1. 环境

- 硬件: 腾讯云2核2G内存弹性服务器
- 系统: Ubuntu 20.04
- web server: Tomcat 9.0.71
- 开发语言: java11
- 开发框架: spring-boot 2.7.6

2. 主要数据结构

保存用户的主要数据结构如下, 包含两重映射

1. `LinkedHashMap`: 负责进行`用户ID->KV表`的映射, 借助`LinkedHashMap`实现LRU算法, 当容量达到上限时, 会自动淘汰最不常使用用户. 该表默认的容量为`10000`
2. `HashMap`: 表示一个KV表, 负责进行`字段名->字段值`的映射

```
private LinkedHashMap<String, HashMap<String, String>> database
```

注意:

- 数据暂存的第一要求是速度快, 对于可靠性要求并不高, 数据只要保存在内存中即可
- **HashMap不是线程安全的**, 这里这是一个demo, 生产环境可以使用redis作为数据库来暂存数据

3. 接口说明

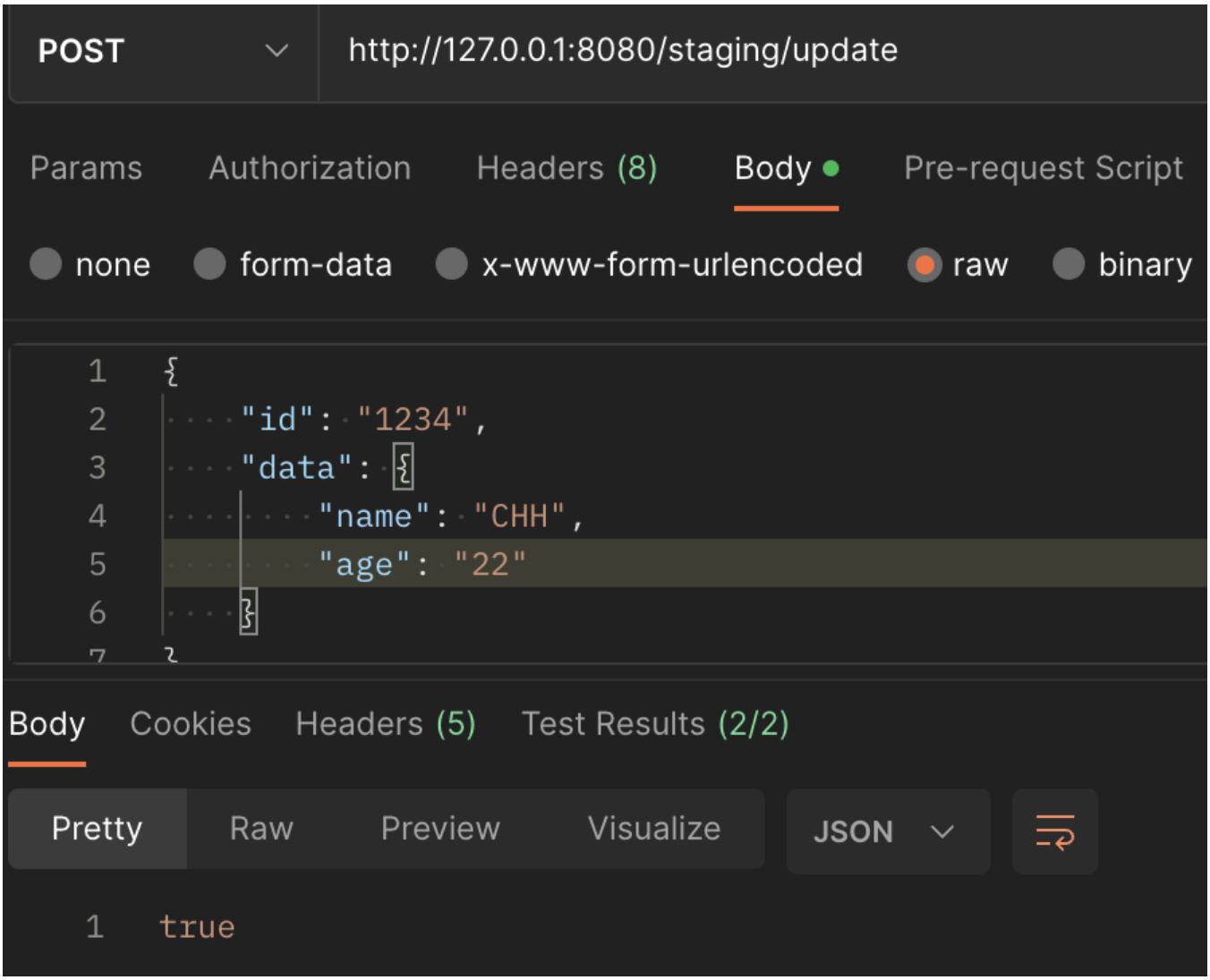
接口URL格式如下

```
http://URL/staging/功能
```

