

【转载】Multithreaded toolkits: A failed dream? From Sun VP Graham Hamilton's Blog

转载

CoderAndy

于 2016-06-26

10:38:16 发布

662

★ 收

藏

文章标签:

Swing multi-threaded

failed dreams

Graham Hamilton

原文地址:

<https://community.oracle.com/blogs/kggh/2004/10/19/multithreaded-toolkits-failed-dream>

Multithreaded toolkits: A failed dream? Blog

发帖人 kggh 于 2004-10-19 2:43:00 在 kggh

The question came up recently of "should we make Swing truly multithreaded?" My personal answer would be "no", and here's why...

The Failed Dream

There are certain ideas in Computer Science that I think of as the "Failed Dreams" (borrowing a term from Vernor Vinge). The Failed Dreams seem like obvious good ideas. So they get periodically reinvented, and people put a lot of time and thought into them. They typically work well on a research scale and they have the intriguing attribute of almost working on a production scale. Except you can never quite get all the kinks ironed out...

For me, multithreaded GUI toolkits seem to be one of the Failed Dreams. It seems like the obvious right thing to do in a multithreaded environment. Any random thread should be able to update the GUI state of buttons, text fields, etc, etc. Damned straight. It's just a matter of having a few locks, what can be so hard? OK, there are some bugs, but we can fix them, right? Unfortunately it turns out not to be so simple...

From observation, there seems to be an amazing tendency towards deadlocks and race conditions in multithreaded GUIs. I first heard about this issue anecdotally from people who had worked with the Cedar GUI libraries at Xerox PARC in the early 80's. That was a community of extremely smart people who really understood threading, so the assertion that they were having regular deadlock issues within GUI code was intriguing. But maybe that was flawed data or an exceptional situation.



CoderAndy

关注

Unfortunately that general pattern has been repeated regularly down the years. People often start off trying for multithreading and then slowly move to an event queue model. "It's best to let the event thread do the GUI work."

We went through this with AWT. AWT was initially exposed as a normal multi-threaded Java library. But as the Java team looked at the experience with AWT and with the deadlocks and races that people had encountered, we began to realize that we were making a promise we couldn't keep.

This analysis culminated in one of the design reviews for Swing in 1997, when we reviewed the state of play in AWT, and the overall industry experience, and we accepted the Swing team's recommendation that Swing should support only very limited multi-threading. With a few narrow exceptions all GUI toolkit work should occur on the event processing thread. Random threads should not try to directly manipulate the GUI state.

Why is this so hard?

John Ousterhout gave a great Usenix talk on **Events versus Threads** in 1995 that explores some of the tradeoffs between thread-driven and event-driven programming and he correctly points out many reasons why multi-threaded programming is hard and why event driven programming can be simpler. I don't necessarily agree with his analysis for all kinds of programs, but I do agree for GUI programs.

The particular threading problems of GUI toolkits seem to me to arise from the combination of input event processing and abstraction.

The problem of input event processing is that it tends to run in the opposite direction to most GUI activity. In general, GUI operations start at the top of a stack of library abstractions and go "down". I am operating on an abstract idea in my application that is expressed by some GUI objects, so I start off in my application and call into high-level GUI abstractions, that call into lower level GUI abstractions, that call into the ugly guts of the toolkit, and thence into the OS. In contrast, input events start off at the OS layer and are progressively dispatched "up" the abstraction layers, until they arrive in my application code.

Now, since we are using abstractions, we will naturally be doing locking separately within each abstraction. And unfortunately we have the classic lock ordering nightmare: we have two different kinds of activities going on that want to acquire locks in opposite orders. So deadlock is almost inevitable.

This problem will initially surface as a series of specific threading bugs. And people's first reaction is to try to adjust the locking behavior to resolve the specific bugs. Let's release that lock there and then lets use more clever locking over here. Well, that is kind of a fun activity, but it is trying to fight back an oceanic tidal force. The cleverer locking typically turns into a combination of subtle races (due to lack of locking) or clever and intricate deadlocks (due to the clever and intricate locking). We went through a bunch of that in 95-97.



