分布式系统
Distributed Systems

陈鹏飞
数据科学与计算机学院
chenpf7@mail.sysu.edu.cn
办公室：超算5楼529d
主页：http://sdcs.sysu.edu.cn/node/3747



第一讲 — 分布式系统简介



1 个人简介

2 课程安排

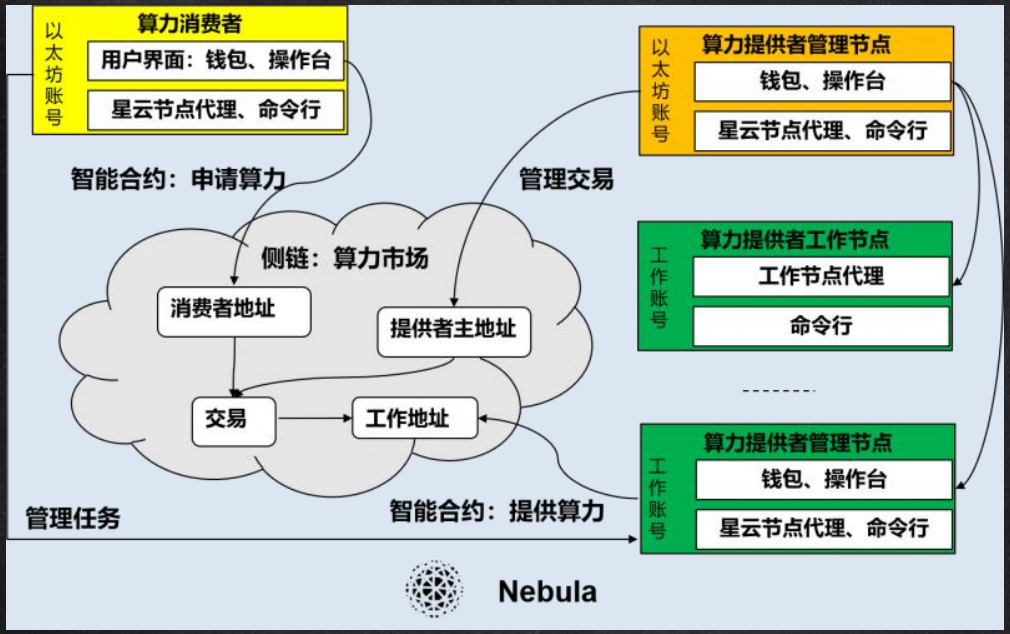
3 课程考核

4 参考资料

5 分布式系统简介

个人简介

主要方向为：**分布式系统、智能运维（ AIOps）、微服务、云计算、软件可靠性、区块链**等。近年来在国际会议和期刊共发表27篇论文，其中SCI论文6篇，中科院一区及CCF A类会议和期刊论文5篇，同时担任多个国际期刊和会议的审稿人。



课程内容

- 课程描述

本课程主要讲授分布式**系统**的相关原理、设计泛型以及前沿研究，涉及**分布式架构、网络通信、命名系统、同步、一致性和复制、共识协议、容错、安全、云计算**等相关原理及技术，同时包括**DevOps、MapReduce、智能运维等操作实践**。课程以讲授为主，实践为辅，同时需要学生阅读相关领域的文献，并进行讨论。 此外，该课程还设计了多个实践项目，以强化学生对分布式系统原理和技术的理解与应用。

- 课程目的

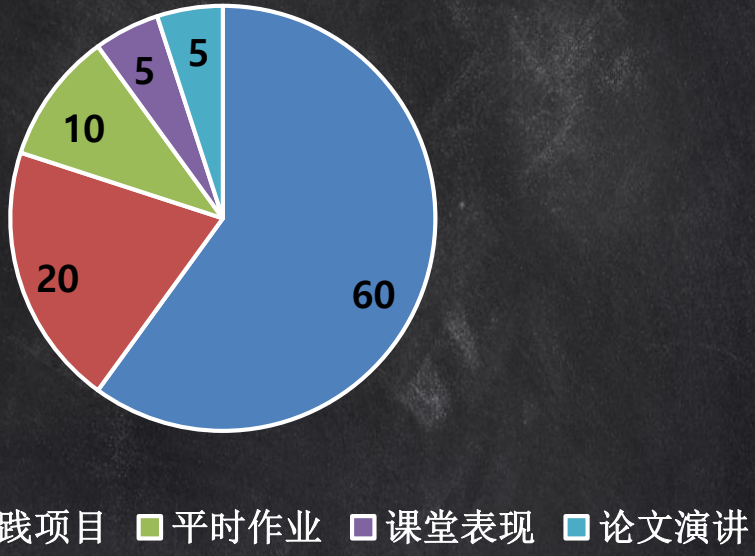
1、 了解分布式系统的相关概念和原理；
2、 熟悉分布式系统的架构设计；
3、 掌握分布式系统的编程方法；
4、 了解分布式系统的运维方法；
5、 了解分布式系统领域的前沿技术；
6、 激发学生对分布式系统的探索兴趣；

课程内容

周次	课程内容	周次	课程内容
第1周	分布式系统概述及体系结构	第11周	基于对象的分布式系统
第2周	分布式系统进程模型及通信	第12周	分布式文件系统
第3周	分布式系统命名及逻辑时钟	第13周	基于Web的分布式系统
第4周	一致性和复制	第14周	基于协作的分布式系统
第5周	互斥及选举算法、Paxos算法	第15周	移动和普适计算
第6周	分布式系统容错及可靠性保障	第16周	MapReduce编程模型
第7周	拜占庭容错机制	第17周	P2P及区块链系统原理
第8周	分布式提交	第18周	云平台架构与虚拟数据中心
第9周	安全及访问控制	第19周	答疑及考试
第10周	面向云环境的开发及测试	第20周	——

课程考核标准

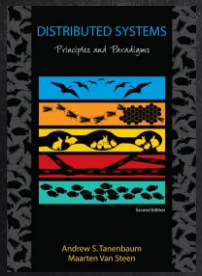
本门课程是计算机科学领域的重点课程，包括理论知识的学习和工程实践。 为了巩固知识、锻炼学生的实操能力，本门课程考核主要包括四部分：**期末考试（闭卷）、实验项目（3次）、论文演讲和平时作业（每两周一次）。** 加分项：课堂积极回答问题、按时上课等。

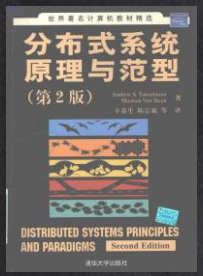


教材及参考资料

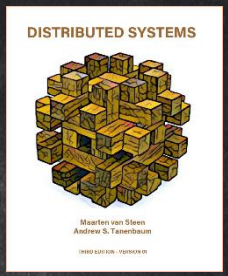
- 教材

分布式系统原理与范型, 第二版； Distributed systems and Paradigms



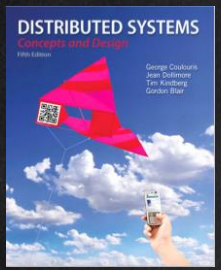


分布式系统原理与范型, 第二版

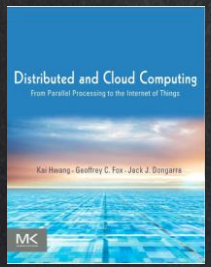


分布式系统原理与范型，英文版，第三版（2017）
https://www.distributed-systems.net/

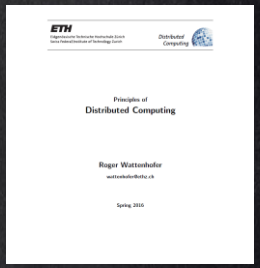
- 参考书



Distributed systems - Concepts and Design (5th)