3.1 `update`

负责更新用户暂存的数据

请求样例如下

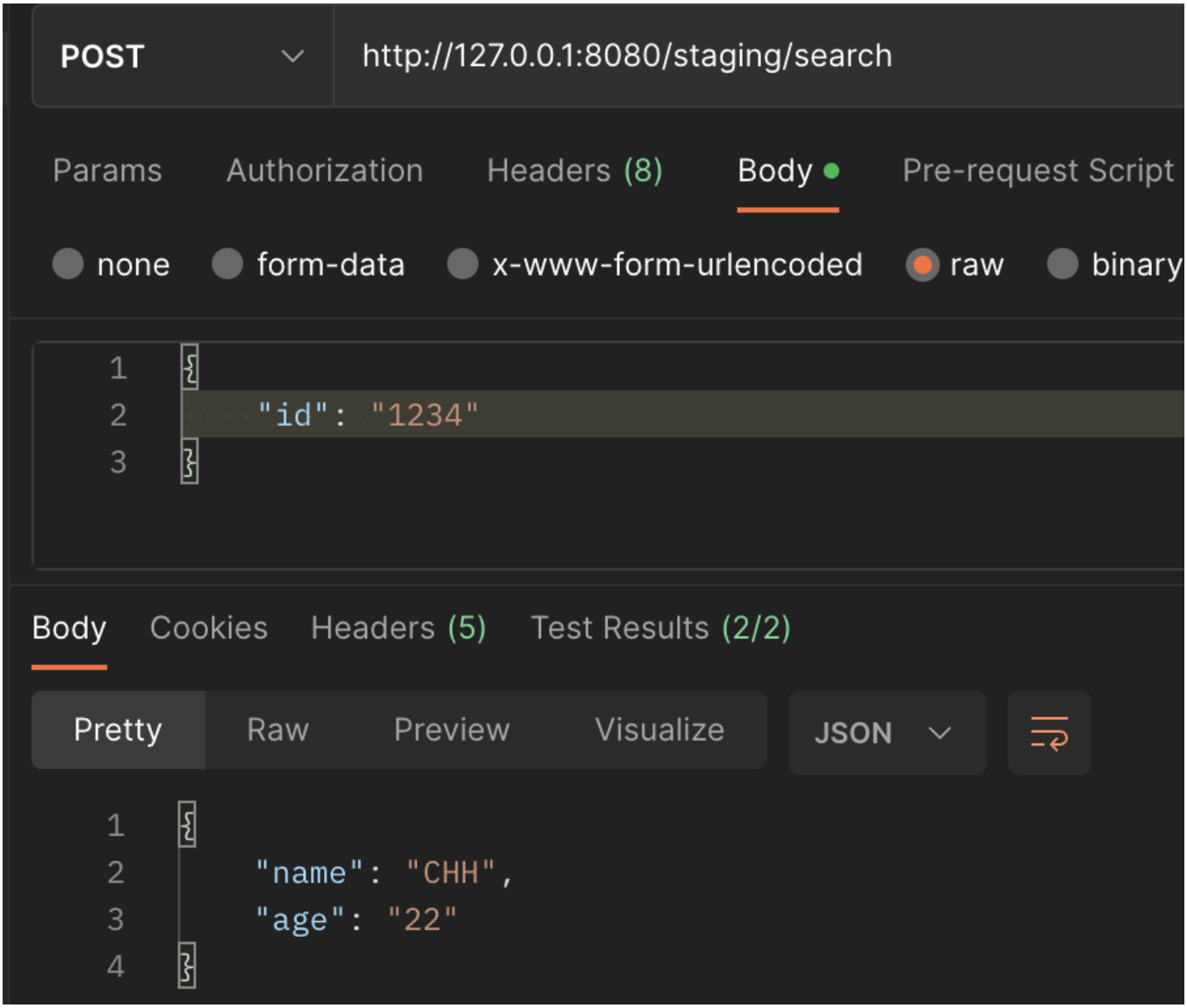


字段说明:

- `"id"`: 表示用户身份, 尽量设置为session id或者其他**不容易被遍历的字段**. 严禁设置为手机号或者用户id这种极易被遍历出的字段, 否则会引起数据泄露问题
- `"data"`: 一个字典对象, 包含所有需要更新的KV对. key与value都是`String`类型. 更新KV对时有两种情况:
