# 使用ES做人脸比对服务测试报告

### 1.环境配置

##### 本地系统配置

系统: windows10 专业版

CPU: Intel Core i7-6700 3.40GHz 4核8线程

内存: 12GB

ES: elasticsearch-7.6.2 单节点部署

数据库: postgresql 10.0 阿里云RDB

##### 线上系统配置

系统: ubutnu 16

CPU: 48核

内存: 96GB

ES: elasticsearch-7.10 单节点部署

数据库: postgresql 10.0 阿里云RDB

### 2.主要功能

人脸特征是通过人脸识别算法和模型提取出来的一组向量,可以通过计算两个向量之间的余弦相似度得到两个人脸特征的匹配程度. 从Elasticsearch 7.2开始，余弦相似度可作为预定义函数使用,所以可以通过Elasticsearch完成人脸匹配功能.

具体实现流程如下:

1.创建特征库.每一个特征库对应一个es索引,特征库的名称即为索引名称,建立索引的同时建立好mapping. mapping定义了索引中存储的数据可能包含的字段和字段类型.由于是测试,所以本次只有两个字段,personId对应抓拍Id(可视为唯一ID),类型为keyword, data对应特征向量,类型为dense\_vector.

2.添加人脸特征.人脸特征从线上公司测试环境的数据中获得,从当天开始向前循环获取抓拍数据,根据抓拍数据取得特征,加入到特征库里.测试时建立三个特征库,大小分别为2000,5000和20000.

3.比对.上传一张人脸图片,首先通过AlgClient提取图片特征,之后通过es预定义函数cosineSimilarity用特征在选择的特征库中进行比对,默认按照分数降序排序,取前40个比对结果的personId并从数据库中找到对应数据后展示.

对比正在使用的比对服务的优势有:

1. 自带高可用性

2. 数据存储在硬盘内,不会存在重启后需要重新建立特征库的问题

3. 自身是分布式系统,方便扩展

4. 容错容灾手段完备

### 3.测试

##### 本地测试

特征库: test\_pool\_2000, test\_pool\_5000, test\_pool\_20000

测试图片:从当天抓拍中取10张不同的人的图片

测试方法:用10张图片对三个库进行第一次匹配,记录时间后进行第二次匹配

test\_pool\_2000 的10次匹配耗时(毫秒)分别为

63 ,279,277,111,992,22,120,91,24,247

第二次测试结果为

100,22,89,121,63,119,60,62,83,104

test\_pool\_5000 的10次匹配耗时(毫秒)分别为

391,340,286,192,53,29,172,222,206,168

第二次测试结果为

18,17,37,16,18,16,16,16,15,15

test\_pool\_20000的10次匹配耗时(毫秒)分别为

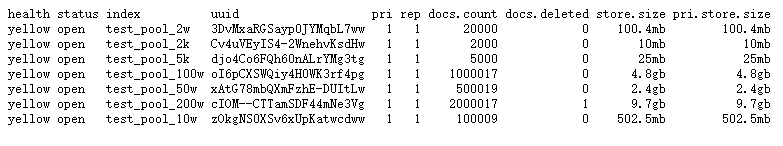
294,251,253,240,71,37,265,267,36,213

第二次测试结果为

48,44,41,61,49,38,57,37,67,35

根据耗时来看无法得出有用的结论

##### 线上测试

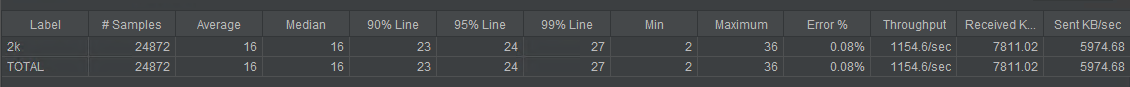
特征库: test\_pool\_2k, test\_pool\_2w, test\_pool\_50w, test\_pool\_100w,test\_pool\_200w  


测试方法:

