Tomcat内存溢出及线程紊乱问题研究

杨 聪[1]，王文永[1]，李 晨[2]
（1.东北师范大学理想信息技术研究院，吉林长春 130117；2.锡根大学，北莱茵-威斯特法伦州，德国）

摘 要 在很多基于B/S结构的网站架构中，WEB容器内存溢出及线程紊乱问题比较隐蔽，很多时候在测试阶段并不能发现，只有在现实中大规模数据和高并发量的情况下问题才逐渐的暴露出来。因此，在网站正式发布前代码进行走查和技术改进，并修改相关服务器软件的配置，可以在很大程度上减少此类事件的发生。本文以Tomcat为例，对WEB容器在数据传输过程中内存溢出及线程紊乱的表现、原因及解决方案作了简要论述。

关键词 Tomcat；WEB容器；内存溢出；线程紊乱
中图分类号 TP **文献标识码** A **文章编号** 1673-9671-(2012)022-0108-02

随着 Internet技术的普及，各地方学校、研究所和商业单位都在积极进行基础教育资源网和资源库的建设。然而，随着资源网使用人数的不断增加，其并发量也在急剧增长，对WEB服务器的承压性和稳定性提出了新的挑战。然而大多数WEB容器均有内存限制，因此，在服务器没有内存还有很大空缺的情况下，WEB容器内存首先溢出，经常报“OutOfMemory”错误，并与其他因素一道引发了线程紊乱，导致应用系统的某些功能重复执行，并且引起了数据库服务器崩溃、系统越来越慢直到死机等问题。

随着互联网技术的发展，基于WEB容器大规模数据传输以及并发量的需求已经日渐突出，而数据传输效率、WEB应用服务器性能以及应用系统的稳定性等因素直接影响了数据传输的质量。在以Tomcat为WEB容器的环境中，若以上问题处理不当，则很多时候表现为Tomcat内存溢出以及线程紊乱，造成服务器宕机，严重影响正常的网站运行。

1 Tomcat内存溢出及线程紊乱的主要表现

Tomcat内存溢出主要是通过系统速度、系统性能表现以及系统日志来反映的。通过对日志文件和系统表现的分析与判断，即可断定是否为内存溢出；线程紊乱是指在Web容器中发生的线程异常的情况，其很多时候是在内存溢出之后出现的，通过对应用系统的操作日志及WEB容器的相关日志即可判断。

1.1 Tomcat内存溢出主要表现

1）系统的速度越来越慢，甚至出现死机的现象。

2）在Tomcat日志中，出现“OutOfMemoryError”提示。

3）设置Xrunpmijvmpiprofiler检测过度使用的对象，两次垃圾回收操作间隔最佳平均时间应至少为每次垃圾回收所用掉的平均持续时间的5-6倍。如果低于这个时长，则可判断系统内存资源已出现紧张。

4）查看GC日志中每次垃圾回收显著增加后立即使用的内存量。如果显示锯齿模式，并且锯齿模式具有不规则的形状，更像楼梯，则存在内存溢出。

5）查看GC日志中已分配的对象数和已释放的对象数之间的差异。如果这个值随着时间的推移是不断增大的，则存在内存溢出。

6）通过任务管理器查看操作系统内存使用情况和CPU使用情况，在工作负载很大的情况下（CPU 100%使用）内存是有可能泄漏。

7）通过任务管理器查看Tomcat进程的内存使用情况，如果出现大于128MB，则有可能是内存溢出。

1.2 Tomcat线程紊乱主要表现

线程紊乱现象是通过对应用系统中的每一个操作进行日志记录，并对该日志研究得到的。线程紊乱现象在日志中主要表现为应用系统的某一功能自发的在短时间内重复执行，而且重复的次数不定。据我们的统计其次数在一秒内的区间为[2，1000]。如图1所示，横坐标为系统并发访问量，纵坐标为系统某一功能自发重复执行的次数。由此可见，随着并发量的增加，重复执行的次数越多。在发生线程紊乱之后，系统的速度明显变慢，而且由于线程紊乱通常发生在内存溢出之后，所以极易引起系统死机。

2 Tomcat内存溢出及线程紊乱产生的主要原因

内存溢出是指应用系统中存在无法回收的内存或使用的内存过多，最终使得程序运行要用到的内存大于虚拟机能提供的最大内存。Tomcat内存的分配是由JVM程序完成的，而内存的释放是由垃圾收集器（Garbage CoHecfion，GC）完成的。

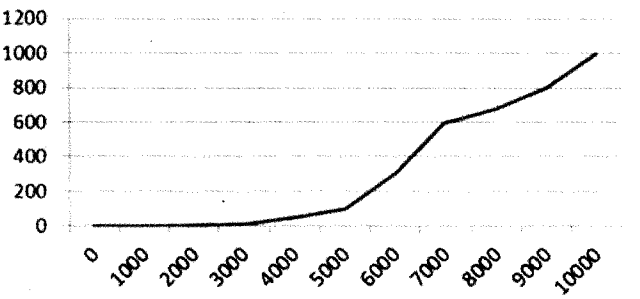

图1 系统并发量与功能自发重复执行次数

线程具有许多传统进行所具有的特征，所以称为轻型进程（Light-Weight Process）。在多线程环境中，为使系统中的多线程能有条不紊地进行，在系统中必须提供用于实现线程间同步和通信的机制，例如互斥锁、条件变量、计数信号以及多读、单写锁等，以上机制如果发生错误或者混乱，都有可能触发线程紊乱。

