JD-hotkey框架处理热key实践

事前准备

1、下载jd-hotkey源码：[gitee.com/jd-platform...](http://cxyroad.com/"https://gitee.com/jd-platform-opensource/hotkey")

2、在etcd下载页面下载对应操作系统的etcd，[github.com/etcd-io/etc...](http://cxyroad.com/"https://gitee.com/link?target=https%3A%2F%2Fgithub.com%2Fetcd-io%2Fetcd%2Freleases") 使用3.4.x以上。

一、安装etcd

详细安装地址可以参考：[tianyalei.blog.csdn.net/article/det...](http://cxyroad.com/"https://tianyalei.blog.csdn.net/article/details/106927733")

1、这里以windows系统，单机启动为例，下载后，解压运行 etcd.exe 即可。

2、运行后出现 ready to serve client requests 字眼，表示启动成功

二、启动worker（集群）

jar 运行方式

下载并编译好代码，将worker打包为jar，启动即可。如：

命令：nohup java -jar worker-0.0.4-SNAPSHOT.jar --etcd.server=${etcdServer} 2>&1 &

这里指定jar运行参数可以参考 worker模块下的 application.yml

idea运行

修改worker模块下的 application.yml的配置，我这里只修改了etcd地址和workrt的路径

```
etcd:
  server: http://127.0.0.1:2379 #etcd的地址，重要！！！
  workerPath: test_hot_key #该worker放到哪个path下，譬如放/app1下，则该worker只能被app1使用，不会为其他client提供服务
```

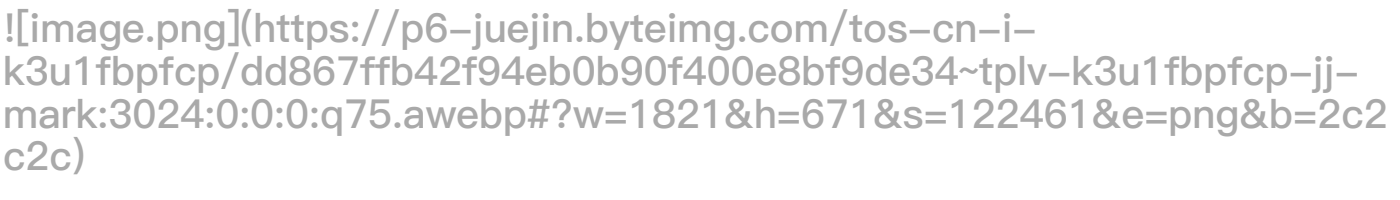
然后找到启动类，运行即可

三、启动控制台

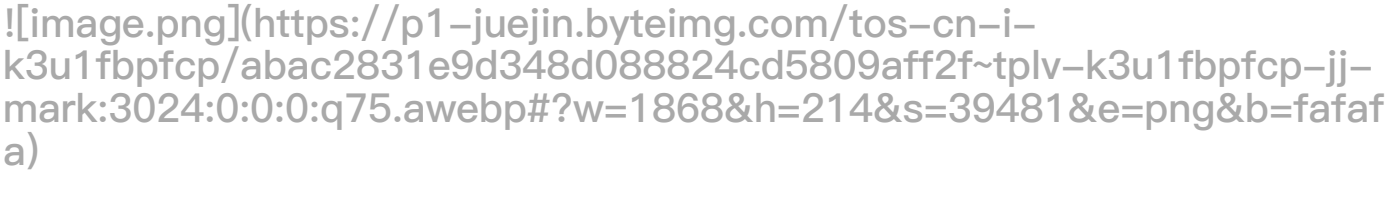
下载并编译好dashboard项目，创建数据库并导入resource下db.sql文件。 配置一下application.yml里的数据库相关和etcdServer地址。 启动dashboard项目，访问ip:8081，即可看到界面。

