

一种预防 XML 文件过大引起 JAVA 内存泄露的方案

李德有, 解晨光, 刘明刚

(哈尔滨金融学院, 黑龙江 哈尔滨 150030)

摘 要: JAVA 内存泄露是对应用系统的稳定性有很大的影响, 文件过大使程序运行要用到的内存大于虚拟机能提供的最大内存而导致内存泄露. 文中提出了通过对 XML 文件拆分与拼装来解决由于 XML 文件过大而导致的内存泄露问题的方法. 测试结果表明可以有效预防内存泄露.

关键词: 内存泄露; XML; 文件拼装

中图分类号: TP311

文献标识码: A

文章编号: 1000-7180(2012)01-0152-03

Scheme to Prevent Memory Leaks Because of XML File is too Large in JAVA

LI De-you, XIE Chen-guang, LIU Ming-gang

(Harbin University of Finance, Harbin 150030, China)

Abstract: Memory leaks in JAVA is a great influence on the stability. It cause memory leaks when the memory running program needs is greater than the maximum available of JVM. Scheme was proposed that to prevent memory leaks because of XML file is too large in JAVA by file splitting and assembling. The results of testing show memory leaks can be effective in preventing.

Key words: memory leaks; XML; file splitting

1 引言

内存泄露的根本原因在于没有释放向系统申请的内存, 从而造成这一部分内存不可被再利用, 使得要使用的内存大于虚拟机能提供的最大内存而导致系统崩溃. Java 本身有内存自动管理机制, Java 中是通过对象的分配和释放来实现内存管理. 在程序中通过 NEW 关键字来实现为对象分配空间, 而内存是否释放以及如何释放是由垃圾收集器 (Garbage Collection, GC) 决定和执行的, 为了达到正确释放对象的目的, GC 要一直监控每一个对象的运行状态, 只要对象不再被其它对象引用, 就释放对象所占用的空间. GC 从程序的最开始的运行对象检查引用链, 遍历一遍之后如果发现存在没有被引用的对象就作为垃圾进行回收.

2 JAVA 内存泄露原因

虽然 JAVA 可以实现内存管理, 但也会出现内存泄露, 主要是由于 GC 的机制造成的, GC 对垃圾的确认方式和程序员是不一样的. GC 认为所有不可达的对象全是应回收的垃圾, 而有句柄指向的对象则被认为是正在使用中, 因此不会被它回收; 但对程序员来说, 程序中所有不再需要使用的对象都是应回收的垃圾, 但如果这些程序员认为应被回收的垃圾还正被其它对象引用着的话, 那么 GC 就不会回收它们^[1]. 除此之外, 由于 GC 参数设置不当的话, 也可能增大内存泄露的机率. 引起内存溢出的原因有很多种, 如: 集合类对对象的引用, 使用后没有清空等都会使 JVM 不能回收. 代码存在死循环的情况或者循环产生了过多的重复对象等. 而在内存

中加载的数据量过于庞大是引起内存泄露的一个常见的原因,XML 文件由于能用来存贮那些对传统的数据库来说难于处理的半结构化的文档的数据,所以 XML 文件的应用也越来越多,而 XML 文件过大就是引发内存泄露的原因之一。

3 JAVA 内存泄露的查找判断

当发生内存泄露时,可以按如下思路查找问题。首先可以通过修改 JVM 启动参数,直接增加内存。JVM 默认情况下可用的内存是 64M, Tomcat 的默认可用的内存是 128M,如果应用系统复杂一些可能就会出现不够用的情况。

其次要经常检查错误日志,查看发生“Out-Of-Memory”错误前是否有其它的异常或者错误信息。查看日志虽然简单但对解决问题很有效,因为可以分析内存溢出的原因,通过查看日志,可以分析发生内存溢出前都做过哪些操作,这样可以确定问题出在哪个模块中^[2-3]。

然后检查对数据库查询中,是否有读取大量数据到内存,是否有使用完成后没有清除不用的数据的问题。但要注意的是集合对象,如 MAP、List 等它们会一直有对对象的引用,因此使这些对象不能被 GC 自动回收。如果读取的是 XML 文件,检查文件是否过大。

最后如果上述方法不能发现问题源头,就要使用内存查看工具动态查看内存使用情况检查是否有缓慢的内存泄漏。目前常用的内存查看工具有:Jin-Sight、JProbeProfiler、Optimizeit Profiler 等。这些工具软件的工作原理相同,都是通过监测 Java 程序在运行时对象的状态,并把当前内存中的所有信息通过统计分析后以可视化界面方式提供给开发人员。用户就可以通过这些信息来进行判断当前运行的程序是否有内存泄漏的情况。正常情况下,一个应用系统在启动后它对内存的使用量是相对稳定的。通过对运行的应用系统观察它所占内存的大小进行分析,如果它所占内存是无限限制的持续增长的,就说明当前系统存在内存泄漏的问题。再通过间隔一段时间做一次内存快照,然后对快照中所有对象的使用与引用情况做对比分析,就可以找出是哪个对象存在泄漏^[4-5]。

通过上面的方法如果确定是由于 XML 文件过大引起的内存泄露,并且修改 JVM 启动参数和修改程序无法解决,针对这种情况,文中设计了基于 XML 文件的拆分与拼装方案来从根本上预防内存

泄露的发生。

4 预防 XML 文件过大解决方案

4.1 设计思想

在设计基于 XML 的不同数据库数据之间转换集成方案时,首先要做的工作就是要把现有的数据库中的数据都导出成指定的 XML 文件。但是由于数据库中的数据可能会有很多,都把数据导出到一个 XML 文件中会造成这个文件太大。由于读写 XML 文件时要在内存中构建它的 Document 对象,因此当 XML 文件太大超过一定限值时会引起内存泄漏的情况。通过实验发现,当一次读取超过十万条的记录数据到内存时就可能引起内存溢出^[6]。

