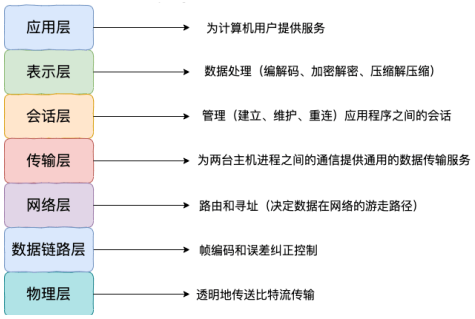
1、什么是负载均衡？

负载均衡指的是将用户**请求分摊**到不同的服务器上处理，以**提高**系统整体的**并发处理**能力以及**可靠性**。例如，系统的商品**服务在不同的服务器上部署**了多份，我们可以使用**负载平衡**对访问商品服务的**请求进行分流**。

2、服务端负载平衡

服务端负载均衡主要应用在**系统外部请求**和**网关层**之间。

根据 OSI 七层模型，服务端负载均衡分为：二层负载均衡、三层负载均衡、**四层**负载均衡、**七层**负载均衡。



1）四层负载均衡：工作在 **OSI 模型第四层**，即**传输层**，这一层的**主要协议是 TCP/UDP**，负载均衡器在传输层基于**数据包里的源端口地址**和**目的端口地址**，通过**负载均衡算法**将**数据包***转发到***后端真实服务器**。

（四层负载均衡的核心就是 **IP+端口层面**的负载均衡，**不涉及具体的报文内容**）

2）七层负载均衡：工作在 **OSI 模型第七层**，即**应用层**，这一层的**主要协议是 HTTP**。这一层的负载均衡比四层负载均衡路由网络请求的方式**更加复杂**，它会**读取报文的数据部分**，然后根据**读取到的数据内容（如 URL、Cookie）**做出负载均衡决策。

（执行第七层负载均衡的设备通常被称为**反向代理服务器**，七层负载均衡器的核心是**报文内容（如 URL、Cookie）层面**的负载均衡）

3、客户端负载均衡

客户端负载均衡：主要应用于系统内部的**不同的服务之间**，可使用**现成的负载均衡组件**来实现。在客户端负载均衡中，客户端会自己维护一份**服务器的地址列表**，发送请求之前，客户端会根据对应的**负载均衡算法**来**选择具体某一台服务器处理请求**。

（Spring Cloud 是通过**组件的形式**实现的负载均衡，属于可选项，比较常用的是 Spring Cloud Load Balancer）

4、负载均衡常见的算法有哪些？

1）随机法

没有配置权重：所有的服务器**被访问到的概率都是相同**的；（适合于**服务器性能相近的集群**，其中每个服务器承载相同的负载）

配置权重：权重越高的服务器被访问的概率就越大。（适合于**服务器性能不等的集群**，权重的存在可以使请求分配更加合理化）

【缺陷】部分机器在**一段时间之内无法被随机到**，毕竟是概率算法

2）轮询法：挨个轮询服务器处理

没有配置权重：每个请求按时间顺序逐一分配到不同的服务器处理。（适合于**服务器性能相近的集群**，其中每个服务器承载相同负载）

配置权重：权重越高的服务器被访问的次数就越多。（适合于**服务器性能不等的集群**）

3）两次随机法

两次随机法在随机法的基础上多增加了一次随机，**多选出一个服务器**。**随后再根据**两台服务器的**负载等情况**，从其中**选择出一个最合适的**服务器。

【好处】可以动态地调节后端节点的负载，使其更加均衡。

4）哈希法

将**请求的参数信息**通过**哈希函数***转换成*一个**哈希值**，然后根据哈希值来**决定请求被哪一台服务器处理**。在**服务器数量不变**的情况下，**相同参数的请求**总是**发到同一台服务器**处理。

【缺陷】**服务器数量改变**时，哈希值会重新落**在不同的服务器**上。

5）一致性哈希法

将**数据和节点**都映射到一个**哈希环上**，然后根据**哈希值的顺序**来确定数据属于哪个节点。当**服务器增加或删除**时，只**影响该服务器的哈希**，而**不会导致整个服务集群**的哈希键值重新分布。

6）最小连接法

当有**新的请求出现**时，**遍历服务器节点列表**并**选取**其中**连接数最小的一台服务器**来响应当前请求。**相同连接**时，可以进行**加权随机**。

【缺陷】**连接数并不能代表**服务器的**实际负载**，有些连接耗费系统资源更多，有些连接不怎么耗费系统资源。

7）最少活跃法

活动连接数可以理解为**当前正在处理的请求数**。活跃数**越低**，说明**处理能力越强**，相同活跃数的情况下，可以进行加权随机。

8）最快响应时间法

最快响应时间法以**响应时间**为标准来**选择具体是哪一台服务器**处理。客户端**会维持每个服务器的响应时间**，每次请求挑选响应时间最短的。相同响应时间时，可以进行加权随机。

（可以使请求**被更快处理**，**但可能**会造成流量**过于集中于高性能服务器**的问题）

5、七层负载均衡可以怎么做？

1）DNS 解析

在 **DNS 服务器**中为同一个主机记录**配置多个 IP 地址**，这些 **IP 地址对应不同的服务器**。当**用户请求域名**的时候，DNS 服务器采用**轮询算法**返回 IP 地址，即实现了轮询版负载均衡。

2）反向代理（Nginx）

客户端将请求发送到**反向代理服务器**，由反向代理服务器**去选择目标服务器**，**获取数据后**再返回给客户端。对外暴露的是反向代理服务器地址，**隐藏了真实服务器 IP 地址**。反向代理**“代理”的是目标服务器**。

