编写说明

标题：系统性能分析手册

类别：文档

存放位置：维护文档\-× -×××××× -系统性能分析-V1.0.0.doc

编辑软件：WPS 中文版

版本历史：

版本	作者	日期	备注
V1.0.0	Tian	2017 年 04 月 14 日	教学手册，内部文件。

前言--漫谈

首先谈谈初衷:应用离不开系统。其实每个运维都知道，影响系统性能的方向体现在系统层面无非是 io 和 cpu。Io 分为网络 io 和系统 io。内存是 io 和 cpu 之前的桥梁。内核管理的所有的内存，而进程只能使用到最大是自己虚拟内存的内存地址空间。下面是一些图作为前言部分可以帮助你思考。

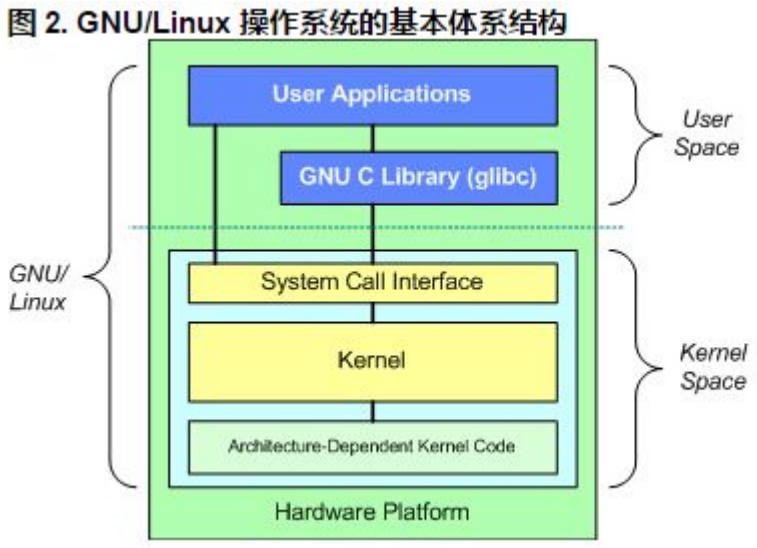


图 1

Linux 内核的主要子系统

现在使用图 3 中的分类说明 Linux 内核的主要组件。

图 3. Linux 内核的一个体系结构透视图

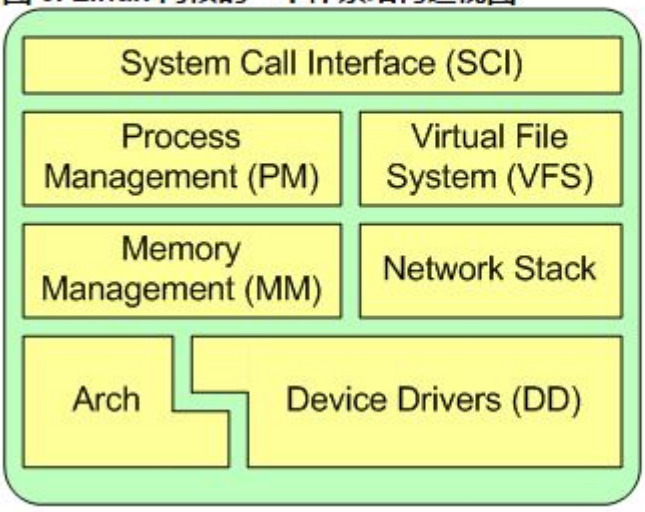