为了解决由于文件过大产生的内存泄露问题,设计方案是在从数据库导出时先导出若干个小的 XML 文件,然后再拼装成最终的 XML 文件。相应的,当在进行导入操作时,可以把这个文件再拆分成若干的小 XML 文件,分别解析,然后再把导出的这些小文件组装成最终的数据库文件。由于这种拆分和拼装要付出一定的性能代价,所以在系统中设定一个进行拆分的阈值。首先对要处理的数据进行预判,只有当数据量超过指定值时才进行 XML 文件的拆分拼装操作。而内存泄露也与不同的运行环境有关,因此这个阈值设计成可以进行配置设定。拆分工作流程如图 1 所示。

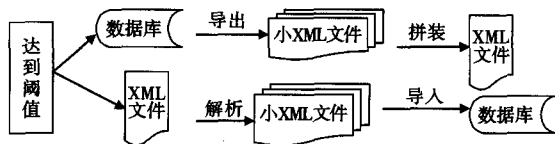


图 1 拆分工作流程

4.2 XML 文件设计

XML 文件中存贮的是一个表或其中的部分数据,如果表中数据超过一定限值就表明数据量过大,就要根据实际情况把数据拆分成若干个小的 XML 文件。由于 XML 文件中表数据是以二维表格行列的方式存贮的,因此可以解析出 XML 文件格式为:

根元素<root>, <root> 中又包含一些元素,我们可以称之为行元素<row>, <row> 行元素中还包含一些子元素<column>, <column> 元素存贮行元素<row> 中一列的值, <column> 中的属性 columnName, 是一个文本结点, 它的属性的值表示<column> 的列名。如果<column> 的类型为 number、data 或者 varchar2, 这个文本结点存贮的

字符串就是数据库中真实的相应字符串,如果 $\langle \text{column} \rangle$ 的类型是 clob 或 blob,由于这两种类型数据存贮的分别是字符文件和二进制文件,则这个文本结点表示的字符串就是一个随机的并且唯一的键值,那么真正的数据就是以这个键值命名的文件.比如 blob 类型列的结点的值为 AV66,则真正的数据文件名为 AV66.data.

假如一个表中的数据太多,那么就分页导出,规定 XML 文件的命名规则是:表名+n+.data.把数据库中的这个数据量大的表导出并形成若干个小的文件后,下一步就要把这些小的文件再组装成最终的文件.建立这个最终文件后,把这些若干个小的文件按顺序逐一追加到这个文件里面,但是要注意的是,在每个小文件前面都要加上一行字符串,字符串的内容是:“文件名”+“-”+“文件字节数”+32 个“=”.

4.3 方案算法描述

- (1) 拆分标志 bool zf;
- (2) 判断是否达到阈值 $zf = \text{ture}$
- (3) 解析表结构文件确定文件名
- (4) 小 XML 文件数 int n;
- (5) 解析 XML 文件确定 n 值
- (6) $\text{for}(i=0; i \leq n; i++)\{$
- (7) 确定拆分的 XML 文件名 // 规则是以表名+n+.data
- (8) 导出小 XML 文件}
- (9) 根据小 XML 文件确定拼装的最终文件名
- (10) $\text{for}(i=0; i \leq n; i++)\{$
- (11) 依次把这些小的文件追加}

5 实验结果分析

在文中设计的不同数据库数据之间通过 XML 文件进行转换集成系统中,实施了对过大 XML 文件进行拆分拼装后,没有再出现由于 XML 文件过大引起内存泄露的问题.测试中发现当读入内存数据量小于设定阈值时,系统整体性能与以前相比几

乎没有变化.当读入内存数据量大于设定阈值时,由于启用了拆分机制使系统整体性能有所下降,根据运行的软硬件环境不同,下降了 3% 到 8%,但这避免了内存泄露,换来了整个系统的稳定,在性能比较高的服务器上下降的影响可以忽略.

6 结束语

JAVA 内存泄露是经常困扰开发人员的一个问题,而且有的内存泄露还有一定的不确定性和隐蔽性.本文就由于 XML 文件过大引起内存泄露的问题进行了探讨,提出了通过对 XML 文件的拆分和拼装预防内存泄露的设计方案,并论述了设计过程.实践证明可以有效地从根本上预防由于文件过大引起 JAVA 内存泄露问题.虽然设计了根据处理的数据量阈值来决定是否进行文件的拆分,但也使系统付出了一定的性能代价.下一步的工作将从提高性能入手改进方案技术策略.

参考文献:

- [1] 朱强. Java 内存泄露问题及对策[J]. 电脑编程技巧与维护, 2009(13):81-83.
- [2] 李伦;陈芳. 内存溢出和数据库锁表的分析与解决[J]. 电脑编程技巧与维护, 2008(11):82-84.
- [3] 张学红,李小慧,杜瑞庆. Java 中内存泄露及垃圾回收机制[J]. 唐山师范学院学报, 2008,30(2):80-82.
- [4] 付晓毓,朱利,顾伟. 基于模型检测的内存泄露静态测试方法[J]. 微电子学与计算机, 2011,27(10):171-173.
- [5] 吴俊军,马鑫龙. 一种混合模式的 Java 卡内字节码校验算法[J]. 微电子学与计算机, 2011,26(4):152-155.
- [6] 李德有. 基于 XML 的银行报表系统数据转换研究[J]. 微计算机信息, 2011,27(5):238-239.

作者简介:

- 李德有 男,(1972—),副教授.研究方向为金融计算机应用.
解晨光 男,(1965—),教授.研究方向为计算机网络.
刘明刚 男,(1977—),讲师.研究方向为数据挖掘.