用户名/密码：admin/123456 （SQL初始导入）

这里也是jar 或者 idea运行都可以，自行选择。



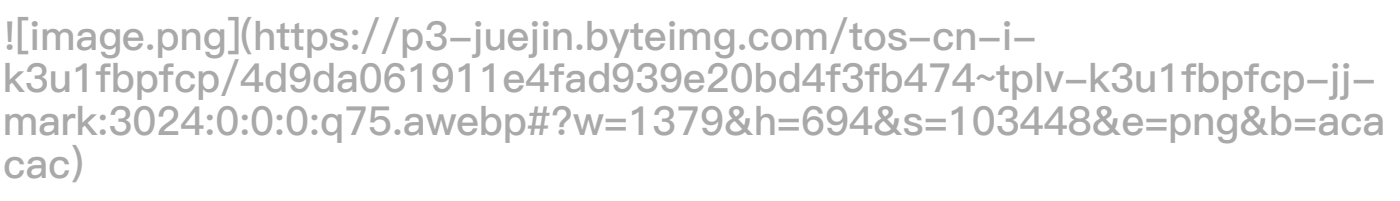
这里可以看到已经运行的worker节点信息



四、规则配置

新建用户

在用户管理菜单中，添加一个新用户，设置他的APP名字，如test_hot_key。之后新添加的这个用户就可以登录dashboard给自己的APP设置规则了。

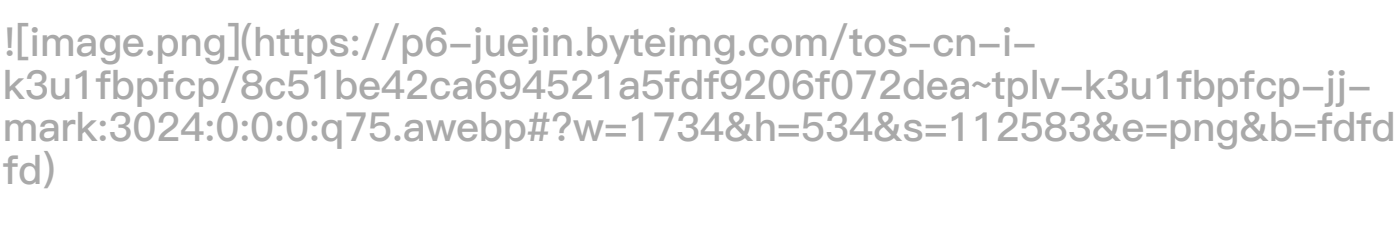


新建规则

选中所属APP，然后配置相应规则

```
[{
"key": "userId__", # key 名称, 结合 prefix 实现精准匹配或者模糊匹配
"prefix": true, # 是否前缀
"interval": "10", # 间隔时间(秒)
"threshold": "5", # 阈值 指在间隔时间内，访问次数超过阈值就会标记为热key
```

”duration”: ”300” # 生成热key 后 缓存时间(秒),默认60
}]

```



五、搭建客户端验证
---

### 搭建脚手架

主要引入springboot + mybatis，可以正常访问 mysql数据库 (也可以用 redis，验证框架是否可用，对数据源不限制)，脚手架地址 ：[start.aliyun.com/?spm=a2c6h....](http://cxyroad.com/”https://start.aliyun.com/?spm=a2c6h.12873639.article-detail.7.4c8b5f3bXNyjzB”)

### hotkey 客户端使用

#### 1、引入依赖

```
<dependency>
    <groupId>com.jd.platform.hotkey</groupId>
    <artifactId>hotkey-client</artifactId>
    <version>${version}</version>
</dependency>
```

#### 2、初始化HotKey

```
// appname 要跟规则对应上
@PostConstruct
```
```

```
public void initHotkey() {

    ClientStarter.Builder builder = new ClientStarter.Builder();
    ClientStarter starter =
builder.setAppName(”test_hot_key”).setEtcdServer(”http://127.0.0.1:237
9”).build();
    starter.startPipeline();
}
```

```

#### 3、使用方式



```
主要有如下4个方法可供使用

1 boolean isHotKey(String key) ，该方法会返回该key是否是热key，如果是返回true，如果不是返回false，并且会将key上报到探测集群进行数量计算。该方法通常用于判断只需要判断key是否热、不需要缓存value的场景，如刷子用户、接口访问频率等。
boolean JdHotKeyStore.isHotKey(String key)

2 Object get(String key)，该方法返回该key本地缓存的value值，可用于判断是热key后，再去获取本地缓存的value值，通常用于redis热key缓存
Object JdHotKeyStore.get(String key)

3 void smartSet(String key, Object value)，方法给热key赋值value，如果是热key，该方法才会赋值，非热key，什么也不做
void JdHotKeyStore.smartSet(String key, Object value)

4 Object getValue(String key)，该方法是一个整合方法，相当于isHotKey和get两个方法的整合，该方法直接返回本地缓存的value。 如果是热key，则存在两种情况，1是返回value，2是返回null。返回null是因为尚未给它set真正的value，返回非null说明已经调用过set方法了，本地缓存value有值了。 如果不是热key，则返回null，并且将key上报到探测集群进行数量探测。
Object JdHotKeyStore.getValue(String key)

