Indian Institute of Information Technology, Vadodara

Parallel Programming(CS403)

Prof. Reshmi Mitra

Name - Dilip Puri
ID - 201351014
Collaborator - Hemant Kumar(201352026)

Lab 01

Submission Date - August 15, 2016 **Deadline** - Due Aug 15, 4:00 PM

Peak Performance

Problem 1 Consider a memory system with a level 1 cache of 32 KB and DRAM of 512 MB with the processor operating at 1 GHz. The latency to L1 cache is 1 cycle and the latency to DRAM is 100 cycles. In each memory cycle, the processor fetches four words (cache line size is 4 words). What is the peak achievable performance of a dot product of two vectors?

```
/* dot product loop */
for (i = 0; i < dim; i++)
   dot_prod += a[i] * b[i];
------------------------------------------------------
Answer:

   Main memory = 512 MB
   Cache = 32 KB
   Processor operating at 1 GHz
   L1 cache latency = 1 cycle
   L2 or main memory latency = 100 cycle
   In 1 memory cycle processor fetch no. of words = 4
   cache line size = 4 words
   Peak Performance = no. of maximum floating pt(arithmatic) operations
   so,

         1 cycle takes 1/(1x10^9) seconds.
         One access to memory takes 100/(1x10^9) seconds. = 100ns.
         Performance = 4 FLOPS/(4x100/(1x10^9)) = 10 MFLOPS.

                                                      -
```

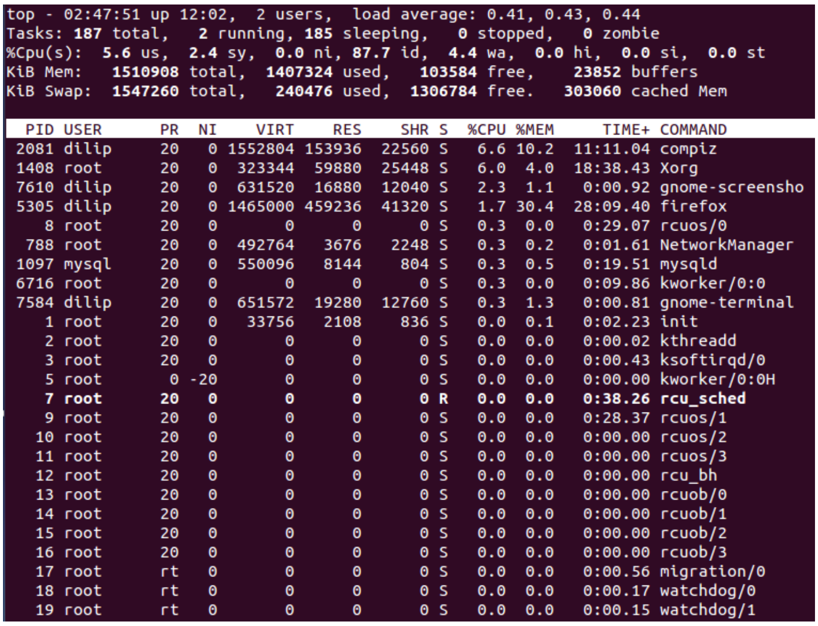
Problem 2 Now consider the problem of multiplying a dense matrix with a vector using a two-loop dot-product formulation. The matrix is of dimension 4K x 4K. (Each row of the matrix takes 16 KB of storage.) What is the peak achievable performance of this technique using a two-loop dot-product based matrix-vector product?

```
/* matrix-vector product loop */
for (i = 0; i < dim; i++)
   for (j = 0; j < dim; j++)
      c[i] += a[i][j] * b[i];
-----------------------------------------------------
Answer:

   Main memory = 512 MB
   Cache = 32 KB
```

Linux commands

```
$top
```



```
top - 02:47:51 up 12:02,  2 users,  load average: 0.41, 0.43, 0.44
Tasks: 187 total,   2 running, 185 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5.6 us,  2.4 sy,  0.0 ni, 87.7 id,  4.4 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem:   1510908 total,  1407324 used,   103584 free,    23852 buffers
KiB Swap:  1547260 total,   240476 used,  1306784 free.   303060 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 2081 dilip     20   0 1552804 153936  22560 S   6.6 10.2  11:11.04 compiz
 1408 root      20   0  323344  59880  25448 S   6.0  4.0  18:38.43 Xorg
 7610 dilip     20   0  631520  16880  12040 S   2.3  1.1   0:00.92 gnome-screensho
 5305 dilip     20   0 1465000 459236  41320 S   1.7 30.4  28:09.40 firefox
    8 root      20   0       0      0      0 S   0.3  0.0   0:29.07 rcuos/0
  788 root      20   0  492764   3676   2248 S   0.3  0.2   0:01.61 NetworkManager
 1097 mysql     20   0  550096   8144    804 S   0.3  0.5   0:19.51 mysqld
 6716 root      20   0       0      0      0 S   0.3  0.0   0:09.86 kworker/0:0
 7584 dilip     20   0  651572  19280  12760 S   0.3  1.3   0:00.81 gnome-terminal
    1 root      20   0   33756   2108    836 S   0.0  0.1   0:02.23 init
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.43 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      20   0       0      0      0 R   0.0  0.0   0:38.26 rcu_sched
    9 root      20   0       0      0      0 S   0.0  0.0   0:28.37 rcuos/1
   10 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuos/2
   11 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuos/3
   12 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/0
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/1
   15 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/2
   16 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcuob/3
   17 root      rt   0       0      0      0 S   0.0  0.0   0:00.56 migration/0
   18 root      rt   0       0      0      0 S   0.0  0.0   0:00.17 watchdog/0
   19 root      rt   0       0      0      0 S   0.0  0.0   0:00.15 watchdog/1
```