  - 如果后端数据库中key已经存在, 则新发送的value会覆盖旧的value. 例如先发送`"name": "C"`, 再发送`"name": "CHH"`, 则`"name"`最终值为`"CHH"`
  - 如果key不存在, 那么会在hash table中插入这个字段及其对应的值, 相当于add功能
- 返回值: 更新成功则返回true

3.2 `search`

查询一个用户所有暂存的字段, 样例如下



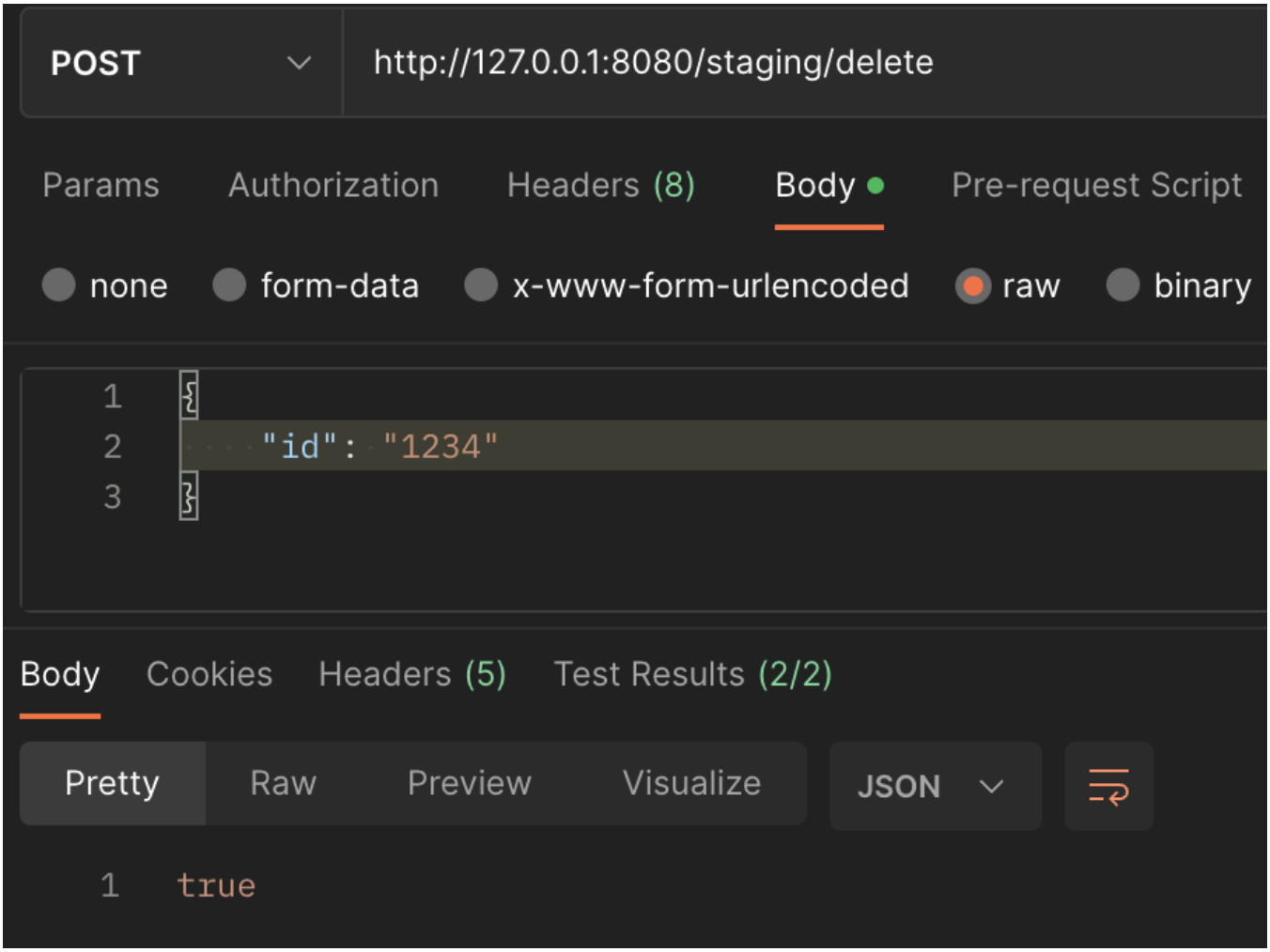
字段说明:

