编译论坛

先进编译实验室
Advanced Compiler

程序性能的分析和测量I

嘉宾：柴晓楠

先进编译实验室
Advanced Compiler

程序性能的分析

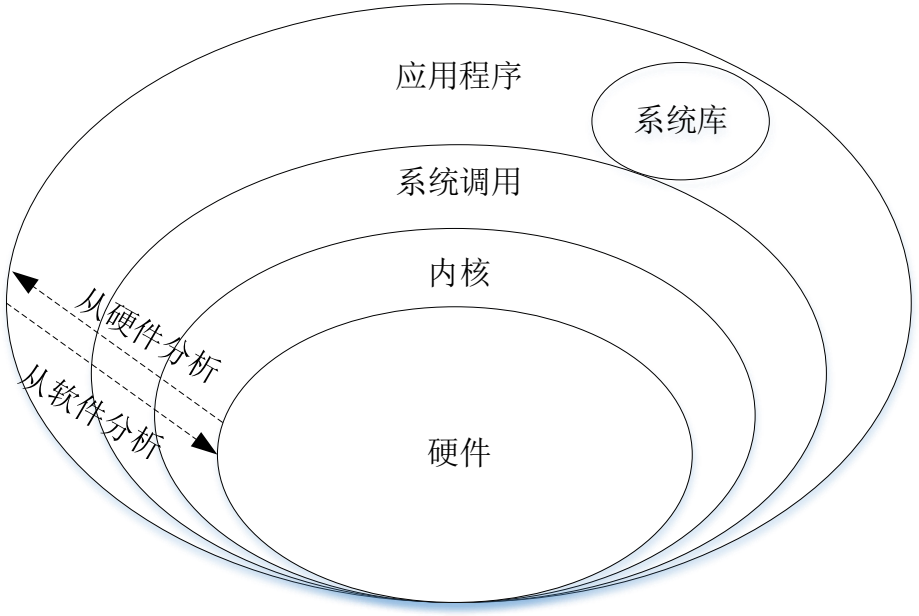
性能分析是程序性能优化的核心，可以为程序的后续优化打下基础，但一般直接通过分析代码是很难确定性能好坏的，通常还需要使用实验或理论方法对程序性能进行较为准确的测量和分析。本节主要介绍的是其中的基础理论部分，主要包括两部分：

（1）程序性能分析的视角

（2）程序性能分析的方法；

程序性能分析的视角

对于程序的性能来说可以从多个角度对它进行分析，其中从硬件层面分析和从软件层面分析是常用的两个视角，如图所示。



程序性能分析的方法

分段查找

进行程序性能分析之前需要首先定位导致程序性能瓶颈产生的原因，才能针对性的开展后续的优化。分段查找是常用定位程序分析代码段的方法之一，可以在时间或空间层面进行，主要思想是根据需要获取某段代码的执行信息并进行分析，查找问题所在。

等待排队

等待排队可以抽象为办理业务排队直到业务办理结束所耗费的整体时间。对应于程序则为程序进入就绪队列到程序执行完成所耗费的时间。等待排队方法主要分析等待进程数以及进程需等待多久。

little估算

little定律的等式为：L=λ*W。其中变量的意思是L 表示在一段时间内排队系统中的平均任务或项目数量即排队队列中的任务数，λ表示在规定的时间间隔内新进入系统的平均任务或项目数量即新任务到达率，W 表示任务或项目在整个系统中花费的平均时间即任务的平均花费时间。

程序性能的测量

程序性能的测量则是利用工具进行较为精准的定位，是对猜测的一种验证。本节主要分为以下三个方面：

- **程序性能测量的信息类型**
- **程序性能测量的工具类型**
- **程序性能测量的工具**

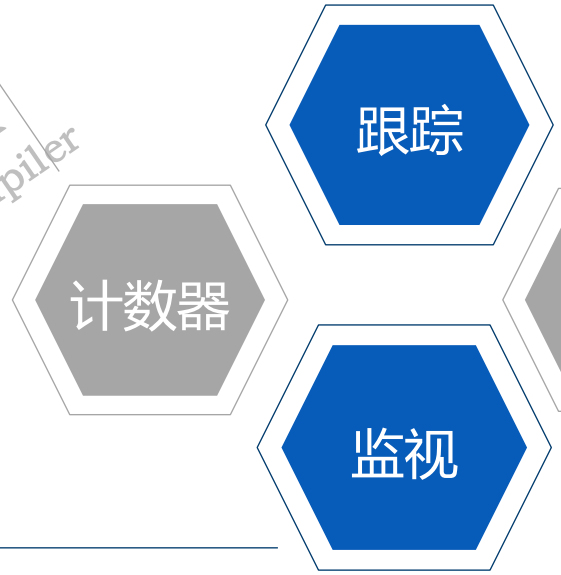
程序性能分析的信息类型

程序性能测量信息按照生成的信息类型可以分为：

- 概要形式是以汇总或者平均值的形式来展示一段时间的程序信息，必须等待一段时间来获取信息，是比较费时的。
- 事件记录形式是逐个记录每个事件，生成系统信息。优点在于可以获得关于时间、位置等详细的信息，缺点是在核对进程到达和出发时会比较费时，效率较低。
- 快照形式是记录当前信息的方式，来生成性能信息。优点是比较适合查找引起性能问题的原因。

程序性能分析的工具类型

在系统内核中，一般会生成一些用于对事件发生次数进行计数的统计数据，称为计数器。通常计数器为无符号的整型数，事件发生时递增。例vmstat、iostat、top、ps



