HTTP服务器设计文档

1. **设计背景**
2. **使用 Java 编写 Http 服务器，考虑服务负载、并发访问、高可用，实现 Http 协议 Authorization 验证方式，支持 html、image、xml、file 的资源访问。**
3. **实现功能**
4. ****Http 服务器****

**已知java中有三种IO方式实现HTTP协议，分别是BIO,NIO,AIO。BIO效率低下已经基本被淘汰，只有NIO和AIO可以选择，其中AIO是NIO在异步方面的改进，由于我们只实现一些静态访问NIO方面最为合适（其实AIO不熟）**

1. ****服务负载****

**负载均衡**

**负载均衡目前较为流行的算法有以下几种：**

**2.1 一致性hash(存在分配不均匀的问题)**

**2.2 IPhash(掉线无法感知)**

**2.3 最小活跃数**

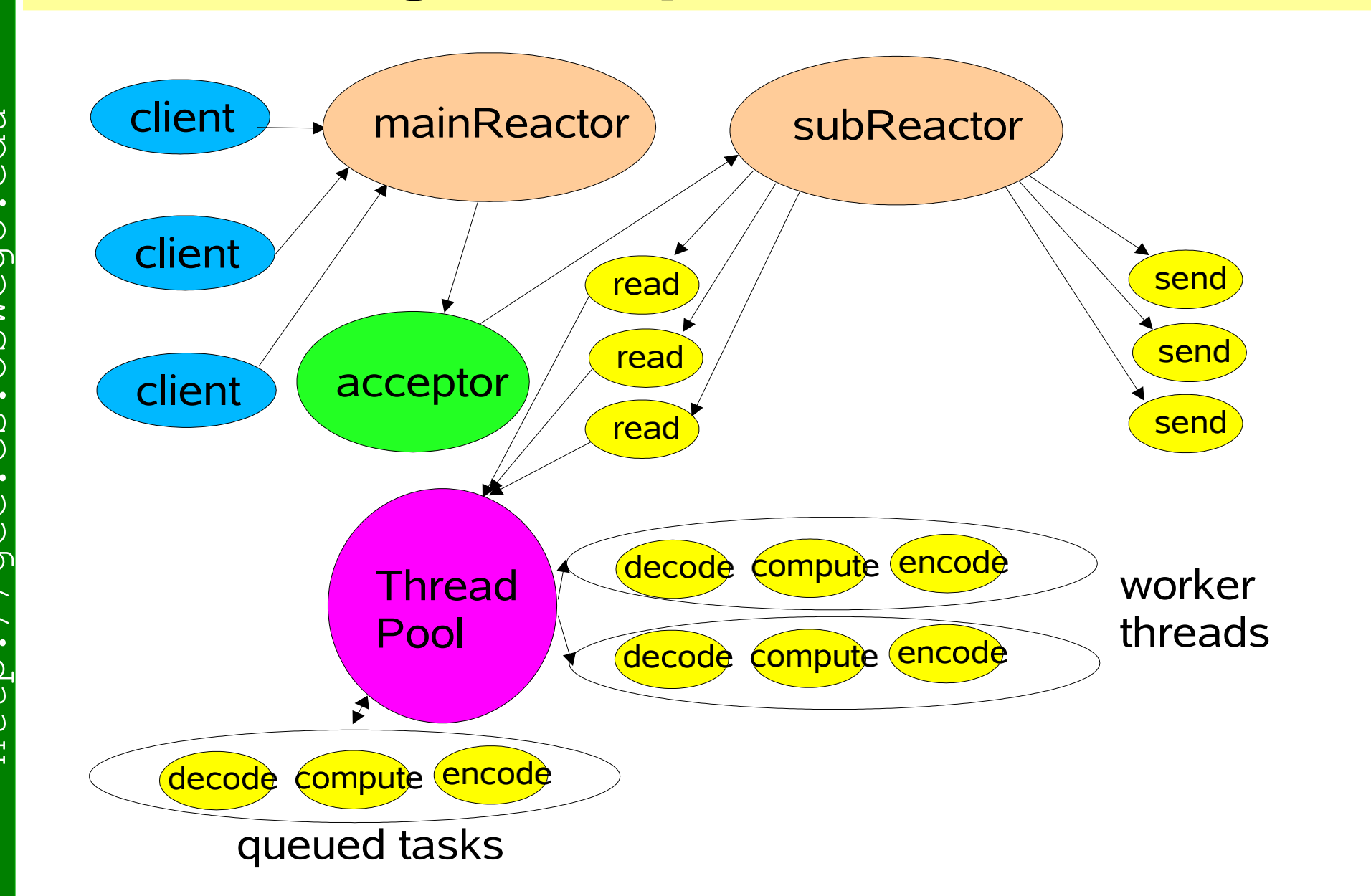
**2.4 随机**

**2.5 轮训**

**2.6 加权轮训**

1. ****并发访问****

**采用Reactor模型 如图：**



**但是做了调整**

1. ****高可用****

**因为http服务器不存在分布式部署，所以采用集群部署，**

1. ****实现 Http 协议 Authorization 验证方式****

**在HTTP请求头之中加入Authorization，Authorization有basic和bearer等。basic方式不建议使用，采用bearer 简单的验证即可 ，既要合理验证也要保证性能**

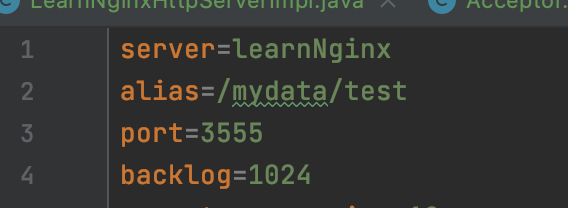
1. ****支持 html、image、xml、file 的资源访问****

**支持以上的文件类型，对于java来说都是静态资源，是存在磁盘上的文件，无关类型，存在解析为字节数组，不存在则返回状态码404 not found.**

**借用nginx中alias的模式，**

**请求路径http://localhost:1222/pom-test.xml**

**alias 配置如下**



**将会匹配到路径为 /mydata/test/pom-test.xml**

**无论静态请求或动态请求，只在于配置,当然只实现了静态请求.**

1. ****CI/CD、日志、监控和报警的设计****

**CI/CD jenkins 感知变化，触发构建上线**

**日志，采用一般的做法，就是创建目录，保存日志，这种做法很多项目都在用tomcat,nginx等等**

**监控,报警： 内存，cpu，硬盘等等，通过jmx方式暴露（这是要逼我熟悉metric）**

1. ****短域名****

**短域名的解决方案也将在次服务器中部署这样能减少一些工作量，短域名解决方案可以简单的理解为key-value 映射关系，生成的短域名对应原本的长域名，在访问短域名的时候解析为长域名，在跳转到长域名地址，在这之中可能存在一些脚本的注入问题，简单的key-value关系可以存储在redis中，key-value结构需要hash算法，好的hash算法产生的冲突就越少，sha1就是其中之一，git就是采用此种hash算法（前七位即可），而一般采用的是md5方式，如果时间允许将提供两种不同的实现**

**可以采用二级缓存的方式保证性能，可以采用guava cache**

**虽然写到了一起但是短域名要保证高可用则需要分布式部署，前文中的静态资源服务器更像nginx报警，而短域名的实现则更像是tomcat,需要分布式部署**