Distributed and cloud co mputing



Principles of Distributed Computing (ETH)

教材及参考资料

➢ 参考资料

- 课程github地址：chen0031/**DistributedComputingCourse, https://github.com/chen0031/DistributedComputingCourse.git** 包含：参考书、相关论文、相关工具及相关代码；
- MIT courses: https://pdos.csail.mit.edu/6.824/schedule.html;
- CMU courses: http://www.andrew.cmu.edu/course/95-702/;
- 清华 courses: http://thu-cmu.cs.tsinghua.edu.cn/curriculum/dscourse/schedule.htm;
- 北京大学 courses: http://net.pku.edu.cn/~course/cs501/2008/schedule.html；
- Cornell courses: http://www.cs.cornell.edu/courses/cs5414/2016fa/;
- NYU courses http://www.news.cs.nyu.edu/~jinyang/fa17-ds/schedule.html http://www.news.cs.nyu.edu/~jinyang/fa17-ds/schedule.html;
- MIT OpenCourseware: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-824-distributed-computer-systems-engineering-spring-2006/;
- 分布式系统资料： https://cloud.tencent.com/developer/article/1085803；
- Awesome-distributed-systems: https://github.com/zhenlohuang/awesome-distributed-systems;
- 知乎分布式系统：https://www.somethingsimilar.com/2013/01/14/notes-on-distributed-systems-for-young-bloods/；

联系方式

分布式系统微信群

助教：余广坝
邮箱：444723257@qq.com



5 分布式系统简介

分布式系统产生的背景

计算机的小型化以及计算性能的提升

高性能主机

高性能服务器

多核高性能CPU

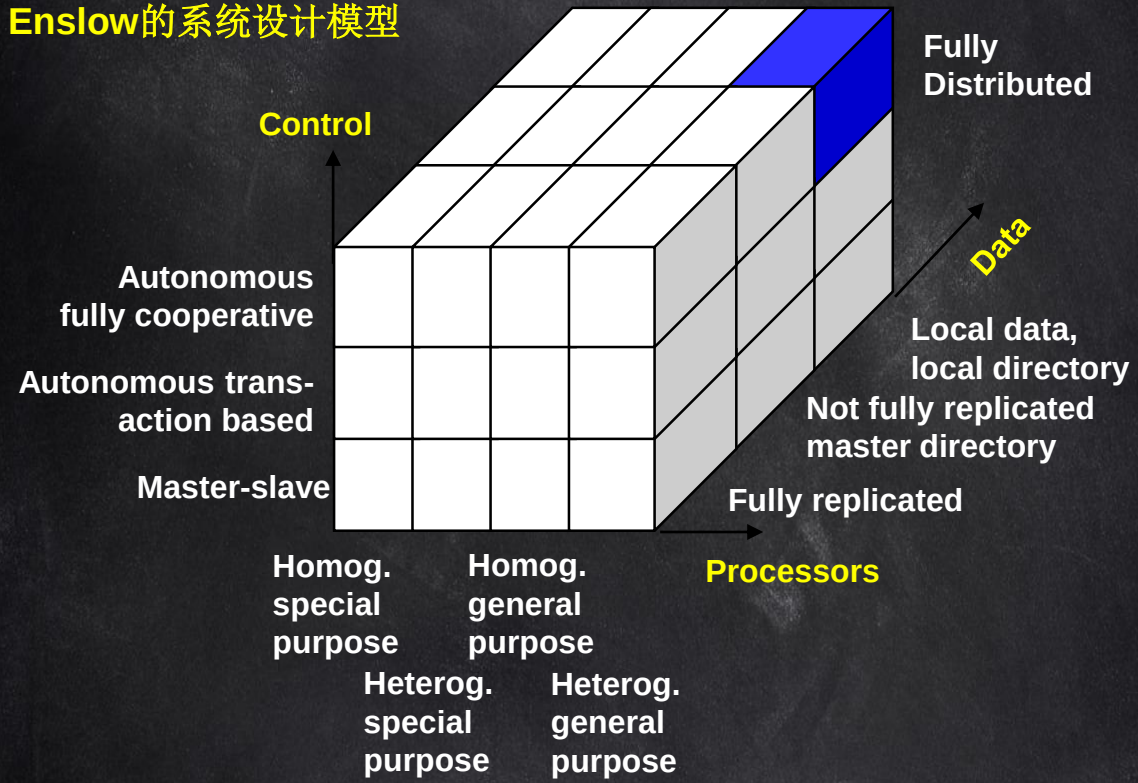
多核高性能CPU

小规模低速网络

大规模高速网络

分布式系统产生的背景

Enslow的系统设计模型



分布式系统定义

- **本书定义：**
  分布式系统是若干**独立自主计算机**的集合，这些计算机对于用于来说像是**单个耦合**系统。

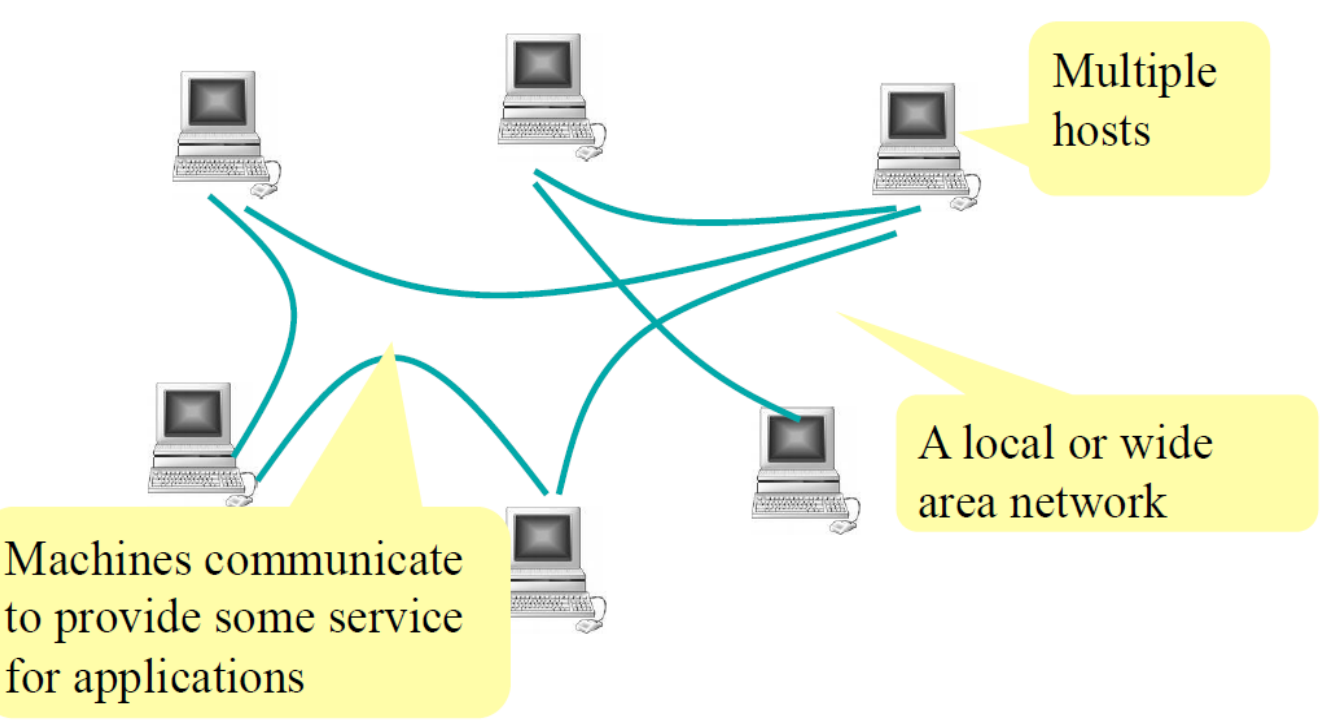
- **Leslie Lamport：** “A distributed system is one in which the failure of a computer you didn’t even know existed can render your own computer unusable.”

