

Java 多线程并发系统中的安全性与公平性

麻孜宁, 伊浩, 李祥

(贵州大学 计算机软件与理论研究所, 贵州 贵阳 550025)

摘要:多线程是 Java 的一个重要方法, 特别有利于在程序中实现并发任务。Java 提供 Thread 线程类, 实现了多线程的并发机制。然而, 程序的并发执行必定会出现多个线程互斥访问临界资源的局面, 因而并发系统解决的关键就是对临界资源的管理和分配问题, 而在进行临界资源分配时有两方面需要考虑, 即安全性和公平性。文中首先讨论了多线程并发系统中的安全性与公平性问题, 指出安全性与公平性在并发系统中访问临界资源时的重要性。并通过火车行驶单行隧道的实例, 演示各种条件下的行驶情况来进一步说明该问题。

关键词:多线程; 并发; 临界资源; 公平性; 安全性

中图分类号: TP311

文献标识码: A

文章编号: 1005-3751(2006)02-0120-03

Safety and Fairness in Java Multithreading Concurrent System

MA Zi-ning, YI Hao, LI Xiang

(Institute of Computer Science, Guizhou University, Guiyang 550025, China)

Abstract: Multithreading is an important method of Java, especially multithreading is propitious to implement concurrent tasks in program. Java offers Thread class, which can implement concurrent mechanism. But the program concurrence must bring the situation that several threads access the same critical resource simultaneously, so the key problem in concurrent system is how to manage and allot critical resources, in which we should pay attention to fairness and safety. In this thesis, discuss the safety and fairness in concurrent system, and point out its importance when multithreading accessing the critical resources. And under the Java Development Platform - NetBean, realized the demo example of trains driving the single-line tunnel to explain the problem.

Key words: multithreading; concurrence; critical resource; safety; fairness

0 引言

并发即多道程序在同一 CPU 上分时运行, 实现多个任务同时执行。但是系统的并发执行必然会引起线程在访问临界资源时冲突的产生, 即多个线程同一时刻申请访问资源。因而并发系统需首要解决的就是对临界资源的管理和分配。然而在进行临界资源分配时有两方面需要考虑, 即安全性和公平性。

文中通过火车行驶单行隧道的实例演示^[1], 模拟各种情况下所产生的结果来进一步说明并发系统中安全性与公平性的重要性, 指出 Java 中对多线程的并发控制应该做到公平性和安全性的理想兼容, 两方面都不可忽视。

1 并发系统的概念

并发^[4], 通俗地讲, 就是在某一时段同时发生几件事的现象, 这是生活中很常见的。但是在计算机系统中, 并

发是相对于多道程序分时地运行在同一个 CPU 上而言的。如果在单 CPU 下同时运行几道程序, 那么在宏观上, 这几道程序是同时向前推进, 但从微观上观察, 则是由单 CPU 按照时间片轮流执行每个程序的一小部分代码, 使每个程序都运行到程序的开始与结束之间的某一处。从逻辑上讲, 这几道程序都在运行, 但从 CPU 的执行轨迹上观察, 却是轮流地为每个程序执行一段时间, 循环往复, 直到所有程序依次完成。目前使用的计算机几乎都是单 CPU 的机器, 但是都能同时完成几件不同的工作, 就是因为采用了 CPU 分时原理, 因此逻辑上的并行称之为“并发”。

为了提高系统的吞吐量, 实现多个任务同时执行, 在 20 世纪 60 年代提出了进程的概念, 每个进程代表一个独立的任务, 是系统可以进行分配调度的基本单位。到 80 年代中期, 人们又提出了比进程更小的能独立运行的基本单位——线程^[3], 用它来提高系统内程序并发执行的速度, 从而可进一步提高系统的吞吐量。线程的引入减少了程序并发执行时所付出的时空开销, 使操作系统具有更好的并发性。线程是进程中的一个实体, 是被系统独立调度和分配的基本单位。多线程的核心在于多个代码块并发执行, 本质特点在于各代码块之间的代码是乱序执行的。

收稿日期: 2005-05-08

基金项目: 贵州省科学基金资助项目(黔科合(2004)GGY002)

作者简介: 麻孜宁(1980—), 女, 河南新乡人, 硕士研究生, 研究方向为计算机软件与应用; 李祥, 教授, 硕士生导师, 研究方向为计算机软件与应用。

2 Java 中的多线程并发执行中的安全性与公平性

Java 提供了线程类 Thread, 其中重要方法是 run(), 它为 Thread 类中的 start() 方法所调用, 提供线程所要执行的代码。建立线程有两种方法: 继承 Thread 类和实现 Runnable 接口^[4]。

Java 提供了线程类 Thread, 实现了多线程并发执行机制, 在 Java 程序并发执行中, 必定会出现多个线程互斥访问临界资源的局面, 因而并发系统首要解决的关键就是对临界资源的管理和分配, 同一时刻只有一个线程能够访问临界资源。然而在进行临界资源分配时有两方面需要考虑, 即公平性和安全性。一方面, 对申请访问临界资源的所有线程要公平对待, 不能对部分线程优先考虑, 使一些线程长时间独占资源, 而使其它的线程进行长时间的等待, 得不到资源; 另一方面, 在考虑了公平性的同时, 也不能忽视了安全性的考虑, 出现某一时刻不止一个线程访问临界资源, 致使程序不能正常运行, 产生严重的后果。

