对缓存击穿的一点思考

张丰哲 (/u/cb569cce501b) +关注
2017.10.28 20:33 字数 772 阅读 326 评论 11 喜欢 31 赞赏 1
(/u/cb569cce501b)

前言

缓存（内存 or Memcached or Redis.....）在互联网项目中广泛应用，本篇博客将讨论下缓存击穿这一个话题，涵盖缓存击穿的现象、解决的思路、以及通过代码抽象方式来处理缓存击穿。

什么是缓存击穿？

```
@Service
public class CacheService {
    @Autowired
    private JedisCluster jedisCluster;
    @Autowired
    private StudentMapper studentMapper;
    public List<Student> query(){
        String key = "studentCache";
        String json = jedisCluster.get(key);
        if(json == null || "".equals(json) || "null".equalsIgnoreCase(json)){
            List<Student> studentList = studentMapper.findStudent();
            //加入缓存
            jedisCluster.set(key,JSON.toJSONString(studentList));
            //设置缓存的有效期
            jedisCluster.expire(key,60);
            return studentList;
        }
        return JSON.parseArray(json,Student.class);
    }

}
```

缓存击穿

上面的代码，是一个典型的写法：当查询的时候，先从Redis集群中取，如果没有，那么再从DB中查询并设置到Redis集群中。

注意，在实际开发中，我们一般在缓存中，存储的数据结构是JSON。（JDK提供的序列化方式效率稍微比JSON序列化低一些；而且JDK序列化非常严格，字段的增减，就很可能导致反序列失败，而JSON这方面兼容性较好）

假设从DB中查询需要2S，那么显然这段时间内过来的请求，在上述的代码下，会全部走DB查询，相当于缓存被直接穿透，这样的现象就称之为“缓存击穿”！

避免缓存击穿的思路分析

加synchronized？

```
public synchronized List<Student> query(){
    String key = "studentCache";
    String json = jedisCluster.get(key);
    if(json == null || "".equals(json) || "null".equalsIgnoreCase(json)){
        List<Student> studentList = studentMapper.findStudent();
        //加入缓存
        jedisCluster.set(key,JSON.toJSONString(studentList));
        //设置缓存的有效期
        jedisCluster.expire(key,60);
        return studentList;
    }
    return JSON.parseArray(json,Student.class);
}
```

同步方式

> **如果synchronized加在方法上，使得查询请求都得排队，本来我们的本意是让并发查询走缓存。也就是现在synchronized的粒度太大了。**

缩小synchronized的粒度？

```
public List<Student> query(){
    String key = "studentCache";
    String json = jedisCluster.get(key);
    if(json == null || "".equals(json) || "null".equalsIgnoreCase(json)){
        synchronized (this){
            List<Student> studentList = studentMapper.findStudent();
            //加入缓存
            jedisCluster.set(key,JSON.toJSONString(studentList));
            //设置缓存的有效期
            jedisCluster.expire(key,60);
            return studentList;
        }
    }
    return JSON.parseArray(json,Student.class);
}
```

缩小粒度

> **上面代码，在缓存有数据时，让查询缓存的请求不必排队，减小了同步的粒度。但是，仍然没有解决缓存击穿的问题。**
>
> **虽然，多个查询DB的请求进行排队，但是即便一个DB查询请求完成并设置到缓存中，其他查询DB的请求依然会继续查询DB！**

synchronized+双重检查机制

```
public List<Student> query(){
    String key = "studentCache";
    String json = jedisCluster.get(key);
    if(json == null || "".equals(json) || "null".equalsIgnoreCase(json)){
        synchronized (this){
            json = jedisCluster.get(key);
            if(json != null && !"".equals(json) && !"null".equalsIgnoreCase(json)){
                JSON.parseArray(json,Student.class);
            }
            List<Student> studentList = studentMapper.findStudent();
            //加入缓存
            jedisCluster.set(key,JSON.toJSONString(studentList));
            //设置缓存的有效期
            jedisCluster.expire(key,60);
            return studentList;
        }
    }
    return JSON.parseArray(json,Student.class);
}
```