使用Jmeter进行压力测试,对每个特征库使用不同数量的线程测试,观察请求所用的时间以及CPU满负荷时线程数量,每个请求采用相同的请求体

1. 对于 test\_pool\_2k 进行性能测试

1). 20线程



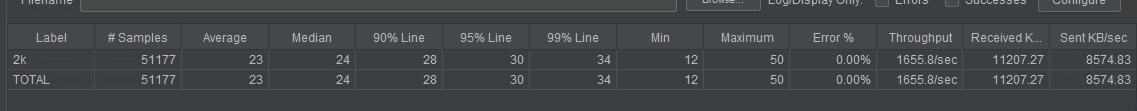
平均响应时间:16ms

最大响应时间:36ms

最小响应时间:2ms

CPU压力: 2000%

2). 40 线程



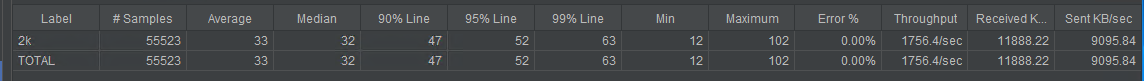
平均响应时间:23ms

最大响应时间:50ms

最小响应时间:12ms

CPU压力:3900%

3). 60线程



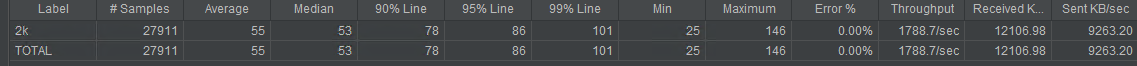
平均响应时间:33ms

最大响应时间:102ms

最小响应时间:12ms

CPU压力:4200%

4). 100 线程



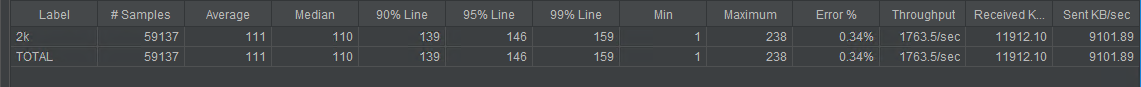
平均响应时间:55ms

最大响应时间:146ms

最小响应时间:25ms

CPU压力:4350%

5.). 200线程



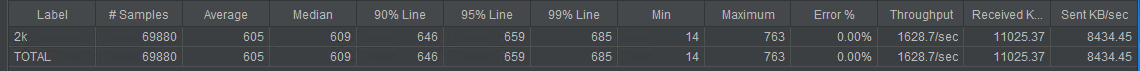
平均响应时间:111ms

最大响应时间:238ms

最小响应时间:1ms

CPU压力:4350%

6). 1000 线程



平均响应时间:605ms

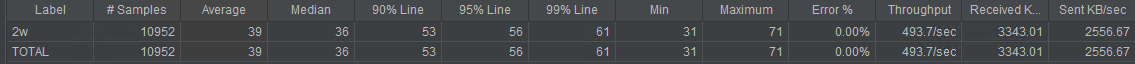
最大响应时间:763ms

最小响应时间:14ms

CPU压力:4150%

2. 对于 test\_pool\_2w 进行性能测试

1). 20 线程



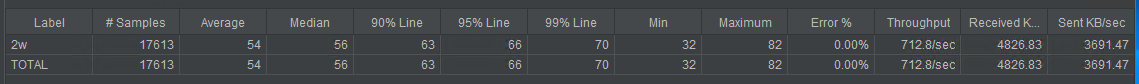
平均响应时间:39ms

最大响应时间:71ms

最小响应时间:31ms

CPU压力:2000%

2). 40 线程



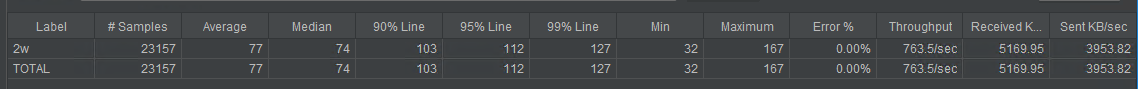
平均响应时间:54ms

最大响应时间:82ms

最小响应时间:32ms