在 Java 中对多线程的并发控制应该做到公平性和安全性的理想兼容, 两方面都不可忽视。并发系统的公平性使得资源在多线程之间得到相当科学和合理的分配, 安全性保证了各线程的正常调度运行, 从而使程序顺利地执行结束, 完成任务。

3 Java 中多线程并发系统的安全性及公平性的具体实现

对于多线程并发系统中的安全性, 应该做好对临界资源的管理, 以及对其访问情况的监督, 充分保证在临界资源空闲时才能被线程申请使用, 即使牺牲点系统运行时间, 也要保证程序的正常执行。下面以一个多线程并发访问临界资源的例子来具体说明并发系统中的公平性和安全性问题。

3.1 问题描述

如图 1 所示, 在单行隧道中, 每次只能容纳一列火车行驶, 本程序就是控制使两边的火车都能安全地通过隧道。如果火车来自同一个方向, 它们可以并行地安全通过隧道, 但是如果是两辆来自不同方向的火车, 在隧道中相遇, 就会发生冲突。

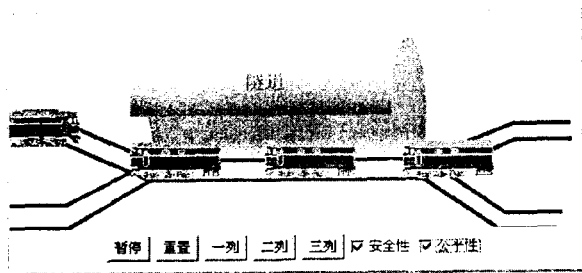


图 1 单行隧道演示界面

3.2 功能实现

本程序简单模拟了上述问题, 假设左边是 LeftTrain, 右边是 RightTrain, 实现了火车为一、二、三列, 以及分别

考虑安全性和公平性的各种情况的实况模拟, 说明并发系统中的公平性与安全性的重要性。

3.3 程序中安全性与公平性的实现

3.3.1 流程分析

1) 假设两边各有三列 Train, 对每列 Train 建立一个线程 (LeftTrain 或 RightTrain), 线程的执行任务就是判断隧道的状态来完成 Train 的进出隧道, 保证互斥访问隧道, 使多个线程正常运行。

2) 对于 Train 过隧道的控制有三种情况:

◆ 只控制 LeftTrain 和 RightTrain 线程类的运行, 不监控隧道的状态, 由 Tunnel 类实现。

◆ 只保证 Train 互斥地通过单行隧道 (安全性), 由 SafeTunnel 类实现, 它继承了 Tunnel 类。

◆ 既保证 Train 互斥通过隧道 (安全性), 又一定程度上保证了公平性, 由 FairTunnel 类实现, 它也继承了 Tunnel 类。

3) 对于界面的控制和各个线程类的建立初始化由类 SingleLaneTunnel 实现。

4) 图形界面的动画处理由 TunnelCanvas 类实现。

3.3.2 关于 SafeTunnel.java 的控制分析

(1) 多线程的并发运行。

在 LeftTrain.java 和 RightTrain.java 中主要实现了多线程的并发运行 (如图 2 所示)。

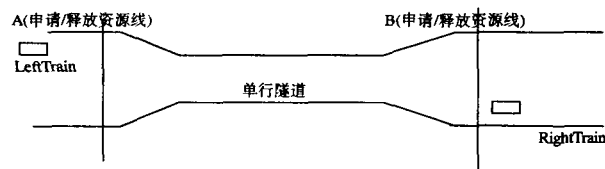


图 2 单行隧道互斥控制图

LeftTrain 和 RightTrain 两类线程竞争资源 (单行隧道), 在线程类运行函数 run() 中,

```
while(true) {
```

```
a) --- while (! display.moveLeft(id)); // not on Tunnel
```

```
b) --- control.LeftEnter ();
```

```
c) --- while (display.moveLeft(id)); // move over Tunnel
```

```
d) --- control.LeftExit ();
```

```
}
```

a) 决定了 LeftTrain 和 RightTrain 谁先申请到资源: 谁先到达各自的申请资源线 (A), 谁就先申请到并占有资源。

b) 使用资源并实现互斥功能: 该函数通过一个整型变量来控制该类线程个数实现。

c) 判断占用资源的线程是否应该释放资源: 当该线程到达自己的释放资源线 (B) 时就应改释放该资源。

d) 执行释放资源的功能, 并唤醒其它所有的线程。

(2) 安全性保证。

在 SafeTunnel.java 中主要实现了资源的互斥访问功能 (安全性)。在该资源中设有全局变量 nLeft 和 nRight 分别代表各类线程占用资源的个数。程序实现如下所示。

```

synchronized void LeftEnter() throws InterruptedException {
a) -- while (nRight > 0) wait ();
    nLeft ++ ; //获取资源, nLeft 加 1
}
synchronized void LeftExit() {
    nLeft -- ; //释放资源线处, nLeft 减 1
b) -- if (nLeft == 0)
    notifyAll();
}