图 2

图 4. VFS 在用户和文件系统之间提供了一个交换层

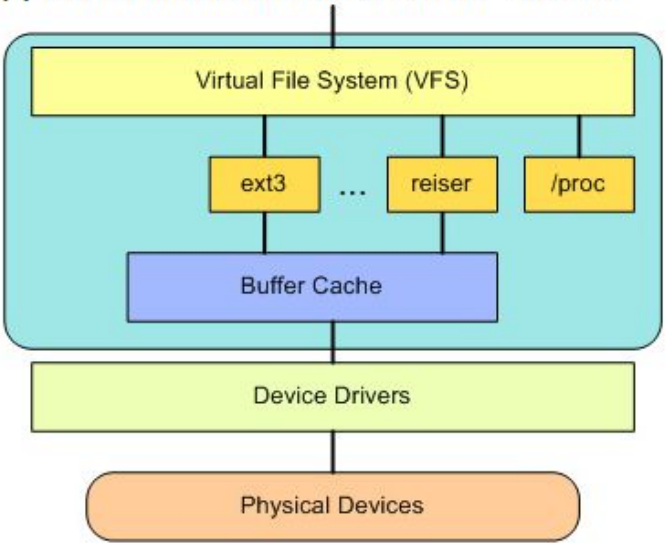


图 3

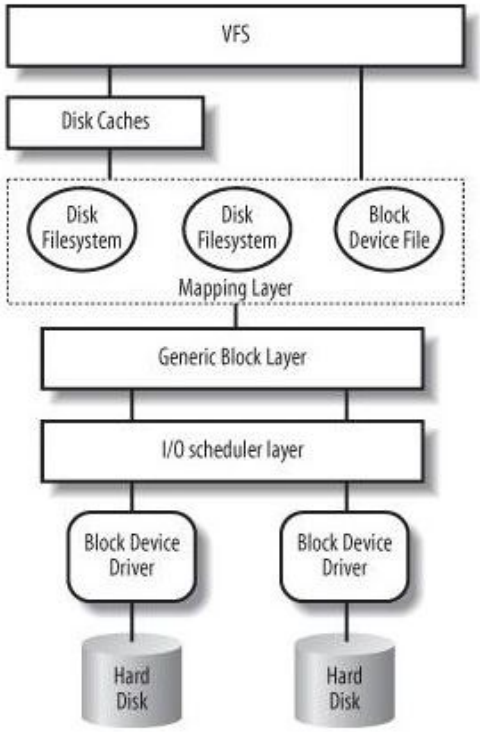


图 4

看了这么多的图。你一定在想这么多的图。那么程序是怎么通过 syscall 完成文件的 io 操作呢。下面就简单介绍一下整个过程吧。其实很简单就是 user space process 通过 syscall 内核方法调用在准备 IO 时，内核会在 buffer 内存开辟一段空间，然后把这段空间的读写权限交给 DMA，DMA 把文件加载到这段内存空间后告诉 CPU 完成任务，然后 CPU 把这段内存空间的访问交给进程。进程通过文件句柄实现 write-copy 到自己的内存空间然后完成文件写操作，读操作直接访问进程 share 就可以了。下面通过各个小的模块认识系统。详情请关注下面的内容。谢谢！

