

## 31 | Guarded Suspension模式：等待唤醒机制的规范实现

2019-05-09 王宝令

Java并发编程实战

[进入课程 >](#)

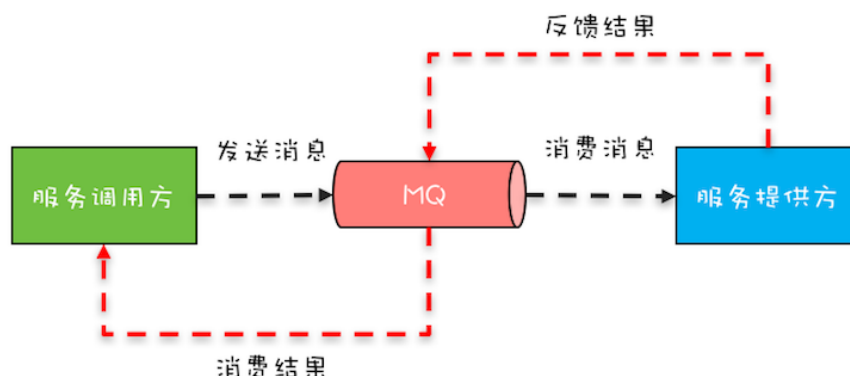


讲述：王宝令

时长 08:25 大小 7.72M




前不久，同事小灰工作中遇到一个问题，他开发了一个 Web 项目：Web 版的文件浏览器，通过它用户可以在浏览器里查看服务器上的目录和文件。这个项目依赖运维部门提供的文件浏览服务，而这个文件浏览服务只支持消息队列（MQ）方式接入。消息队列在互联网大厂中用的非常多，主要用作流量削峰和系统解耦。在这种接入方式中，发送消息和消费结果这两个操作之间是异步的，你可以参考下面的示意图来理解。



消息队列（MQ）示意图

在小灰的这个 Web 项目中，用户通过浏览器发过来一个请求，会被转换成一个异步消息发送给 MQ，等 MQ 返回结果后，再将这个结果返回至浏览器。小灰同学的问题是：给 MQ 发送消息的线程是处理 Web 请求的线程 T1，但消费 MQ 结果的线程并不是线程 T1，那线程 T1 如何等待 MQ 的返回结果呢？为了便于你理解这个场景，我将其代码化了，示例代码如下。

 复制代码

```
1 class Message{
2     String id;
3     String content;
4 }
5 // 该方法可以发送消息
6 void send(Message msg){
7     // 省略相关代码
8 }
9 //MQ 消息返回后会调用该方法
10 // 该方法的执行线程不同于
11 // 发送消息的线程
12 void onMessage(Message msg){
13     // 省略相关代码
14 }
15 // 处理浏览器发来的请求
16 Respond handleWebReq(){
17     // 创建一消息
18     Message msg1 = new
19         Message("1","{...}");
20     // 发送消息
21     send(msg1);
22     // 如何等待 MQ 返回的消息呢？
23     String result = ...;
24 }
```

看到这里，相信你一定有点似曾相识的感觉，这不就是前面我们在[《15 | Lock 和 Condition（下）：Dubbo 如何用管程实现异步转同步？》](#)中曾介绍过的异步转同步问题吗？仔细分析，的确是这样，不过在那一篇文章中我们只是介绍了最终方案，让你知其然，但是并没有介绍这个方案是如何设计出来的，今天咱们再仔细聊聊这个问题，让你知其所以然，遇到类似问题也能自己设计出方案来。

