Atitit.软件可靠性原理与概论attilax总结

[1. 资源死锁 1](#_Toc3534)

[1.1. 解除死锁策略 1](#_Toc29049)

[2. 死锁的检测 1](#_Toc11782)

[3. ----影响稳定性几个类别 2](#_Toc7076)

[4. -----影响稳定性的因素 2](#_Toc6950)

[5. 为什么httpclient会不能自动关闭连接 2](#_Toc27901)

[6. 资源泄露 2](#_Toc18962)

[6.1. 内存泄露 2](#_Toc12281)

[6.2. Conn泄露（常见有http连接和数据库连接泄露 3](#_Toc26624)

[6.3. 句柄泄露（io 文件handle等） 3](#_Toc7974)

[6.4. tcp连接阻塞（http） 3](#_Toc18098)

[6.5. Stream流式api阻塞 3](#_Toc26880)

[7. 为什么http会采用阻塞设计 3](#_Toc30576)

[8. 资源死锁与泄露解决模式 3](#_Toc27526)

[8.1. 设置连接超时，与stream读取超时时间 3](#_Toc29380)

[8.2. 每次关闭连接 4](#_Toc11636)

[8.3. futureTask（设置一个超时时间，强制释放资源即可） 4](#_Toc23383)

[8.4. Timer 4](#_Toc2633)

[8.5. Watchdog看门狗技术wdt技术 4](#_Toc7251)

[8.6. 自定义Gc 垃圾回收 4](#_Toc28548)

[8.7. 智能指针（c++） 4](#_Toc26679)

[9. Httpclient的关闭连接与设置超时设置 4](#_Toc15547)

[10. 《工程设计中可靠性验证、试验与分析》(盖瑞S...) 5](#_Toc778)

[11. 可靠性工程(第2版) 6](#_Toc9593)

[12. 《系统软件可靠性》(陆民燕...) 6](#_Toc23365)

[13. 软件可靠性模型综述  7](#_Toc11018)

[14. 《软件可靠性方法（图灵奖得主Clarke作序、北大南大教授联袂推荐，软件形式化方法经典著作）》(（以）佩莱得...) 8](#_Toc20207)

# 资源死锁

## [解除死锁策略](http://blog.csdn.net/attilax/article/details/42202307" \l "t8)

* 1. [1 程序复位](http://blog.csdn.net/attilax/article/details/42202307" \l "t9)
  2. [2      撤消线程剥夺资源](http://blog.csdn.net/attilax/article/details/42202307" \l "t10)
  3. [3 进程回退策略](http://blog.csdn.net/attilax/article/details/42202307" \l "t11)

# 死锁的检测

数据库死锁可以使用数据库工具，代码死锁使用代码profile工具

或者自己写watchdog检测程序即可

# [----影响稳定性几个类别](http://blog.csdn.net/attilax/article/details/27129709" \l "t0)

* 1. [资源和内存泄漏溢出](http://blog.csdn.net/attilax/article/details/27129709" \l "t1)
  2. [数据库文件死锁](http://blog.csdn.net/attilax/article/details/27129709" \l "t2)
  3. [类库冲突](http://blog.csdn.net/attilax/article/details/27129709" \l "t3)
  4. [热更新热部署业务可用性](http://blog.csdn.net/attilax/article/details/27129709" \l "t4)
  5. [程序崩溃](http://blog.csdn.net/attilax/article/details/27129709" \l "t5)
  6. [磁盘空间cpu内存占用过高](http://blog.csdn.net/attilax/article/details/27129709" \l "t6)

# [-----影响稳定性的因素](http://blog.csdn.net/attilax/article/details/27129709" \l "t7)

* 1. [内存泄漏溢出](http://blog.csdn.net/attilax/article/details/27129709" \l "t8)
  2. [数据库连接泄漏](http://blog.csdn.net/attilax/article/details/27129709" \l "t9)
  3. [数据库死锁](http://blog.csdn.net/attilax/article/details/27129709" \l "t10)
  4. [类库冲突造成部署问题](http://blog.csdn.net/attilax/article/details/27129709" \l "t11)
  5. [热更新的支持不足部署比较麻烦](http://blog.csdn.net/attilax/article/details/27129709" \l "t12)
  6. [Web服务跟数据库服务崩溃](http://blog.csdn.net/attilax/article/details/27129709" \l "t13)
  7. [非托管资源的释放](http://blog.csdn.net/attilax/article/details/27129709" \l "t14)
  8. [其他的潜在隐患](http://blog.csdn.net/attilax/article/details/27129709" \l "t15)
  9. [多线程并发读写死锁](http://blog.csdn.net/attilax/article/details/27129709" \l "t16)
  10. [子线程异常造成主线程崩溃java不影响net有这个问题](http://blog.csdn.net/attilax/article/details/27129709" \l "t17)
  11. [文件并发读写](http://blog.csdn.net/attilax/article/details/27129709" \l "t18)
  12. [别的网络socket连接释放问题](http://blog.csdn.net/attilax/article/details/27129709" \l "t19)
  13. [直接内存读写](http://blog.csdn.net/attilax/article/details/27129709" \l "t20)
  14. [Stream的关闭释放](http://blog.csdn.net/attilax/article/details/27129709" \l "t21)
  15. [native method调用的内存](http://blog.csdn.net/attilax/article/details/27129709" \l "t22)
  16. [磁盘空间不足造成许多的莫名其妙的问题也许提示连接耗尽](http://blog.csdn.net/attilax/article/details/27129709" \l "t23)

