深入理解JVM（二）——内存模型、可见性、指令重排序

2017-08-14 09:16 by 飘扬的红领巾, 7584 阅读, 1 评论, 收藏, 编辑

上一篇我们介绍了JVM的基本运行流程以及内存结构，对JVM有了初步的认识，这篇文章我们将根据JVM的内存模型探索java当中变量的可见性以及不同的java指令在并发时可能发生的指令重排序的情况。

内存模型

首先我们思考一下一个java线程要向另外一个线程进行通信，应该怎么做，我们再把需求明确一点，一个java线程对一个变量的更新怎么通知到另外一个线程呢？我们知道java当中的实例对象、数组元素都放在java堆中，java堆是线程共享的。（我们这里把java堆称为主内存），而每一个线程都是自己私有的内存空间（称为工作内存），如果线程1要向线程2通信，一定会经过类似的流程：

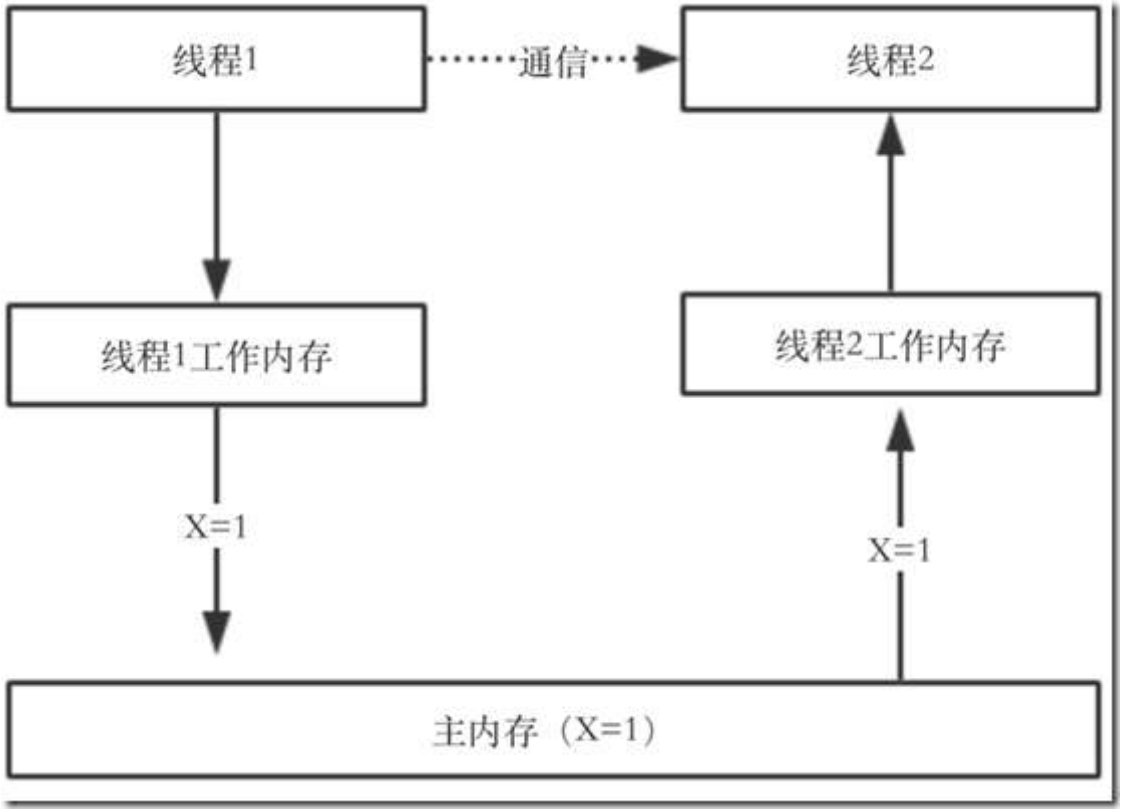
About

李平，目前在一家O2O互联网公司从事设计、开发工作。业余时间喜欢跑步、看书、游戏。

喜欢简单而高效的工作环境，熟悉JavaEE、SOA、数据库架构、优化、系统运维，有大型门户网站，金融系统建设经验。RHCE、MySQL OCP。MyCAT开源项目成员。