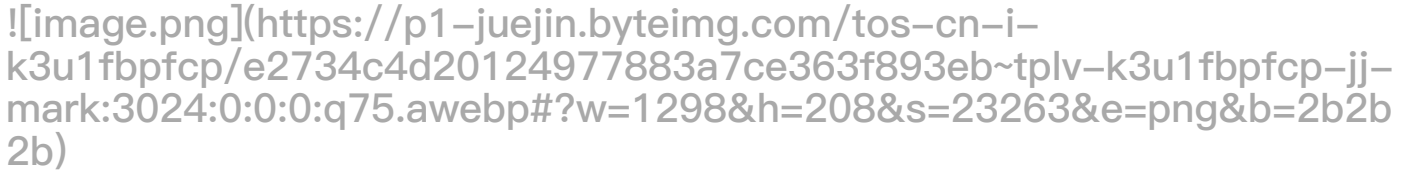
```

```

3、验证demo

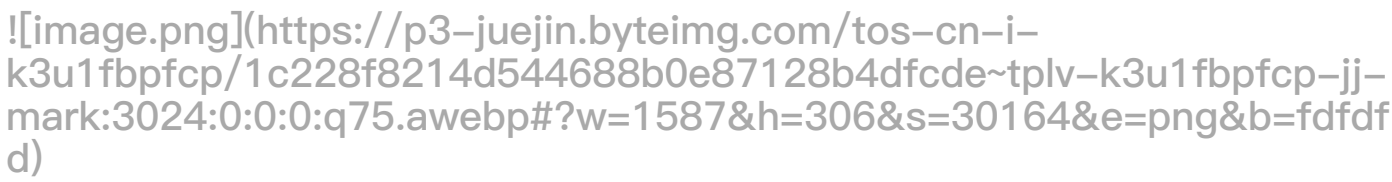

```
@Controller
public class PathVariableController {
```

```
    @Resource
    private UserMapper userMapper;

    public static volatile AtomicInteger count = new AtomicInteger(0);

    @RequestMapping(value = "/insert/{password}", method =
RequestMethod.GET)
    @ResponseBody
    public String testInsert(@PathVariable("password") String password) {
        UserDO user = new
UserDO().setUsername(UUID.randomUUID().toString())
                .setPassword(password).setCreateTime(new Date());
        userMapper.insert(user);
        return user.getId().toString();
    }

    @RequestMapping(value = "/selectById/{userId}", method =
RequestMethod.GET)
    @ResponseBody
    public UserDO testSelectById(@PathVariable("userId") Integer userId)
{
        System.out.println("累计请求多少次 count = " +
count.incrementAndGet());
        Object userInfo = JdHotKeyStore.getValue("userId_" + userId);
        if (userInfo == null) {
            System.out.println("通过数据库查询：" + count.get());
            UserDO theUserInfo = userMapper.selectById(userId);
            JdHotKeyStore.smartSet("userId_" + userId, theUserInfo);
            return theUserInfo;
        } else {
            System.out.println("热key探测查询：" + count.get());
            return (UserDO) userInfo;
        }
    }

}

```

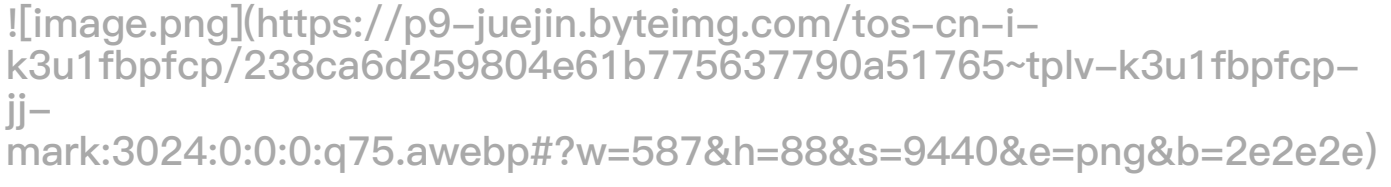
4、验证结果



可以看到第七次请求的时候，已经直接从jvm缓存中获取。



缓存过期后，重新查询，又开始从数据源获取源数据。



六、官方最佳实践

1 判断用户是否是刷子

```
if (JdHotKeyStore.isHotKey(“pin__” + thePin)) {
    //限流他，do your job
}
```