系统性能分析总纲：
1，整体系统 cpu 利用率
2，内存利用率
3，磁盘 I/O 的利用率和延迟
4，网络利用率

注明：os 是一个大的系统工程。分析问题必须分析整体才能确认问题。切勿知其一不知其二，一叶遮目。

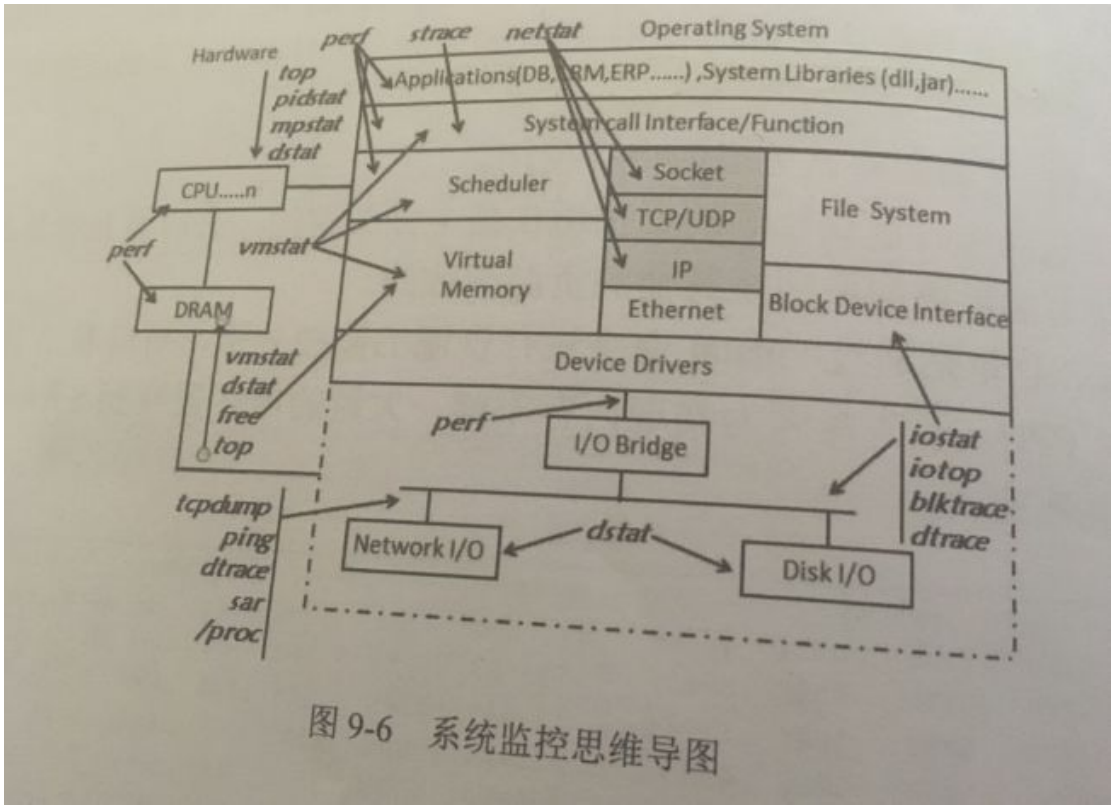
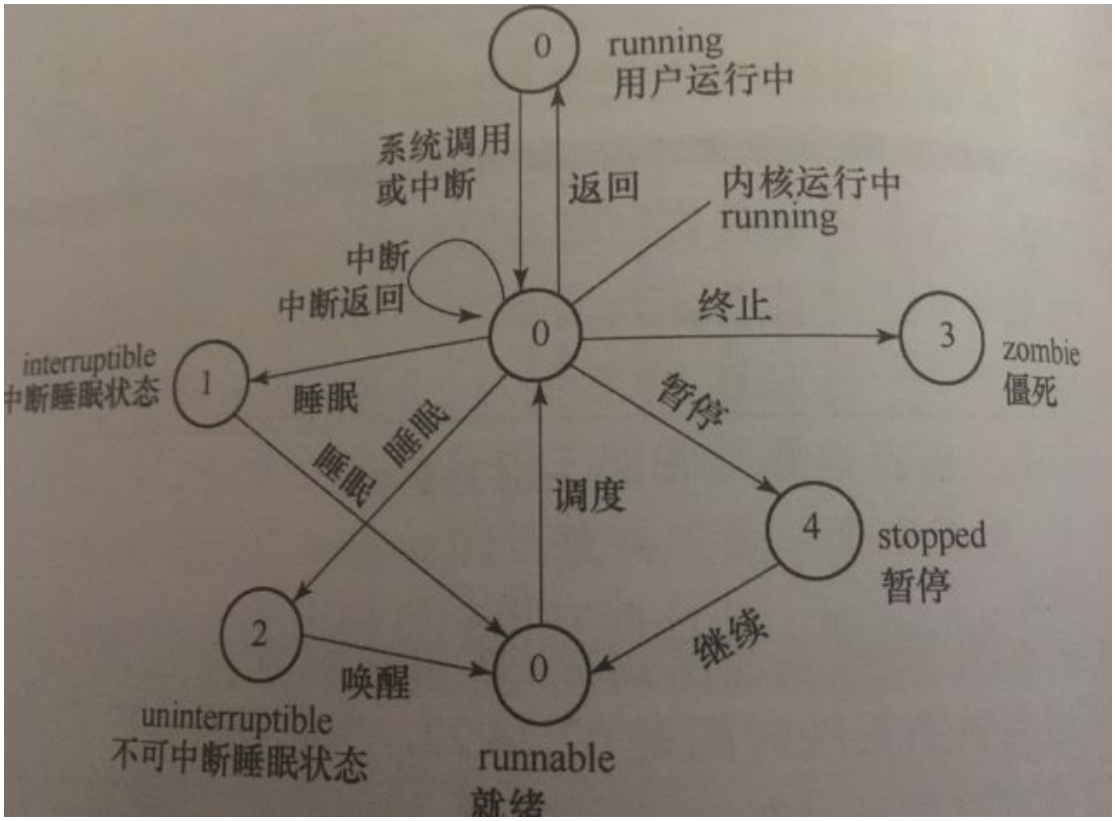


