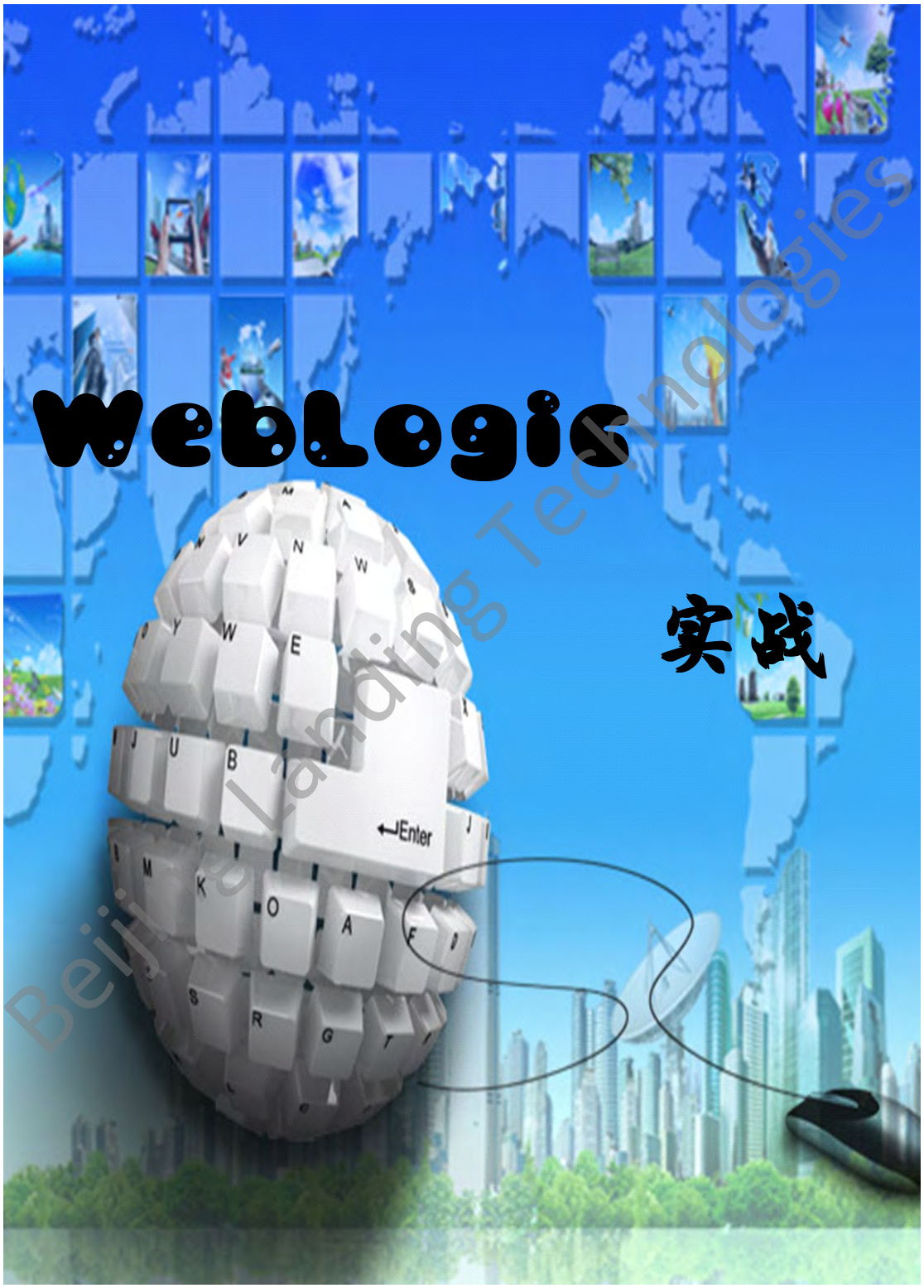
联动北方
LANDING

企业级 IT 运维宝典系列



联动北方 出品

目　录

第 14 章 执行线程丢失故障

在前面章节中，已经从各方面提到了观察系统运行线程状态的方法，尤其是 Thread Dump 这一分析利器，但在某些情况下，我们会发现有些线程在列表中丢失了，本章就重点来分析这方面的知识。

