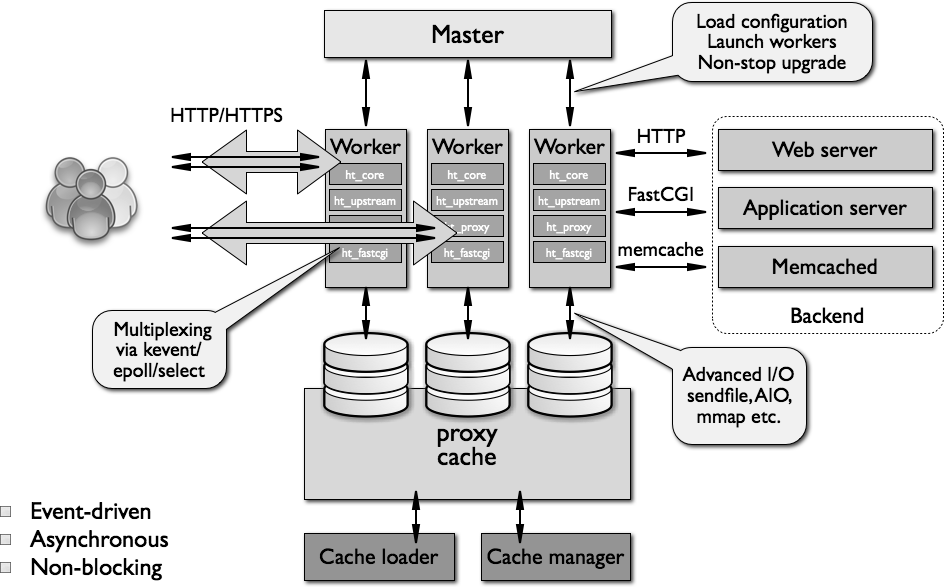
**Nginx Load Balancer**

简单说一下Nginx架构[1][2]。Nginx是一个多进程服务，包含一个Master进程和多个Worker进程，进程间通过共享内存通信。对于网络请求，各个worker共享一个监听socket，独立接收（accept）网络连接请求，并处理随后的数据信息。具体架构分析可参考[2]，我们主要看一下Nginx作为HTTP(S)负载均衡的配置[1]。



1. Nginx架构图

先配置承担业务流量的一组上游服务器：

http {

upstream backend {

# no load balancing method is specified for Round Robin

*#least\_conn|ip\_hash|random two least\_conn*

server backend1.example.com **weight=5 max\_conns=3**;

server backend2.example.com;

server 192.0.0.1 **backup**;

}

server {

listen 80;

location / {

proxy\_pass http://backend;

}

}

}

1. *Nginx配置*

负载均衡方法：

使用least\_conn和random负载均衡方式不可以保证同一个客户端请求发送到同一个服务器处理，ip\_hash可以保证同一个客户端请求发送到同一个服务器处理。

权重：

权重值代表对应服务器应处理的请求数份额。

健康监测

[max\_fails](https://nginx.org/en/docs/http/ngx_http_upstream_module.html" \l "max_fails" \t "/Users/lihanhui/Documents\\x/_blank), [fail\_timeout](https://nginx.org/en/docs/http/ngx_http_upstream_module.html" \l "fail_timeout" \t "/Users/lihanhui/Documents\\x/_blank)，等参数设置判定服务失效的标准：max\_fails表示服务连续失败的最大次数；fail\_timeout表示失败的时间间隔判定标准。backup参数表示在只有其他服务全部不可用的情况时，此服务才可以启用。

其他调优参数：

net.core.somaxconn：最大连接数

net.core.netdev\_max\_backlog：数据缓存优化

[worker\_processes](https://nginx.org/en/docs/ngx_core_module.html" \l "worker_processes" \t "/Users/lihanhui/Documents\\x/_blank)：worker数

[worker\_connections](https://nginx.org/en/docs/ngx_core_module.html" \l "worker_connections" \t "/Users/lihanhui/Documents\\x/_blank)：worker同时处理的连接数

[keepalive\_requests](https://nginx.org/en/docs/http/ngx_http_core_module.html" \l "keepalive_requests" \t "/Users/lihanhui/Documents\\x/_blank)：单条连接支持的最大请求数

[keepalive\_timeout](https://nginx.org/en/docs/http/ngx_http_core_module.html" \l "keepalive_timeout" \t "/Users/lihanhui/Documents\\x/_blank)：连接失效的超时时间

client\_max\_body\_size：上传文件限制

1. <https://docs.nginx.com/nginx/admin-guide/load-balancer/>
2. <http://www.aosabook.org/en/nginx.html>
3. <https://nginx.org/en/docs/http/load_balancing.html>
4. <https://docs.nginx.com/nginx/admin-guide/load-balancer/http-load-balancer/>
5. <https://www.nginx.com/blog/tuning-nginx/>