- **特性：**
  - 自主性：
    计算节点硬件或者软件进程是独立的；
  - 单体耦合性：
    用户或者应用程序感觉系统是一个系统——节点之间需要相互协作；

优点？

优点？

分布式系统定义

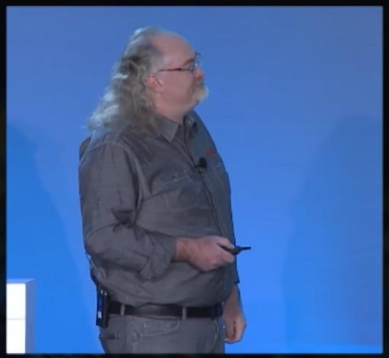

Multiple hosts
A local or wide area network
Machines communicate to provide some service for applications

一个典型的数据中心

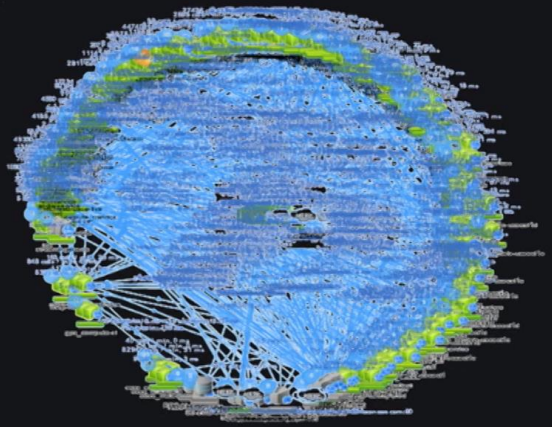
分布式系统定义

aws re:Invent

amazon web services



Netflix architecture



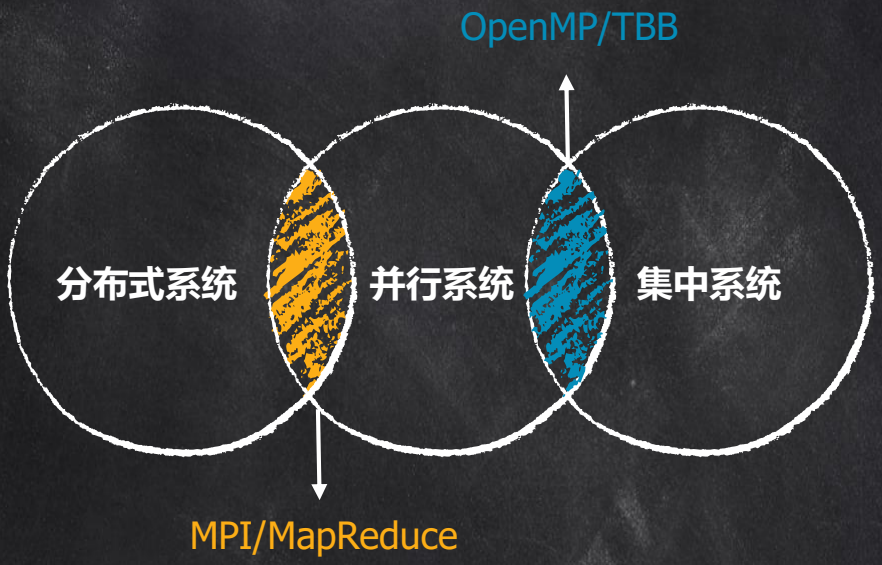
分布式系统定义

- 更多特性：
  - 构成组件并被所有用户共享；
  - 系统资源可能不允许访问；
  - 软件运行在不同处理器上的多个并发进程上；
  - 允许多点控制；
  - 允许多点失效；

分布式系统定义

- 集中式系统特性：
  - 仅由单个组件构成；
  - 单个组件被用户一直占用；
  - 所有的资源都是可访问的；
  - 软件运行在单个进程中；
  - 单点控制；
  - 单点失效；

不同系统比较

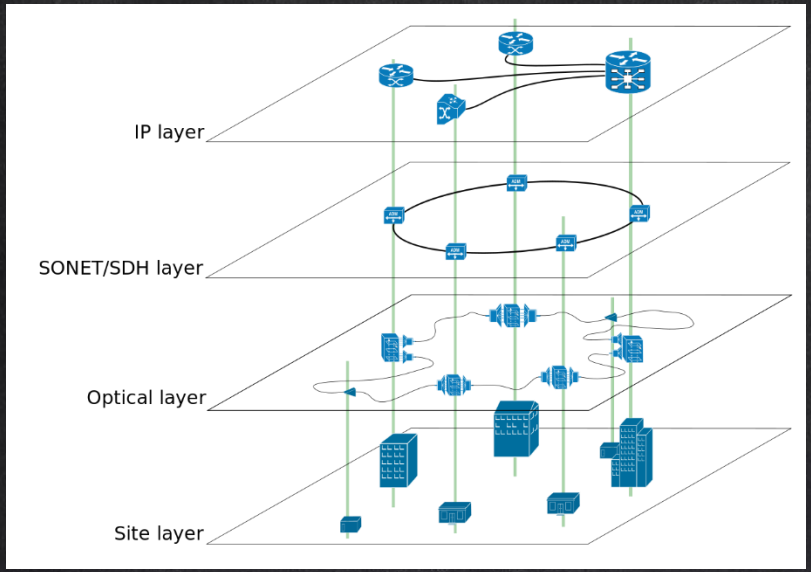


自主节点集合

- 独立行为：
  - 每个节点都是独立的，有自己的本地时间；
  - 没有全局锁；
  - 存在基本的同步和协同的问题；
- 节点集合行为：
  - 如何管理集合中的节点之间的关系？
    — 开放集合、封闭集合；
  - 如何知道确实是在跟一个授权（非授权的）成员通信？
    — 信任、安全机制；

节点之间的组织形式

- 覆盖网络（**Overlay network**）：
  - 实践表明覆盖网络最为常用；



节点之间的组织形式

- 覆盖网络（Overlay network）
  - 最佳实践表明覆盖网络最为常用；
  - 每个节点仅和邻居节点通信；
  - 邻居节点是动态的甚至只能通过查询获得；
- 覆盖网络类型
  - P2P网络（Peer-toPeer）
  - 结构型的P2P网