我的开源项目：
mycat-eye
nosql-eye

昵称：飘扬的红领巾
园龄：6年9个月
荣誉：推荐博客
粉丝：910
关注：0
+加关注



1、 线程1将自己工作内存中的X更新为1并刷新到主内存中；

2、 线程2从主内存读取变量X=1，更新到自己的工作内存中，从而线程2读取的X就是线程1更新后的值。

从上面的流程看出线程之间的通信都需要经过主内存，而主内存与工作内存的交互，则需要Java内存模型（JMM）来管理器。下图演示了JMM如何管理主内存和工作内存：

Re:InnoDB一棵B+树可以存放多少行数据?
好文 -- icycheng

Re:深入理解JVM（八）——java堆分析
@zhoumy 应该还有其他对象占用空间吧，比如这个类的一些元数据 -- xiaoli2333

Re:大型网站的灵魂——性能
mark -- xiaoli2333

Re:深入理解JVM（七）——性能监控工具
mark -- xiaoli2333

Re:MySQL在并发场景下的问题及解决思路
大神在吗，怎么联系你啊 -- duchaochen

日历

< 2018年8月 >

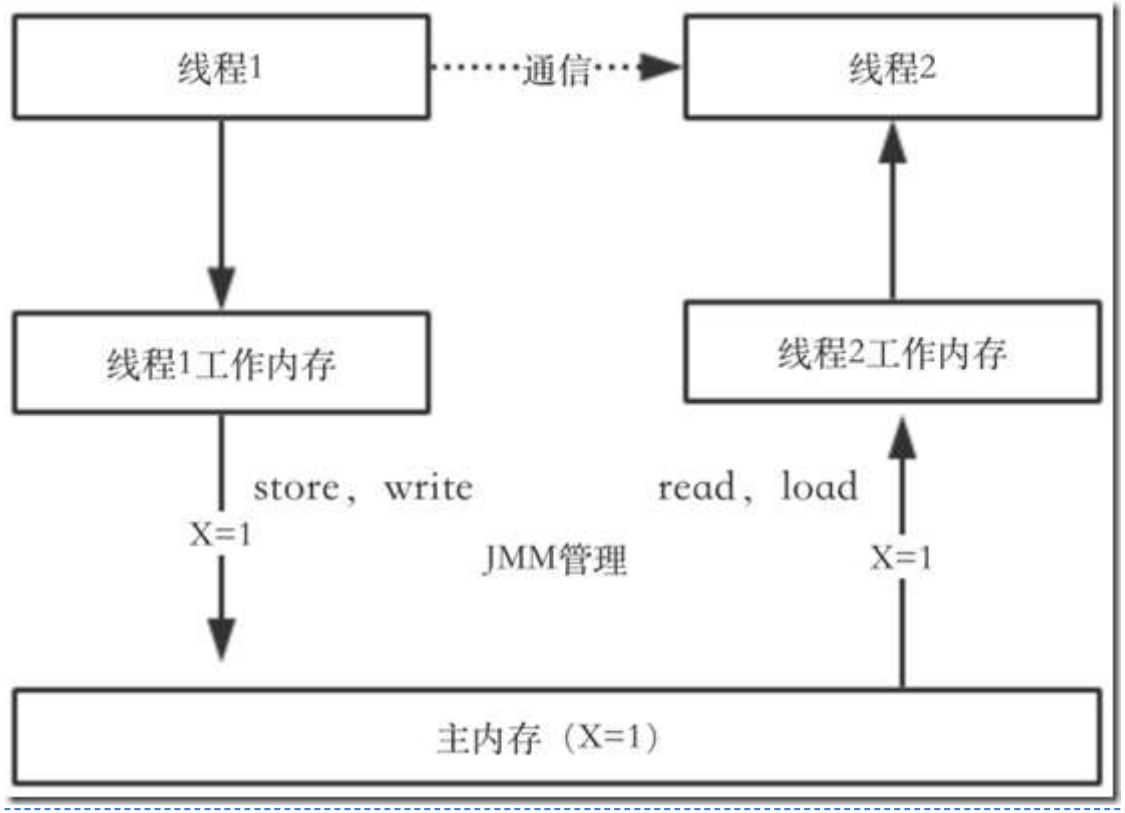
日	一	二	三	四	五	六
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

随笔档案

2018年1月(2)
2017年10月(1)
2017年9月(4)
2017年8月(7)
2015年6月(1)
2015年1月(2)
2014年10月(2)
2014年9月(2)
2014年5月(1)
2014年3月(2)
2014年1月(1)
2013年9月(1)
2013年8月(2)
2013年5月(1)
2013年4月(1)
2013年3月(1)

我的标签

Maven(3)
Jenkins(2)
Nexus(2)
Sonar(2)
Svn(2)
Tomcat(2)