Notice that the problem extends beyond the GUI toolkit layers and also appears between the toolkit layer and the application level. With great difficulty one might try to adopt a single lock for all activity within the GUI layer, but the same problem then resurfaces a level up.

So what's the answer? Well, at some point you have to step back and observe that there is a fundamental conflict here between a thread wanting to go "up" and other threads wanting to go "down", and while you can fix individual point bugs, you can't fix the overall situation.

This led to the solution that the Swing team adopted and which is used by most leading GUI toolkits: run all GUI activity on a single event thread. This means that in some sense all GUI activity becomes event driven, and the "down" threads become just a new kind of event.

This demonstrably works. It is possible to write complex GUI apps that work reliably. Hurrah! But it does make managing long running activities tougher. I wrote a smallish Swing program that I use periodically to selectively zap large boring attachments from my email archives. I don't want to hang the GUI while it reads tens of megabytes of emails, and I also want to display a progress monitor, so I ended up having to carefully balance handing off big activities to worker threads and handing GUI activities back to the event thread. It is probably more complicated than it would be if I had a magic multi-threaded library, but it has the significant saving grace that it actually seems to work reliably.

Subtleties

Are things really so black and white? Surely there have been people who have used multi-threaded toolkits successfully? Yes, but I think this demonstrates one of the characteristics of the Failed Dreams.

I believe you can program successfully with multi-threaded GUI toolkits if the toolkit is very carefully designed; if the toolkit exposes its locking methodology in gory detail; if you are very smart, very careful, and have a global understanding of the whole structure of the toolkit. If you get one of these things slightly wrong, things will mostly work, but you will get occasional hangs (due to deadlocks) or glitches (due to races). This multithreaded approach works best for people who have been intimately involved in the design of the toolkit.

Unfortunately I don't think this set of characteristics scale to widespread commercial use. What you tend to end up with is normal smart programmers building apps that don't quite work reliably for reasons that are not at all obvious. So the authors get very disgruntled and frustrated and use bad words on the poor innocent toolkit. (Like me when I first started using AWT. Sorry!)

Another wrinkle: it is possible to have multiple simultaneous GUI activities within a Java VM by using multiple event threads. That works provided the different activities are almost entirely isolated, have their own distinct GUIs (no shared components or mixed hierarchies) and provided they dispatch events to the right thread. This is useful in (for example) ru



CoderAndy

关注

But it isn't a very general solution - most applications need to live within the constraint of only a single event thread.

In this note I've most been covering why Swing and other toolkits are essentially single-threaded. Chet recently blogged on some related topics around [why multi-threading complicates user programs](#) and normally won't help raw graphics performance.

Also, before I forget, some people are probably remembering that "processes and monitors are duals". Well, yes, it's true. In some sense we are using the event thread to implement a global lock. We could invert things, and create a global lock that is equivalent to the event queue. This would be fairly ugly and would require wide coordination and undermine a lot of abstractions. But the larger problem is that Java developers tend to use multiple locks and if they are to preserve the equivalence with an event queue model, they will need to follow various non-obvious rules about how they interact with these other locks. The event queue model makes the central single lock much more visible and explicit, and on the whole that seems to help people to more reliably follow the model and thus construct GUI programs that work reliably.

Conclusion

I guess the bottom line is that like many others I would really like to see a flexible, powerful, truly multi-threaded GUI toolkit. But I don't know how to get there - at this point there is fairly strong experience that the obvious approaches for multi-threading don't work. Maybe in future years people will come up with a radically new and better approach, but for now the answer seems to be that events are our friends.

Graham

多线程工具包：一个失败的梦？ 永无止境 52
此文翻译自：<http://weblogs.java.net/blog/kgf/archive/2004/10/multithread...>

spring-batch-multithreaded:spring-batch-多线程 06-29
Spring Batch 多线程示例有两个 RDBMS（它们在应用程序的 Oracle 模式...

Android GUI 单线程消息队列机制 —— 多线程... Elsa's Blog 1281
前言：Android、Swing、MFC等的GUI库都采用了单线程消息队列机制来...

multitjreaded-multithreaded.zip_class A_parallel... 最新发布 07-13
A multithreaded example of the C # language, using the Parallel class

statisticsCalculator_Multithreaded.c:CS423 的作业 #2 07-05
例如，假设你的程序传入以下整数作为输入（假设a.out是你的程序） a.out...

