记一次java进程频繁挂掉问题排查修复

前言

最近业务部门有个java服务进程会突然无缘无故的挂掉，然后这个服务会产生一堆类似hs_err_pid19287.log这样的日志。业务部门负责人就把hs_err_pidxxx的日志发给我，让我帮忙看下问题。本文就来回顾一下，我是如何帮业务部门进行问题排查

排查历程

首先hs_err_pidxxx的日志有提示如下内容

我就让业务部门那边配置下ulimit 。具体步骤如下

```
vim /etc/security/limits.conf

# 在最后追加
* soft nofile 327680
* hard nofile 327680
```

不过业务部门负责人跟我反馈说，他们早就加过了，但是不管用。

接着继续分析hs_err_pidxxx的日志内容

k3u1fbpfcp/8aae08080f884a81b78993720d105ee7~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1210&h=441&s=66730&e=png&b=fffefe)

看到新生代的内存出现100%，就问业务负责人说，你们是不是jvm内存设置比较小，结果得到的反馈是他们最近刚进行内存扩容，也进行了相应的jvm设置，内存肯定够，而且程序日志并没有oom的相关日志信息。

于是继续分析hs_err_pidxxx的日志内容，

看着大量的thread_blocked，感觉看到问题要被修复的曙光，于是跟业务负责人说，你们代码可能存在堵塞了，业务负责人说这个服务运行多年了，而且其他机子好好的，如果有这个问题，正常早就暴露出来了。

从这个hs_err_pidxxx日志，我能得到就这些信息。看着业务负责人的由期待再到眼神无光，我大大的感受他有一种所托非人的感觉。后面我跟他说，不然你jdk升级一个小版本，其实我只是试探，毕竟升级jdk，带来收益的时候，可能也带来风险，尤其在运行多年的项目上。没想到业务负责人回答说正有此意。

后面业务负责人就拿那台有问题机子进行jdk升级，事情就暂时告了一段落

问题后续

后来同部门的架构师在某次吃饭和我提起业务部门这个问题，我才发现业务部门升级jdk后，仍然没用。于是业务负责人找了部门的架构师进行求助。知道这个事后，我就主动去找业务部门负责人，问他问题解决没啊，得到他否定的答案后。

本着负责到底的心，我先向他们要来他们宿主机的messages日志。就是位于/var/log/messages。看到如下信息

k3u1fbpfcp/45450fe47dc44882b09ea3752e779415~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1504&h=80&s=13777&e=png&b=fffefe)

里面日志有个abrt-server。这边科普一下。

> 什么是abrt-server

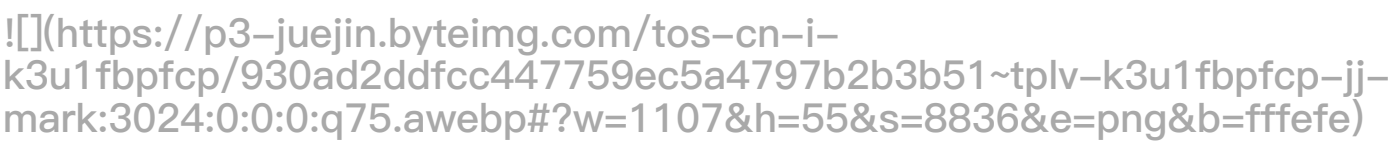
abrt是centos操作系统中的一个错误报告和跟踪工具。它可以自动收集应用程序和系统的错误信息，并生成错误报告。当系统发生错误时，abrt会收集相关的信息，如错误消息、堆栈跟踪、核心转储等。它会生成一个错误报告，包含了这些信息以及其他有用的调试信息。

它的记录保存在内核core文件，随着时间推移，core文件会不断变大，会占用磁盘空间。我们可以使用 **abrt-cli list** 确认core状态对应的进程及触发时间。并通过abrt-cli rm 【文件包】进行删除

