paxos
      - 读列表中的文章
    - 实践：
      - 高可用 Key-Value 存储
- **4.分区与复制**
- **5.共识算法**
- **6.分布式事务**

分布式系统课程安排

- **系统软件质量保障**
  - 读论文
    - 超越专家测试
      - 人工构造测试用例的方法
    - 模拟，随机化，验证，形式化
  - 实践：
    - 应用前沿技术
    - 测试，验证前面的系统
- **7.传统质量保障技术**
  - 传统测试方法
  - 随机化方法
- **8.现代质量保障技术**
  - 确定性模拟
  - 形式化验证

期中作业

- **1.寻找一个需求场景**
  - 必须要开发一个分布式数据管理系统
  - 而不是，一台机器撑起世界
- **2.描述系统规范和规格**
  - 非形式化、形式化地
- **3.实现它**
  - 用你认为合适的技术
  - AI Helps
- **4.保障它的质量**
  - 你写的是一个软件，而不只是一个程序
- **可以（鼓励）你使用AI**
- **要把你自己想象成，**
  - 管理/拥有这个软件项目的经理
  - 而不是，精通某种语言/平台的程序员
- **在AI的时代**
  - 完成1 ~ 4的闭环更重要

你不是写代码的人

你是一个系统产品的负责人

分布式系统学习路径

- **读书：**
  - 了解基础概念
- **读论文：**
  - 前沿技术、方法
- **动手实践：**
  - 编写，调试，测试，验证代码

- **在AI时代，**
- **你可以把这件事倒过来**
  - 先找到你要解决的问题
  - 从问题出发找你需要的技能
  - 让AI工具填补你空缺的技能
  - 对结果负责
- **“两端”比“中间”更重要**
- **“以终为始”**

为什么一台机器撑不起世界?

故事一：双十一

如果只有一台服务器，会发生什么？

故事二：抢红包

全球几十亿用户，如何同时在线?

故事三：GitHub

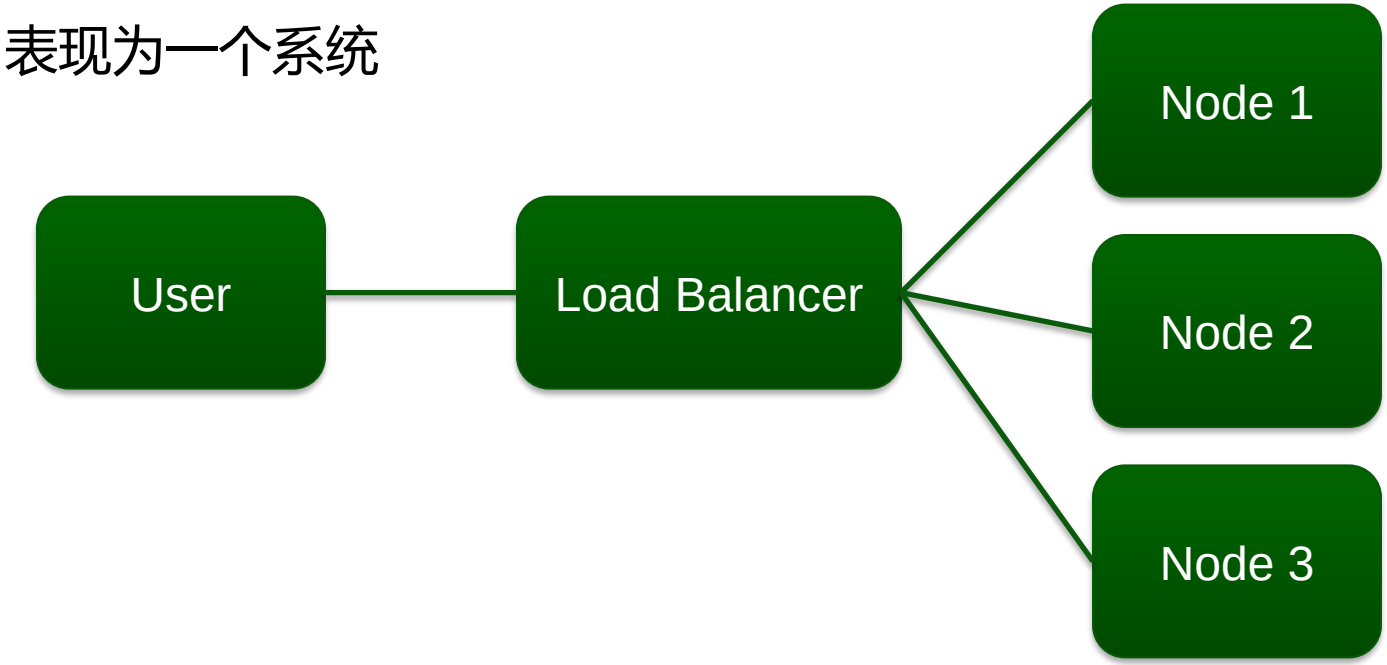
全球协作，代码如何保持一致?