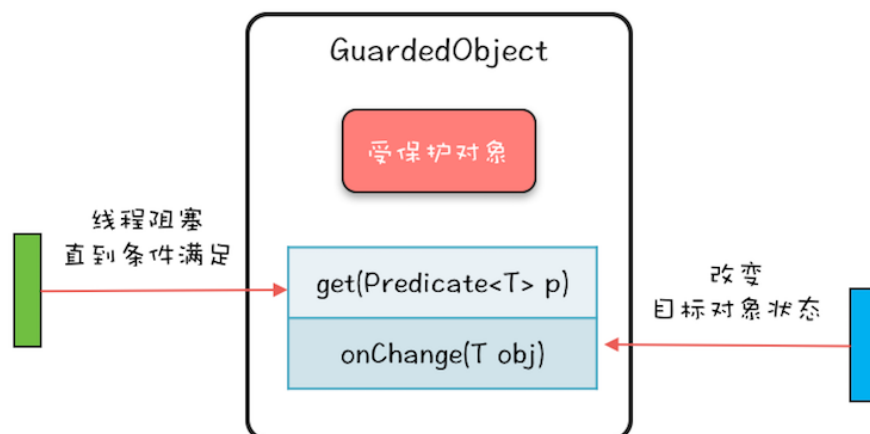
## Guarded Suspension 模式

上面小灰遇到的问题，在现实世界里比比皆是，只是我们一不小心就忽略了。比如，项目团建要外出聚餐，我们提前预订了一个包间，然后兴冲冲地奔过去，到那儿后大堂经理看了一眼包间，发现服务员正在收拾，就会告诉我们：“您预订的包间服务员正在收拾，请您稍等片刻。”过了一会，大堂经理发现包间已经收拾完了，于是马上带我们去包间就餐。

我们等待包间收拾完这个过程和小灰遇到的等待 MQ 返回消息本质上是一样的，都是**等待一个条件满足**：就餐需要等待包间收拾完，小灰的程序里要等待 MQ 返回消息。

那我们来看看现实世界里是如何解决这类问题的呢？现实世界里大堂经理这个角色很重要，我们是否等待，完全是由他来协调的。通过类比，相信你也一定有思路了：我们的程序里，也需要这样一个大堂经理。的确是这样，那程序世界里的大堂经理该如何设计呢？其实设计方案前人早就搞定了，而且还将其总结成了一个设计模式：**Guarded Suspension**。所谓 Guarded Suspension，直译过来就是“保护性地暂停”。那下面我们就来看看，Guarded Suspension 模式是如何模拟大堂经理进行保护性地暂停的。

下图就是 Guarded Suspension 模式的结构图，非常简单，一个对象 GuardedObject，内部有一个成员变量——受保护的對象，以及两个成员方法——`get(Predicate<T> p)` 和 `onChanged(T obj)` 方法。其中，对象 GuardedObject 就是我们前面提到的大堂经理，受保护对象就是餐厅里面的包间；受保护对象的 `get()` 方法对应的是我们的就餐，就餐的前提条件是包间已经收拾好了，参数 `p` 就是用来描述这个前提条件的；受保护对象的 `onChanged()` 方法对应的是服务员把包间收拾好了，通过 `onChanged()` 方法可以 fire 一个事件，而这个事件往往能改变前提条件 `p` 的计算结果。下图中，左侧的绿色线程就是需要就餐的顾客，而右侧的蓝色线程就是收拾包间的服务员。



Guarded Suspension 模式结构图

GuardedObject 的内部实现非常简单，是管程的一个经典用法，你可以参考下面的示例代码，核心是：get() 方法通过条件变量的 await() 方法实现等待，onChange() 方法通过条件变量的 signalAll() 方法实现唤醒功能。逻辑还是很简单的，所以这里就不再详细介绍了。

复制代码

```

1 class GuardedObject<T>{
2     // 受保护的對象
3     T obj;
4     final Lock lock =
5         new ReentrantLock();
6     final Condition done =
7         lock.newCondition();
8     final int timeout=1;
9     // 获取受保护对象
10    T get(Predicate<T> p) {
11        lock.lock();
12        try {
13            //MESA 管程推荐写法
14            while(!p.test(obj)){
15                done.await(timeout,
16                    TimeUnit.SECONDS);
17            }
18        }catch(InterruptedException e){
19            throw new RuntimeException(e);
20        }finally{
21            lock.unlock();
22        }
23        // 返回非空的受保护对象

```

```

24     return obj;
25 }
26 // 事件通知方法
27 void onChanged(T obj) {
28     lock.lock();
29     try {
30         this.obj = obj;
31         done.signalAll();
32     } finally {
33         lock.unlock();
34     }
35 }
36 }

```

## 扩展 Guarded Suspension 模式

上面我们介绍了 Guarded Suspension 模式及其实现，这个模式能够模拟现实世界里大堂经理的角色，那现在我们再来看看这个“大堂经理”能否解决小灰同学遇到的问题。

Guarded Suspension 模式里 GuardedObject 有两个核心方法，一个是 get() 方法，一个是 onChanged() 方法。很显然，在处理 Web 请求的方法 handleWebReq() 中，可以调用 GuardedObject 的 get() 方法来实现等待；在 MQ 消息的消费方法 onMessage() 中，可以调用 GuardedObject 的 onChanged() 方法来实现唤醒。