    节点之间的连接具有特定规则的结构；
  - 非结构性的P2P网络

    节点之间的连接具有随机和任意性；

一致(Coherent)系统

- **本质**

  节点无论在什么地方，用户无论何时访问，节点集合对于用户来讲是一个整体；

- 例子
  - ✓ 终端用户不知道计算发生在什么地方；
  - ✓ 用户也不知道与应用相关的数据存储在什么地方；
  - ✓ 数据拷贝完全是隐藏的；

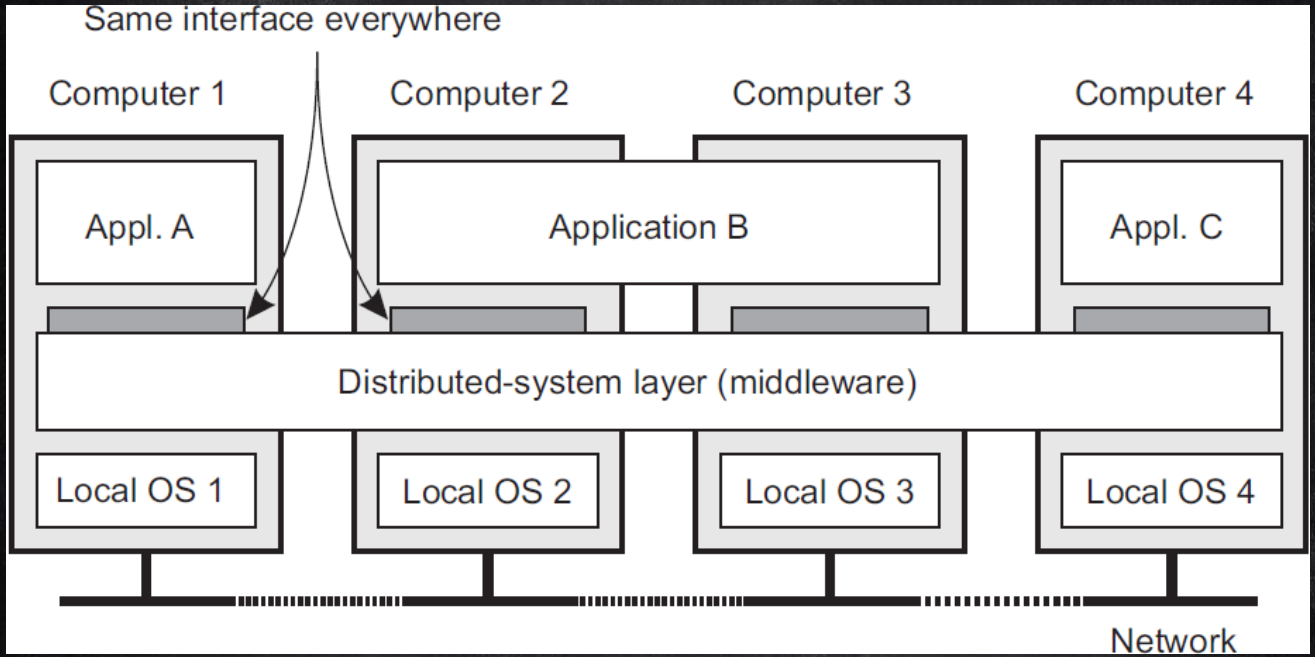
  （核心是分布式透明性）

- 挑战

  部分失效： 不可避免地，分布式系统的某一部分会失效，部分失效以及恢复很难做到对用户的透明性；

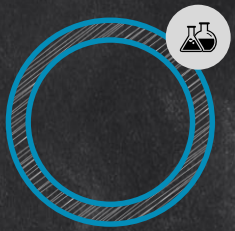
中间件（Middleware）

为了让分布式系统呈现为单个系统，分布式系统需要中间件组织起来

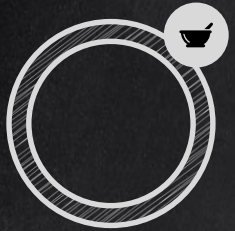


分布式系统的目标

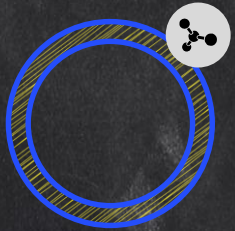
使资源可访问
让用户方便地访问资源



透明性
隐藏资源在网络上的分布



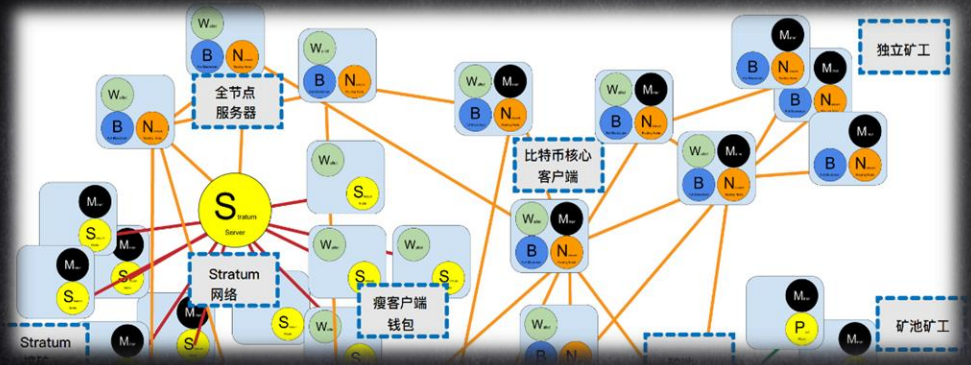
开放性
访问接口的标准化



可扩展性
系统在规模、地域、管理上的可扩展性

资源访问共享

- 典型样例
  - ✓ 基于云的存储和文件系统；
  - ✓ 基于P2P的流媒体系统；
  - ✓ 共享邮件系统（外包的邮件系统）；
  - ✓ 共享的Web支撑系统（CDN）；
  - ✓ ……



比特币网络

"**The network is the computer**"
—John Gage, then at Sun Microsystems

透明性

- 隐藏进程和资源在多台计算机上分布这一事实；
- 透明的类型

透明性	说明
访问	隐藏数据表示形式的不同以及资源访问方式的不同
位置	隐藏资源所在位置
迁移	异常资源是否移动到另一个位置
重定位	隐藏资源是否在使用过程中移动到另一个位置
复制	隐藏是否对资源进行复制
并发	隐藏资源是否由相互竞争的用户共享
故障	隐藏资源的故障和恢复
持久化	隐藏数据在主存和磁盘这一事实

透明度

- 观点

完全透明性是不可取的也是难以实现的，主要因为：

- 可能隐含通信的性能问题；
- 完全隐藏网络和节点的失效是不能的；
  - ✓ 不能区分失效和性能变慢的节点；
  - ✓ 不能确定系统失效之前的操作是什么；
- 完全的透明性可能牺牲性能，暴露系统分布特征；
- 保证复制节点与主节点的一致性需要时间；
- 为了容错需要立即将内存修改的内容同步到磁盘上；

透明度

- **暴露系统的分布有一定使用场景**
  - 利用基于位置的服务（如：找到附近的朋友）
  - 当与不同时区的用户交互时；
  - 当让用户理解系统发生了什么时，如当一台服务器不响应时，报告失效；
- **结论**

  分布式透明性是一个较好的属性，但是需要区别对待。

分布式系统的开放性

- 什么是分布式系统的开放性？

分布式系统的开放性指：系统根据一系列准则来提供服务，这些准则描述了所提供服务的语法和语义。

- 讨论分布式系统开放性的那些方面？

  - 系统应该具有良好定义的接口；
  - 系统应该容易实现互操作性；
  - 系统应该支持可移植性；
  - 系统应该容易实现可扩展性；

策略与机制

重点： 策略与机制分离

- **实现开放性：策略**
  - 需要为客户端的缓冲数据设置什么级别的一致性?
  - 我们允许下载的程序执行什么操作?
  - 当出现网络带宽波动的时候如何调整QoS需求？
  - 通信的安全水平设置多高？
- **实现开放性：机制**
  - 允许动态设定缓冲策略；
  - 支持为移动代码设置不同的信任级别；
  - 为每个数据流提供可调整的QoS参数；
  - 提供不同的加密算法；