图 9-6　系统监控思维导图



一，主题：分析问题判断依据：

Cpu:

1,运行的任务队列长度，uptime（读取的是/proc/loadavg）　　sar -q　　vmstat 的 r 列

2,cpu 空闲比，us，sy，ni 改变过优先级的进程占用 CPU 的百分比

Wa IO 等待占用 CPU 的百分比

hi　硬中断（Hardware　IRQ）硬件中断占用 CPU 的百分比

si 软中断占用 CPU 的百分比 （主要是来自网络的中断 ksoftirqd 进程）

3,注意大量的网络吞吐量会导致占用 cpu 的资源增大。此时系统要分出部分资源去进行软中断处理

4,大量的 cpu 操作会尝试使用更多的内存

5，IO 范畴的应用不对 cpu 及网络发起更多请求。除非是 NAS 网络存储。IO 范畴的应用通常使用 cpu 资源是为了产生 IO 请求以及进入内核调度的 sleep 状态。 MySQL

6，CPU 范畴的应用。一般都是高 cpu 负载。通常就是一个批处理 CPU 请求以及数学计算的过程。Web server

7，统计那些进程占用了 CPU 时间。Top 命令

8，perf 捕获处理器的错误信息

9，亲缘性查看。Top 设置 Last used CPU。可以观察到进程运行到内个逻辑 cpu 上。

IO：

1，需要考虑 IO 的 TPS，平均 IO，平均队列长度，平均服务时间，平均登录时间，IO 利用率

2，iostat -xz　　%util

Sar -d　　%util

Iotop　　利用率很高

iostat -xz　　avgqu-sz>1

dmesg |grep IO　　IO 报错

Net：

1，统计收发包的速率，以及总共大小

2，通过 sar -n EDEV 5 3 EDEV 是网络错误统计，初步估计查看是否有丢包和阻塞情况

3，top 观察是否有大量的软中断，si 软中断

4，统计 sar -n DEV 和 ifconfig RX/TX 等信息

5，iocstat 命令可以直接观察 uti 是否占满

Mem：

1，free 查看内存使用情况

2，vmstat 查看 swap 的 si（换入）　so（换出）

3， sar -r 5 5　　// 内存和 swap 空间使用情况和 free 基本一样

4，sar -W swap 换入换出情况

5，sar -B 5 5　　// 页交换速率

6，OOM　　dmesg |grep kill

7，buffer 和 cache 的作用是缩短 I/O 系统调用的时间，如果 cache 很大。Buffer 和 cache 是可以自动回收的。例如 mysql 设置的虚拟内存是 23G，这个时候你需要关注的是 RES 和 mem total。