 复制代码

```

1 // 处理浏览器发来的请求
2 Respond handleWebReq(){
3     // 创建一消息
4     Message msg1 = new
5         Message("1","{...}");
6     // 发送消息
7     send(msg1);
8     // 利用 GuardedObject 实现等待
9     GuardedObject<Message> go
10         =new GuardObjec<>();
11     Message r = go.get(
12         t->t != null);
13 }
14 void onMessage(Message msg){
15     // 如何找到匹配的 go?
16     GuardedObject<Message> go=???
17     go.onChanged(msg);
18 }


```



但是在实现的时候会遇到一个问题，handleWebReq() 里面创建了 GuardedObject 对象的实例 go，并调用其 get() 方等待结果，那在 onMessage() 方法中，如何才能找到匹配的 GuardedObject 对象呢？这个过程类似服务员告诉大堂经理某某包间已经收拾好了，大堂经理如何根据包间找到就餐的人。现实世界里，大堂经理的头脑中，有包间和就餐人之间的关系图，所以服务员说完之后大堂经理立刻就能把就餐人找出来。

我们可以参考大堂经理识别就餐人的办法，来扩展一下 Guarded Suspension 模式，从而使它能够很方便地解决小灰同学的问题。在小灰的程序中，每个发送到 MQ 的消息，都有一个唯一性的属性 id，所以我们可以维护一个 MQ 消息 id 和 GuardedObject 对象实例的关系，这个关系可以类比大堂经理大脑里维护的包间和就餐人的关系。

有了这个关系，我们来看看具体如何实现。下面的示例代码是扩展 Guarded Suspension 模式的实现，扩展后的 GuardedObject 内部维护了一个 Map，其 Key 是 MQ 消息 id，而 Value 是 GuardedObject 对象实例，同时增加了静态方法 create() 和 fireEvent()；create() 方法用来创建一个 GuardedObject 对象实例，并根据 key 值将其加入到 Map 中，而 fireEvent() 方法则是模拟的大堂经理根据包间找就餐人的逻辑。

 复制代码


```
1 class GuardedObject<T>{
2     // 受保护的對象
3     T obj;
4     final Lock lock =
5         new ReentrantLock();
6     final Condition done =
7         lock.newCondition();
8     final int timeout=2;
9     // 保存所有 GuardedObject
10    final static Map<Object, GuardedObject>
11    gos=new ConcurrentHashMap<>();
12    // 静态方法创建 GuardedObject
13    static <K> GuardedObject
14        create(K key){
15        GuardedObject go=new GuardedObject();
16        gos.put(key, go);
17        return go;
18    }
19    static <K, T> void
20        fireEvent(K key, T obj){
21        GuardedObject go=gos.remove(key);
22        if (go != null){
23            go.onChanged(obj);
```

```

24     }
25 }
26 // 获取受保护对象
27 T get(Predicate<T> p) {
28     lock.lock();
29     try {
30         //MESA 管程推荐写法
31         while(!p.test(obj)){
32             done.await(timeout,
33                 TimeUnit.SECONDS);
34         }
35     }catch(InterruptedException e){
36         throw new RuntimeException(e);
37     }finally{
38         lock.unlock();
39     }
40     // 返回非空的受保护对象
41     return obj;
42 }
43 // 事件通知方法
44 void onChanged(T obj) {
45     lock.lock();
46     try {
47         this.obj = obj;
48         done.signalAll();
49     } finally {
50         lock.unlock();
51     }
52 }
53 }

```

这样利用扩展后的 `GuardedObject` 来解决小灰同学的问题就很简单了，具体代码如下所示。

 复制代码

```

1 // 处理浏览器发来的请求
2 Respond handleWebReq(){
3     int id= 序号生成器.get();
4     // 创建一消息
5     Message msg1 = new
6         Message(id,"{...}");
7     // 创建 GuardedObject 实例
8     GuardedObject<Message> go=
9         GuardedObject.create(id);
10    // 发送消息
11    send(msg1);
12    // 等待 MQ 消息

```

```
13     Message r = go.get(
14         t->t != null);
15 }
16 void onMessage(Message msg){
17     // 唤醒等待的线程
18     GuardedObject.fireEvent(
19         msg.id, msg);
20 }
```