策略与机制严格分离

- **观察**

  策略和机制之间分离的越严格，越需要设计合适的机制，这样会导致出现很多配置参数和复杂的管理。

- **策略和机制之间的平衡**

  硬编码某些策略可以简化管理，减少复杂性，但是会导致灵活性降低。没有放之四海而皆准的方法。

分布式系统的可扩展性

➢ 观察

现代分布式系统的很多开发人员经常使用“可扩展性”，但不清楚为什么要进行扩展。

➢ 三个方面的扩展性

- 规模可扩展性：用户数量和进程数量增加；
- 地理可扩展性：节点之间的最大物理位置；
- 管理可扩展性：管理域的数量；

➢ 观点

- 大部分系统关心的是规模可扩展性；
- 解决方案：多个服务器独立并行运行；
- 地理可扩展性和管理可扩展性仍然充满挑战；

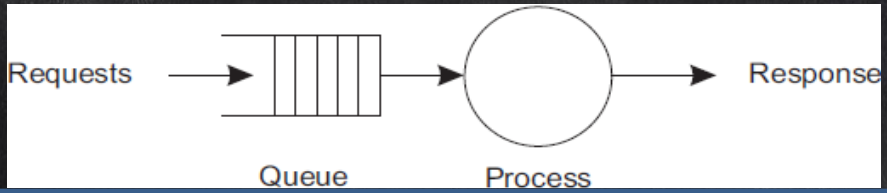
规模可扩展性

- 集中式解决方案可扩展性问题的根因
  - 计算容量受到CPU性能的限制；
  - 存储容量局限，包括CPU与磁盘之间的传输速率；
  - 网络局限：用户与集中服务之间的网络带宽；

概念	实例
集中式服务	供所有用户访问的单个服务器
集中式数据	单个在线电话簿
集中式算法	根据完整信息来安排路由

形式化分析

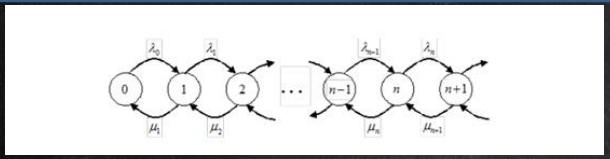
一个集中的服务可以建模为一个简单的排队系统（M/M/1）



假设与定义

- 队列具有无限长度；
- 到达率为：λ
- 服务率为：μ

系统中包含 k 个请求的概率为：



$$p_k = (1 - \frac{\lambda}{\mu})(\frac{\lambda}{\mu})^k$$

形式化分析

Utilization U of a service is the fraction of time that it is busy

$$U = \sum_{k>0} p_k = 1 - p_0 = \frac{\lambda}{\mu} \Rightarrow p_k = (1-U)U^k$$

Average number of requests in the system

$$\overline{N} = \sum_{k\geq 0} k \cdot p_k = \sum_{k\geq 0} k \cdot (1-U)U^k = (1-U)\sum_{k\geq 0} k \cdot U^k = \frac{(1-U)U}{(1-U)^2} = \frac{U}{1-U}$$

Average throughput

$$X = \underbrace{U \cdot \mu}_{\text{server at work}} + \underbrace{(1-U)\cdot 0}_{\text{server idle}} = \frac{\lambda}{\mu} \cdot \mu = \lambda$$

形式化分析

- 响应时间：处理一个请求的总的时间

$$R = \frac{\overline{N}}{X} = \frac{S}{1-U} \Rightarrow \frac{R}{S} = \frac{1}{1-U}$$

- 观点
  - 如果 U 较小， 响应时间趋近于1；意味着服务被立即处理；
  - 如果U增加到1，系统处于挂起的状态；应该减少S；

地理可扩展性的问题

- 不能简单从LAN扩展到WAN：很多分布式系统假设客户端-服务器之间的交互是同步的即客户端发送请求等待结果。广域环境中的延迟问题限制扩展性。

- WAN中的连接常常是不可靠的：简单地将流视频从LAN移动到WAN会导致失效；

- 缺少多点通信，导致一个简单的搜索广播不能执行。解决方案是将命名服务和目录服务分离。

管理可扩展性的问题

- **本质**
  与使用方法、管理和安全相关的策略冲突；
- 例子
  - ✓ 计算网格：在不同的域之间共享昂贵的计算资源；
  - ✓ 共享设备：如何控制、管理、使用共享无线望远镜等？
- 例外
  - 文件共享系统（BitTorrent）；
  - P2P电话（Skype）；
  - 基于P2P的音频流数据（Spotify）；

扩展技术

- **隐藏通信延迟**

  基本想法很简单：尽量避免等待远程服务对请求的响应。

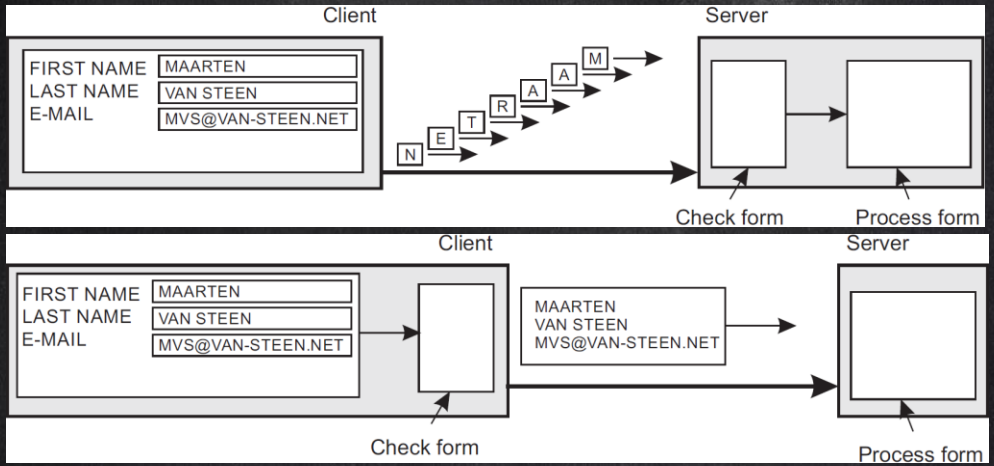
- 利用异步通信技术；
- 设计分离的响应消息处理器；
- 问题：并不是所有应用都适合这种模式
- 某些交互式应用的用户发出请求后，处于等等无所事事状态