8，VIRT 虚拟内存　RES 进程使用的内存　　share 共享内存　PR 优先级　NI 定义的优先级

二，命令展示

CPU 定位分析

```
[root@localhost ~]# vmstat 2 3
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 2  0 622584 175668      0 473336    5   14    97    42   79  172  1  0 99  0  0
 0  0 622584 175668      0 473352    0    0     0     0   77  193  0  0 100  0  0
 0  0 622584 175668      0 473352    0    0     0     0   76  192  1  0 100  0  0
[root@localhost ~]#
```

```
[root@localhost ~]# uptime
 23:15:10 up 1 day,  6:21,  1 user,  load average: 0.00, 0.01, 0.05
[root@localhost ~]#
```

```
[root@localhost ~]# sar -u 2 3
Linux 3.10.0-229.el7.x86_64 (localhost)         04/12/2017      _x86_64_        (1 CPU)

11:15:54 PM     CPU     %user     %nice   %system   %iowait    %steal     %idle
11:15:56 PM     all      0.50      0.00      0.50      0.00      0.00     99.00
11:15:58 PM     all      0.00      0.00      0.00      0.00      0.00    100.00
11:16:00 PM     all      0.00      0.00      0.00      0.00      0.00    100.00
Average:        all      0.17      0.00      0.17      0.00      0.00     99.67
[root@localhost ~]#
```

```
top - 23:21:19 up 1 day,  6:27,  1 user,  load average: 0.00, 0.01, 0.05
Tasks: 357 total,   2 running, 355 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.3 us,  0.3 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  1003164 total,   174224 free,   354744 used,   474196 buff/cache
KiB Swap:  2097148 total,  1474712 free,   622436 used.   470620 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
 1190 root      20   0  560824  12336   2128 S  0.3  1.2   0:24.25 dockerd-current
 5883 root      20   0 1079708  26544   3992 S  0.3  2.6   1:10.92 java
15199 root      20   0  146296   2272   1432 R  0.3  0.2   0:00.04 top
    1 root      20   0   44092   4468   2752 S  0.0  0.4   0:03.40 systemd
    2 root      20   0       0      0      0 S  0.0  0.0   0:00.04 kthreadd
    3 root      20   0       0      0      0 S  0.0  0.0   0:06.68 ksoftirqd/0
```

主要判断：

1，idle 空闲

2，vmstat -r 计数

3，sar -q　runq-sz 计数

4，task 队列信息

，perf 命令可以获取处理器的错误信息

NET 定位分析

主要是看设备接收和发送的数据包。DEV 是网络接口信息。EDEV 是网络错误统计，SOCK 是套接字句柄统计

```
[root@localhost ~]# sar -n DEV 2 3
Linux 3.10.0-229.el7.x86_64 (localhost)         04/12/2017      _x86_64_        (1 CPU)

11:24:41 PM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
11:24:43 PM eno16777736      1.51      0.50      0.09      0.08      0.00      0.00      0.00
11:24:43 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
11:24:43 PM   docker0      0.00      0.00      0.00      0.00      0.00      0.00      0.00

11:24:43 PM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
11:24:45 PM eno16777736      1.01      1.01      0.06      0.26      0.00      0.00      0.00
11:24:45 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
11:24:45 PM   docker0      0.00      0.00      0.00      0.00      0.00      0.00      0.00

11:24:45 PM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
11:24:47 PM eno16777736      0.50      0.50      0.03      0.23      0.00      0.00      0.00
11:24:47 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
11:24:47 PM   docker0      0.00      0.00      0.00      0.00      0.00      0.00      0.00

Average:        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s
Average:    eno16777736      1.01      0.67      0.06      0.19      0.00      0.00      0.00
Average:           lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:      docker0      0.00      0.00      0.00      0.00      0.00      0.00      0.00
[root@localhost ~]#
```

```
[root@localhost ~]# sar -n EDEV 2 3
Linux 3.10.0-229.el7.x86_64 (localhost)         04/12/2017      _x86_64_        (1 CPU)

11:25:05 PM     IFACE   rxerr/s   txerr/s    coll/s  rxdrop/s  txdrop/s  txcarr/s  rxfram/s  rxfifo/s  txfifo/s
11:25:07 PM eno16777736      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
11:25:07 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
11:25:07 PM   docker0      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00

11:25:07 PM     IFACE   rxerr/s   txerr/s    coll/s  rxdrop/s  txdrop/s  txcarr/s  rxfram/s  rxfifo/s  txfifo/s
11:25:09 PM eno16777736      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
11:25:09 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
11:25:09 PM   docker0      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00

11:25:09 PM     IFACE   rxerr/s   txerr/s    coll/s  rxdrop/s  txdrop/s  txcarr/s  rxfram/s  rxfifo/s  txfifo/s
11:25:11 PM eno16777736      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
11:25:11 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
11:25:11 PM   docker0      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00

Average:        IFACE   rxerr/s   txerr/s    coll/s  rxdrop/s  txdrop/s  txcarr/s  rxfram/s  rxfifo/s  txfifo/s
Average:    eno16777736      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:           lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
Average:      docker0      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00
[root@localhost ~]#
```

```
[root@localhost ~]# sar -n SOCK 2 3
Linux 3.10.0-229.el7.x86_64 (localhost)         04/12/2017      _x86_64_        (1 CPU)

11:25:27 PM    totsck    tcpsck    udpsck    rawsck   ip-frag    tcp-tw
11:25:29 PM       448         7         0         0         0         0
11:25:31 PM       448         7         0         0         0         0
11:25:33 PM       448         7         0         0         0         0
Average:          448         7         0         0         0         0
[root@localhost ~]#
```