6、什么是 docker？

Docker：一个**快速交付**应用、**运行应用**的技术；

1）可以将**程序**及其**依赖**、**运行环境**一起打包为一个镜像，**可迁移到任意 Linux 操作系统**；

2）运行时利用**沙箱机制**形成**隔离容器**，各个**应用互不干扰**；

3）**启动、移除**都可以通过**一行命令完成**。

7、Docker 如何解决依赖的兼容问题？

例如**不同的软件**所需要的**函数库**或**依赖版本不同**，相互之间会产生干扰。

将应用的 libs 函数库、deps 依赖、配置与应用**一起打包**；并将每个应用**放到一个隔离容器去运行**，避免互相干扰。

8、Docker 如何解决不同系统环境的问题？

例如 Ubuntu 上的程序在 centOS 上跑不了，因为有些库函数在 centOS 上没有。

Docker 将用户程序与所需要调用的**系统函数库一起打包**；运行到不同操作系统时，直接**基于打包的库函数**，**借助于**操作系统的 **Linux 内核**来运行。

9、Docker 和虚拟机的差异

Docker 是一个**系统进程**，体积小、启动速度快、性能好；

虚拟机是在操作系统中的**操作系统**，体积大、启动慢、性能一般。

10、镜像(Image):一个特殊的文件系统

Docker 将应用程序及其所需的依赖、函数库、环境、配置等文件**打包在一起**，成为镜像。

（镜像不包含任何动态数据，其内容在构建之后也不会被改变。**是只读的**，被读到容器中）

11、容器(Container):镜像运行时的实体

镜像中的应用程序**运行后形成的进程**就是容器，但 docker 会给容器做隔离，对外不可见。

（容器存储层的生存周期和容器一样，容器消亡时，容器存储层也随之消亡。）

12、仓库(Repository):集中存放镜像文件的地方（集中的**存储、分发镜像**的服务）

一个 Docker Registry 中可以包含多个仓库（Repository）；每个仓库可以包含多个标签（Tag）（**同一个软件**不同版本的镜像）；每个标签对应一个镜像（**标签对应该软件的各个版本**）。可以通过**<仓库名>:<标签>**的格式来指定具体是这个软件哪个版本的镜像。如果不给出标签，将以 **latest 作为默认标签**。

13、Docker 架构

Docker 是一个 **CS 架构**的程序，有：

1）服务端：docker 守护进程，负责处理 docker 指令，管理镜像、容器等；

2）客户端：通过**本地命令**或**远程 RestAPI** 向 docker 服务端发送指令。

两种构建镜像的方式：

1）通过命令 docker build 发送数据到服务端，服务端通过 **docker 守护进程**接受命令，并根据发送的数据构建镜像；

2）通过命令 docker pull 发送请求到服务端，服务端通过 docker 守护进程接受命令，并从仓库中拉去指定的镜像；

3）通过命令 docker run 发送请求到服务端，服务端通过 docker 守护进程接受命令，帮助完成容器创建并运行。

14、镜像操作命令

docker build（构建）、docker pull（拉取）、docker push（推送）、docker save（保存镜像为压缩包）、docker load（加载压缩包为镜像）

docker run（运行）、docker pause（暂停）、docker unpause（恢复运行）、docker stop（停止）、docker start（开始运行）

docker ps：查看所有运行的容器及状态；

docker logs：查看容器运行日志；

docker exec：进入容器执行命令；

docker rm：删除指定容器；

15、数据卷

数据卷是一个**虚拟目录**，指向**宿主机文件系统中**的某个目录。这样**容器**就可以加载数据卷，从而**和宿主机文件进行关联**。

【作用】将容器和数据分离，解耦合，方便操作容器内数据，保证数据安全。

1）数据卷基本操作：docker volume + 命令

1> create：创建一个 volume；

2> inspect：显示一个或多个 volume 的信息；

3> ls：列出所有的 volume；

4> prune：删除未使用的 volume；

5> rm：删除一个或多个指定的 volume；

2）挂载数据卷：-v

1>挂载数据卷到某个容器目录：

-v [数据卷]: [容器内文件目录]

2>挂载宿主机目录到某个容器目录：

-v [宿主机文件] : [容器内文件]

16、镜像结构

分层结构，每一层称为一个 Layer

1）BaseImage 层：包含基本的系统函数库、环境变量、文件系统；

2）Entrypoint：入口，镜像中应用启动的命令；

3）其他：在 BaseImage 层基础上添加依赖、安装程序、完成整个应用的安装和配置。

17、DockerCompose-部署微服务集群

1）编写 docker-compose 文件，.yml 格式

```
version: "3.2"

services:
  nacos:
    image: nacos/nacos-server
    environment:
      MODE: standalone
    ports:
      - "8848:8848"
  mysql:
    image: mysql:5.7.25
    environment:
      MYSQL_ROOT_PASSWORD: 123
    volumes:
      - "$PWD/mysql/data:/var/lib/mysql"
      - "$PWD/mysql/conf:/etc/mysql/conf.d/"
  userservice:
    build: ./user-service
  orderservice:
    build: ./order-service
  gateway:
    build: ./gateway
    ports:
      - "10010:10010"
```

2）将项目中数据库、nacos 地址都命名为 docker-compose 中的服务名；

3）maven 打包工具将每个微服务打包 app.jar；并将其 target 文件夹中的 app.jar 拷贝到对应 Dockerfile 所在文件夹；

```
FROM java:8-alpine
COPY ./app.jar /tmp/app.jar
ENTRYPOINT java -jar /tmp/app.jar
```

4）上传虚拟机，并 docker-compose up -d 部署