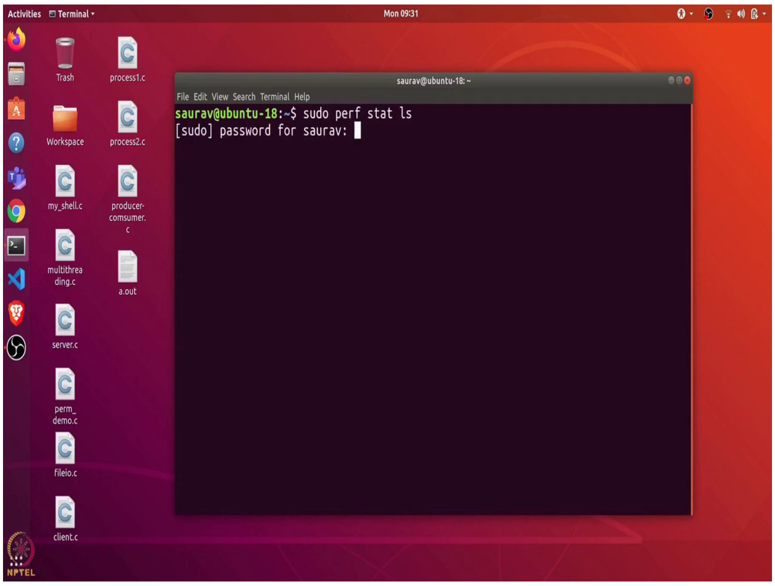
Design and Engineering of Computer Systems
Professor Mythili Vutukuru
Computer Science and Engineering
Indian Institute of Technology, Bombay
Lecture 50 (Week 7, Tutorial 1)
Basics of Perf

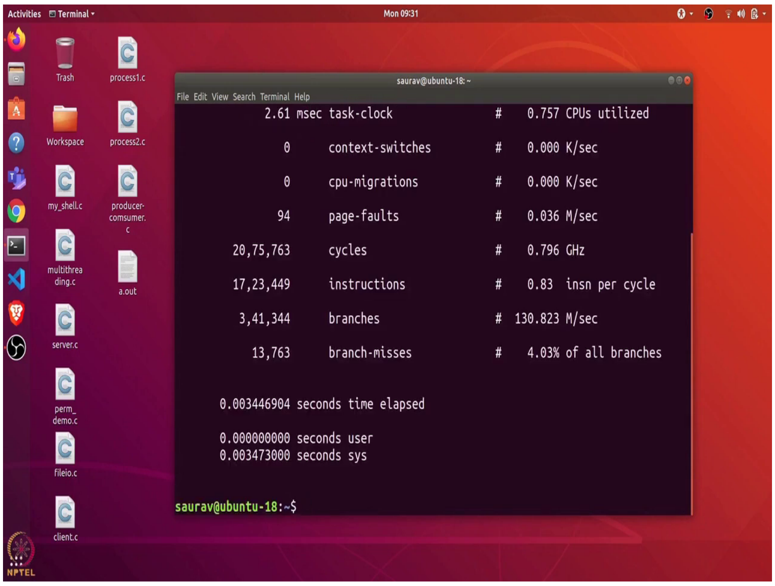
(Refer Slide Time: 0:16)



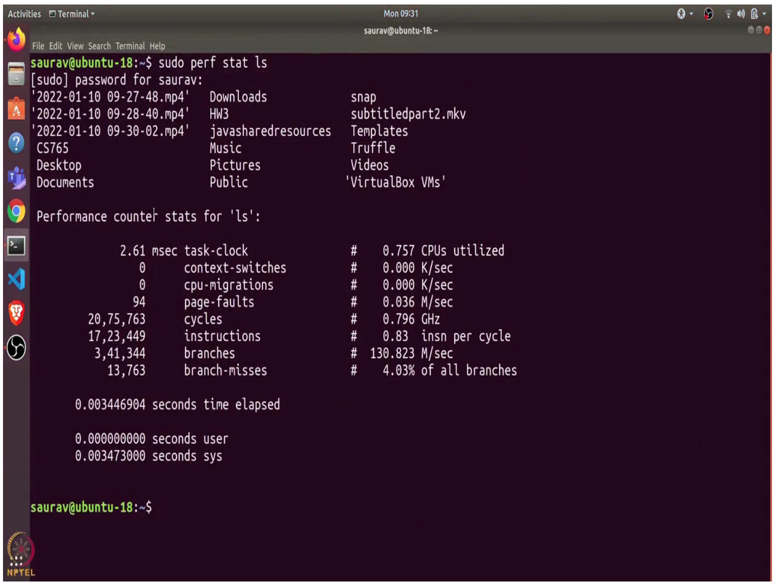
```
saurav@ubuntu-18:~$ sudo perf stat ls
[sudo] password for saurav: 
```

Hi everyone. In this video we will learn about Perf. Perf is a tool which offers a rich set of commands to collect and analyze performance data. If we want to know which parts of the programs are taking a lot of CPU time, then we can use perf. And let us open a terminal. And let us go through some of the very basic commands of perf. So, I will start with the stat command. Perf stat is used to collect statistics about any program. So, we can give it any command, let us say ls, and it will show us the statistics when ls is run. So, we will use sudo.