Ifconfig 命令 RX/TX

```
eno16777736: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.18.11.100  netmask 255.255.255.0  broadcast 172.18.11.255
        inet6 fe80::20c:29ff:fe38:289b  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:38:28:9b  txqueuelen 1000  (Ethernet)
        RX packets 510899  bytes 671680047 (640.5 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 202307  bytes 38614470 (36.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

VFS IO 定位分析

Mem 内存定位分析

```
[root@localhost ~]# sar -r
Linux 3.10.0-229.el7.x86_64 (localhost)         04/12/2017      _x86_64_        (1 CPU)

10:30:01 PM kbmemfree kbmemused  %memused kbbuffers  kbcached  kbcommit   %commit  kbactive   kbinact   kbdirty
10:40:02 PM    195644    807520     80.50         0    370332   4554608    146.91    216860    445132        36
10:50:01 PM    195500    807664     80.51         0    370360   4554608    146.91    216948    445260        32
11:00:01 PM    191600    811564     80.90         0    373512   4556092    146.96    218536    447324        32
11:10:01 PM    175264    827900     82.53         0    389652   4554680    146.91    233048    449100        32
11:20:01 PM    174664    828500     82.59         0    389900   4554680    146.91    243584    439068        32
Average:       186534    816630     81.41         0    378751   4554934    146.92    225795    445177        33
[root@localhost ~]#
```

```
[root@localhost ~]# free
              total        used        free      shared  buff/cache   available
Mem:        1003164      339956       65204        1436      598004      489632
Swap:       2097148      637644     1459504
[root@localhost ~]# free
```

Mem buffer　用作内核缓存的内存量
Mem cache　用作缓冲的交换区总量

很多人说不清楚是什么是 buffer 和 cache。其实很简单，你要这么想。内存是干嘛的。内存是用来存储各种变量的。学过 python 的人都知道。函数在内存中存储的是函数体，而函数体现给用户的也是变量。这个变量就是函数的门牌号也是函数名。变量调用也是通过内存寻址完成，变量的运维返回就是这个变量所在的程序进程的内存虚拟空间。说了这么多为了更好的理解内存。下面我就举一个 swap 例子：

一个程序运行最要指标就是虚拟内存空间。当一个程序运行时系统的物理内存空间不够时，就需要将物理内存中的一些很长时间没有什么操作的程序释放出来。这些被释放的空间被临时保持到 swap 内存中，等待那些程序需要运行时，再从 swap 内存中恢复保持的数据到物理内存中。这样，系统总是在物理内存不够时，才进行内存之间的交换。

三，Tomcat 监控指标（prode）：

JVM:

1，jvm 内存 关注 GC 回收，full GC 次数

2，最大线程　线程池链接数长期大于 80%建议优化

3，数据库连接数 活动链接数大于 80% 建议优化

4，请求状态和请求数　线程数，线程状态，大量的 Blocked 状态线程 wait timeoutx 线程可以 Dump 线程堆信息进行分析。堆中的对象合并分析。

5，对 tomcat 的 jdbc 的 bool 监控　zabbix gateway jar

注明 jvm 是解释性语言。有一个预编译成 jvm 的二进制码提高效率

Jvm 内存存储：

堆　对象

栈　内存地址　对象引用

方法区　存储方法体 class　　class 不执行不会报错。

四，MySQL 监控：

Buffer

Key cache 相关

Index cache 相关

Tmp　相关

Type　ALL 统计

Tps

Qps

Theard

Lock lockwait lockTX

Mem buffer cache cache 是否稳定

Swap 是否有被使用的情况　内存不够

当前 cpu task 长度是否超过了稳定值得 20%

Io 队列大小及队列长度 及 io util

审计 init common

慢查询日志 过滤分析。及没有索引的日志记录　自动优化可以参考美团

业务监控和 mysql 主机的系统资源监控合并