为什么需要分布式系统?

- **扩展性**
  - 增加节点横向扩展性能
  - 处理海量请求或数据
  - 数据库分片（sharding）
- **可靠性**
  - 冗余设计避免单点故障
  - 提升系统可用性
  - 高可用，容错
- **资源共享**
  - 整合分散的资源
    - 云计算
    - 共享文档编辑
    - Git
- **低延迟**
  - 将服务部署在靠近用户的地理位置
    - CDN
- **不得不分布式**
  - 地理分布
    - World Wide Web（WWW）
    - 点对点系统

什么是分布式系统?

- **多台计算机**
  - 通过网络连接
  - 协同完成任务
  - 对用户表现为一个系统



什么是分布式系统?

- **分布式系统**
  - 由多台独立的计算机（称为**节点**）
  - 通过网络连接协同工作
  - 共同完成任务
  - 这些节点分布在不同的地理位置
  - 对外表现为一个统一的整体

多机协作 → 对外统一

真正的难题是什么?

- **网络会延迟**
- **机器会宕机**
- **数据会冲突**
- **你无法预测未来**
  - 即，无法区分“慢”和“挂了”

Latency Numbers Everyone Should Know

Operation	Time in ns	Time in ms (1ms = 1,000,000 ns)
L1 cache reference	1	
Branch misprediction	3	
L2 cache reference	4	
Mutex lock/unlock	17	
Main memory reference	100	
Compress 1 kB with Zippy	2,000	0.002
Read 1 MB sequentially from memory	10,000	0.010
Send 2 kB over 10 Gbps network	1,600	0.0016
SSD 4kB Random Read	20,000	0.020
Read 1 MB sequentially from SSD	1,000,000	1
Round trip within same datacenter	500,000	0.5
Read 1 MB sequentially from disk	5,000,000	5
Read 1 MB sequentially from 1Gbps network	10,000,000	10
Disk seek	10,000,000	10
TCP packet round trip between continents	150,000,000	150

Source: Jeff Dean, "Latency Numbers Every Programmer Should Know"