跟踪工具即跟踪收集每一个事件的数据以供性能分析。一般默认不启用，跟踪捕获储存数据会有开销。例gdb，pstack，strace

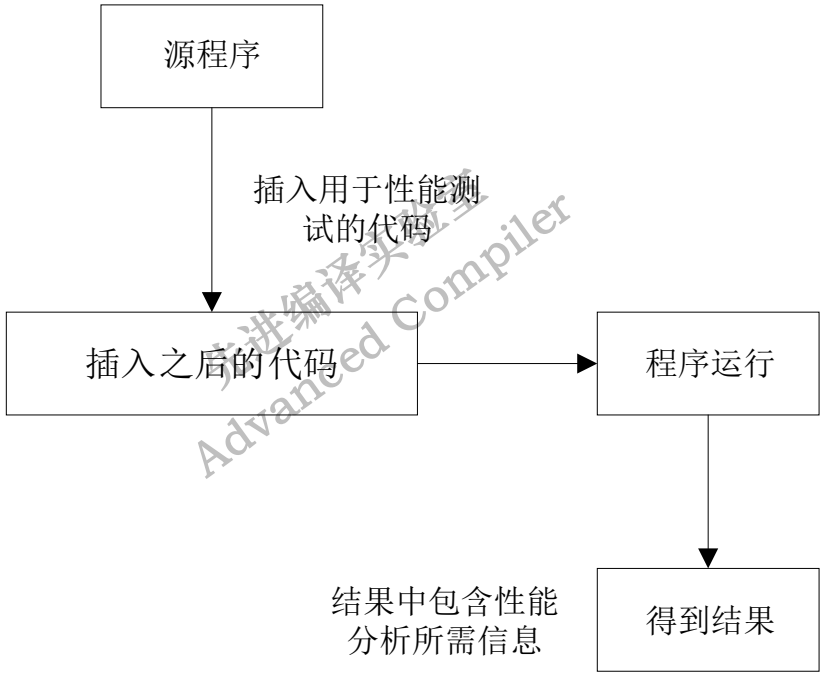
性能剖析通常是按照特定的时间间隔对系统的状态进行采样，然后对这些样本进行分析与研究。例gprof，oneAPI，perf，nvprof

性能监视记录了一段时间内的性能统计数据。通过性能监视，可以将过去的记录信息和现在的做比较，这样就能够找出程序基于时间的运行规律。例netstat、sar、iotop、mpstat

程序性能分析的工具类型

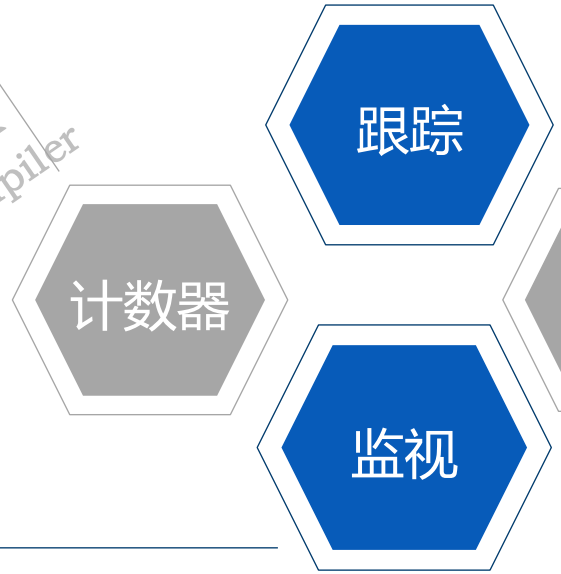
性能剖析通常是按照特定的时间间隔对系统的状态进行采样，然后对这些样本进行分析与研究。性能剖析的目标是寻找性能瓶颈，查找引发性能问题的原因及热点代码。

源程序首先被插入将用于性能测试的代码，代码插入的工作原理是让编译器修改函数调用，并插入代码以记录这些调用、调用者或者完整调用栈以及可能需要的时间信息。



程序性能分析的工具类型

在系统内核中，一般会生成一些用于对事件发生次数进行计数的统计数据，称为计数器。通常计数器为无符号的整型数，事件发生时递增。例vmstat、iostat、top、ps



跟踪工具即跟踪收集每一个事件的数据以供性能分析。一般默认不启用，跟踪捕获储存数据会有开销。例gdb，pstack，strace

性能剖析通常是按照特定的时间间隔对系统的状态进行采样，然后对这些样本进行分析与研究。例gprof，oneAPI，perf，nvprof

性能监视记录了一段时间内的性能统计数据。通过性能监视，可以将过去的记录信息和现在的做比较，这样就能够找出程序基于时间的运行规律。例netstat、sar、iotop、mpstat