- `"id"` 字段表示要查询的用户
- 返回值: 会返回一个用户暂存的所有字段组成的字典对象. 如果该用户没有保存数据, 则返回空字典对象 `{}`

> 为何不支持单个字段查询?
>
> 因为单独一个KV对的数据量比较小, 通常少于或约等于HTTP头部的长度, 因此如果为每一个KV读都单独发起一个请求的话会有大量的带宽浪费在HTTP头部上, 因此直接一次性查询所有字段效率更高.

3.3 `delete`

从数据库中删除一个用户, 样例如下



字段说明:

- `"id"`: 需要删除的用户
- 返回值: 删除成功返回true, 如果用户不存在则返回false

在明确该用户暂存的数据不会再被使用时调用此接口从数据库中删除该用户, 以清理无用数据.

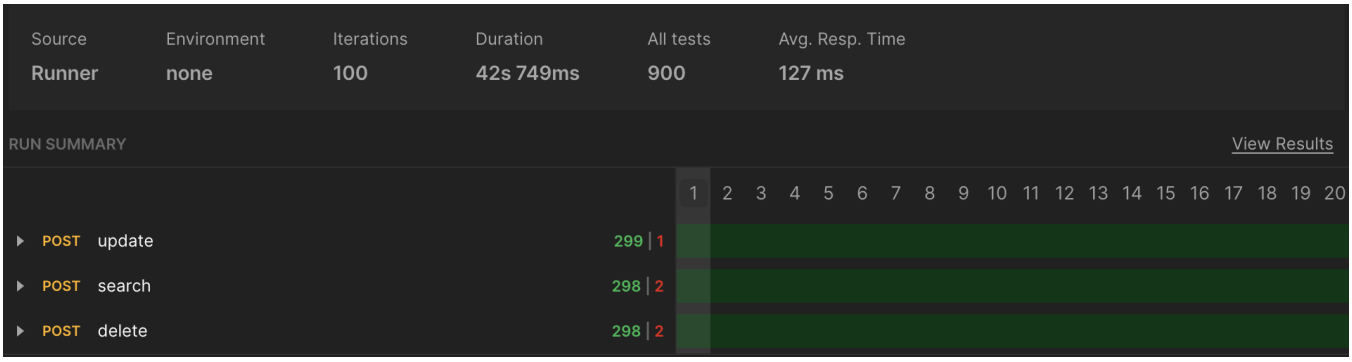
这个接口不是必须调用的, 因为在数据库已满时会自动淘汰最不常用的用户, 但是调用此接口可以帮助后端或者更好的性能