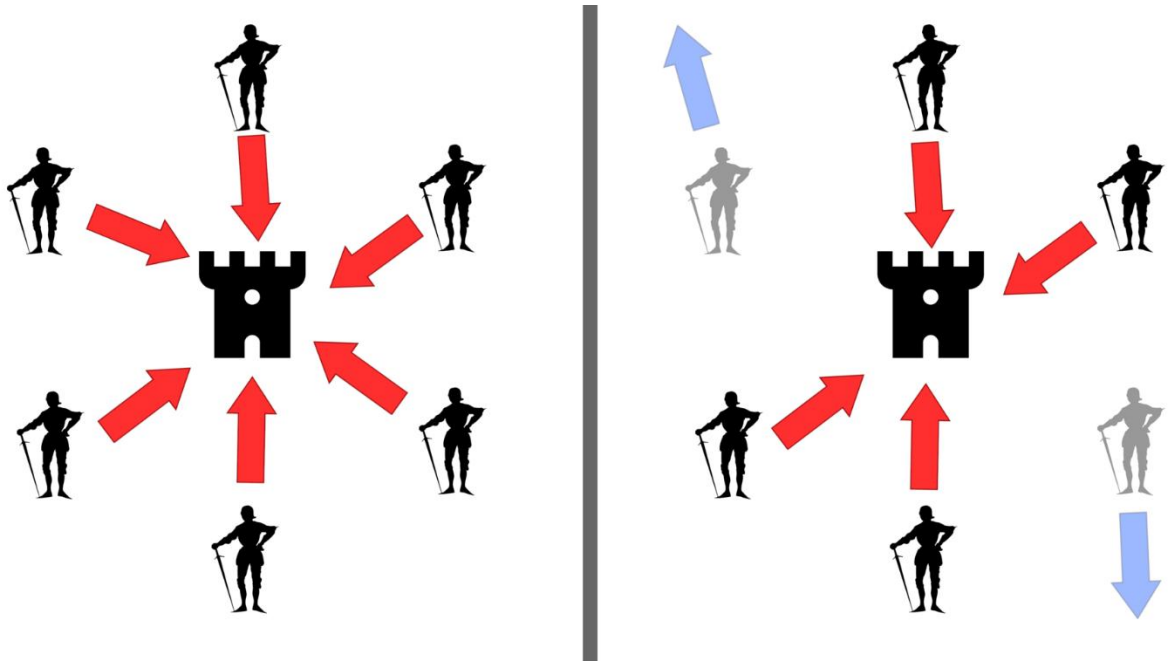
分布式系统抽象模型

- **失败模型(这里介绍两种典型模型)**
  - 失败停止（Fail-stop）模型
    - 无恶意者
      - 消息会丢，但不会被篡改
    - 例：高可用数据库
  - 拜占庭失败Byzantine failure模型
    - 存在恶意者
      - 消息可能会被篡改
    - 例：区块链
- **同步或异步模型**
  - 同步模型
    - 通信和任务存在时间上界约束
    - 消息延迟超过上界 N，则被认为丢失
  - 异步
    - 通信和任务不存在时间上界约束
    - 消息延迟，丢失不确定

