2 分布式系统架构

Author： 中山大学 17数据科学与计算机学院 YSY

https://github.com/ysyisyourbrother

2 分布式系统架构
- 体系结构样式
- 系统体系结构
  - 集中式体系结构
    - 分层分布式系统
      - 两层体系结构
      - 三层体系结构：
  - 非集中式体系结构
    - P2P系统
      - 结构化P2P系统
      - 非结构化P2P系统
      - 超级对等节点

体系结构样式

样式包含以下几点：

1. 组件具有定义良好的接口
2. 组件之间互联的方式
3. 组件之间交换的数据
4. 组件以及连接器是如何协同配置的

连接器：

连接器提供了一种**机制**，在组件之间传递通信、使组件相互 协调和协作。

根据组件和连接器的使用，可以出现不同的配置，从而划分 不同的体系结构，包括：

1. 分层体系结构；
2. 基于对象的体系结 构；
3. 以数据为中心的体系结构；
4. 基于事件的体系结构

系统体系结构

确定软件组件、这些组件的交互以及它们的位置就是软件 体系结构的一个实例，称为系统体系结构

集中式体系结构：整个系统包含一个控制中心，协同系统的运行

非集中式组织结构 ： 系统没有一个整体的控制中心，各个节点独立自主运行

混合组织结构： 系统中既包含集中式结构也包含了非集中式结构；

集中式体系结构

分层分布式系统