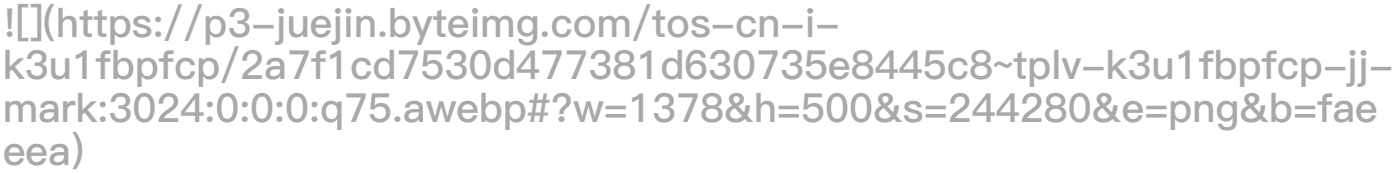
示例：

```
abrt-cli rm /var/spool/abrt/oops-2022-09-27-14:22:55-13596-0
```

回归正题，我们通过/var/log/messages里面的内容



通过搜索资料，得知这个错误是因为**无法创建ccpp文件导致**。但这个是不是导致java进程频繁挂掉的原因，于是我们做了这么一步，将无法创建ccpp文件的时间点和生成的hs_err_pidxxx时间点做个对比



时间点基本上是吻合的，而且/var/log/messages里面还有一段

```
Executable '/usr/local/tomcat/jdk1.8.0_291/bin/java' doesn't belong to any package and ProcessUnpackaged is set to 'no'
```

后与业务负责人确认，这个jdk确实是目前这个业务所使用的jdk。综上基本上可以确定是因为无法创建ccpp文件导致，导致该业务的java进程频繁挂掉的原因之一

如何修复

> 方法一：将ProcessUnpackaged改为yes

这个参数的意思是表示ABRT将非rpm安装程序（如源代码包等）识别为未打包程序，会生成相关的警告和错误日志，因而会更好地抓住一些程序的 bug。如果为no，表示ABRT将不会跟踪和报告那些在未打包的应用程序中发生的崩溃信息，而只针对现有的软件包进行跟踪。因此，使用yes选项能够扩展 ABRT 的范围，提高异常程序的捕捉数量，但同时可能也会导致一些误报

```
sed -i 's/ProcessUnpackaged = no/ProcessUnpackaged = yes/g' /etc/abrt/abrt-action-save-package-data.conf&& systemctl restart abrtd.service
```

或者分步执行也可以

```
vim /etc/abrt/abrt-action-save-package-data.conf
ProcessUnpackaged = yes
systemctl restart abrtd.service
```

```

不过这边还有个细节要注意，核心转储文件的默认最大大小为5000，我们可以按实际情况调整，也可以设置为0，为0表示核心转储文件的大小不作限制，不过设置为0有个风险点是可能会磁盘空间占满，因为core的文件正常比较大

可以通过如下配置，修改MaxCrashReportsSize参数

```
vim /etc/abrt/abrt.conf

MaxCrashReportsSize = 0

systemctl restart abrtd.service
```

或者执行如下命令

```
sed -i "s/MaxCrashReportsSize = 5000/MaxCrashReportsSize = 0/g" /etc/abrt/abrt.conf && systemctl restart abrtd.service
```

> 方法二：禁用abrtd

abrt-hook-ccpp在进行执行崩溃转储操作时，使用的内存有可能会超过预期或系统能够提供的内存限制，导致影响其他应用程序。因此我们也可以直接执行如下命令，禁用abrtd

```
systemctl stop abrt-ccpp.service
systemctl disable abrt-ccpp.service
systemctl status abrt-ccpp.service
```
```

总结

执行了如上操作，业务部门观察了一段时间，没有再发现java进行频繁挂掉问题。

此外不管是容器化部署还是传统的宿主机部署，当出现问题时，没头绪时，我们可以通过查看/var/log底下的各种日志进行梳理。如下链接

[www.cnblogs.com/deverz/p/87...](http://cxyroad.com/”https://www.cnblogs.com/deverz/p/8779357.html”)

是各种var/log的描述说明，感兴趣的小伙伴可以看看