```

a) 判断另一类线程是否占用资源: 如果 nRight 等于 0 说明没有占用该资源, 如果 nRight 大于 0, 说明另一类线程正在占用该资源, 这样就挂起本线程。

b) 释放资源并唤醒线程: 如果 nLeft(nRight) 等于零, 就说明该类线程已经全释放了资源, 重新唤醒所有线程继续竞争资源, 这样就保证了多线程的延续性。

(3) synchronized 的作用^[5]。

Java 提供了专门机制以解决这种冲突, 即 synchronized 关键字, 它有效避免了同一个数据对象被多个线程同时访问。synchronized 方法使每个类实例对应一把锁, 每个 synchronized 方法都必须获得调用该方法的类实例的锁方能执行, 否则所属线程阻塞本程序就是使用第一种方法来实现对公共资源的互斥访问。

3.3.3 关于 FairTunnel.java 的控制分析

FairTunnel 类是对 SafeTunnel 类的改进, 在实现多线程并发执行中的安全性的同时, 也保证了多个线程对资源访问的公平性。

```

private int nLeft = 0, nRight = 0, waitRight = 0, waitLeft = 0;
private boolean Rightturn = true;
synchronized void LeftEnter() throws InterruptedException {
++ waitLeft; //等待的 LeftTrain 线程的个数加 1
a) -- while (nRight > 0 || (waitRight > 0 && Rightturn)) wait();
-- waitLeft; //等待的 LeftTrain 线程个数减 1
++ nLeft; //占用该资源的线程个数随之加 1
}
synchronized void LeftExit() {
-- nLeft; //释放资源, nLeft(nRight) 减 1
b) -- Rightturn = true;
c) -- if (nLeft == 0)
    notifyAll();
}

```

(上接第 119 页)

717-728.

- [6] Eisen M B, Spellman P T, Brown P O, et al. Cluster analysis and display of genome-wide expression patterns[A]. Proc. Natl. Acad. Sci USA, 95 [C]. USA: [s. n.], 1998. 14863-14868.
- [7] Duda R O, Hart P E. Pattern Classification and Scene Analysis

在这个类中引用了以下几个公共变量:

* nLeft, nRight: 代表占用资源的同类线程数;

* waitLeft, waitRight: 代表申请资源的同类线程数;

* Rightturn: 是一个时间片开关, 它实现了两边的两类线程公平地访问资源, 两类线程申请资源获得允许使用权力的时间片就是一列 Train 通过单行隧道的的时间。

a) 挂起线程: 当 nRight 大于 0 时说明 RightTrain 线程正在占用资源, 当 waitRight 大于 0 时, 说明有新的 RightTrain 线程将准备使用资源, Rightturn 等于 true 时, 说明 RightTrain 线程有权继续使用资源。在上述情况下, 挂起 LeftTrain 线程。

b) 当第一个使用资源的 LeftTrain 线程到达资源释放处时, 使得 Rightturn 改为 true, 使得 LeftTrain 线程失去申请资源获得允许使用权力。

c) 释放资源并唤醒线程: 如果 nLeft(nRight) 等于零, 就说明该类线程已经全释放了资源, 重新唤醒所有线程继续竞争资源, 这样就保证了多线程的延续性。

4 结束语

文中简单模拟火车行驶单行隧道, 通过这个实验模拟了多线程并发系统中的安全性与公平性问题, 分别演示了在不同情况下产生何种后果: 只考虑安全性, 将会有一类 Train(LeftTrain 或者 RightTrain) 长时间等待, 申请不到资源; 只考虑公平性, 会出现两边火车相撞的情况。由此, 可充分认识打到并发系统中的安全性与公平性二者的辩证关系, 通过两者的结合, 设计解决实际问题的最优方案。

参考文献:

- [1] Magee J, Kramer J. Concurrency: State Models & Java Programs[M]. [s. l.]: Wiley publishing company, 1999.
- [2] 任爱华, 王雷. 操作系统实用教程(第 2 版)[M]. 北京: 清华大学出版社, 2004.
- [3] Stallings W. 操作系统精髓与设计原理(第 3 版)[M]. 北京: 清华大学出版社, 1998.
- [4] 陶冶. Java 多线程学习笔记[EB/OL]. http://www.cs-dn.com.cn/program/index.htm, 2003-06-18.
- [5] Deitel H M, Deitel P J. Java 程序设计教程[M]. 施平安, 施惠琼译. 北京: 清华大学出版社, 2004.

[M]. [s. l.]: John Wiley and Sons, 1973.

- [8] Xu Ying, Olman V, Xu Dong. Clustering gene expression data using a graph-theoretic approach: an application of minimum spanning trees[J]. Bioinformatics, 2002, 18: 536-545.
- [9] Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by simulated annealing[J]. Science, 1983, 220: 671-680.