2 判断商品id是否是热点

```
Object skuInfo = JdHotKeyStore.getValue(”skuId__” + skuId);
if(skuInfo == null) {
    JdHotKeyStore.smartSet(”skuId__” + skuId, theSkuInfo);
} else {
    //使用缓存好的value即可
}

// 或者这样
 if (JdHotKeyStore.isHotKey(key)) {
```

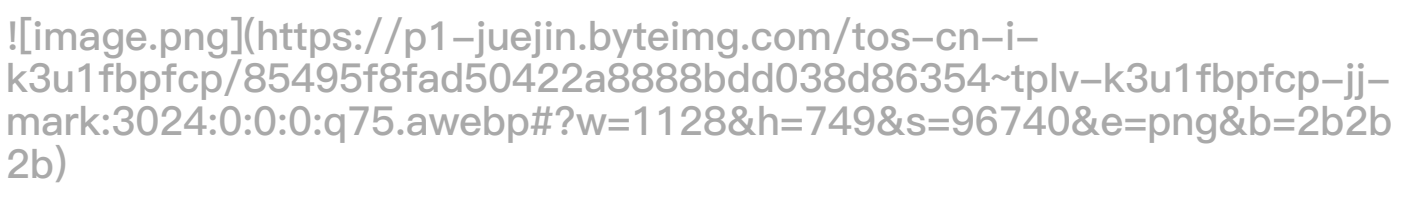
```
    //注意是get，不是getValue。getValue会获取并上报，get是纯粹的本地获取
    Object skuInfo = JdHotKeyStore.get("skuId__" + skuId);
    if(skuInfo == null) {
        JdHotKeyStore.smartSet("skuId__" + skuId, theSkuInfo);
    } else {
        //使用缓存好的value即可
    }

}
```

七、疑问

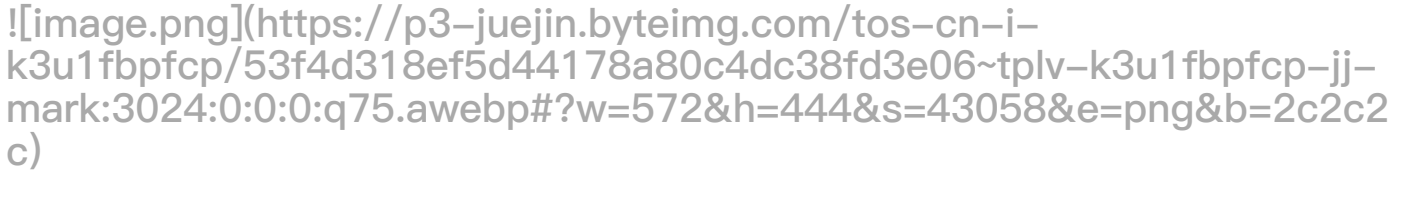
为什么不是超过阈值就直接从缓存获取?

hotkey统计是采用 批量推送的，间隔时间默认500ms，该值越小，上报热key越频繁，相应越及时，建议根据实际情况调整，如单机每秒qps10个，那么0.5秒上报一次即可，否则是空跑。该值最小为1，即1ms上报一次

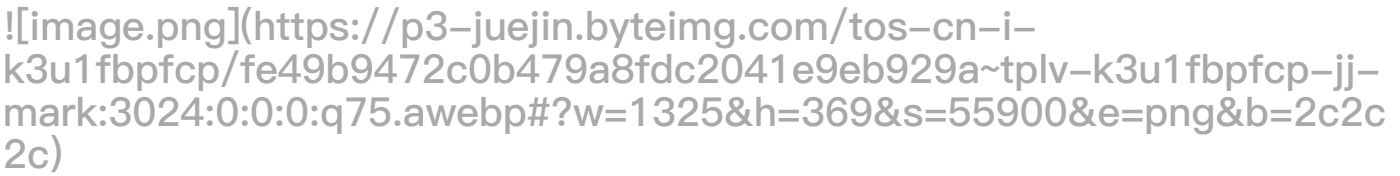
1、每次调用方法时，仅进行累计。



2、应用启动时初始化hotkey，会启动一个定时调度 com.jd.platform.hotkey.client.ClientStarter#startPipeline



com.jd.platform.hotkey.client.core.key.PushSchedulerStarter#startPusher 本质是使用 ScheduledExecutorService 周期线程池，根据设置的周期，进行数据上送



原文链接: https://juejin.cn/post/7352075813259411506