AdvancedCompiler
Tel: 13839830713

编译论坛

先进编译实验室
Advanced Compiler

程序性能的分析和测量II

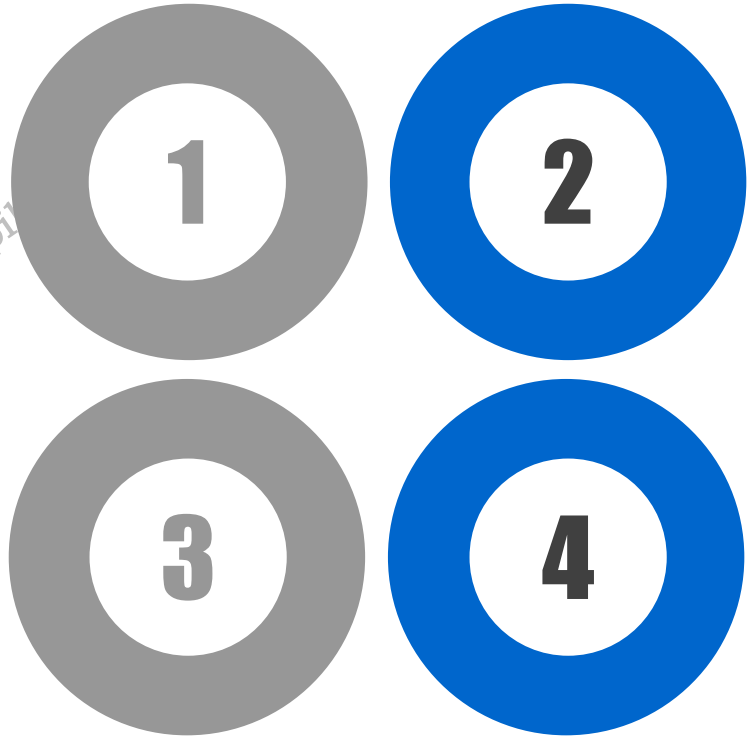
嘉宾：柴晓楠

先进编译实验室
Advanced Compiler

程序性能测量的工具 › 计数器类

虚拟内存统计工具vmstat

显示Linux系统虚拟内存状态，还可以统计关于进程 、内存、处理器使用率、I/O、对swap空间的I/O、通常的I/O等系统整体运行状态。



实时状态工具top

显示系统当前的进程以及其它的一些信息，是一个动态显示过程，即可以通过用户按键来不断刷新当前状态。

输入/输出统计工具iostat

是对系统的磁盘操作活动进行监视的一个工具。它的特点是汇报磁盘活动统计情况，同时也会汇报处理器使用情况。

当前进程信息统计工具ps

是Linux中用来列出系统中当前运行进程的指令工具，使用该命令可以确定有哪些进程正在运行以及运行的状态、进程是否结束、进程有没有僵死、哪些进程占用了过多的资源等。

Vmstat常用格式及使用方法为：vmstat - [选项] [interval] [times]

即每隔interval秒采样一次，一共采样times次，如果省略了times，则vmstat会一直采集数据，直到用户手动停止，或使用ctrl+c进行停止。具体选项可以使用man vmstat命令来查看。

```
[@localhost Desktop]$ vmstat -a 2 10
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free  inact active   si   so    bi    bo   in   cs us sy id wa st
 1  0 180812 133144 352932 358280   41 1041  5360  1510  720 1391 24 11 65  0  0
 0  0 180804 132740 353064 358380   16    0    78     0  148  243 11  1 89  0  0
 0  0 180744 132252 353420 358512  160    0   258   103  169  284 11  1 88  1  0
 1  0 180520 128040 355648 360128  972    0  1914     0  260  457 13  3 85  0  0
 0  0 180512 128048 355732 360124   16    0    42     0  162  281 11  1 89  0  0
 1  0 180512 128048 355952 360128    0    0   112     0  163  281 10  1 89  0  0
 2  0 180444 127180 356752 360132  240    0   404     0  152  259  9 1 91  0  0
 0  0 180444 127180 356752 360132    0    0     0     0  149  264 11  1 89  0  0
 1  0 180440 126684 357208 360196   32    0   252     0  287  523 15  2 83  0  0
 0  0 180436 126684 357284 360204   16    0    16     0  140  263 10  1 89  0  0
```

输出字段一共分为6大部分：procs、memory、swap、io、system和cpu。

输出字段	输出字段释义
procs	r：等待运行的进程数目b：处在非中断睡眠状态的进程数
memory	swpd：虚拟内存的使用情况，单位为：KBfree：空闲的物理内存的大小 buff：用来做buffer（缓存，主要用于块设备缓存）的内存数，单位： KBcache:用作缓存的内存大小，单位：KB
swap	si：从磁盘交换到swap虚拟内存的交换页数量，单位：KB/sso：从swap 虚拟内存交换到磁盘的交换页数量，单位：KB/s
io	bi：每秒从块设备接收到的块数，单位：块/秒，也就是读块设备bo：每 秒发送到块设备的块数，单位：块/秒，也就是写块设备
system	in：每秒的中断数，包括时钟中断cs：每秒的环境（上下文）切换次数
cpu	us：用户CPU使用时间(非内核进程占用时间)（单位为百分比）sy：系统 内核使用的CPU时间（单位为百分比）id：空闲的CPU的时间(百分比)wa: 等待I/O的CPU时间st:虚拟机占用CPU时间的百分比

Iostat常用格式及使用方法为：iostat [options] [<interval> [<count>]

Options 表示操作项参数，interval指定统计时间间隔，count表示总共输出次数，具体的参数可以使用iostat -help来查看，参数的意义可以使用man iostat来查看。

```
@-virtual-machine$ iostat -d 2 3
Linux 5.11.0-43-generic (mk-virtual-machine)        2022年01月01日      _x86_64_  (2 CPU)

Device            tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
loop0            0.00         0.00         0.00         0.00         17          0          0
loop1            0.01         0.04         0.00         0.00        347          0          0
loop10           0.01         0.04         0.00         0.00        348          0          0
loop11           0.07         2.26         0.00         0.00      17997          0          0
loop12           0.01         0.14         0.00         0.00       1093          0          0
loop13           0.00         0.00         0.00         0.00         18          0          0
loop2            0.01         0.04         0.00         0.00        358          0          0
loop3            0.01         0.13         0.00         0.00       1067          0          0
```

Iostat性能分析方法如下：

查看I/O等待时间较高，表示磁盘存在I/O瓶颈。

一般地，系统I/O响应时间应该低于5ms，如果大于10ms就比较慢了。

或者查看设备是否已经接近满负荷运行。

如果在iostat级别出现了性能下降的情况，一般可以查看一下存储方面的性能信息，存储也是影响I/O性能的一个主要方面。

top命令提供了对系统处理器的实时状态监视，它将显示系统中使用处理器最密集的任务列表。该命令可以按内存使用和执行时间对任务进行排序。

Top命令的使用格式为：top [options]

```
@virtual-machine:~/Desktop$ top
top - 15:55:15 up  2:21,  1 user,  load average: 0.01, 0.04, 0.00
Tasks: 290 total,   1 running, 288 sleeping,   1 stopped,   0 zombie
%Cpu(s):  1.0 us,  1.7 sy,  0.0 ni, 97.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3896.7 total,   2092.7 free,    970.4 used,    833.6 buff/cache
MiB Swap:    923.3 total,    923.3 free,      0.0 used.   2695.0 avail Mem

  PID USER     PR  NI    VIRT    RES    SHR S %CPU  %MEM  TIME+COMMAND
  2551 mk       20   0 3999752 257360108460 S   3.7  6.4   0:49.20 gnome-shell
  2026 mk       20   0  299552  76344  39528 S  2.3  1.9   0:32.80 Xorg
  2916 mk       20   0 1058152  63900  46064S   0.7  1.6   0:14.82 gnome-terminal-
  3636 root     20   0       0      0       0I0.3  0.0   0:00.78 kworker/1:1-events
     1 root     20   0  168776  12668   8196 S 0.0  0.3   0:02.14 systemd
     2 root     rt   0       0      0      0S 0.0   0.0   0:00.04migration/0
     3 root      0  -20      0      0      0I  0.0   0.0   0:00.00rcu_gp
```