CPU压力:4000%

3). 60 线程



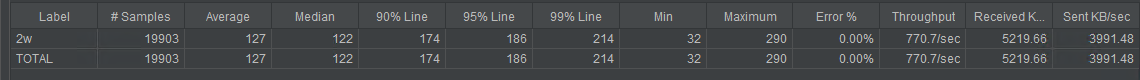
平均响应时间:77ms

最大响应时间:167ms

最小响应时间:32ms

CPU压力:4550%

4). 100线程



平均响应时间:127ms

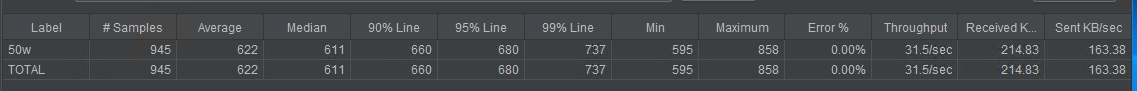
最大响应时间:290ms

最小响应时间:32ms

CPU压力:4590%

3. 对于 test\_pool\_50w 进行性能测试

1). 20 线程



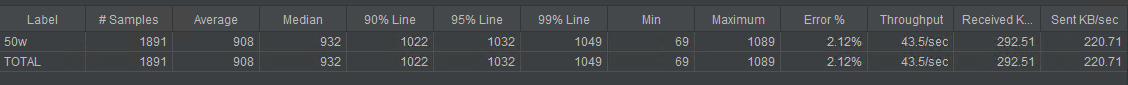
平均响应时间:622ms

最大响应时间:858ms

最小响应时间:594ms

CPU压力:2000%

2). 40线程



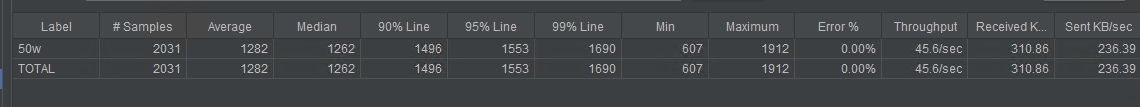
平均响应时间:908ms

最大响应时间:1089ms

最小响应时间:69ms

CPU压力:4000%

3). 60 线程



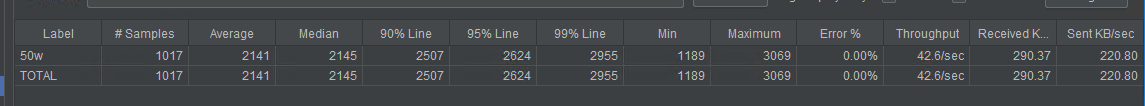
平均响应时间:1280ms

最大响应时间:1912ms

最小响应时间:607ms

CPU压力:4780%

4). 100线程



平均响应时间:2071ms

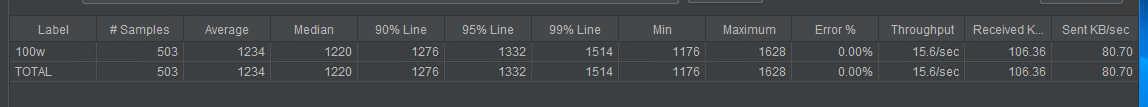
最大响应时间:3069ms

最小响应时间:1189ms

CPU压力:4780%

4. 对于 test\_pool\_100w 进行性能测试

1). 20 线程



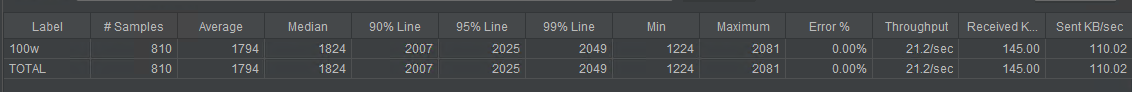
平均响应时间:1234ms

最大响应时间:1628ms

最小响应时间:1176ms

CPU压力:2000%

2). 40线程



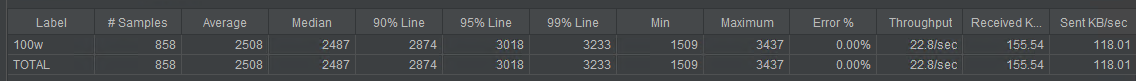
平均响应时间:1794ms

最大响应时间:2081ms

最小响应时间:1224ms

CPU压力:4000%

3). 60线程



