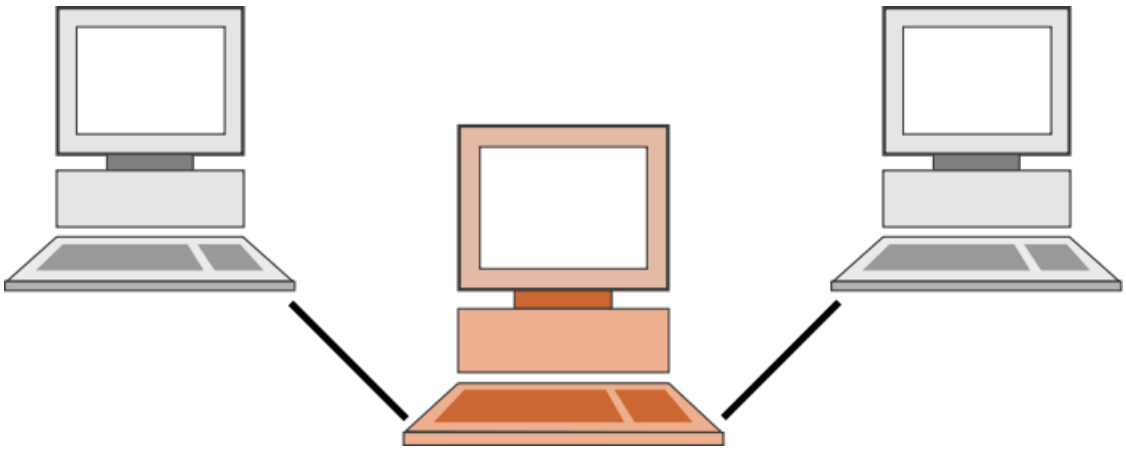
代理简介



正向代理，是一个位于客户端和原始服务器之间的服务器，为了从原始服务器取得内容，客户端向代理发送一个请求并指定目标(原始服务器)，然后代理向原始服务器转交请求并将获得的内容返回给客户端。通常客户端需要进行一些设置，比如指定代理的ip和端口,用户名密码等。

- 联想场景：比如小明有个项目，想向马云借钱获得启动资金，但知道马云不认识他不会借给他。这时小明可以联系他的导师，他导师刚好认识马云。他导师借到钱后转手就把钱给了小明。马云自始至终不知道实际借钱的人是谁。

反向代理 正好相反，对于客户端而言它就像是原始服务器，并且客户端不需要进行任何特别的设置。客户端向反向代理 的命名空间中的内容发送普通请求，接着反向代理将判断向何处(原始服务器)转交请求，并将获得的内容返回给客户端，就像这些内容原本就是它自己的一样。

- 联想场景：我们在打10086查询话费的时候，我们始终关心的是自己到底消费了多少钱。至于一个10086号码拨通后，到底是哪个业务员告诉我消费结果我是不知道的，我也不关心。

负载均衡

通过load balance将http请求分发到不同的server上，来达到方便扩展，提升性能和提高可靠性的目的。常见load balance方法如下：

- round-robin（轮询调度）

    - **算法原理**：相当于循环哈希，每来一个请求，在上一个hash值的基础上做一次调整，重新计算哈希。
    - **特点**：每个服务器处理请求的机会均等。
    - **优点**：算法实现简单，当其中某台服务器出现故障时，可轻易将该台服务器切出服务。整体服务的可靠性能得到保障。
    - **缺点**：服务器之间的性能差异较大，或者请求服务的时间长短不一时，容易导致服务器间的负载不平衡。另外，同一个client的多次请求，无法保证会打到同一个server上，session的保持就会出现问题。

- Least connected load balancing

    - **算法原理**：每次选择最少并发量请求的server，当server间连接数量相等时，退化成round-robin。
    - **特点**：并发数量少的服务器优先级高于并发数量多的服务器。
    - **优点**：server间的负载会比较均衡。
    - **缺点**：session保持同round-robin算法。

- Weighted load balancing

    - **算法原理**：通过为不同处理能力的server设置权值，按照权值的大小关系，对client的请求进行分配。
    - **特点**：服务器设置的权值越大，分配的请求越多
    - **优点**：可以充分利用处理能力较强的server
    - **缺点**：由于需要事先确定权值，如果请求的服务不同，每种服务处理时长不同时，还是会造成负载不均。

- ip hash

    - **算法原理**：将client请求的ip作为哈希的key，这样同一个client的多次请求会严格转发到同一个server上。
    - **优点**：session保持的问题解决了。
    - **缺点**: 会造成负载不均。

- url hash（第三方模块）

    - 和ip_hash算法类似，是对每个请求按url的hash结果分配，使每个URL定向到一个同 一个后端服务器，但是也会造成分配不均的问题。

- Fair（第三方模块）

    - 按后端服务器的响应时间来分配请求，响应时间短的优先分配。

Health checks

- 主要通过配置max_fails（最大尝试失败次数）和fail_timeout（失效时间）命令来实现。
    - 当达到最大尝试失败次数后，在fail_timeout的时间范围内节点被置为失效。当超过失效时间后，节点会重新置为有效，nginx会重新探测。
    - 当测探到所有的节点均失效时，nginx会对所有节点恢复为有效，重新尝试探测有效节点。

nginx代理配置

在nginx中配置proxy_pass代理转发时，如果在proxy_pass后面的url加/，表示绝对根路径；如果没有/，表示相对路径。

1. 绝对路径

```
location /proxy/ {
          proxy_pass http://127.0.0.1:8000/;
    }
```

- 当访问http:/127.0.0.1/proxy/example.html时，实际请求代理服务器的url为：http:/127.0.0.1:8000/example.html
- 即: **转发的url** = proxy_pass + url[url.index(location) + len(location):]

2. 相对路径

```
location /proxy/ {
          proxy_pass http://127.0.0.1:8000;
    }
```

- 当访问http:/127.0.0.1/proxy/example.html时，实际请求代理服务器的url为：http:/127.0.0.1:8000/proxy/example.html
- 即: **转发的url** = proxy_pass + url[url.index(location):]