14.1 WebLogic 的执行线程

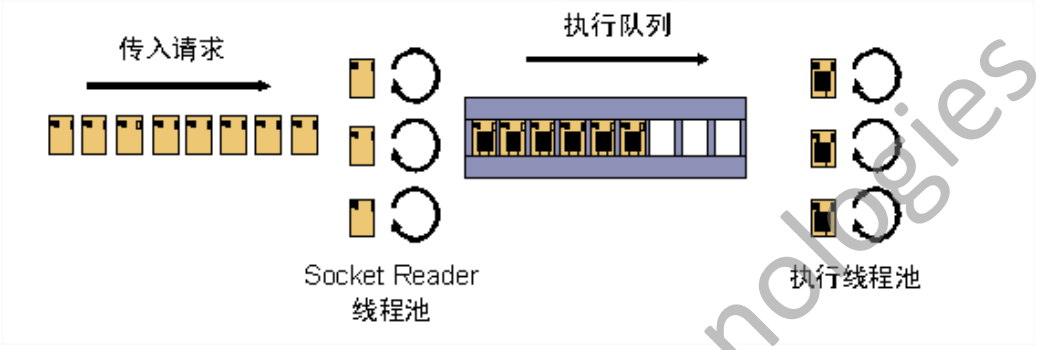


图 14-1 WebLogic 执行线程示意图

WebLogic 实例在线程池中执行所有处理，线程有如下分工：

- 执行线程：负责处理用户请求，执行应用程序要求的具体任务；默认个数：15。
- SocketReader 线程：负责从 socket 读取信息，处理网络通信量；默认个数：执行线程数的 33%(非启动 NativeIO 的情况下)。

SocketReader 线程负责接收进来的请求，然后将其放如执行队列中，最后由执行线程做执行具体的任务。

WebLogic 线程在某些情况下可能会出现丢失现象，正常情况下 WLS 的 Thread Dump 信息中应该显示所有的线程运行信息。

应用程序也可以创建线程，用于执行特定的工作，用户创建的执行线程也可能发生丢失现象。

14.2 丢失线程时的故障症状

14.2.1 故障症状概述

线程丢失时，可能会出现下面的这些故障症状：

（1）可能抛出了未捕获的异常或错误；

（2）通常不会向服务器日志输出任何异常消息、堆栈跟踪或通知；

（3）会从服务器的 Thread Dump 中消失；

（4）当某些线程等待来自其它（已消失的）线程的响应时，可能会导致服务器挂起；

（5）可能会导致服务器挂起，并因此被检测到；

（6）可能会导致意外超时或其它异常行为；

（7）可能是应用程序创建的线程，这些线程会在许多方面对应用程序产生影响；

一般，线程丢失问题会在分析服务器挂起问、未解释的超时或其它令人费解和未解释的行为时被发现。当线程抛出未捕获的异常或错误时，线程可能就会消失。这可能会使服务器挂起，因为可能有其它线程正在等待针对 monitor 的、但是永远不会被调用的 notify()，因为本应调用 notify()的线程消失了，导致其它线程一直在等待就有可能出现挂起现象。

14.2.2 线程丢失信息示例

这里，我们看一下线程丢失时，Thread Dump 信息示例，如下是显示线程 13 和线程 11 丢失的 Thread Dump 示例：

```
    "ExecuteThread: '14' for queue: 'default'" daemon prio=5 tid=0x7bc140
nid=0x13c runnable [0x157ef000..0x157efdc0]
    at weblogic.socket.NTSocketMuxer.getNextSocket(Native Method)
    at weblogic.socket.NTSocketMuxer.processSockets(NTSocketMuxer.java:589)
    at weblogic.socket.SocketReaderRequest.execute(SocketReaderRequest.java: 24)
    at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)
    at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)

    "ExecuteThread: '12' for queue: 'default'" daemon prio=5 tid=0x7bb8e0
nid=0xc5 r unnable [0x1576f000..0x1576fdc0]
    at weblogic.socket.NTSocketMuxer.getNextSocket(Native Method)
    at weblogic.socket.NTSocketMuxer.processSockets(NTSocketMuxer.java:589)
    at weblogic.socket.SocketReaderRequest.execute(SocketReaderRequest.java: 24)
    at weblogic.kernel.ExecuteThread.execute(ExecuteThread.java:139)
    at weblogic.kernel.ExecuteThread.run(ExecuteThread.java:120)

    "ExecuteThread: '10' for queue: 'default'" daemon prio=5 tid=0x7b90c0
nid=0x1a2 runnable [0x156ef000..0x156efdc0]
    at weblogic.socket.NTSocketMuxer.getNextSocket(Native Method)
    at weblogic.socket.NTSocketMuxer.processSockets(NTSocketMuxer.java:589)
    at . . .
```

