**软件架构与中间件-（作业4）**

学号：L170300901 学生：卢兑玧

分散系统

分散系统的定义

分布式系统( distributed systems)的核心是多个计算机在互联网上相互通信合作，以处理相同的测试。例如，巨大的网站存储库、地图(MapReduce)等大数据计算、peer-to-peer(P2P)文件共享等。在当今时代，基础设施(infrastructure)为了做一件事(job)，需要多个电脑。因此，人们开始研究这些方法。 如果大家可以用一台电脑来解答某些问题，就没有必要建立分散系统了。实际上，用一台电脑也能做的事情真的很多。 分散系统不容易。因此，大家最好研究用一台电脑完成作业。

分布式系统的原因

high-performance

人们构建分散系统的原因是，通过多个电脑的合作，可以展示high-performance。 为此需要paralleism这个概念。 这意味着许多CPU、 许多磁盘、 内存等并联连接在一起 。

fault tolerance

构建分散系统的另一个原因是可以忍受缺点（tolerate faults 当两台电脑做同样的事情时， 其中一台失败时， 另一台可以覆盖。

Physical reason

当服务器处于物理距离时， 也会使用 。 比如说，如果叫银行服务器，一台可以部署在纽约，另一台也可以部署在伦敦。

security

存在安全隐患时，可根据不同的权限、协议进行不同的配置。 即，可以单独组成孤立的(isolated)电脑。

分散系统困难的原因

有很多部分，同时实行。 因此，发生某种问题时，电脑之间会有复杂连接的部分，也有依赖时间（timing dependent stuff）。 分散系统困难的另一个原因是分散系统有很多部分和网络，大家会遇到很多意想不到的失败。 如果用一个计算机操作， 则运行良好， 分散系统可能会发生Crash， 或者因为未知原因而失败 。 您在分布式系统中的许多计算机中， 某些部分会成功， 有些部分会失败。 这种局部失败（partial failure）是分散系统困难的另一个原因。 （challenges concurrency partial failure）分散系统之所以困难，是因为使用1000台电脑并不意味着1000倍的性能提高。 以前用一个个人电脑建立网站没有问题，但是网站越来越大，为了储存和处理大量数据，经过研究，分散系统在当今时代成为了非常重要的基础设施结构。 关于分散系统，有很多部分已经解决，但还有很多部分尚未解决，需要研究。 如果大家想构建分散系统的话，这个课程会是个很好的课程。

实现分布式系统所需的( implementation)

1. Remote Procedure Call(RPC)

2. Threads

3. Concurrency Control

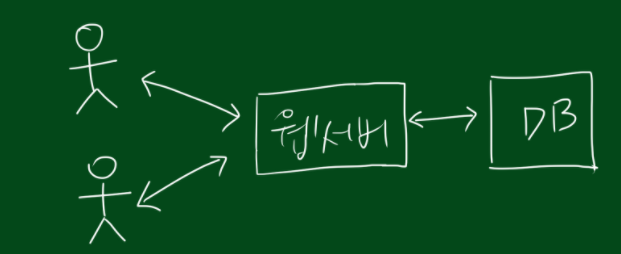
性能(performance)

- 扩展性( Scalability)

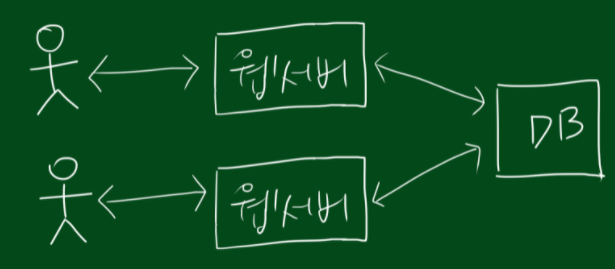
通常分散系统的高目标(high-level goal)是scalable spped-up。 具体来说，为了解答什么问题，用一个电脑需要一个小时。 如果我买第二台电脑，那么答题的时间就会减少一半，期待30分钟就能答完。