## 总结

Guarded Suspension 模式本质上是一种等待唤醒机制的实现，只不过 Guarded Suspension 模式将其规范化了。规范化的好处是你无需重头思考如何实现，也无需担心实现程序的可理解性问题，同时也能避免一不小心写出个 Bug 来。但 Guarded Suspension 模式在解决实际问题的時候，往往还是需要扩展的，扩展的方式有很多，本篇文章就直接对 GuardedObject 的功能进行了增强，Dubbo 中 DefaultFuture 这个类也是采用的这种方式，你可以对比着来看，相信对 DefaultFuture 的实现原理会理解得更透彻。当然，你也可以创建新的类来实现对 Guarded Suspension 模式的扩展。

Guarded Suspension 模式也常被称作 Guarded Wait 模式、Spin Lock 模式（因为使用了 while 循环去等待），这些名字都很形象，不过它还有一个更形象的非官方名字：多线程版本的 if。单线程场景中，if 语句是不需要等待的，因为在只有一个线程的条件下，如果这个线程被阻塞，那就没有其他活动线程了，这意味着 if 判断条件的结果也不会发生变化了。但是多线程场景中，等待就变得有意义了，这种场景下，if 判断条件的结果是可能发生变化的。所以，用“多线程版本的 if”来理解这个模式会更简单。

## 课后思考

有同学觉得用 done.await() 还要加锁，太啰嗦，还不如直接使用 sleep() 方法，下面是他的实现，你觉得他的写法正确吗？

 复制代码

```
1 // 获取受保护对象
2 T get(Predicate<T> p) {
3     try {
4         while(!p.test(obj)){
5             TimeUnit.SECONDS
6                 .sleep(timeout);
7         }
8     }
```



```
8     }catch(InterruptedException e){
9         throw new RuntimeException(e);
10    }
11    // 返回非空的受保护对象
12    return obj;
13 }
14 // 事件通知方法
15 void onChanged(T obj) {
16     this.obj = obj;
17 }
```

欢迎在留言区与我分享你的想法，也欢迎你在留言区记录你的思考过程。感谢阅读，如果你觉得这篇文章对你有帮助的话，也欢迎把它分享给更多的朋友。



# Java 并发编程实战

全面系统提升你的并发编程能力

王宝令  
资深架构师



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 30 | 线程本地存储模式：没有共享，就没有伤害

下一篇 32 | Balking模式：再谈线程安全的单例模式



青莲

2019-05-09

👍 10

sleep 无法被唤醒，只能时间到后自己恢复运行，当真正的条件满足了，时间未到，接着睡眠，无性能可言

---



刘章周

2019-05-09

👍 6

当从消息队列接收消息失败时，while循环会一直执行下去，永远不会结束，回占用大量资源。

作者回复: 👍



zhangwei

2019-05-18

👍 4

老师，我有个疑问，希望帮忙解答。如果Web应用是集群的，A节点处理HTTP请求后发了MQ，B节点的onMessage消费了回执消息，那么A节点怎么把结果响应给客户端呢？疑问好久了，希望老师给个思路，谢谢！

展开 ▾

作者回复: 我了解有人是这么做的：把回执消息放到redis的list中，按照ip重新分组之后从redis中再次消费。

也可以按照ip建立不同的topic。



Felix Env...

2019-05-12

👍 4

老师，感觉如果有方法调用了GuardedObect.create方法但是没有任何其他线程调用fireEvent方法会造成内存泄漏啊，这种情况需要考虑吗？

作者回复: 👍 需要，等待超时后要把他移除。



Mr.Brooks

2019-05-10

👍 3

没有锁也无法保证内存可见性吧

展开 ▾

作者回复: 📬



张三

2019-05-10

👍 2

接入微信支付支付宝支付里边，也需要提供一个回调函数，onChange()就是一个回调函数吧，不过微信支付宝支付是异步回调，是不是也可以改成这种？微信支付宝里边的其它第三方支付是不是就是这种模式，因为支付成功之后跳转到它们自己的页面，而不是微信支付支付宝官方的支付成功界面

作者回复: 这个回调函数和mq的回调函数从服务接入方的角度看是一样的



zero

2019-05-09

👍 2

wait会释放占有的资源，sleep不会释放

展开 ▾



null

2019-06-03

👍 1

老师，您好！

我想到了一个场景：线程 t1 提交了消息 m1，线程 t2 提交了消息 m2，此时都在 get() 方法处等待结果返回。m2 先被处理完，this.obj 对应的是消息 m2 的结果，调用 fireEvent() 唤醒 t1 和 t2，t1 竞争到锁资源，消费了 m2 的结果 this.obj。

...