本课程绝大多数情形：失败停止+异步模型

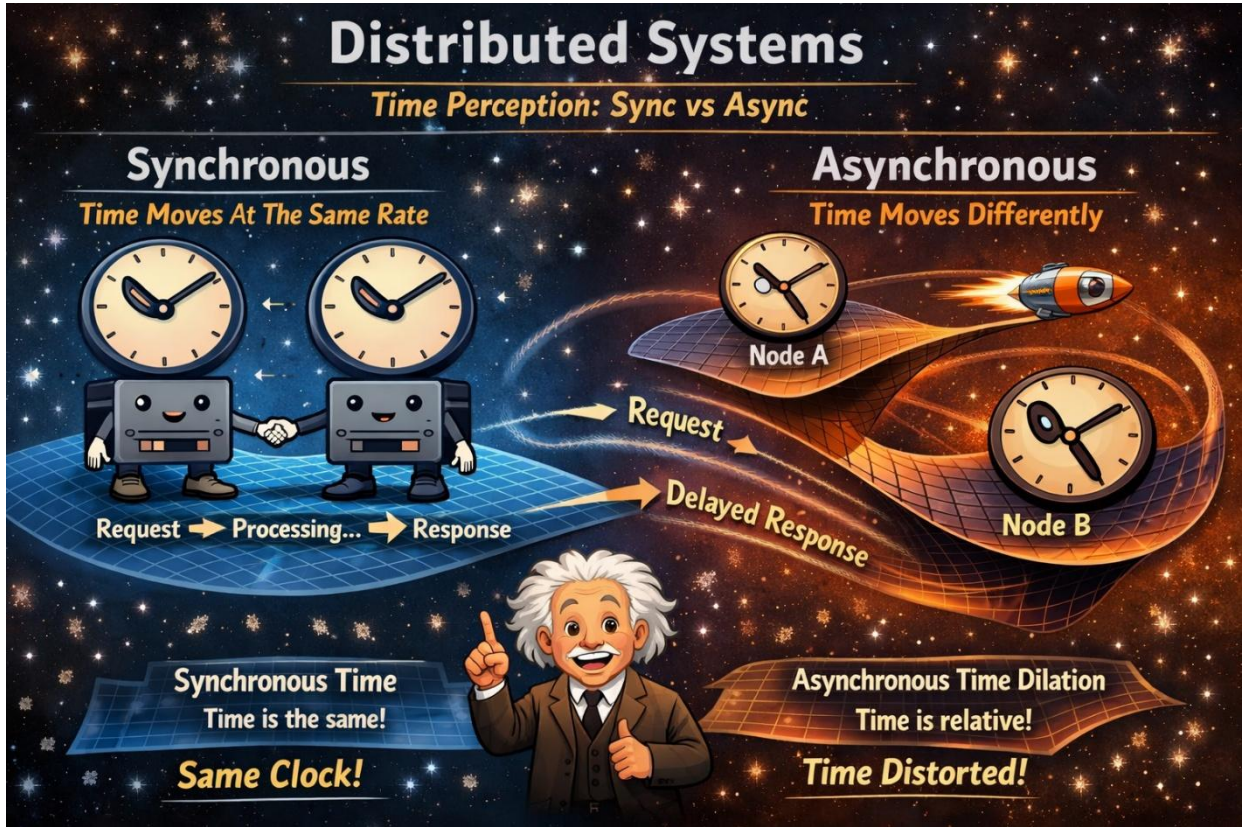
两种失败

- Fail-stop：机器挂了，大家知道
- Byzantine：机器说谎，你不知道



时间问题

- **同步模型：**
  - 存在已知时间上界
- **异步模型：**
  - 不存在已知时间上界



分布式系统的本质

不确定性 + 失败 + 网络

CAP定理（简单理解）

- **CAP**
  - **C**onsistency（一致性）
  - **A**vailability（可用性）
  - **P**artition tolerance（分区容忍）
- **在发生网络分区时，只能在 C 与 A 中二选一**

Harvest, Yield, and Scalable Tolerant Systems

分布式系统问题

- 分区
- 复制
- 共识
- 并发
- 同步
- …

是否需要分布式系统?