2 × computers -> 2 × hroughput

这个方法很便宜，就是多买一台电脑来解决问题。 如果不想再买电脑，要想解决问题，就要雇佣更有能力的程序员。



一两个人接触网络服务器时，一台电脑就足够了。 但是几百万人登录时该怎么办呢？ 简单增加网页服务器，将用户分成下图，让用户连接到不同的网页服务器即可。



当然，用户需要看同样的画面，因此网页服务器需要看同样的DB。 但如果在上述系统中无限增加网页服务器，网页服务器在读取数据库数据的过程中会产生瓶颈。 因此DB也要一起增加。

Fault Tolerance

用一台普通电脑组成服务器时，不用Crash，可以坚持一年左右。 但是，如果大家用1000台电脑进行分散处理，一天会出现3~4台左右的故障。 也就是说，每天都有电脑出故障。 因为出现Crash，未知的错误，或者变慢等原因。 各个系统都连接了非常多的网络电缆和开关。 电缆连接出错， 冷却开关的 fan 可能因故障发热或发生各种问题。

Availability

即使整个系统出现故障，服务也不能停止。 即使一个服务器出现故障， 其他服务器也必须代替它 。

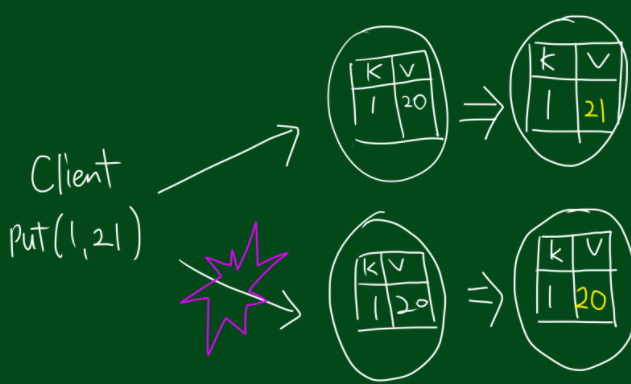
Recoverability

即使服务有任何问题， 恢复后仍然需要中断的元素( component) 。 应该维持最新的唱片。

为了以上两个目标， 最重要的是非挥发性存储空间( Non-volatile storage) 和复制( Replication

Consistency

在仓库里存一对钥匙吧。 即Put(k, v)。 那么用户可以通过密钥获得值。 这个就叫get->v吧。 默认密钥值服务( key- value service) 非常有用 。



看看上面的画吧。 如果有两台服务器，各有一张钥匙桌。 这时身高1，价格20。 但是用户想要将密钥值从 (1,20) 改为 (1, 21) 。 当然两台服务器都要变更。 如果用户成功请求到服务器1， 但请求到服务器2 出错， 服务器2的密钥值将无法更新 。 在这种情况下，如果服务器1在转弯处被服务器2取代，会怎么样呢？ 密钥值的一致性将被取消。 这种情况不会consistency。 相反，strong consistency的意思是能够看到最新的数据，weak consistency的意思是无法保证能够看到最新的数据。 而且维持strong consistency需要很多费用。 这个难的理由就当是服务器离得很远吧。 一个在亚洲，一个在美洲，就当一个地方发生地震而发生故障吧。 在这种情况下，电波移动的速度非常短，在这种情况下维持strong consistency需要很多费用。

地图编辑器( MapReduce)

地图编辑器是谷歌在2004年设计的系统。 他们本应该用天字节巨大的数据来计算的。 为计算整个网站最重要的页面是什么， 创建了所有网站内容的索引。 对网页创建索引的意义等于对所有数据进行排序 。 假设这个电脑就按部就班。 也许需要几天，几个月几年的时间。 工程师们不能坐以待毙。 所以诞生了地图教程。

地图概念

地图是框架。 对每个Infoot文件适用map function。 output是身高值对。 举个简单的数字的例子。 那么地图函数的结果就是（单词，1）形态。 映射框架的一个重要概念是将来自所有 map 的实例汇总在一起。 看下面的图片，你会很快理解的。 这是MapReduce Job。 此时对一个input文件的处理称为task，如下图所示，地图整体称为Job。