扩展技术

- **隐藏通信延迟**

  基本想法很简单：尽量避免等待远程服务对请求的响应。

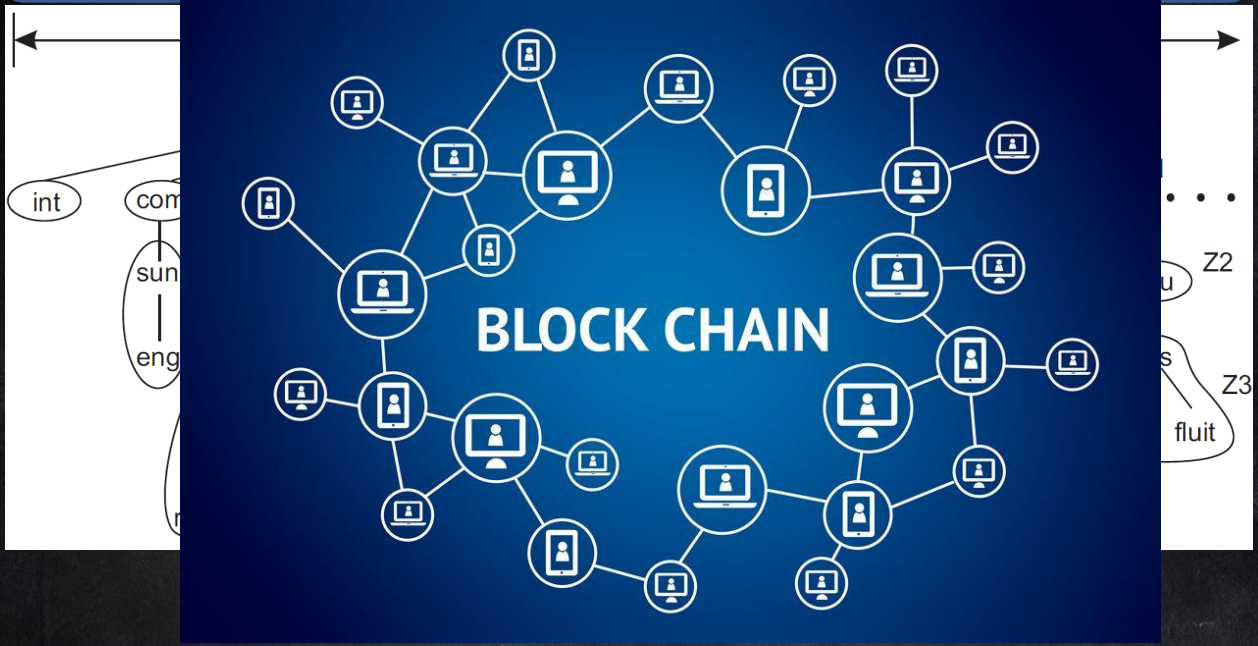
- 将计算从服务器端移动到客户端



扩展技术

- 分布

  在多个机器上划分数据和计算。



扩展技术

- **复制**

  复制和缓存：在多个不同的机器上创建多个数据副本。

复制的重要作用？

- 复制文件服务器和数据库；
- Web站点进行镜像；
- Web缓存（在浏览器或者代理位置）；
- 文件缓存（在服务器和客户端）；

扩展技术：复制存在的问题

- 复制的使用简单，但是存在一致性问题：
  - 设计多个副本（缓存或者复制），导致不一致：修改一个副本后会让该副本与其他副本内容不一致；
  - 总是保证副本的一致性需要对每次修改进行全局同步；
  - 全局同步由于其高昂的代价导致难以应用到大规模的解决方案中。
- 观察：

  如果可以容忍不一致，我们可以减少全局同步，但是这由应用程序决定。

构建分布式系统：陷阱

- 观察

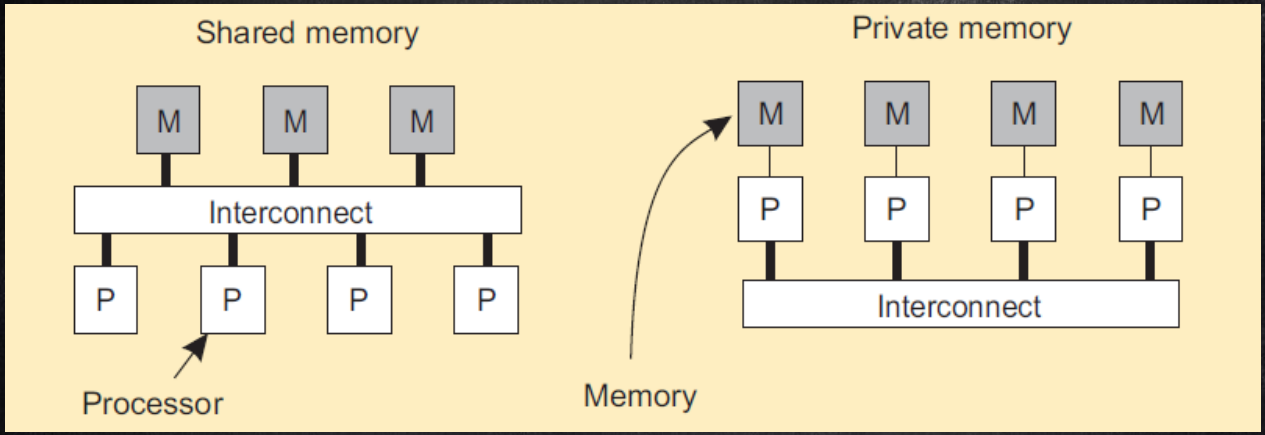
  很多分布式系统没有设计充分，出现错误的假设，包括：

- 网络是可靠的；
- 网络是安全的；
- 网络是同构的；
- 拓扑关系不发生变化；
- 响应延迟为0；
- 带宽是无限的；
- 传输代价为0；
- 仅存在一个管理员；

三种类型分布式系统

- **观察**

  高性能的分布式计算开始于并行计算。



多核、多处理器 VS 多计算机

分布式共享内存系统

- **观察**

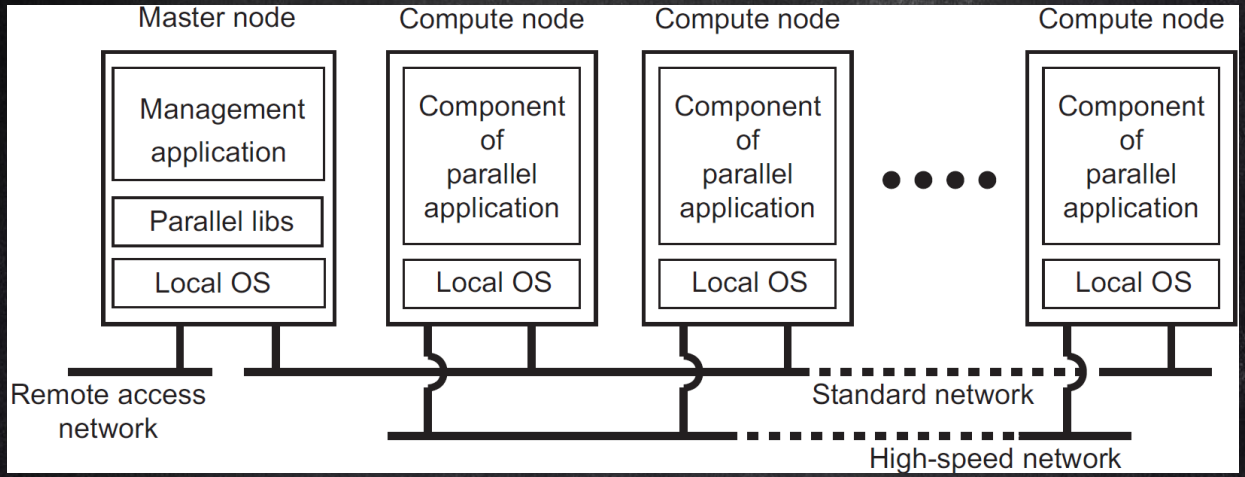
  多处理器与多计算机的系统相比，编程相对简单，然而随着处理器或者核心数的增加也会遇到各种问题。方案：在多计算机的基础上实现共享内存的模型。

- 利用虚拟内存技术的例子
  - 将所有的内存页映射到单个虚拟地址空间；
  - 如果处理器 A上的进程访问处理器B的内存页面P，OS A会陷入并从B上获取页面P，就像页面P位于本地磁盘上。