ps命令是最简易操作又非常强大的进程查看命令，使用该命令可以确定有哪些进程正在运行以及运行的状态、进程是否结束、进程有没有僵死、哪些进程占用了过多的资源等。

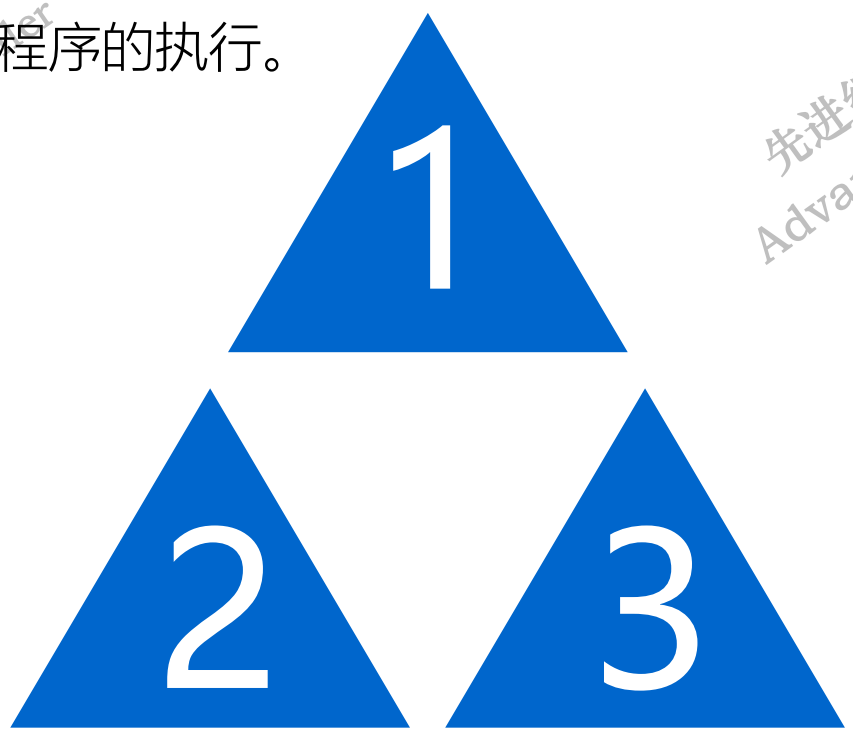
Ps常用格式及使用方法为：ps -[选项] 或者ps [选项]

Ps命令最常用的选项就是e、f、a、u，相应的选项组合为ps -aux，ps -ef，可以通过这些组合准确定位系统进程状态。

```
[root@localhost stmk]# ps -ef
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 12:51 ?        00:00:02 /usr/lib/systemd/systemd --switched-r
root         2     0  0 12:51 ?        00:00:00 [kthreadd]
root         3     2  0 12:51 ?        00:00:00 [ksoftirqd/0]
root         5     2  0 12:51 ?        00:00:00 [kworker/0:0H]
root         7     2  0 12:51 ?        00:00:00 [migration/0]
root         8     2  0 12:51 ?        00:00:00 [rcu_bh]
root         9     2  0 12:51 ?        00:00:00 [rcu_sched]
```

程序调试工具gdb

是GNU开源组织发布的一款功能强大的UNIX/Linux下的程序调试工具，主要功能就是监控程序的执行。



堆栈统计信息工具pstack

pstack工具可对指定进程号的进程输出函数调用栈，对于排查进程问题时非常有用。

跟踪系统调用工具strace

是一种相当有效的跟踪工具，它的主要特点是可以用来监视系统调用。可以跟踪要运行或者已经运行的进程。

gdb的主要功能就是监控程序的执行。当源程序文件编译为可执行文件并执行时，并且该文件中必须包含必要的调试信息，gdb才会派上用场。所以在编译时需要使用-g选项编译源文件，才可生成满足gdb要求的可执行文件。

一般来说，gdb主要完成以下四个方面的功能：

（1）启动程序，可以按照自定义的要求随心所欲的运行程序。

（2）可让被调试的程序在所设置的指定断点处停住。

（3）当程序被停住时，可以检查此时程序中所发生的事件。

（4）动态的改变程序的执行环境。

b 3表示调试的代码第三行设置为断点，r或run表示执行程序，whatis iNum表示查看iNum的数据类型，c表示继续执行程序，-n一步步执行程序，其中p iNum表示输出变量iNum的值。

```
[@localhost ~]$ gcc -o test2 -g test.c
[@localhost ~]$ gdb test2
(gdb) b 3
Note: breakpoints 1 and 2 also set at pc 0x4005a8.
Breakpoint 3 at 0x4005a8: file test.c, line 3.
(gdb) r
Starting program: /home/stmk/test2
Please input a number :100

Breakpoint 1, ShowRevertNum (iNum=100) at test.c:4
4               while (iNum > 10)
(gdb) whatis iNum
type = int
(gdb) p iNum
$1 = 100
```