当线程1需要将一个更新后的变量值刷新到主内存中时，需要经过两个步骤：

1、 工作内存执行store操作；

2、 主内存执行write操作；

完成这两步即可将工作内存中的变量值刷新到主内存，即线程1工作内存和主内存的变量值保持一致；

当线程2需要从主内存中读取变量的最新值时，同样需要经过两个步骤：

1、主内存执行read操作，将变量值从主内存中读取出来；

2、工作内存执行load操作，将读取出来的变量值更新到本地内存的副本；

完成这两步，线程2的变量和主内存的变量值就保持一致了。

并发(1)
并发 乐观锁 悲观锁(1)
大型网站(1)
代码质量 Checkstyle PMD JDepend Eclemma Metric(1)
更多

随笔分类

Apache Mina(1)
Eclipse(1)
Hibernate(2)
Java(19)
JVM(8)
MongoDB(2)
MySQL(4)
RCP/SWT/Jface(1)
SOA(1)
Spring(3)
持续集成(4)
大型网站(3)
多线程(1)
开源项目(2)
敏捷(1)
其他(7)
设计模式(1)
数据结构/算法(1)
系统架构(3)
支付(1)

2012年12月(1)
2012年11月(1)
2012年9月(1)
2012年6月(2)
2012年5月(4)
2012年3月(1)

可见性

Java中有一个关键字volatile，它有什么用呢？这个答案其实就在上述java线程间通信机制中，我们想象一下，由于工作内存这个中间层的出现，线程1和线程2必然存在延迟的问题，例如线程1在工作内存中更新了变量，但还没刷新到主内存，而此时线程2获取到的变量值就是未更新的变量值，又或者线程1成功将变量更新到主内存，但线程2依然使用自己工作内存中的变量值，同样会出问题。不管出现哪种情况都可能导致线程间的通信不能达到预期的目的。例如以下例子：

```
//线程1 boolean stop = false; while(!stop){ doSomething(); } //线程2

stop
= true;
```

这个经典的例子表示线程2通过修改stop的值，控制线程1中断，但在真实环境中可能会出现意想不到的结果，线程2在执行之后，线程1并没有立刻中断甚至一直不会中断。出现这种现象的原因就是线程2对线程1的变量更新无法第一时间获取到。

但这一切等到Volatile出现后，再也不是问题，Volatile保证两件事：

1、 线程1工作内存中的变量更新会强制立即写入到主内存；

2、 线程2工作内存中的变量会强制立即失效，这使得线程2必须去主内存中获取最新的变量值。

所以这就理解了Volatile保证了变量的可见性，因为线程1对变量的修改能第一时间让线程2可见。

指令重排序

关于指令排序我们先看一段代码：

重构(1)

推荐排行榜

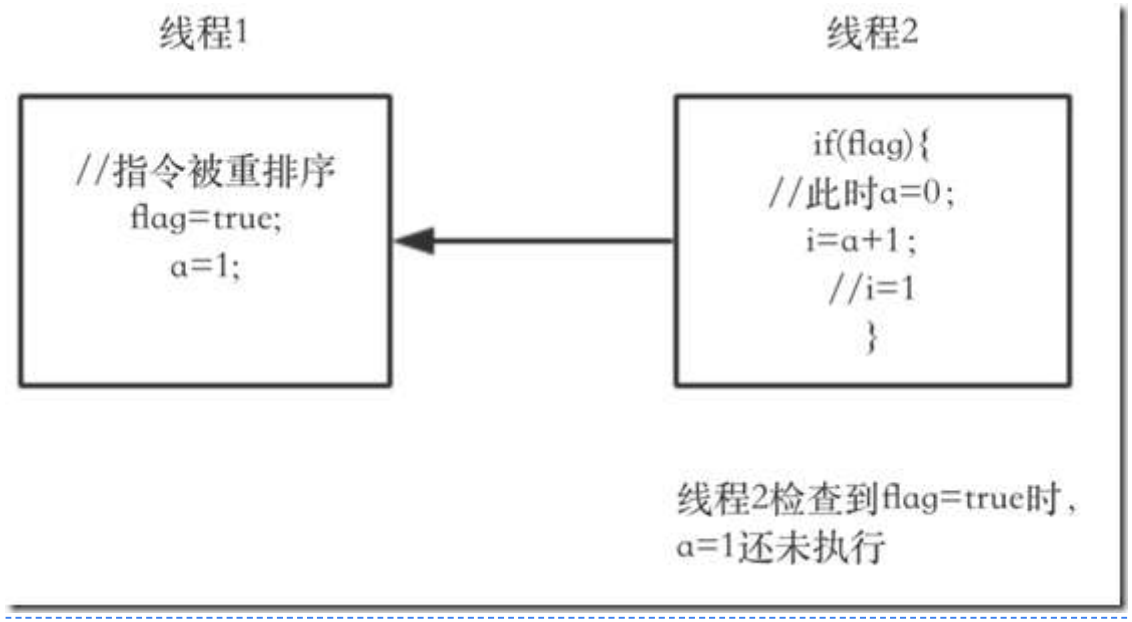
1. 大型网站系统架构的演化(211)
2. 大型网站的灵魂——性能(63)
3. 电商系统中的商品模型的分析与设计—续(51)
4. 电商系统中的商品模型的分析与设计(47)
5. 做了两款数据库监控工具，打算在近期开源(39)