Figure 1: top: provides dynamic real-time view of individual jobs running on the system

```
$gnome-system-monitor
```

```
$lscpu
We can also get the same information from $cat /proc/cpuinfo
==================================
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                2
On-line CPU(s) list:   0,1
Thread(s) per core:    1
Core(s) per socket:    2
Socket(s):             1
NUMA node(s):          1
Vendor ID:             AuthenticAMD
CPU family:            18
Model:                 1
```

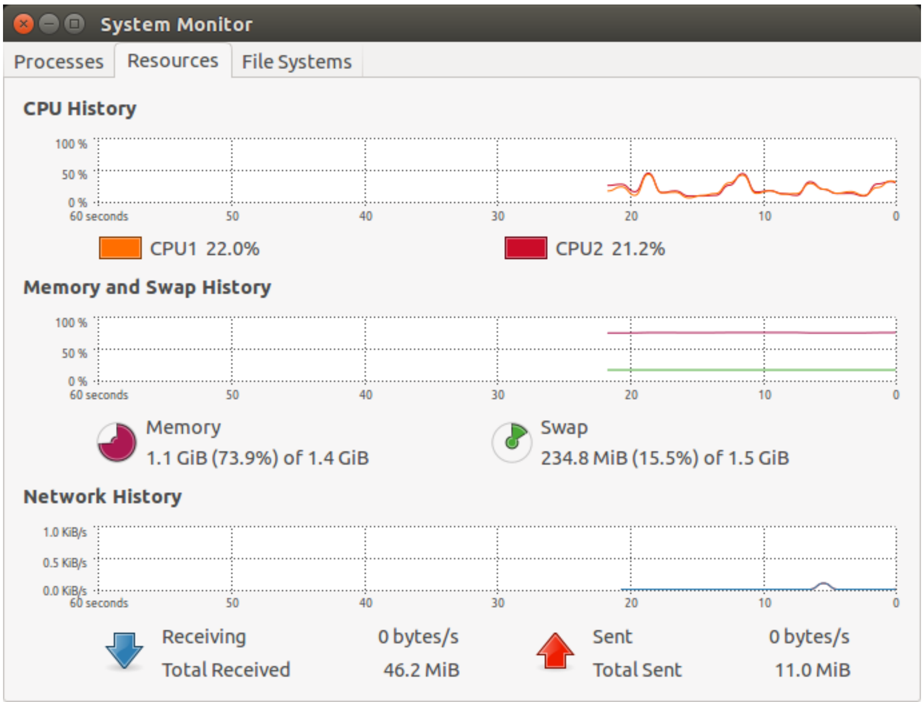


Figure 2: gnome-system-monitor: shows which programs are running and how much processor time, memory, and disk space are being used. This gives an overall system view whereas the “top” instruction represents a detailed perspective

```
Stepping:              0
CPU MHz:               800.000
BogoMIPS:              4392.08
Virtualization:        AMD-V
L1d cache:             64K
L1i cache:             64K
L2 cache:              1024K
NUMA node0 CPU(s):     0,1
dilip@dilip-notebook-pc:~$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                2
On-line CPU(s) list:   0,1
Thread(s) per core:    1
Core(s) per socket:    2
Socket(s):             1
NUMA node(s):          1
Vendor ID:             AuthenticAMD
CPU family:            18
Model:                 1
Stepping:              0
CPU MHz:               800.000
BogoMIPS:              4392.08
Virtualization:        AMD-V
L1d cache:             64K
L1i cache:             64K
L2 cache:              1024K
NUMA node0 CPU(s):     0,1
```

```
$time ./a.out
```

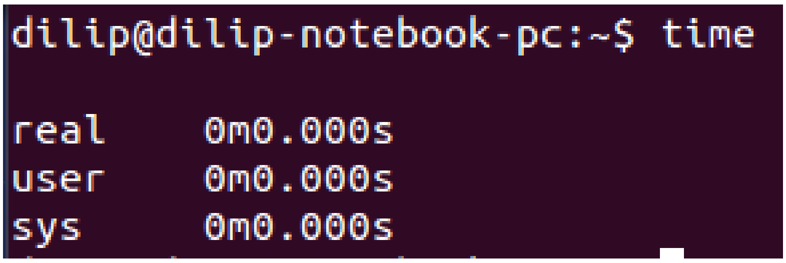


Figure 3: time: Get total program execution time in the shell

Lab Problems

1. Familiarize yourself with the Linux commands and POSIX thread code given in this handout.
2. Solutions for Problems 1-2 on peak performance.
3. Using the basic Linux commands find the cache size, bandwidth number of processors on your system.
4. Write a C-code using POSIX threads to create an unbalanced load using sleep command and hello.c. The sample sleep times are given below:
   (a) thread-1: 1000 sec, thread-2: 5000 sec, thread-3: 20 sec, thread-4: 1200 sec.
   (b) Measure the total time taken for the complete execution of code with and without the additional sleep command.
   Hint: You will require to include unistd.h for successful compilation.
5. Write a C-code using POSIX threads about matrix multiplication.
   (a) Take overall execution time measurement using time command for different application size and thread count for the serial and parallel code.
   (b) Observe gnome-system-monitor output as your fire up different thread counts.
   (c) Use the overall execution time measurements to plot and comment upon the speed-up.