2.1 内存溢出的主要原因

1）内存中加载的数据量过于庞大，如一次取十万条记录到内存，就可能引起内存溢出。

2）集合类中有对对象的引用，使用完后未清空，使得JVM不能回收。

3）使用第三方软件中的BUG。例如在Tomcat中VF数据库与ORACLE数据库进行数据导入导出，主要是通过javaDBF实现的，而javaDBF的线程本来就是不安全的，在每次导入导出的数据量超过10万条时，很可能会出现内存溢出现象。

4）Tomcat与JVM启动参数内存值的设定过小。

5）在B/S系统开发模式下，数据库连接不当。频繁的建立、关闭连接，数据库服务器必须建立大量的连接，这样必然会极大的降低系统的性能，并引发内存溢出。

6）应用系统代码问题。例如代码中有死循环或递归调用、有大循环重复产生新对象实体都有可能引起内存溢出。

2.2 线程紊乱产生的主要原因

1）通过对应用系统的操作日志以及Tomcat的系统日志分析发现，线程紊乱主要发生在Tomcat内存溢出之后，即内存溢出能够导致线程紊乱。

2）客户端的操作不当。例如客户对于重要的提交按钮在很短的时间内多次点击，而每点击一次触发的数据量过大，就可能引起线程紊乱。

3）网络环境引起的线程紊乱。通过对实际应用系统运行的观察发现，在高并发量的情况下，如果客户端网络环境不好，其部分操作可引发线程紊乱。

3 解决Tomcat内存溢出及线程紊乱的方法

内存溢出是一个系统性能方面的复杂顽症，但是如果解决得当，并根据需要进行相应的参数设置，可在很大程度上避免。对于线程紊乱，在解决内存溢出的基础上对应用程序的功能进行相应的改进，也可降低出现线程紊乱发生的频率。

3.1 内存溢出的解决方案

1）修改JVM启动参数，直接增加内存。这一点看上去似乎很简单，

但很容易被忽略。JVM默认可以使用的内存为64MB，Tomcat默认可以使用的内存为128MB（图2），对于稍复杂一点的系统就会不够用。因此，首次安装用户可在安装过程中调整Tomcat的默认内存大小，对于已经安装的用户或者内存需求改变的用户，可以通过修改<CATALINA_HOME>\bin\catalina.bat文件的内容，并且修改注册表中的参数，加入JVM初始化内存的参数值即可。

2）释放无用的对象的引用。使用临时变量时，让引用变量在退出活动域后，自动设置为null，暗示垃圾收集器来收集该对象，防止内存泄露发生。

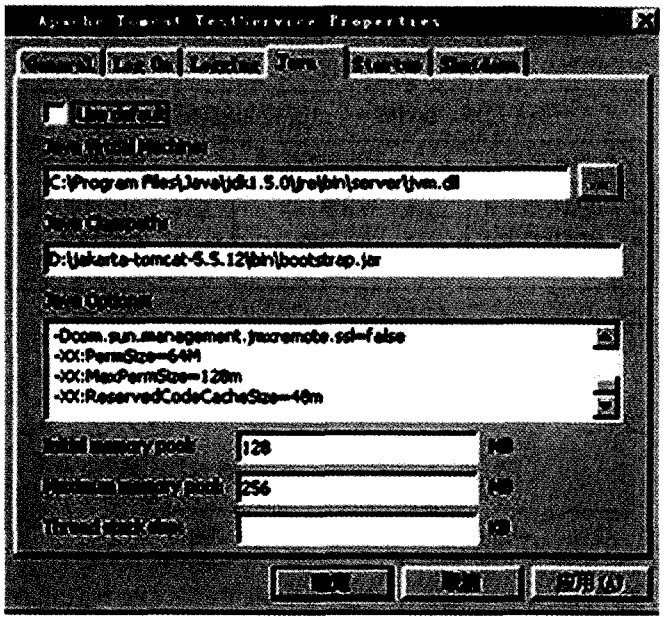

图2 Tomcat默认可以使用的内存为128MB

3）遇到问题时注意检查错误日志，查看OutOfMemory错误前是否有其它异常或错误。通过检查错误日志，可以大致定位有问题的模块，并对该模块进行修改和处理。

4）使用内存查看工具动态查看内存使用情况，比较较有名的有：Optimizeit Profiler、JProbe Profiler、JinSight和Java1.5的Jconsole，通过间隔一段时间取一次内存快照，然后对内存快照中对象的使用与引用等信息进行比对与分析，可以找出是哪个类的对象在泄漏。