两层体系结构

Client/Server模式：

- Server进程 实现特定服务的进程；
- Client进程 通过往服务器发送请求来请求服务、然后等待服务器回复的进程；（请求响应模式）

三层体系结构：

1. 用户接口层：

   用户接口层包含系统与用户直接交互的单元；例如：显示 管理

2. 应用处理层

   包含应用的主要函数，但是，不与具体的数据绑定

3. 数据层

   数据层管理应用使用的实际数据

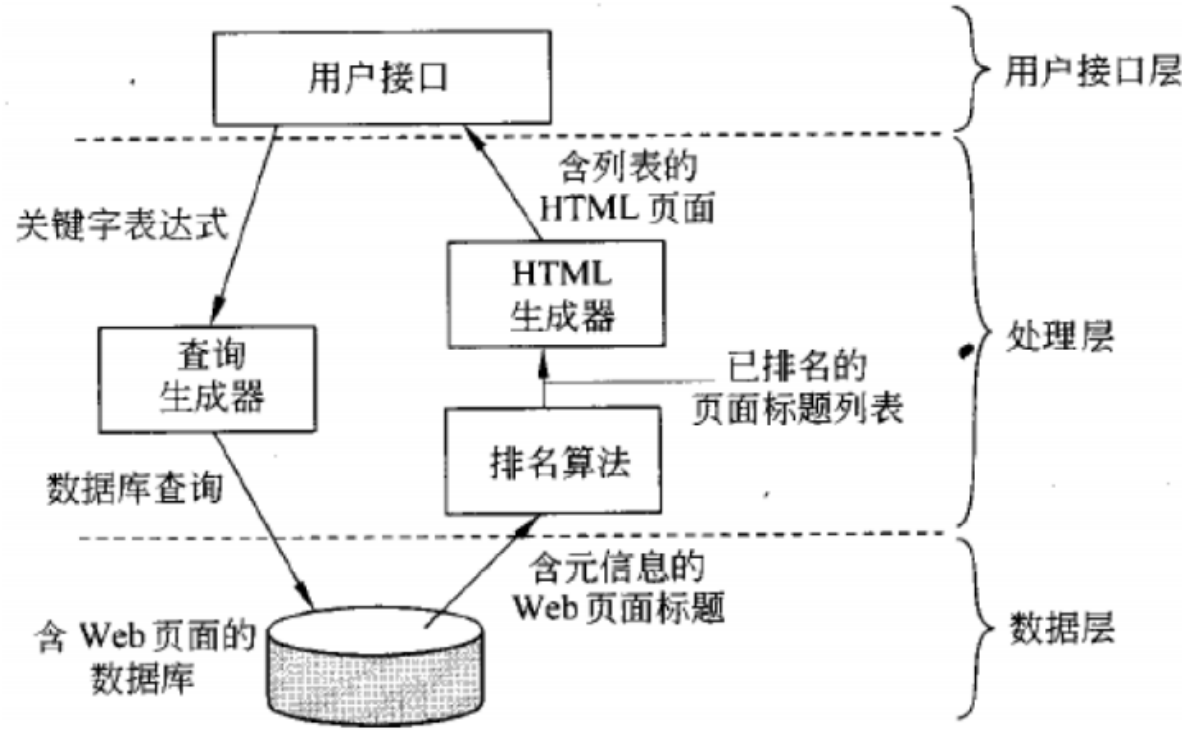
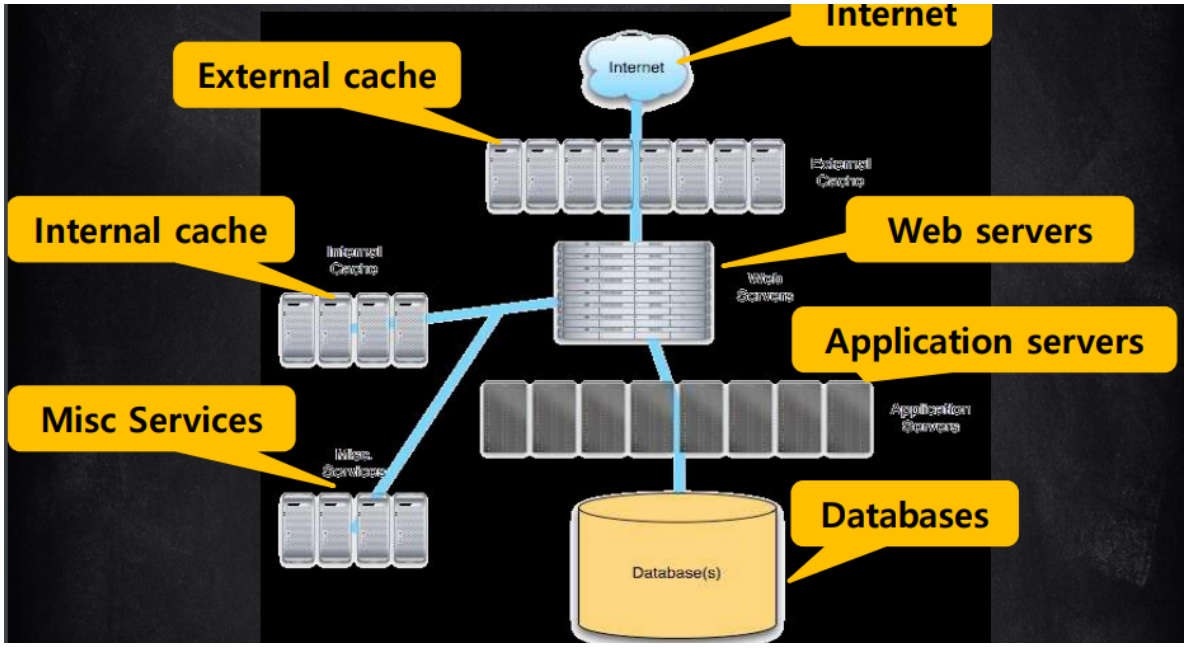


图 2.4　因特网搜索引擎简化成三个不同的层

有些服务端同时成为客户端，演变出三层甚至多层架构

真实的系统架构为：



非集中式体系结构

主要类型分为：

1. 垂直分布： 系统逻辑分层，不同层次分布于不同的机器上
2. 水平分布： 客户或者服务器在物理上分成逻辑上相等的几个部分，每个部分 相对独立，且分布在不同的机器上
3. 点对点系统： 水平分布，构成点对点的系统的进程完全相同（既是客户端又是 服务器、无中心化系统）

P2P系统

P2P系统主要分为以下几种类型：

1. 结构化P2P系统： P2P系统中的节点按照特定的分布式数据结构组织
2. 无结构化P2P系统： 节点的邻居是随机选择的
3. 混合P2P系统： 系统中的某些节点具有良好的组织方式