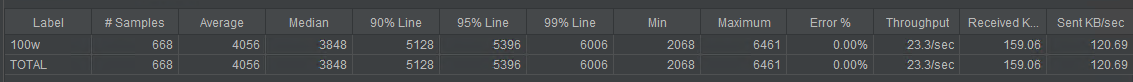
平均响应时间:2508ms

最大响应时间:3437ms

最小响应时间:1509ms

CPU压力:4780%

4). 100 线程



平均响应时间:4056ms

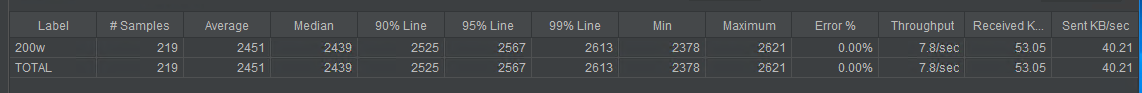
最大响应时间:6461ms

最小响应时间:2068ms

CPU压力:4780%

4. 对于 test\_pool\_200w 进行性能测试

1). 20 线程



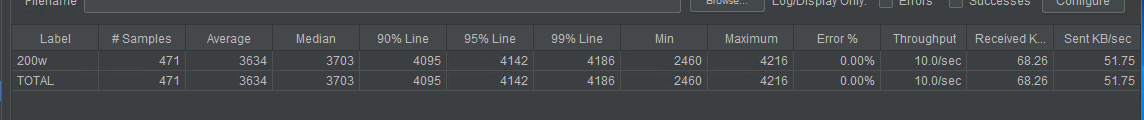
平均响应时间:2451ms

最大响应时间:2621ms

最小响应时间:2378ms

CPU压力:2000%

2). 40 线程



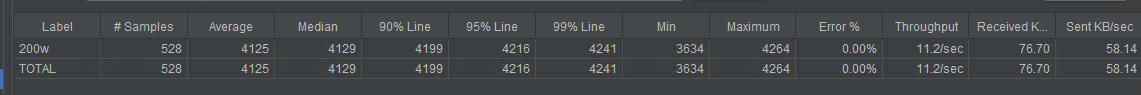
平均响应时间:3634ms

最大响应时间:4216ms

最小响应时间:2460ms

CPU压力:4000%

3). 48线程



平均响应时间:4125ms

最大响应时间:4264ms

最小响应时间:3634ms

CPU压力:4780%

### 4.结论

1.响应时间随特征库数量增大而增加,百万级以下的响应时间和目前使用的差别不大(目前是32核),百万级以上时优于目前服务(与青岛社保项目对比,服务器配置应该不同)

2.根据线程数和CPU的压力来看, 1个线程的请求占用1核的算力,理论上最优线程数应该是48个,线程继续增加的话会使响应时间增加

3.2000数量特征库的CPU压力和其他库不同应该是由于处理时间较短,导致最后几个请求未达到服务器时之前的请求已经处理完了.

### 5.面临的问题

未找到如何在一个文档中存储多个向量.目前的匹配服务可以将多个人的特征向量放在一个数组中,以此表示是同一个人,匹配时对数组中的所有向量进行计算,以最高分作为特征对于这个人的分数,最后返回以人为单位如分数最高的五个人.一个规避方法是用personId来记录人的person\_unid,用es文档的id来记录图片id,由此也能认为是同一个人,但是最后的返回结果只能以图片为单位例如返回分数最高的五张图片.