Pstack在排查进程问题时非常有用，比如发现一个服务一直处于工作状态，如假死状态，好似死循环，使用这个命令就能轻松定位问题所在。

Pstack命令使用格式为：pstack pid

```
[root@localhost Desktop]# ps -aux |grep bash
root      973  0.0  0.0 115216   896 ?        S    18:02   0:00 /bin/bash /usr/sbin/ksmtuned
aubin    13082  0.0  0.2 116260  2916 pts/0    Ss   18:03   0:00 /bin/bash
root     13185  0.1  0.2 116264  2928 pts/0    S    18:04   0:00 bash
root     13213  0.0  0.0 112640   964 pts/0    R+   18:04   0:00 grep --color=auto bash
[root@localhost Desktop]# pstack 13082
#0  0x00007fb6a506d4bc in waitpid () from /lib64/libc.so.6
#1  0x00000000004406d4 in waitchld.isra.10 ()
#2  0x000000000044198c in wait_for ()
#3  0x00000000004337ee in execute_command_internal ()
#4  0x0000000000433a1e in execute_command ()
#5  0x000000000041e205 in reader_loop ()
#6  0x000000000041c88e in main ()
```

strace是一种相当有效的跟踪工具，它的主要特点是可以用来监视系统调用。

Strace工具有两种运行模式：

（1）通过它启动要跟踪的进程。用法很简单，在原本的命令前加上strace即可。例如，要跟踪“ls -al”这个命令的执行，只需要输入指令strace ls -al即可。

（2）另外一种运行模式是跟踪已经在运行的进程，在不中断进程执行的情况下，了解程序的运行过程。此时，只需要给strace传递一个-p pid选项即可。

需要跟踪程序add.out的运行过程，则输入命令strace ./add.out即可

```
root@-virtual-machine# strace ./add.out
```

了解正在运行的进程号为2403的进程情况，只需要输入命令strace –p 2403

```
[root@localhost Desktop]# strace -p 2403
```

AdvancedCompiler
Tel: 13839830713

编译论坛

先进编译实验室
Advanced Compiler

程序性能的分析和测量III

嘉宾：柴晓楠

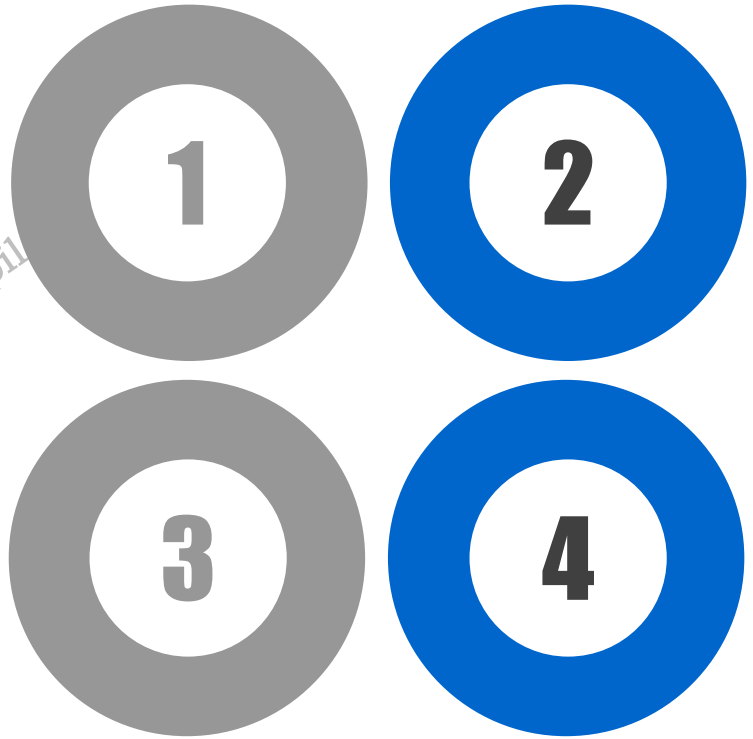
先进编译实验室
Advanced Compiler

函数剖析工具gprof

是GNU编译器工具包提供的性能分析工具。使用代码抽样和插入技术，收集函数剖析信息和调用图文件，从而给出函数调用关系、调用次数、执行时间等信息。此外还会列出各个函数的耗时比例。

可视化软件性能分析工具oneAPI

基于 Intel oneAPI 这一编程模型开发的产品之一，包含编译器、函数库、预先优化框架以及先进的分析调试工具等组件。包含了常用的性能分析工具。



性能分析工具perf

是基于用户空间数据上的命令行性能分析工具，支持软硬件计数器，并可以像strace工具一样跟踪内核调用。可以分析程序的性能或者内核的性能。

CUDA程序性能分析工具nvprof

是NVIDIA开发的用于分析CUDA程序性能的工具，可以统计各个内核和CUDA函数的运行效率，还可以给出硬件计数器的值。

gprof使用代码抽样和插入技术，收集函数剖析信息和调用图文件，从而给出函数调用关系、调用次数、执行时间等信息。gprof是通过在编译时插入代码来分析程序，因此在一些情况下，其给出的结果会有不准确的地方。

Gprof通常和GCC、LLVM等编译器配合使用，在使用GCC或LLVM编译链接程序时，添加选项-pg，表示产生的程序可以使用gprof分析。程序运行结束后，会在程序退出的路径下生成一个gmon.out文件。

```
[root@localhost Desktop]# ls
test.c
[root@localhost Desktop]#gcc test.c -o test_gprof -pg
[root@localhost Desktop]# ./test_gprof
[root@localhost Desktop]# ls
gmon.out  test.c  test_gprof
```