# 为什么httpclient会不能自动关闭连接

Httpclient的设计基于阻塞式的api接口。所以会遇到连接阻塞，以及读取stream阻塞。。

需要设置其超时时间，以及及时关闭连接即可了。。

# 资源泄露

## 内存泄露

一般vm语言java c#以及实现了自动的内层释放。。解决了大部分的内寸泄露情况。。但是任然有少部分static 或者集合导致的内存泄露

## Conn泄露（常见有http连接和数据库连接泄露

## 句柄泄露（io 文件handle等）

## tcp连接阻塞（http）

## Stream流式api阻塞

# 为什么http会采用阻塞设计

他从TCP原理出发说了两个现象，感觉很有道理，

他这样说的，他说TCP是面向连接的，你接收的数据的缓冲如果比实际数据大时，就会阻塞掉，

要么除非关掉浏览器，或者别的方式断掉连接

阻塞的效率肯定要比非阻塞高，但是使用阻塞时，当没有事件发生时，就会一直等待下去。而很多时候，我们需要随时知道当前的状态，或者需要及时返回状态。这就是非阻塞，   
一般来说实现方式是，当发现阻塞进程一直在等待，未返回结果，等待超时时，会发一个信号给阻塞进程，阻塞进程收到信号后，会主动结束等待，返回结果。   
非阻塞的好处就是响应及时，并发

# 资源死锁与泄露解决模式

## 设置连接超时，与stream读取超时时间

## 每次关闭连接

## futureTask（设置一个超时时间，强制释放资源即可）

## Timer

## Watchdog看门狗技术wdt技术

在最外层增加一个watchdog，内层需要定时喂狗，否则内层资源就被狗吃了。。达到自动回收的目的。

## 自定义Gc 垃圾回收

## 智能指针（c++）

# Httpclient的关闭连接与设置超时设置

• **socket阻塞**   
　　参考了相关文章, 基本认可博文<<[Apache HttpClient 没有设置time out导致应用长时间阻塞的问题](http://blog.csdn.net/hengyunabc/article/details/22716911)>>中的说法: **socket没有设置超时选项导致**.  
　　具体解读, 可以理解为服务端对端不回应, 或者因网络异常, 导致socket一直阻塞于读. Apache HttpClient默认设置的socket为**SO\_TIMEOUT为0**, 既无限等待.  
　　结合之上的代码分析, 确实忽视了超时设定, 陷入了Apache HttpClient隐藏很深的坑中. ^\_^!  
　　解决方案是：**为对应的httpclient设置超时即可**.

|  |  |
| --- | --- |
| 1  2  3 | httpClient = new DefaultHttpClient();  httpClient.getParams().setIntParameter(CoreConnectionPNames.SO\_TIMEOUT, 2000);  httpClient.getParams().setIntParameter(CoreConnectionPNames.CONNECTION\_TIMEOUT, 2000); |

　　这样阻塞的问题就可以暂时告一段落了.   
• **句柄泄露(CLOSE\_WAIT半连接关闭)**  
　　同样参考了博文: <<[HttpClient容易忽视的细节——连接关闭](http://seanhe.iteye.com/blog/234759)>>与<<[HttpClient 与 Close\_Wait](http://network.51cto.com/art/201406/442097.htm)>>.   
　　按apache httpclient的设计理念, 当http client 处于高并发时, 默认机制导致的CLOSE\_WAIT会影响服务的可用性.  
　　对比之上的代码, 确实没有找到主动关闭连接的代码.  
Apache HttpClient本身变动多, 接口又杂, 以至每个版本的解决方案又有所不同, 有些让人眼花缭乱.  
　　比较一致的解决方式是，在处理完单个请求后, 调用如下清理代码:

|  |  |
| --- | --- |
| 1  2 | httpmethod.releaseConnection();  client.getConnectionManager().shutdown(); |

　　但不管怎么说, Apache HttpClient 4.x的关闭机制, 还是值得大书特书的, 这边显得简略, 希望以后有机会能深入研究.