- 问题
  - ✓ 分布式共享内存的性能难以与多处理器性能相媲美；
  - ✓ 已经被丢弃；

集群计算系统

- 集群计算系统本质上是通过LAN连接起来的高端计算系统
  - 同构： 相同的OS， 近乎相似的硬件；
  - 单个管理节点；



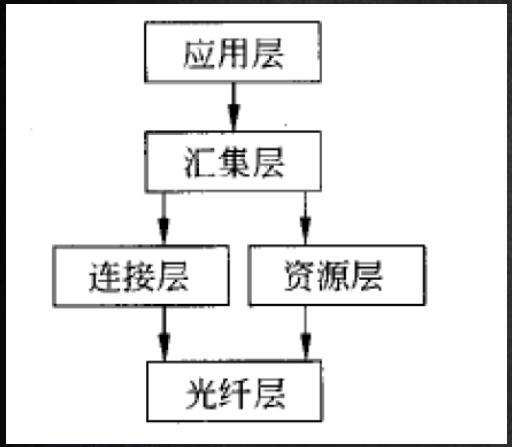
网格计算系统

- 由各地的节点构成的系统
  - 异构；
  - 包含多个组织；
  - 容易扩展到广域网的环境中；

- 要点

  为了允许合作，网格通常利用虚拟组织。

网格计算系统架构



分层

- 光纤层

  提供逻辑资源的接口；
- 连接层

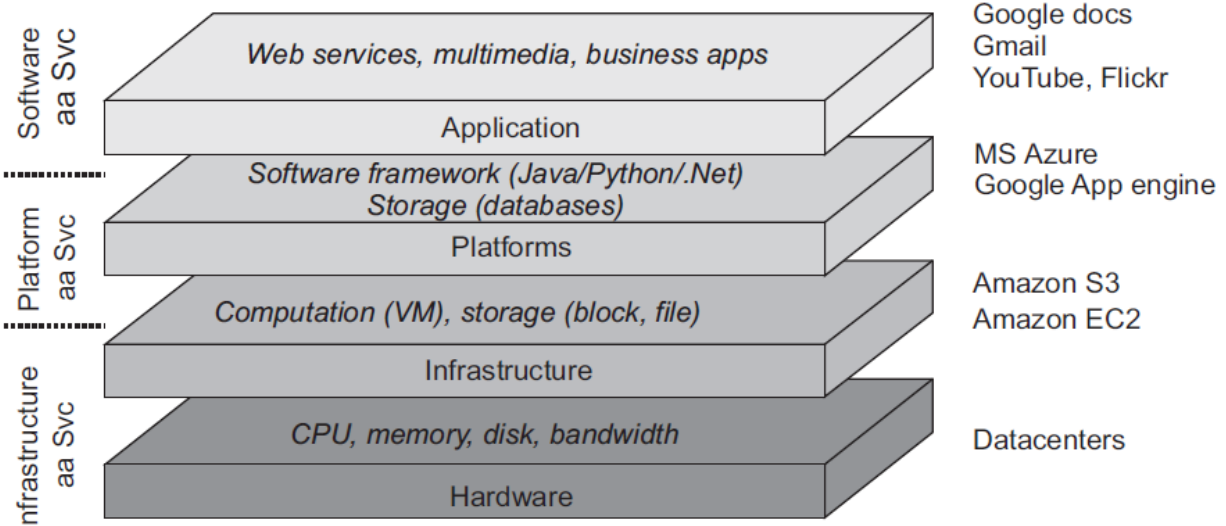
  通信/传输协议，以及授权协议；
- 资源层

  管理单个字元，如创建进程或者读数据；
- 汇聚层

  处理访问多种资源：发现、调度和复制 ；
- 应用层

  包含在同一个组织下的实际的网格应用；

云计算



云计算是否可以减少成本？

云计算

分为四层：

- 硬件层

  处理器、路由、电源和冷却系统，普通消费用户不可见；

- 基础设施层

  部署虚拟化技术，管理虚拟存储、计算、网络；

- 运行平台层

  平台为存储等资源提供高层抽象，例如Amazon S3提供API用于存储和访问数据；

- 应用程序层

  实际的应用程序，例如Office办公软件等；

集成应用

➢ 场景

- 组织面临很多网络应用，但是完成互操作却很复杂；

➢ 基本方法

- 网络应用以服务的形式运行允许远端的客户端访问；
- 简单集成：客户端合并请求，收集请求结果；

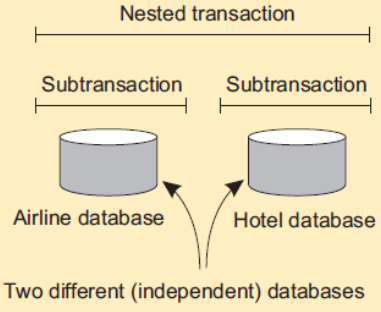
➢ 下一步

- 允许应用程序之间进行通信，这就产生了企业应用集成系统（EAI）

EAI：事务系统

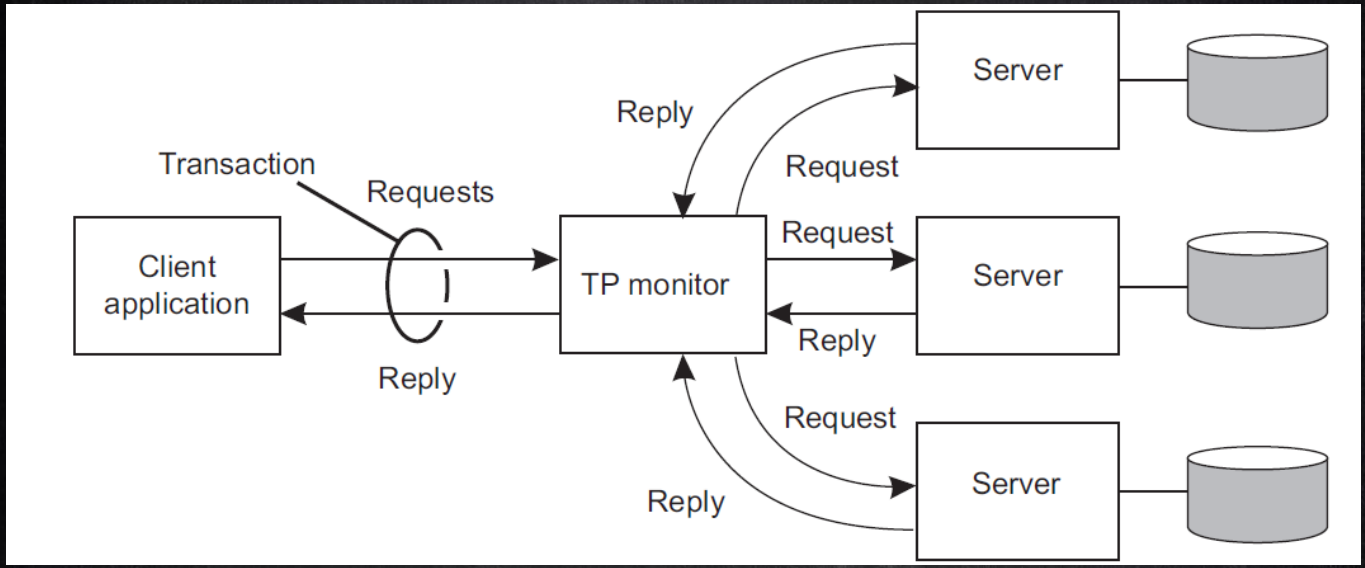
原函数	说明
BEGIN_TRANSACTION	标识一个事务处理的开始
END_TRANSACTION	终止事务处理并试图提交
ABORT_TRANSACTION	杀死事务处理并恢复旧值
READ	从文件、表或其他地方读取数据
WRITE	往文件、表或其他地方写入数据