Gprof输出结果主要包含两部分，剖析文件Flat profile和调用图Call graph。Flat profile中每一行代表一个函数，各列参数含义如下：

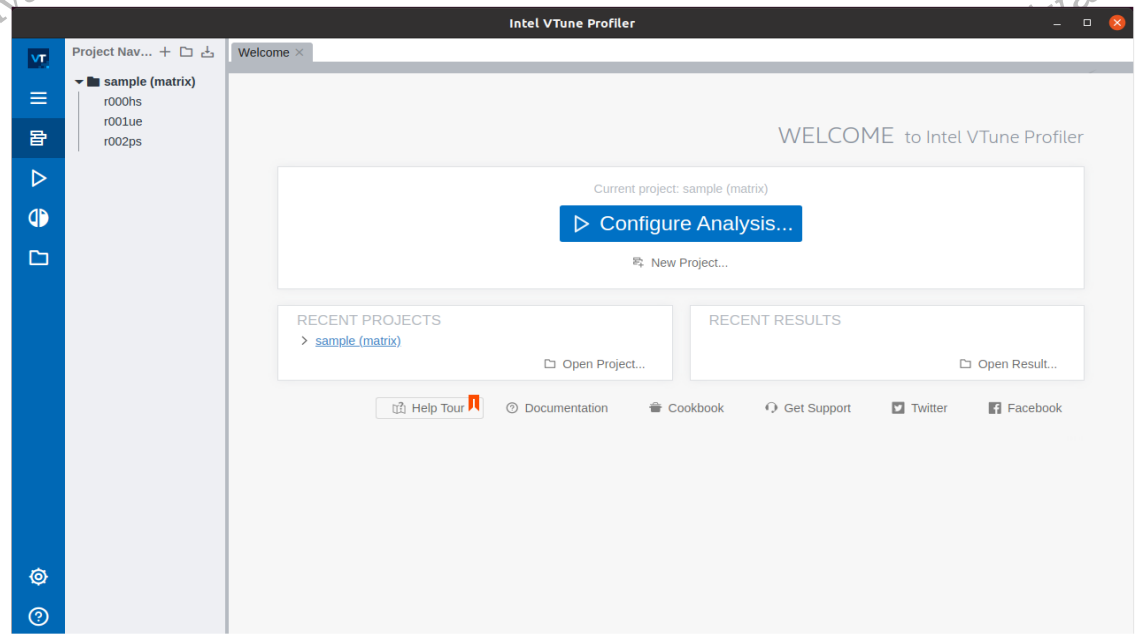
```
Flat profile:
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  s/call   s/call  name
101.69      2.45     2.45      110     0.02     0.02  longa
 0.00      2.45     0.00        1     0.00     0.22  funcA
 0.00      2.45     0.00        1     0.00     2.23  funcB
```


%time	函数独立运行时间，不包含被该函数调用的其它函数的运行时间，占总运行时间的百分比
Cumulative seconds	所有函数的累积运行时间，包括自身
Self seconds	函数的独立运行时间，单位为秒
Calls	函数被调用的次数
Selfs/call	函数每次调用的平均独立运行时间，单位为秒
Totals/call	函数每次调用的平均整体运行时间，包括被它调用的其它函数的运行时间，单位为秒

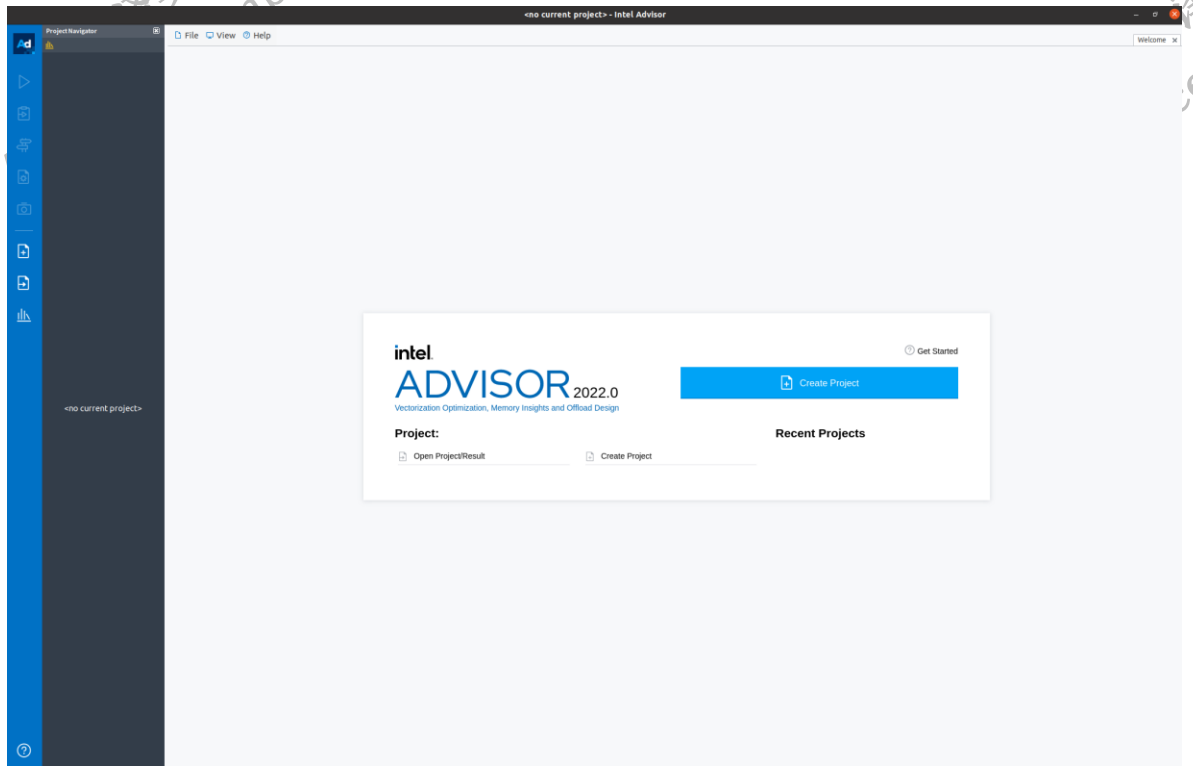
调用图的每一部分代表一个函数，最左边显示了该函数的运行索引。位于该索引之上的是调用当前函数的函数，之下的是被当前函数调用的函数。

```
Call graph (explanation follows)
granularity: each sample hit covers 2 byte(s) for 0.41% of 2.45 seconds
index % time    self  children    called     name
0.22    0.00      10/110        funcA [4]
2.23    0.00     100/110        funcB [3]
[1]   100.0    2.45    0.00     110        longa [1]
-----------------------------------------------
<spontaneous>
[2]   100.0    0.00    2.45                main [2]
0.00    2.23      1/1           funcB [3]
0.00    0.22      1/1           funcA [4]
```