双重检查

> 通过synchronized+双重检查机制：
>
> **在同步块中，继续判断检查，保证不存在，才去查DB。**

代码抽象

> **发现没有，其实我们处理缓存的代码，除了具体的查询DB逻辑外，其他都是模板化的。下面我们就来抽象下！**

一个查询DB的接口：

```
public interface CacheLoader<T>{
    public T load();
}
```

CacheLoader

> 既然查询具体的DB是由业务来决定的，那么暴露这个接口让业务去实现它。

一个模板：

```
@Repository
public class CacheTemplateService {

    @Autowired
    private JedisCluster jedisCluster;

    /**
     *
     * @param key 查询的KEY
     * @param seconds 设置到缓存的过期时间
     * @param clazz 返回的数据类型
     * @param cacheLoader 业务侧从DB加载数据的实现
     * @param <T>
     * @return
     */
    public <T> T findData(String key,int seconds,TypeReference<T> clazz,CacheLoader<T> cacheLoader){
        String json = jedisCluster.get(key);
        if(json == null || "".equals(json) || "null".equalsIgnoreCase(json)){
            synchronized (this){
                json = jedisCluster.get(key);
                if(json != null && !"".equals(json) && !"null".equalsIgnoreCase(json)){
                    JSON.parseArray(json, Student.class);
                }
                T t = cacheLoader.load();
                //加入缓存
                jedisCluster.set(key,JSON.toJSONString(t));
                //设置缓存的有效期
                jedisCluster.expire(key,seconds);
                return t;
            }
        }
        return JSON.parseObject(json,clazz);
    }

}
```

Template

> **Spring不是有很多Template类么？我们也可以通过这种思想对代码进行一个抽象，让外界来决定具体的业务实现，而把模板步骤写好。（有点类似AOP的概念）**

改进后的代码：

```java
public List<Student> query(){
    String key = "studentCache";
    String json = jedisCluster.get(key);
    if(json == null || "".equals(json) || "null".equalsIgnoreCase(json)){
        synchronized (this){
            json = jedisCluster.get(key);
            if(json != null && !"".equals(json) && !"null".equalsIgnoreCase(json)){
                JSON.parseArray(json,Student.class);
            }
            List<Student> studentList = studentMapper.findStudent();
            //加入缓存
            jedisCluster.set(key,JSON.toJSONString(studentList));
            //设置缓存的有效期
            jedisCluster.expire(key,60);
            return studentList;
        }
    }
    return JSON.parseArray(json,Student.class);
}

public List<Student> queryNew(){

    String key = "studentCache";
    List<Student> data = cacheTemplateService.findData(key, 60, new TypeReference<List<Student>>() {
    }, new CacheLoader<List<Student>>() {
        @Override
        public List<Student> load() {
            return studentMapper.findStudent();
        }
    });
    return  data;
}
```

改进后的调用代码

> **从这里可以看出，我们并不关心缓存的数据从哪里加载，而是交给具体的使用方，而且使用方在使用时再也不必关注缓存击穿的问题，因为我们都给抽象了。**

好了，到这里，关于缓存击穿就讨论到这里。

2017.10.28 zhangfengzhe

轩辕宇虹 (/u/cdceb286d024)
2楼 · 2017.10.29 17:31
(/u/cdceb286d024)
这篇文章写得真好，周五晚上去网易面试，二面，正好考察到这个问题，虽然换了种方式去问～不过，synchronized+双重检查机制，确实考察到了，由于我个人对缓存木有太多应用，现在看到这篇文章，豁然开朗□□

赞 回复

> 张丰哲 (/u/cb569cce501b)： 常来看看~
> 2017.10.29 21:37 回复
>
> 轩辕宇虹 (/u/cdceb286d024)： @张丰哲 (/users/cb569cce501b) 你的文章很好，我基本都看完了，MQ系列，多分享一些噢□□
> 2017.10.30 07:58 回复
>
> 张丰哲 (/u/cb569cce501b)： @轩辕宇虹 (/users/cdceb286d024) 恩恩，持续更新，
> 2017.10.30 11:25 回复
>
> 添加新评论

那个码农 (/u/f11440d436ec)
3楼 · 2017.10.29 23:11
(/u/f11440d436ec)
双重检查是不是少了个return。□□

赞 回复

> 张丰哲 (/u/cb569cce501b)： 恩恩，火眼金睛，谢谢指出，
> 2017.10.30 11:24 回复
>
> 添加新评论

秋田君l (/u/2bb84e08d233)
4楼 · 2017.10.30 08:50
(/u/2bb84e08d233)
赞一个□，看到cacheloader突然想起来guava好像也有一个

赞 回复

> 张丰哲 (/u/cb569cce501b)： 恩恩，是的，guava的本地缓存很好用~
> 2017.10.30 11:25 回复
>
> 添加新评论

藤伦柳揶 (/u/56a64219b193)
5楼 · 2017.10.31 09:07
(/u/56a64219b193)
越看越像guava的cache了

赞 回复

你的小李 (/u/4f448009f2d9)
6楼 · 2017.10.31 18:27
(/u/4f448009f2d9)
学习了□

赞 回复

xiangtan (/u/15d31a7cfdaa)
7楼 · 2017.11.01 11:19
(/u/15d31a7cfdaa)

写得很棒！给您赏一个

赞 回复

被以下专题收入，发现更多相似内容

 程序员 (/c/NEt52a?utm_source=desktop&utm_medium=notes-included-collection)

 @IT·互联网 (/c/V2CqjW?utm_source=desktop&utm_medium=notes-included-collection)

 缓存 (/c/efd45005e307?utm_source=desktop&utm_medium=notes-included-collection)