- 特点：**all-or-nothing**



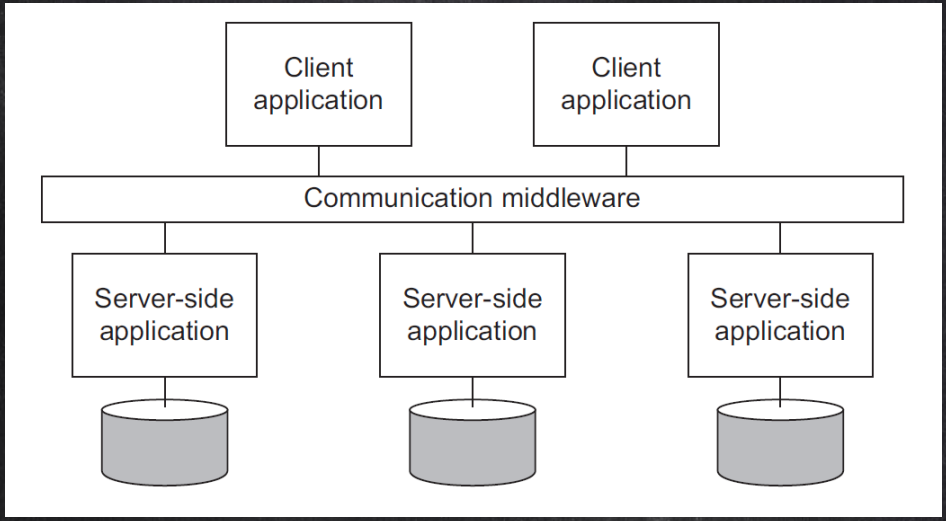
- **Atomic**: happens indivisibly (seemingly)
- **Consistent**: does not violate system invariants
- **Isolated**: not mutual interference
- **Durable**: commit means changes are permanent

TPM：事务处理监控器



- 观察
  - 在很多场景中，事务所涉及的数据是分布在多个服务器上
  - TPM负责协调事务的执行；

中间件和EAI



- 中间件为系统集成提供通信设施
- RPC：请求通过本地调用发出，结果以函数返回值的方式接收；
- 面向消息的中间件（MOM）：Pub, Sub。

如何集成应用

- 文件传输：
  技术实现简单，但是不够灵活；需要了解文件的格式和部署方式，了解文件的管理方法，更新传播和更新通知；
- 共享数据库：
  更加灵活，但是仍然需要通用的模式，导致出现瓶颈；
- 远程过程调用：
  当需要执行一系列的行为时非常有效，但是需要caller和callee同时在线；
- 消息传递：
  允许caller和callee在时间和空间上解耦；

分布式普适系统（分布式嵌入系统）

- 观察

  近年来涌现出了下一代分布式系统，特点：节点较小、移动、嵌入型，常与用户的环境融合在一起；

- 三种类型
  - 普适计算系统

    普适、连续计算，与用户连续交互；
  - 移动计算系统

    普适、计算设备是移动的；
  - 传感网络

    普适、强调与环境的感知和作用；

普适计算系统

- 分布式

  设备是通过网络连接、分布并且是透明访问的；

- 交互

  用户和设备之间的交互是高度隐蔽的；

- 上下文可感知

  系统知晓用户的上下文以便于优化交互行为；

- 自主性

  设备自主运行，不需要人为干预，因此具有高度的自管理能力；

- 智能

  系统作为一个整体可以处理一系列的动态行为和交互；

移动计算系统

- 特征：

- 大量的不同的移动设备（智能手机、平板、GPS设备、遥控器等）；
- 移动预示着设备的位置随着时间的变化而变化 => 本地服务、可连接性的变化，关键词：“发现”；
- 设备之间的通信变得很困难：没有稳定的路由，而且没有可保证的连接性，这也就要求网络连接可容错；

移动计算模式

- 问题：
  - 信息传播与人们的移动之间的关系是什么？基本的观点是：触碰型的信息交互（口袋交换网络）；
- 比较成功的策略：
  - Alice的世界包含朋友和敌人；
  - 如果 Alice想要发送消息到Bob：Alice首先将消息传递到她的所有的朋友；
- 观察：
  - 的朋友将消息传递给Bob；
  - 这个策略可以奏效是因为朋友之间会行程紧密的社区；

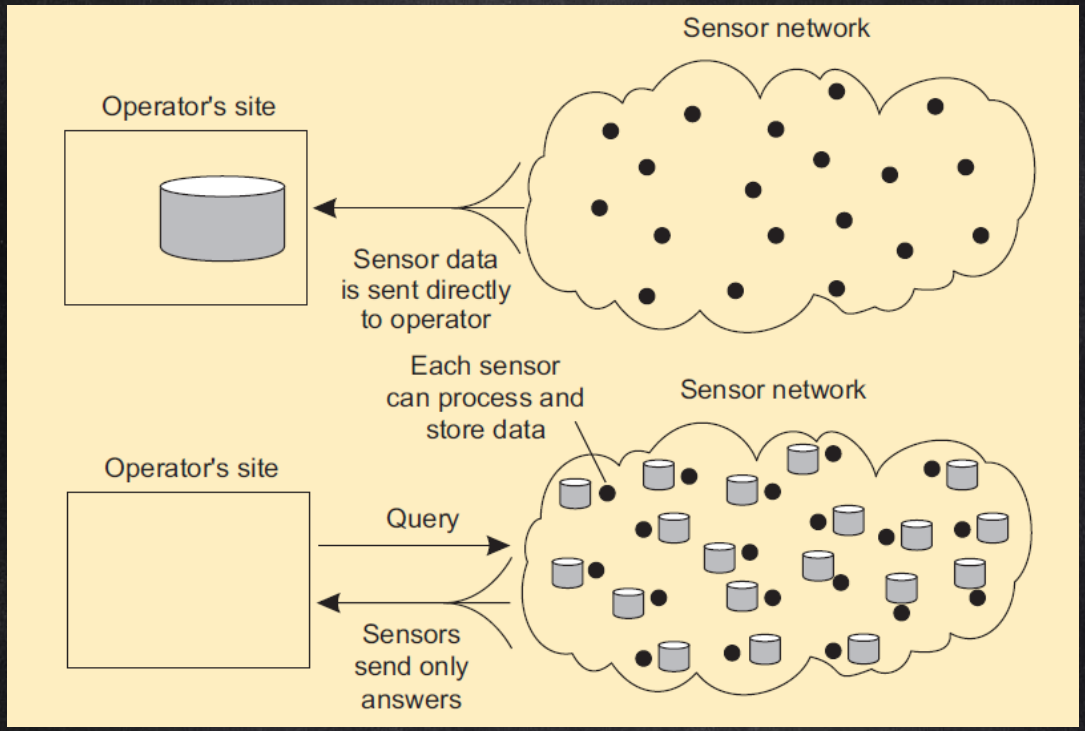
传感网络

➢ 特点：

- 这节点数量众多（10~1000）；
- 简单（内存小、计算能力低、通信效率低 ）；
- 通常是电池驱动的（甚至没有电池）；

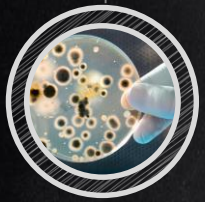
Samsung SmartHome

传感网络作为分布式数据库



两个极限

分布式系统面临的挑战

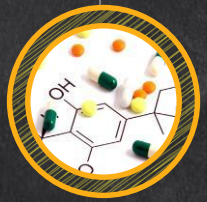


系统设计

—正确的接口设计和抽象；
—如何换分功能和可扩展性；

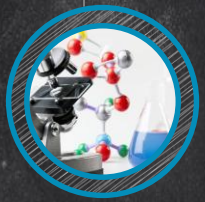
一致性

—如何一致性共享数据；



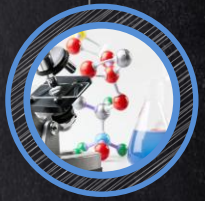
容错

—如何保障系统在出现失效情况下正常运行；



不同的部署场景

—集群；
—广域分布；
—传感网络；



实现

—如何最大化并行；
—性能瓶颈是什么；
—如何平衡负载；

总结

- 分布式系统是由自主计算机组成的系统，但是对用户看来是一个单一耦合的系统；
- 优点 1： 分布式程序集成；
- 优点 2： 扩展性好；
- 为了屏蔽分布式系统的底层异构性，系统设计与实现面临诸多挑战；

C.A.P. ?