Intel推出的oneAPI 2022工具是基于 Intel oneAPI 这一编程模型开发的产品之一，包含编译器、函数库、预先优化框架以及先进的分析调试工具等组件。Intel oneAPI toolkits中的VTune是一个用于分析和优化程序性能的工具，可以通过从系统中收集性能数据、从系统到源代码不同的层次以及形式上组织和展示数据、发现潜在的性能问题并提出改进措施这三个方面，帮助优化人员找到性能不理想的原因并针对性的对程序进行优化。



Intel® Advisor由一组工具或视角组成，Intel Advisor能够从向量化、CPU/GPU /内存性能上限、卸载建模以及线程角度分析代码，可以帮助优化人员分析程序中影响大、优化不足的循环优化瓶颈，还可以根据硬件施加的性能上限可视化实际性能，给出下一步的优化步骤建议等，从而帮助Fortran、C、C++等多种语言构建的应用程序达到较好的性能。



perf的功能实现支持软硬件计数器，并可以像strace工具一样跟踪内核调用。借助perf工具，应用程序可以使用性能监控单元（performance monitoring unit,PMU）、tracepoint和内核中的特殊计数器来进行性能统计。

```
[root@localhost Desktop]# vim test.c
[root@localhost Desktop]# gcc test.c -o test_perf.out -g
[root@localhost Desktop]#perf stat ./test_perf.out
Performance counter stats for './test_perf.out':

        756.74 msec task-clock                #    0.993 CPUs utilized
            33      context-switches          #    0.044 K/sec
             1      cpu-migrations            #    0.001 K/sec
            45      page-faults               #    0.059 K/sec
 1,346,078,844      cycles                    #    1.779 GHz                      (49.24%)
             0      stalled-cycles-frontend                                     (49.73%)
             0      stalled-cycles-backend    #    0.00% backend cycles idle     (50.25%)
             0      instructions              #    0.00  insn per cycle          (50.76%)
             0      branches                  #    0.000 K/sec                   (50.27%)
             0      branch-misses             #    0.00% of all branches         (49.75%)
   0.761973407 seconds time elapsed
   0.752836000 seconds user
   0.004004000 seconds sys
```

程序性能测量的工具 › 剖析类 › nvprof


```
[wh@localhost Nvproftest]$ nvprof ./sumMatrixGPUManaged
==6418== NVPROF is profiling process 6418, command: ./sumMatrixGPUManaged
./sumMatrixGPUManaged Starting using Device 0: NVIDIA GeForce GT 730
Matrix size: nx 4096 ny 4096
initialization:          0.572687 sec
sumMatrix on host:       0.039688 sec
sumMatrix on gpu :       0.009752 sec <<<(128,128), (32,32)>>>
==6418== Profiling application: ./sumMatrixGPUManaged
==6418== Profiling result:
            Type  Time(%)      Time     Calls       Avg       Min       Max  Name
 GPU activities:  100.00%  9.5120ms         2  4.7560ms  2.5826ms  6.9295ms  sumMatrixGPU(float*, float*, float*, int, int)
      API calls:   45.54%  133.87ms         4  33.467ms  10.342ms  102.73ms  cudaMallocManaged
                   28.07%  82.494ms         2  41.247ms  7.0270us  82.487ms  cudaLaunchKernel
                   13.26%  38.990ms         4  9.7475ms  9.2694ms  10.151ms  cudaFree
                    9.70%  28.504ms         1  28.504ms  28.504ms  28.504ms  cudaDeviceReset
                    3.31%  9.7429ms         1  9.7429ms  9.7429ms  9.7429ms  cudaDeviceSynchronize
                    0.04%  121.64us       101  1.2040us      95ns  52.647us  cuDeviceGetAttribute
                    0.04%  105.01us         1  105.01us  105.01us  105.01us  cudaGetDeviceProperties
                    0.02%  58.184us         1  58.184us  58.184us  58.184us  cuDeviceTotalMem
                    0.01%  23.236us         1  23.236us  23.236us  23.236us  cudaSetDevice
                    0.01%  21.094us         1  21.094us  21.094us  21.094us  cuDeviceGetName
                    0.00%  4.7980us         1  4.7980us  4.7980us  4.7980us  cuDeviceGetPCIBusId
                    0.00%     809ns         3     269ns     126ns     512ns  cuDeviceGetCount
                    0.00%     541ns         2     270ns     127ns     414ns  cuDeviceGet
                    0.00%     416ns         1     416ns     416ns     416ns  cudaGetLastError
                    0.00%     174ns         1     174ns     174ns     174ns  cuDeviceGetUuid
```

程序性能测量的工具 › 监控类

01

系统活动情况报告工具sar

是Linux系统上最为全面的系统性能测量工具之一，可以从多方面对系统的活动进行报告，包括文件的读写情况、系统调用的使用情况、磁盘I/O、CPU效率、内存利用率、进程活动及网络流量等。

02

监控网络工具netstat

是一个监控TCP/IP网络的非常有用的工具，它可以显示路由表、实际的网络连接以及每一个网络接口设备的状态信息，也可以显示与IP、TCP、UDP和ICMP协议相关的统计数据，一般用于查询本机各端口的网络连接情况。

03

监控硬盘工具iotop

用来监控硬盘 I/O的使用情况，其界面和top类似，包括 PID、用户、IO、进程等相关信息。

04

实时系统监控工具mpstat

实时系统监控工具，用于报告CPU相关的一些统计信息，这些信息存放在/proc/stat文件中。在多CPU系统中，其不但能查看所有CPU的平均状态信息，而且能够查看特定CPU的信息。