4. 性能测试

4.1 时延测试

以update, search, delete三个请求为一组, 循环执行100次. 共计耗时42749ms, 平均

- **每个请求的响应时间为127ms**
- **处理每个请求耗时142.5ms**



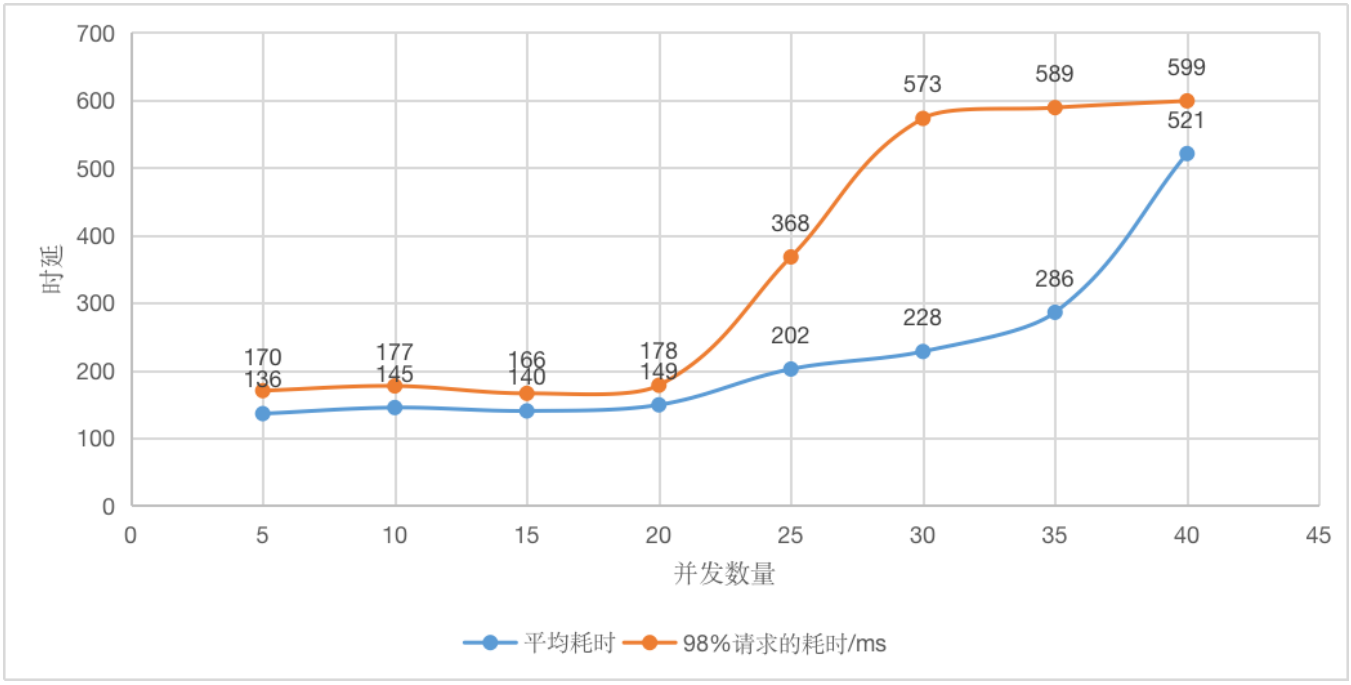
(使用Postman测试, 时延与地理位置相关, 上述数据仅代表一个参考值)

4.2 并发测试

apache bench测试, 测试命令如下

```
ab \
    -t 5 \
    -c 并发数量 \
    -T application/json \
    -p /tmp/post \
    http://101.34.91.153:8080/staging/update
```

横坐标为并发数量, 纵坐标为耗时, 不断调整并发数量, 测试5s, 得到的请求时延与并发数量的关系如下图


700
600
500
400
300
200
100
0
时延
0
5
10
15
20
25
30
35
40
45
并发数量
170
136
177
145
166
140
178
149
202
368
228
573
286
589
521
599
平均耗时
98%请求的耗时/ms