作者:: 绰号:老哇的爪子 （ 全名：：Attilax Akbar Al Rapanui 阿提拉克斯 阿克巴 阿尔 拉帕努伊 ）

汉字名：艾提拉（艾龙），   EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

# 《工程设计中可靠性验证、试验与分析》(盖瑞S...)

第1章 当代视角的可靠性概念及可靠性设计

第2章 可靠性评估定义、顺序统计入门

第3章 可靠性评价中的统计分布概述

第4章 参数估计技术概述

第5章 分布拟合检验

第6章 试验样本量的确定

第7章 加速试验

第8章 设计验证的工程方法

第9章 似然估计（提高篇）

第10章 对比设计

# 可靠性工程(第2版)

作者:（美）[赛义德](http://www.dangdang.com/author/%C8%FC%D2%E5%B5%C2_1" \t "http://product.dangdang.com/_blank)

Line 103: 第一部分 可靠性与系统设计

Line 104: 第1章 可靠度和失效率函数

Line 139: 第2章 系统可靠度评估

Line 179: 第3章 时间和失效相关可靠度

Line 210: 第二部分 参数估计及可靠性试验

Line 211: 第4章 失效时间分布参数估计方法

Line 228: 第5章 参数可靠性模型

Line 276: 第6章 加速寿命试验模型

Line 317: 第三部分 可靠性的提高：保修期及预防维修

Line 318: 第7章 更新过程和预计失效数

Line 341: 第8章 预防性维修与检测

Line 367: 第9章 担保模型

Line 385: 第10章 案例分析

Line 395: 第11章 附录

# 《系统软件可靠性》(陆民燕...)

第1章 绪论

第2章 系统可靠性概念

第3章 估计理论

第4章 软件开发生命周期和数据分析

第5章 软件可靠性建模

第6章 不完美排错模型

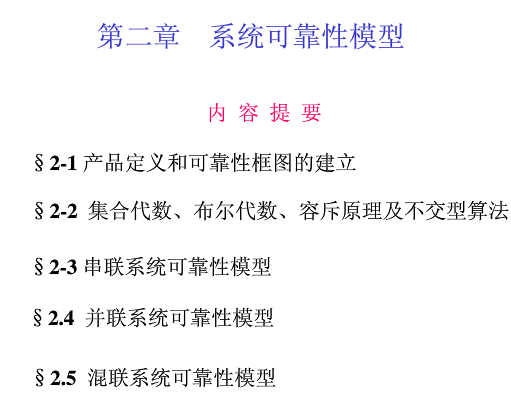
第7章 测试覆盖率与错误移除模型

第8章 考虑环境因子的软件可靠性模型

第9章 软件可靠性模型的校准

第10章 最优软件发布策略

第11章 复杂容错系统可靠性建模



# 软件可靠性模型综述

可靠性是衡量所有软件系统最重要的特征之

1软件失效过程

1.1软件失效的定义及机理

当软件发生失效时，说明该软件不可靠，发生的失效数越多，发生失效的时间间隔越短，则该软件越不可靠。软件失效的机理如下图所示：

1）软件错误（Software error）：指在开发人员在软件开发过程中出现的失误，疏忽和错误，包括启动错、输入范围错、算法错和边界错等。

2）软件缺陷（Software defect）：指代码中存在能引起软件故障的编码，软件缺陷是静态存在的，只要不修改程序就一直留在程序当中。如不正确的功能需求，遗漏的性能需求等。 3）软件故障（Software fault）：指软件在运行期间发生的一种不可接受的内部状态，是软件缺陷被激活后的动态表现形式。

4）软件失效（Software failure）：指程序的运行偏离了需求，软件执行遇到软件中缺陷可能导致软件的失效。如死机、错误的输出结果、没有在规定的时间内响应等

# 《软件可靠性方法（图灵奖得主Clarke作序、北大南大教授联袂推荐，软件形式化方法经典著作）》(（以）佩莱得...)

第1章　引言

第2章　预备知识

第3章　逻辑和定理证明

第4章　软件系统建模

第5章　形式化规约

第6章　自动验证

第7章　演绎式软件验证

第8章　进程代数与等价关系

第9章　软件测试

第10章　组合形式化方法

第11章　可视化

参考资料

Atitit.线程 死锁 跑飞 的检测与自动解除 与手动解除死锁 java c# .net php javascript. - attilax的专栏 - 博客频道 - CSDN.NET.html

软件可靠性模型综述\_百度文库.html

HTTP&POST使用阻塞之郁闷 - L\_yangliu的专栏 - 博客频道 - CSDN.NET.html

浏览器请求阻塞到底是怎么回事\_我们为什么要把静态资源分服务器放置\_ - 活雷锋 - 博客园.html

TCP的阻塞机制 - 勇者无敌 - 博客频道 - CSDN.NET.html

TCP的阻塞和重传机制 - 轩脉刃 - 博客园.html

解决C#网络通信编程的阻塞问题 - onedime - 博客园.html

TCP\_IP总结1：阻塞socket和非阻塞socket \_ 学步园.html

试论软件的可靠性及其保证 - GerryZhu的专栏 - 博客频道 - CSDN.NET.html

Atitit.http httpclient实践java c# .net php attilax总结-布布扣-bubuko.com.html

Atitit 网络爬虫与数据采集器的原理与实践attilax著 v2 - attilaxAti - 博客园.html

atitit 提升数据库死锁处理总结 - phlsheji - 博客园.html

Atitit.软件开发提升稳定性总结 - attilax的专栏 - 博客频道 - CSDN.NET.html

Atiend