Linux系统中的sar命令可以从多方面对系统的活动进行报告，包括文件的读写情况、系统调用的使用情况、磁盘I/O、CPU效率、内存利用率、进程活动及网络流量等。

Sar命令常用格式为：sar -[选项] [时间间隔] [次数]。

sar -u 1 3 为每间隔1秒钟统计一次，总共统计三次。

```
[root@localhost Desktop]# sar -u 1 3
Linux 5.11.0-41-generic (virtual-machine)   2022年01月02日      _x86_64_   (2 CPU)

20时36分48秒    CPU    %user    %nice   %system   %iowait    %steal    %idle
20时36分49秒    all     0.00     0.00     1.02     0.00     0.00    98.98
20时36分50秒    all     0.00     0.00     1.01     0.00     0.00    98.99
20时36分51秒    all     0.50     0.00     1.01     0.00     0.00    98.49
平均时间:    all     0.17     0.00     1.01     0.00     0.00    98.82
```

要判断系统瓶颈问题，有时需几个sar命令选项结合起来。

（1）如果怀疑CPU存在瓶颈，可用sar -u和sar -q等来查看；

（2）如果怀疑内存存在瓶颈，可用sar -B、sar -r和sar -W等来查看；

（3）如果怀疑I/O存在瓶颈，可用sar -b、sar -u和sar -d等来查看。

指标性能分析技巧：

（1）若%iowait的值过高，表示硬盘存在I/O瓶颈；

（2）若%idle 的值高但系统响应慢时，有可能是CPU等待分配内存，此时应加大内存容量；

（3）若%idle 的值持续低于1，则系统的 CPU 处理能力相对较低，表明系统中最需要解决的资源是CPU。

netstat是一个监控TCP/IP网络的非常有用的工具，它可以显示路由表、实际的网络连接以及每一个网络接口设备的状态信息，netstat用于显示与IP、TCP、UDP和ICMP协议相关的统计数据，一般用于查询本机各端口的网络连接情况。

Netstat命令使用格式如下：

NETSTAT [-a] [-b] [-e] [-f] [-n] [-o] [-p proto] [-r] [-s] [-t] [interval]

以下为netstat -a 列出了所有连接示例

```
@virtual-machine:~/Desktop$ netstat -a | head
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN
tcp        0      0 localhost:ipp           0.0.0.0:*               LISTEN
tcp        0      0 mk-virtual-machin:42460 82.221.107.34.bc.g:http ESTABLISHED
tcp        0      0 mk-virtual-machin:54406 17.111.232.35.bc.g:http TIME_WAIT
tcp        0      0 mk-virtual-machin:42466 82.221.107.34.bc.g:http TIME_WAIT
tcp        0      0 mk-virtual-machin:42468 82.221.107.34.bc.g:http TIME_WAIT
tcp        0      0 mk-virtual-machin:35158 ec2-44-228-106-27:https ESTABLISHED
```

netstat输出字段释义如下表所示：

输出字段	输出字段释义
Proto	Protocol 的简称，它可以是 TCP 或 UDP
Recv-Q	接收队列，数字一般都应该是0，如果不是，则表示软件包正在队列中堆积
Send-Q	发送队列，数字一般都应该是0
Local Address	本机的 IP 和端口号
Foreign Address	指所要连接的主机名称和服务
State	指现在连接的状态

Iotop用来监控硬盘 I/O的使用情况，其界面和top类似，包括 PID、用户、IO、进程等相关信息。Linux下系统自带的I/O统计工具如iostat等大多数是只能统计到单碟设备的读写情况，如果想知道每个进程是如何使用I/O的就比较麻烦，使用iotop命令可以很方便的查看，命令iotop -h可以查看使用帮助，最简单的方法就是直接使用iotop命令。

```
Total DISK READ :      9.73 M/s | Total DISK WRITE :       0.00 B/s
Actual DISK READ:      9.73 M/s | Actual DISK WRITE:      435.27 K/s
  TID  PRIO USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
 2598 be/4 stmk       9.73 M/s    0.00 B/s  0.00 %  1.48 % firefox
    1 be/4 root       0.00 B/s    0.00 B/s  0.00 % 0.00 % systemd --sw~eserialize 21
    2 be/4 root       0.00 B/s    0.00 B/s  0.00 %  0.00 % [kthreadd]
    3 be/4 root       0.00 B/s    0.00 B/s  0.00 %  0.00 % [ksoftirqd/0]
    5 be/0 root       0.00 B/s    0.00 B/s  0.00 % 0.00 % [kworker/0:0H]
    7 rt/4 root       0.00 B/s    0.00 B/s  0.00 %  0.00 % [migration/0]
    8 be/4 root       0.00 B/s    0.00 B/s  0.00 %  0.00 % [rcu_bh]
    9 be/4 root       0.00 B/s    0.00 B/s  0.00 %  0.00 % [rcu_sched]
   10 rt/4 root       0.00 B/s    0.00 B/s  0.00 %  0.00 % [watchdog/0]
   12 be/4 root       0.00 B/s    0.00 B/s  0.00 %  0.00 % [kdevtmpfs]
```

Mpstat是实时系统监控工具，用于报告CPU相关的一些统计信息，这些信息存放在/proc/stat文件中。在多CPU系统中，其不但能查看所有CPU的平均状态信息，而且能够查看特定CPU的信息。例如命令 mpstat -P ALL 2 3，表示每2秒对所有处理器统计数据报告，采集三次信息并输出。


```
[root@localhost ]# mpstat -P ALL 2 3
Linux 3.10.0-693.el7.x86_64 (localhost.localdomain)    01/07/2022          _x86_64_  (1 CPU)

09:25:43 PM  CPU  %usr  %nice  %sys  %iowait  %irq  %soft  %steal  %guest  %gnice  %idle
09:25:45 PM  all   3.52   0.00  0.50    0.00  0.00   0.00    0.00    0.00    0.00  95.98
09:25:45 PM    0   3.52   0.00  0.50    0.00  0.00   0.00    0.00    0.00    0.00  95.98
```

AdvancedCompiler
Tel: 13839830713