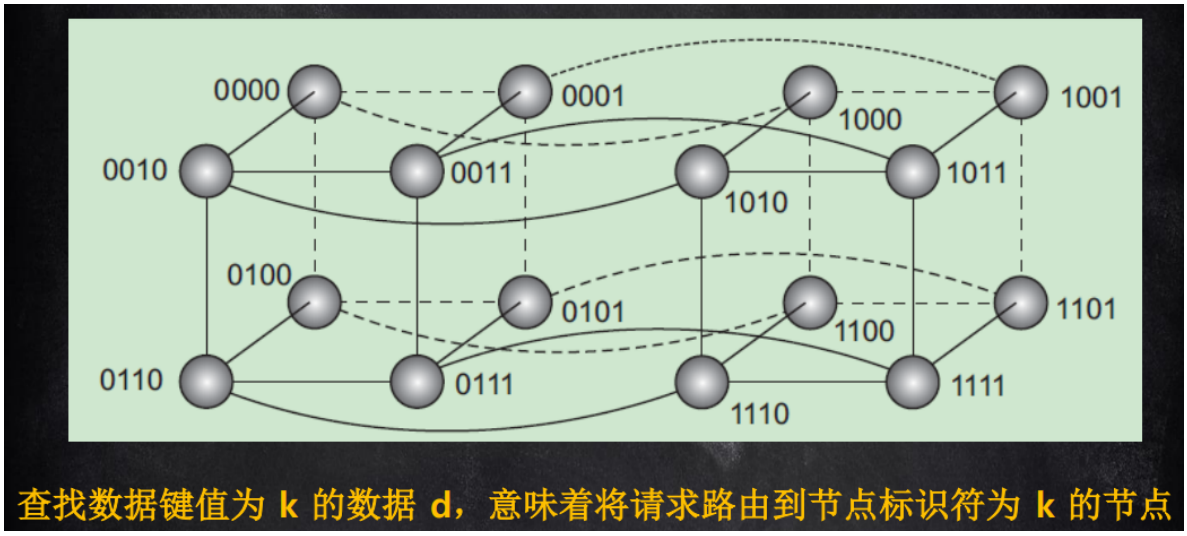
在所有的P2P场景中，我们都会讨论“覆盖网络”：数据通过节点之间的连接传输，与应用层的**广播**类似。

结构化P2P系统

将节点组织在一个**特定结构的覆盖网络**中，如**环形结构**，让这些节点根据自己的ID提供相应的职能

利用Hash函数 key(数据项) = hash(数据项的值)

P2P系统用于存储（key, value）对



chord结构，详情见第五章

非结构化P2P系统

本质

- 每个节点维护一张动态随机的含有c个节点的邻接表，目标是构建一个随机图
- 边<u,v>存在的概率为P(<u,v>)
- 部分视图 ： 从当前节点集合中随机选择出来 c 个“活”节点
- **节点 P 周期性地从部分视图中选择节点 Q**， 并且交换信息

部分视图构造算法

主动进程推送到对等节点 c/2 + 1 个表项；被动进程推送同样数 量的表项给对等节点

```
主动线程的动作（周期性重复）：
从当前的部分视图中选择一个对等体 P
if PUSH_MODE{
    mybuffer=[(MyAddress,0)];
    permute partial view;
    move H oldest entries to the end;
    append first c/2 entries to mybuffer;
    send mybuffer to P;
} else {
    send trigger to P;
}
if PULL_MODE {
    receive P's buffer;
}
从当前视图和 P 的缓冲区构建一个新的部分视图；
增加新的部分视图中每个项目的年龄
```

(a)

```
被动线程的动作：
从任意进程 Q 中接收缓冲
if PULL_MODE{
    mybuffer=[(MyAddress,0)];
    permute partial view;
    move H oldest entries to the end;
    append first c/2 entries to mybuffer;
    send mybuffer to P;
}
从当前视图和 P 的缓冲区构建一个新的部分视图；
增加新的部分视图中每个项目的年龄
```

(b)

P2P系统数据搜索方式

- 泛洪：请求发出节点 u 会其向其所有邻居节点发出数据搜索请求。 如果之前已收到节点请求，则请求会被忽略（环检测）。否则，节点会在本 查找数据项。 请求会迭代传递下去
- 随机游走： 请求发出节点 u 从其邻居节点中随机选择一个节点，然后进 行本地搜索。如果没有完成，继续从其邻居节点中选择一个随机 节点向下进行，一般需要一个终止

超级对等节点

非结构化点对点系统中，随着网络的增大，相关数据的定位非常困难。不存在一种确定的方法能把请求路由到特定的数据项，结点所能借助的唯一技术就是泛洪，不过泛洪的危害有很多。作为替代方法，很多点对点系统提出了使用**能够维护数据项索引的特别结点**。

核心思想：

1. 非结构化的点对点系统中，数据项的定位非常困难
2. 打破点对点网络的对称性
3. 设置代理程序

超级对等节点：能维护一个索引或充当一个代理程序的结点。 在非结构化的P2P进行搜索时，索引服务器会提高性能， 通过代理程序可以更加有效地决定在哪里放置数据

所有往来于常规对等体的通信都是先通过与该对等体关联的超级对等体。一个对等体只要加入到网络中它就只归依某个超级对等体。

更好的方法是，让客户归依的超级对等体维护着客户感兴趣的文件索引。如果客户发现了一个更好的超级对等体，可以改变当前关系。