示例 14-1

以上信息是出现线程丢失的 Thread Dump 示例，由于只出现了执行线程 10、12、14 的信息，而没有执行线程 11 和 13 的信息，所以可以判断执行线程 11 和 13 丢失（异常退出）了。

14.3 线程丢失原因及相应解决方法分析

在出现这种状况后，需要对线程丢失的具体原因进行分析，并找到有针对性的应对和处理办法。

14.3.1 线程丢失原因概述

线程丢失的可能原因包括：

（1）出现内存不足情况，这中情况会导致线程终止时得不到通知；

（2）处理异常时可能会出现问题；

（3）未捕获的异常或错误。

下面来针对不同的情况，一步步进行分析。

14.3.2 JVM 堆内存不足造成的线程丢失

如果通过分析，发现 JVM 堆内存不足是成因，那么：

（1）可能会抛出 OutOfMemoryException，而在处理该异常访问时又再次抛出同一异常，因而导致线程退出；

（2）此时请检查：

- JVM 堆的最大大小
- 实际使用的堆大小
- 堆内分配的永久生成空间的大小

（3）使用以下参数将永久空间设置为 128MB 或更大：

```
-XX:MaxPermSize=128m
```

示例 14-2

如果在设置了 MaxPermSize 以后，问题解决了，那么可能的解释是：不足的 MaxPermSize 设置在某种程度上造成了 OutOfMemoryException 异常。

另外，在这里补充说明一下，如何设置 JVM 的内存大小：

- 在 commEnv 中设置：

找到在..\bea\weblogic\common\bin 目录下的 commEnv.cmd 文件，用文本编辑器编辑，会看到 bea 或是 sun 的 jdk，选择自己所用的 jdk 修改一下代码段（以 sun JDK 为例），重启生效。

```
:sun
if "%PRODUCTION_MODE%" == "true" goto sun_prod_mode
set JAVA_VM=-client
set MEM_ARGS=-Xms32m -Xmx200m -XX:MaxPermSize=128m -XX:+UseSpinning
set JAVA_OPTIONS=%JAVA_OPTIONS% -Xverify:none
goto continue

:sun_prod_mode
set JAVA_VM=-server
set MEM_ARGS=-Xms32m -Xmx200m -XX:MaxPermSize=128m -XX:+UseSpinning
goto continue
```

示例 14-3

- 在 setDomainEnv.cmd 中设置：

找到在..\bea\user_projects\domains\landingbj(域名)\bin 目录下的 setDomainEnv.cmd 文件，修改方法同上。

```
if "%JAVA_VENDOR%"=="Sun" (
   set MEM_ARGS=%MEM_ARGS% %MEM_DEV_ARGS% -XX:MaxPermSize=128m
)
```

示例 14-4

14.3.3 应用程序的异常处理造成的线程丢失

在排查问题时，还要分析应用程序的异常处理，因为应用程序的异常处理也可能会造成线程丢失现象出现。

（1）如果应用程序创建的线程丢失，则可略微缩小搜索范围；

（2）执行代码审核，以充分了解错误的处理方法，特别是对复合错误的处理方法；

（3）检查异常处理代码处理后续异常的方法；

（4）使用调试器检查代码，查找未捕获的异常。

另外，在必要时可修改代码，以免异常处理的方法不正确。

14.4 故障排除检查清单

对这类问题的故障排除策略如下：

（1）使用服务器 Thread Dump 确定线程是否消失；

（2）探查任何 JVM 堆内存不足状态，并根据需要进行矫正；

（3）分析应用程序的异常处理，使用调试器和/或更正任何错误的处理方法；

（4）检查是否有与线程消失有关的任何其它异常事件；

（5）继续监视，看是否还会发生故障。