3.2 线程紊乱的解决方案

1）安排有经验的编程人员对代码进行走查和分析，找出可能发生内存溢出的位置，包括循环或递归调用、大循环重复产生新对象实体、一次性获得全部数据的查询等。这个问题比较隐蔽，在上线前，数据库中数据较少，不容易出问题，上线后，数据库中数据多了，一次查询就有可能引起内存溢出。因此对于数据库查询尽量采用分页的方式查询。

2）检查List、MAP等集合对象是否有使用完后未清除的问题。

3）对于按钮等控件，点击后使其立刻失效，不让用户重复点击，避免同时对同一条记录操作。

4）与数据库的交互中，尽量使用数据库连接池技术。

5）在往数据库插入数据（Insert）之前，可以依照需求删除（Delete）相同内容或类型的数据，即向数据库插入数据之前，为了保持数据的唯一性，先删除一些有可能重复的数据。这样可以减少在线程紊乱中的重复执行同一插入动作问题，保护了数据的正确性。例如在考生基本信息征集系统中，考生通过网页提交上来自己的基本数据，在将这些数据插入数据库之前，先将该考号的考生信息进行删除，然后再插入，这样就可以避免线程紊乱或其他原因而引起的重复插入和数据重复问题。

其实内存溢出的解决方案与线程紊乱是互补的，开发人员与系统管理人员运用以上方法进行相应的检查、操作和处理，可以大大减少内存溢出，进而减少了线程紊乱。

4 结束语

Tomcat内存溢出及线程紊乱问题比较隐蔽，很多时候在测试阶段并不能发现，只有在现实中大规模数据和高并发量的情况下问题才逐渐的暴露出来。因此，在正式发布前依据经验对代码进行走查和技术改进，并且修改相关服务器软件的配置，可以在很大程度上减少此类事件的发生。

很多时候Tomcat只适合于开发阶段使用，基于WEB的企业级应用程序的开发人员中，WebLogic应用程序服务器可完成Tomcat服务器所能提供的一切并提供更多的功能，因此在完成开发之后，将程序从Tomcat迁移到WebLogic上，能够使系统更加稳定、安全的运行。

参考文献

[1]刘英,冯云. WebSphere Application Server内存溢出问题初探[J].甘肃科技,2008,24(15):27-29.
[2]李伦,陈芳.内存溢出和数据库锁表的分析与解决[J].计算机安全技术,2008,11:82-84.
[3]张昊,潘祖烈.Java软引用防止内存泄漏技术的研究[J].安徽教育学院学报,2006,24(6):42-45.
[4]汤小丹,梁红兵,哲凤屏,等.计算机操作系统[M].北京:西安电子科技大学出版社,2007.
[5](美)Bruce Eckel著,陈昊鹏 译.Java编程思想[M].北京:机械工业出版社,2007.

作者简介

杨聪（1987—），甘肃省庆阳市人，东北师范大学理想信息技术研究院硕士研究生，主要研究方向：云计算，多媒体数据挖掘，信息管理系统。

（上接第106页）

元，通过监控单位可以监控整个系统。一方面，监控单元对煤矿生产设备设备实时采集和整理；另一方面，对监控点收集到的数据进行管理，并将系统中的实时数据传输给远程计算机。同时，监控单元也接收远程计算机发送过来的指令，对监控内容进行调整。

监控单元微控制器的设计：监控单元微控制器采用 SAMSUNG 公司的 S3C44B0X处理器，由于SAMSUNG公司的S3C44B0X 在现场总线上使用并行机制和存储器等特点，在一定程度上降低了指令周期，提升了系统的运行速度，具有很强的数据运算处理能力和控制能力，实现了对煤矿监控系统数据的实时采集和处理，并且SAMSUNG公司的S3C44B0X 微控制器具有较高的集成程度，具备异步串行通信接口，可实现系统的远程通信，同时也使系统整体性能和集成性得到了提高，使系统更为适应于煤矿生产的使用需求，也更为安全可靠。

点对点通信是监控单元和远程计算机的通信方式，期间的传输通道可为电缆、电话线、光纤通道、互联网系统等。

5 结束语

在煤矿生产作业过程中，监控系统有着举足轻重的地位，它不仅是煤矿矿井安全生产的重要保障，也是保护煤矿工人生命安全的重要措施。随着自动化的飞速发展，相信有关人员在今后能够研究出更好的煤矿监控系统软件，为我国煤矿行业的发展保驾护航。

参考文献

[1]魏乐平.煤矿安全监控系统应用中存在的问题与对策[J].煤矿安全,2006,11:71-74.
[2]许洪华.现场总线与工业以太网技术[M].北京:电子工业出版社,2007.
[3]KeC,ZhenWW.Researeh and Implemetation of Remote Monitoring System Based onReal—TimeuClinux[J].IEEE,2005.
[4]赵延明,等.煤矿安全监控系统的现状与发展.煤矿机电[J].2007,03:39-41.

Tomcat内存溢出及线程紊乱问题研究

作者： 杨聪， 王文永， 李晨
作者单位： 杨聪,王文永(东北师范大学理想信息技术研究院,吉林长春,130117)， 李晨(锡根大学,北莱茵-威斯特法伦州,德国)

刊名： 科技与生活
英文刊名：
年，卷(期)： 2012(4)