- **设计系统之前考虑**
- **是否需要分布式系统?**
  - 能不要就不要

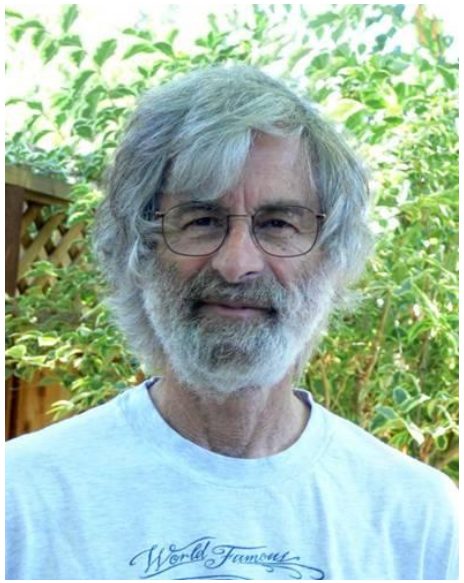
分布式系统挑战

- **一致性**
- **容错性**
- **可扩展性**
- **安全性**
- **复杂性**
  - 分布式系统是“复杂系统”
- **…**

复杂性来自：网络 + 时间 + 不确定性

Leslie Lamport说

- **A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable.**

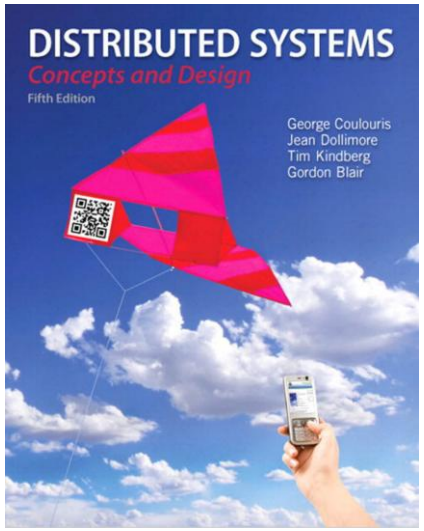


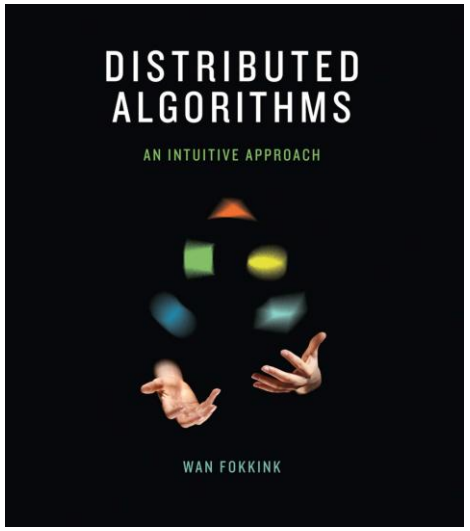
—— Leslie Lamport

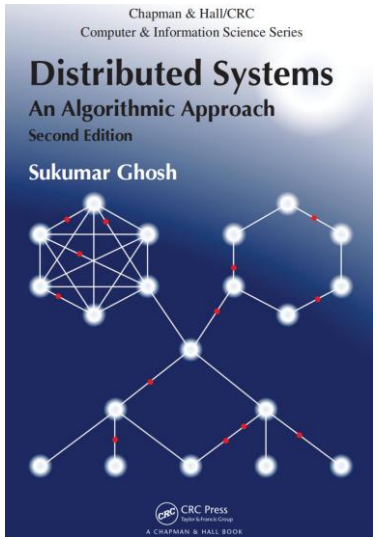
你将学习什么?

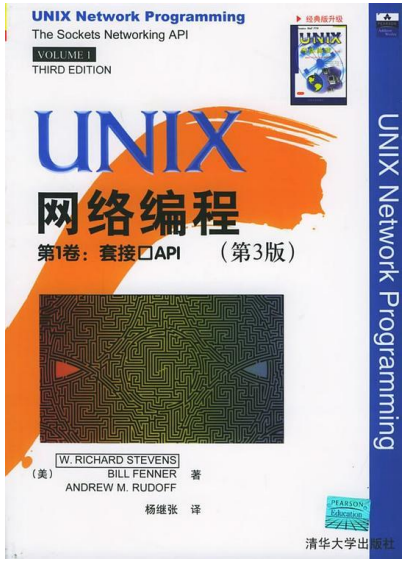
抽象模型 → 具体技术 → 质量保障

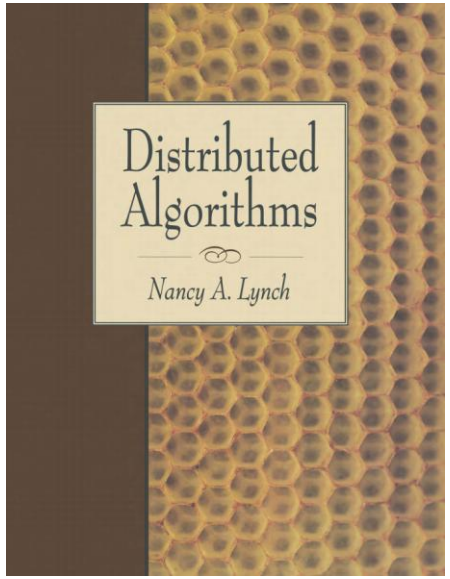
参考书

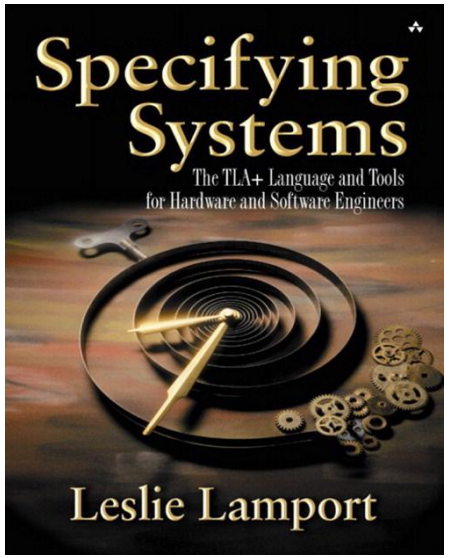

DISTRIBUTED SYSTEMS
Concepts and Design
Fifth Edition
George Coulouris
Jean Dollimore
Tim Kindberg
Gordon Blair


DISTRIBUTED ALGORITHMS
AN INTUITIVE APPROACH
WAN FOKKINK


Chapman & Hall/CRC
Computer & Information Science Series
Distributed Systems
An Algorithmic Approach
Second Edition
Sukumar Ghosh
CRC Press


UNIX Network Programming
The Sockets Networking API
VOLUME 1
THIRD EDITION
UNIX
网络编程
第1卷：套接口API （第3版）
W. RICHARD STEVENS
BILL FENNER
ANDREW M. RUDOFF
杨继张 译
清华大学出版社
UNIX Network Programming


Distributed Algorithms
Nancy A. Lynch


Specifying Systems
The TLA+ Language and Tools for Hardware and Software Engineers
Leslie Lamport

欢迎进入复杂世界

你准备好了吗?