展开 ▾

作者回复: 只要唤醒的时候能找到正确的线程就可以，不知道你的方法是不是能做到



Rancood

👍 1

2019-05-22

这个模式了解了，但是实例中业务有点懵，handleWebReq方法最终拿到的是自己发送出去的消息；是不是应该在onMessage方法唤醒等待线程之前进行业务处理，生成新的Message消息newMsg，然后把newMsg传到fireEvent里面，这样拿到的是反馈的结果。请老师指点一下



晓杰

2019-05-09

👍 1

用sleep的话只能等睡眠时间到了之后再返回while循环条件去判断，但是wait相当于和singal组成等待唤醒的机制，这样满足条件的概率更大一些，性能也更好

展开 ∨



朱晋君

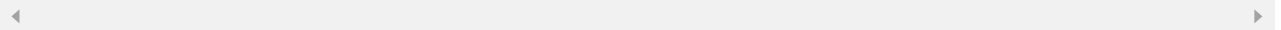
2019-05-09

👍 1

如果以文中的最后一段示例代码来看，每一个请求生成一个id，对应一个GuardedObject，并没有线程安全问题。我觉得可以去掉锁。但是加sleep的话，没有办法唤醒，只能等到超时。

展开 ∨

作者回复: await和notify获取锁才能调用，所以不能去掉锁



linqw

2019-05-25

👍

总结：Guarded Suspension模式，要解决的是，发送消息的线程和消费消息结果的线程不是同一个，但是消息结果又需要由发送的线程进行处理，为此需要为每个消息创建出类似大堂经理，生活中一般是只有一个大堂经理，但是在编程世界里需要为每个分配一个大堂经理，大堂经理主要做的事情就是发送线程发送完消息时，将其阻塞，提供消息结果的回调接口，通知阻塞的发送线程消费消息结果。...

展开 ∨



cricket198...

2019-05-24

👍

想问一下分布式环境下，异步转同步的方法有哪些？例如，数据服务部署多个instance，客户在Web UI上点击外部数据源试用，后端通过一个数据服务instance请求外部数据源，

外部数据源会异步回调结果(LB地址)返回，怎么样将结果显示在请求数据服务的Web UI上？客户试用的过程是同步的，但请求外部数据源操作流程是异步的。谢谢！

展开 ∨

作者回复: websocket可以吗，没了解到你说的难点在哪里

◀ ▶



andy

2019-05-20



我有个疑惑就是，这里所说的MQ可以是RabbitMQ或者是其他类型的MQ吗？还是说这个MQ其实就JAVA中的一个数据结构？

作者回复: 前者

◀ ▶



三木禾

2019-05-19



老师，您这代码能不能写的完整一点啊，前面有个onMessage ,后面有个onChange

作者回复: onmessage调用 onchange,他们是两个类里的方法

◀ ▶



Zach\_

2019-05-15



await(timeout); 本来是去等待队列里面呆timeout长的时间，如果到timeout之前没有被signal就出队;如果被signal了就是机制的正常运行。最终都return obj;

sleep(timeout); 守护条件不成立就sleep timeout, 还不成立再sleep timeout, ... 如果在发生死锁或者onchange失败的情况下， get()方法会因为状态改变的失败而一直 判断 slee...

展开 ∨



Zach\_

2019-05-15



额 我看有个同学说 可见性也无法保证。

锁是保证可见性的嘛？我记得锁是保证原子性的哇，可见性不是volatile变量来保证的嘛？

---



**Geek\_ebda9...**

2019-05-14



sleep方法，sleep期间，不会释放对象的锁，线程T1执行get，获取的lock在sleep期间他的锁不会释放，如果t2线程去执行onchanged方法，这时候获取不到lock的锁的，会导致程序死锁，

await方法，线程T1执行get,在等待期间，他会释放掉锁，这时t2执行onchanged，可以获取到lock然后给obj赋值，get1就可以获取到最新的obj...

展开 ∨

---



**佑儿**

2019-05-14



学习心得：每一种设计模式都适应某种场景，个人认为Guarded Suspension模式的根本在于get数据的时候一定能够获取，如果获取不到就等待，保护性暂停也就是这个意思，阻塞队列大部分都是这种模式。

展开 ∨

---



**Kaleidosco...**

2019-05-14



想问下老师，Create和fireEvent方法不加锁可以么，像Create方法不加锁指令重排优化，先返回object再加进map里会不会有问题

作者回复: 线程安全的容器能避免你说的这种情况

