案例一：Java服务响应变慢，传统工具查不出来

某个Java服务响应从50ms涨到200ms，CPU、内存、IO全都正常，日志也没报错。用jstack看线程也没发现死锁。

同事甩给我一个bpftrace✦脚本，追踪read系统调用的延迟分布：

```
bpftrace -e '
tracepoint:syscalls:sys_enter_read /comm == "java"/ {
    @start[tid] = nsecs;
}
tracepoint:syscalls:sys_exit_read /comm == "java" && @start[tid]/ {
    @usecs = hist((nsecs - @start[tid]) / 1000);
    delete(@start[tid]);
}
'
```

跑了几秒钟，输出一个直方图，128-256微秒这个区间的调用数量异常偏多。正常应该是单峰分布，这里出现了双峰。

顺着这个线索往下查，发现是读配置文件的时候有锁竞争。问题定位花了不到5分钟。

换strace能不能查？能，但strace开销太大，挂上去服务更慢了，生产环境不敢用。eBPF几乎零开销，直接在生产跑。

案例二：TCP延迟偶发飙高

生产环境有台机器偶尔TCP延迟飙到几百毫秒，网络组说线路没问题。

用BCC✦自带的tcpretrans追踪重传：

```
/usr/share/bcc/tools/tcpretrans
```

输出：

```
TIME     PID    IP LADDR:LPORT          T> RADDR:RPORT          STATE
14:23:15 0      4  10.0.1.5:443         R> 10.0.2.8:52341       ESTABLISHED
14:23:15 0      4  10.0.1.5:443         R> 10.0.2.8:52341       ESTABLISHED
14:23:16 0      4  10.0.1.5:443         R> 10.0.2.8:52341       ESTABLISHED
```

同一个连接连续重传，问题明显在10.0.2.x这个网段。

再用tcpconnlat看连接延迟：

```
PID    COMM         IP SADDR            DADDR            DPORT LAT(ms)
1892   curl         4  10.0.1.5         10.0.2.8         443   245.12
1893   curl         4  10.0.1.5         10.0.2.8         443   312.45
1894   curl         4  10.0.1.5         10.0.3.9         443   1.23
```

对比明显，连10.0.2网段延迟高了两个数量级。拿着这个证据去找网络组，最后查出来是那个机房的交换机有问题。

案例三：容器CPU高但不知道谁在吃

有个容器CPU一直70%+，top看不出来具体哪个函数在消耗。

用profile做CPU采样生成火焰图：

```
/usr/share/bcc/tools/profile -p $(pgrep -f myapp) -f 30 > profile.out
./FlameGraph/flamegraph.pl profile.out > cpu.svg
```

火焰图一看，某个JSON解析函数占了40%的CPU。原来是每次请求都在重复解析同一个大配置文件，解析完也不缓存。加个缓存，CPU直接降到20%。

案例四：查谁在频繁读写某个文件

有段时间某个日志文件疯狂写入，不知道是哪个进程干的。

```
bpftrace -e '
kprobe:do_sys_openat2 {
    @files[str(arg1)] = count();
}
interval:s:5 {
    print(@files);
    clear(@files);
}
'
```

每5秒输出一次文件打开统计，很快就抓到了那个"幽灵进程"。

为什么说它强大

传统排查方式的问题：

strace：开销大，挂上去服务变慢，生产环境不敢用
perf：功能强但学习曲线陡
tcpdump：只能抓包，看不到内核内部
加日志：要改代码、重新部署，等不及

eBPF的优势：

开销极低：生产环境直接用，不影响业务
内核级观测：系统调用、网络栈、文件系统，什么都能看
动态加载：不用重启服务，不用改代码
安全：内核会校验你的程序，不会搞崩系统

Cloudflare✦用它扛DDoS，Facebook用它做负载均衡，这些大厂早就在生产环境大规模用了。