阅读排行榜

1. 大型网站系统架构的演化(51186)
2. 深入理解JVM（一）——基本原理(34265)
3. 电商系统中的商品模型的分析与设计(16784)
4. 大型网站的灵魂——性能(15766)
5. 使用Maven+Nexus+Jenkins+Svn+Tomcat+Sonar搭建持续集成环境（一）(15198)

```
int a = 0;boolean flag = false;

//线程1

public void writer() {

a = 1;

flag = true;

}

//线程2

public void reader() {

if (flag) {

int i= a+1;

..... }

}
```

线程1依次执行a=1,flag=true；线程2判断到flag==true后，设置i=a+1，根据代码语义，我们可能会推断此时i的值等于2，因为线程2在判断flag==true时，线程1已经执行了a=1；所以i的值等于a+1=1+1=2；但真实情况却不一定如此，引起这个问题的原因是线程1内部的两条语句a=1；flag=true；可能被重新排序执行，如图：



这就是指令重排序的简单演示，两个赋值语句尽管他们的代码顺序是一前一后，但真正执行时却不一定按照代码顺序执行。你可能会说，有这个指令重排序那不是乱套了吗？我写的程序都不按我的代码流程走，这怎么玩？这个你可以放心，你的程序不会乱套，因为java和CPU、内存之间都有一套严格的指令重排序规则，哪些可以重排，哪些不能重排都有规矩的。下列流程演示了一个java程序从编译到执行会经历哪些重排序：

在这个流程中第一步属于编译器重排查，编译器重排序会按JMM的规范严格进行，换言之编译器重排序一般不会对程序的正确逻辑造成影响。第二、三步属于处理器重排序，处理器重排序JMM就不好管了，怎么办呢？它会要求java编译器在生成指令时加入内存屏障，内存屏障是什么？你可以理解为一个不透风的保护罩，把不能重排序的java指令保护起来，那么处理器在遇到内存屏障保护的指令时就不会对它进行重排序了。关于在哪些地方该加入内存屏障，内存屏障有哪些种类，各有什么作用，这些知识点这里就不再阐述了。可以参考JVM规范相关资料。

下面介绍一下在同一个线程中，不会被重排序的逻辑：

名称	代码示例	说明
写后读	a = 1; b = a;	写一个变量之后，再读这个位置。
写后写	a = 1; a = 2;	写一个变量之后，再写这个变量。
读后写	a = b; b = 1;	读一个变量之后，再写这个变量。

这三种情况中，任意改变一个代码的顺序，结果都会大不相同，对于这样的逻辑代码，是不会被重排序的。注意这是指单线程中不会被重排序，如果在多线程环境下，还是会产生逻辑问题，例如我们一开始举的例子。

结语

本文简单介绍了java在实现线程间通信时的简单原理，并介绍了volatile关键字的作用，最后介绍了java当中可能会出现指令重排序的情况。下一篇将介绍JVM中的参数设置对java程序的影响。

参考资料：

《实战Java虚拟机》 葛一鸣

《深入理解Java虚拟机（第2版）》 周志明

《深入理解Java内存模型》 程晓明

飘扬的红领巾
关注 - 0
粉丝 - 910

2 0

荣誉：推荐博客

+加关注

« 上一篇：深入理解JVM（一）——基本原理
» 下一篇：深入理解JVM（三）——配置参数

分类：Java,JVM

#1楼 Mainthing
2018-03-23 15:15

写得挺好的

ADD YOUR COMMENT

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

最新IT新闻:

- IBM发明咖啡无人机：能配送还能预测何时想喝咖啡
- 陈天桥：人脑研究或迎来重大突破 脑对脑沟通不是梦
- G Suite用户即将获得Gmail的侧边工具栏
- 从首富到逃犯的彭小峰 在江西起高楼，在苏州楼塌了
- 这家估值32亿美元的无人驾驶汽车公司把它的CEO赶跑了

» 更多新闻...

最新知识库文章:

- 一个故事看懂"区块链"
- 被踢出去的用户
- 成为一个有目标的学习者
- 历史转折中的"杭派工程师"
- 如何提高代码质量?

» 更多知识库文章...