A Guide to Multithreaded Programming 04-23
A Guide to Multithreaded Programming

multithreaded-renderer:使用专用线程中的渲染测试应用... 03-31
multithreaded-renderer:使用专用线程中的渲染测试应用程序以进行实验

multithreaded-chatroom:在终端中运行的用 Java 编写的... 06-04
多线程聊天室 我使用 Java 中的 TCP/IP 编程构建了一个聊天室。特征 wh...

java-multithreaded-chat:该I
多线程-Java-Chat 该项目包含一



CoderAndy

关注

Pthreads Primer: a guide to **multithreaded** programmingr 12-22
介绍**pthread**s多线程编程的一本相当不错的书。网上的电子书版本大多没...

Java: A Beginner's Guide 5th **Edition** 06-22
You'll also find coverage of **some** of Java's most advanced**ed** features, inclu...

multithreaded-download-manager:具有多线程支持的Firef... 05-07
多线程下载管理器 下载适用于Firefox的管理器扩展，具有多线程支持。建...

ValueError: cannot have a **multithread**... qq_43828004的博客 3145
最近再写一个flask项目中需要用到框架本身自带的多进程，于是就设置pro...

l/dalvikv... forlong401的专栏--有问题上: http://www.androidren.com 2170
http://**androidren.com**/index.php?qa=390&qa_1=i-dalvikvm-total-arena-pa...

multithreaded-sorting-:多个线程并行执行以高效排序 06-17
多线程排序- 多个线程并行执行以高效排序

multithreaded-xml-parser:2014 年 12 月 07-11
多线程-xml-orders-parser 应用程序使用 WatchService 监视带有订单的 x...

Flask 之 ValueError: cannot have a **multithreaded** ... 俗人 3277
之前启动项目一直起不来，报cannot have a **multithreaded** and **multi** proc...

multithreaded-sudoku:一个用 C 编写的多线程数独解谜器... 07-12
用 C 编写的多线程数独求解器。解决方法如下： 创建 1 个线程以检查所...

“相关推荐”对你有帮助么？


 非常有帮助  有帮助  一般  没帮助  非常没帮助

©2022 CSDN 皮肤主题：大白 设计师：CSDN官方博客 返回首页

关于 招贤 商务 寻求 400-
我们 纳士 合作 报道 660-
公安备案号11010502030143 京ICP备19004658号 京网文〔2020〕1039-165号
经营性网站备案信息 北京互联网违法和不良信息举报中心 家长监护 网络110报警服务
中国互联网举报中心 Chrome商店下载 账号管理规范 版权与免责声明 版权申诉 出版物许可证
营业执照 ©1999-2022北京创新乐知网络技术有限公司



CoderAndy
码龄14年 暂无认证

12 110万+ 170万+ 2万+ 
原创 周排名 总排名 访问 等级

388 3 2 1 3
积分 粉丝 获赞 评论 收藏

私信

关注


搜博文文章




CoderAndy 关注

热门文章

用apache的Mail包（commons-email-1.2.jar），发送邮件  3345

并发引起的java.lang.NullPointerException  3239

【转】Spring security3 sec:authorize url 无效的问题  2937

“伪集群”导致的Hibernate主键increment生成策略异常  2258

jbpm-6.3.0.Final-installer-full在Windows上的部署、数据库由H2切换为MySql、Linux上的部署全过程  2125

最新评论

并发引起的java.lang.NullPointerException
CoderAndy: 勉强算吧。但这个问题可以规避，那就是别new那么多对象，而是重用 ...

您愿意向朋友推荐“博客详情页”吗？



强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

jbpm-6.3.0.Final-installer-full在Windows上的部署、数据库由H2切换为MySql、Linux上的部署全过程

“伪集群”导致的Hibernate主键increment生成策略异常

Coding中的低级问题总结【开个头先】

2016年 6篇

2015年 10篇



CoderAndy

关注