功能问题，通过日志，单步调试相对比较好定位。

性能问题，例如线上服务器 CPU100%，如何找到相关服务，如何定位问题代码，更考验技术人的功底。

58 到家架构部，运维部，58 速运技术部联合进行了一次线上服务 CPU 问题排查实战演练，同学们反馈有收获，特将实战演练的试题和答案公布出来，希望对大家也有帮助。

题目

某服务器上部署了若干 tomcat 实例，即若干垂直切分的 Java 站点服务，以及若干 Java 微服务，突然收到运维的 CPU 异常告警。

问：如何定位是**哪个服务进程**导致 CPU 过载，**哪个线程**导致 CPU 过载，**哪段代码**导致 CPU 过载？

步骤一、找到最耗 CPU 的进程

工具：top

方法：

- 执行 top -c ，显示进程运行信息列表
- 键入 P (大写 p)，进程按照 CPU 使用率排序

图示：

```
[work@37-8-12 ~]$ top -c
top - 20:27:36 up 501 days,  7:25,  1 user,  load average: 0.51, 0.39, 0.27
Tasks: 187 total,   1 running, 186 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.9%us,  1.2%sy,  0.1%ni, 97.7%id,  0.2%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  32880592k total, 32105700k used,   774892k free,   397648k buffers
Swap:        0k total,        0k used,        0k free, 24724980k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
10765 work      20   0 15.2g 3.7g  48m S 19.9 12.0  15151:26 java -Xms4g -Xmx4g -Xmn1g -Xss1024K -XX:PermSize=256m -X
21919 work      22   2 5849m 1.5g  10m S  1.7  4.7   3804:29 /opt/soft/java/bin/java -Xms1024m -Xmx1024m -cp /opt/log
17604 root      20   0  118m  13m 8576 S  1.0  0.0   6:50.94 /usr/local/aegis/aegis_client/aegis_10_23/AliYunDun
 1743 work      20   0 4722m 249m  11m S  0.3  0.8 206:11.87 java -XX:PermSize=128m -XX:MaxPermSize=256m -Xms512m -Xm
17551 root      20   0 24608 2648 2028 S  0.3  0.0   1:21.93 /usr/local/aegis/aegis_update/AliYunDunUpdate
    1 root      20   0 19356 1360 1060 S  0.0  0.0   5:02.92 /sbin/init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.26 [kthreadd]
    3 root      RT   0     0    0    0 S  0.0  0.0   0:15.64 [migration/0]
    4 root      20   0     0    0    0 S  0.0  0.0  19:06.37 [ksoftirqd/0]
```


如上图，最耗 CPU 的进程 PID 为 10765

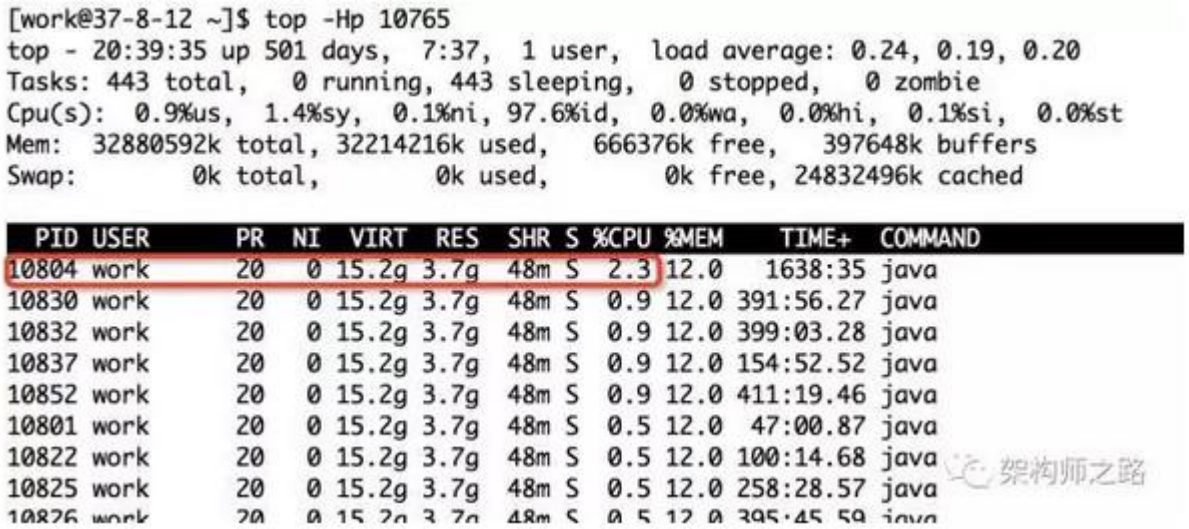
步骤二：找到最耗 CPU 的线程

工具：top

方法：

- top -Hp 10765 ，显示一个进程的线程运行信息列表
- 键入 P (大写 p)，线程按照 CPU 使用率排序

图示：



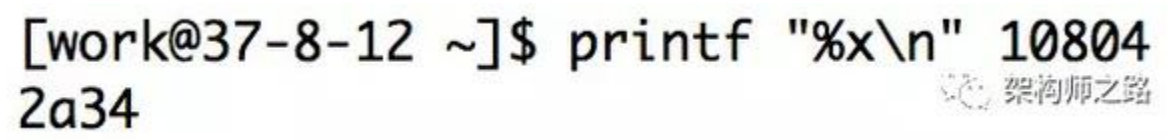
如上图，进程 10765 内，最耗 CPU 的线程 PID 为 10804

步骤三：将线程 PID 转化为 16 进制

工具：printf

方法：printf “%x\n” 10804

图示：



如上图，10804 对应的 16 进制是 0x2a34，当然，这一步可以用计算器。

之所以要转化为 16 进制，是因为堆栈里，线程 id 是用 16 进制表示的。

步骤四：查看堆栈，找到线程在干嘛

工具：pstack/jstack/grep

方法：jstack 10765 | grep ‘0x2a34’ -C5 --color

- 打印进程堆栈
- 通过线程 id，过滤得到线程堆栈

图示：

如上图，找到了耗 CPU 高的线程对应的线程名称“AsyncLogger-1”，以及看到了该线程正在执行代码的堆栈。