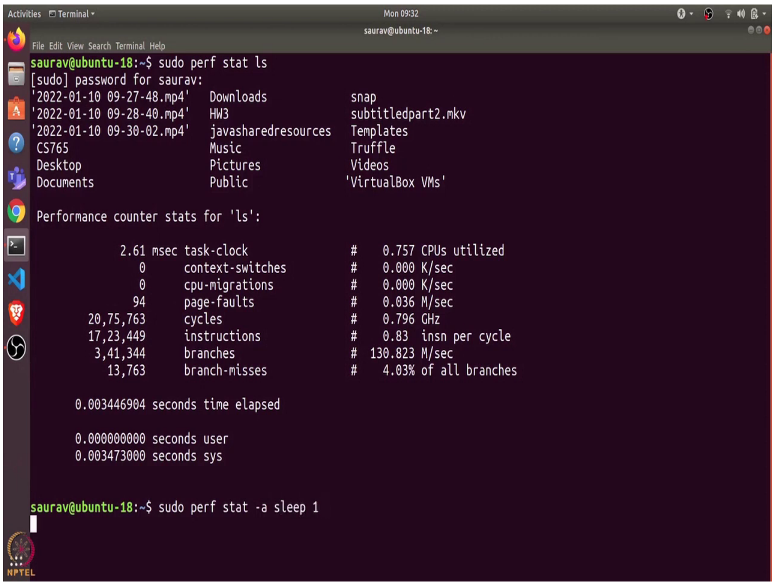
(Refer Slide Time: 0:55)



```
              2.61 msec task-clock                #    0.757 CPUs utilized
                 0      context-switches          #    0.000 K/sec
                 0      cpu-migrations            #    0.000 K/sec
                94      page-faults               #    0.036 M/sec
         20,75,763      cycles                    #    0.796 GHz
         17,23,449      instructions              #    0.83  insn per cycle
          3,41,344      branches                  #  130.823 M/sec
            13,763      branch-misses             #    4.03% of all branches

       0.003446904 seconds time elapsed

       0.000000000 seconds user
       0.003473000 seconds sys

saurav@ubuntu-18:~$
```



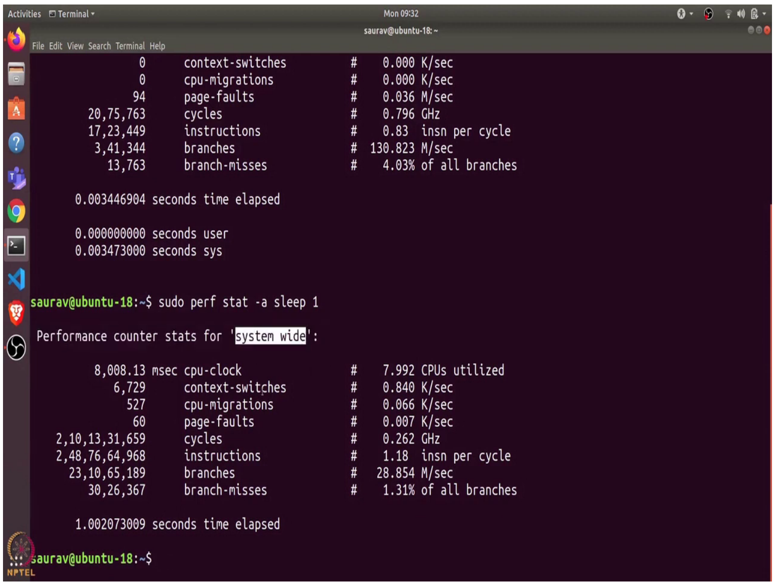
```
saurav@ubuntu-18:~$ sudo perf stat ls
[sudo] password for saurav:
'2022-01-10 09-27-48.mp4'   Downloads             snap
'2022-01-10 09-28-40.mp4'   HW3                   subtitledpart2.mkv
'2022-01-10 09-30-02.mp4'   javasharedresources   Templates
 CS765                      Music                 Truffle
 Desktop                    Pictures              Videos
 Documents                  Public               'VirtualBox VMs'

 Performance counter stats for 'ls':

              2.61 msec task-clock                #    0.757 CPUs utilized
                 0      context-switches          #    0.000 K/sec
                 0      cpu-migrations            #    0.000 K/sec
                94      page-faults               #    0.036 M/sec
         20,75,763      cycles                    #    0.796 GHz
         17,23,449      instructions              #    0.83  insn per cycle
          3,41,344      branches                  #  130.823 M/sec
            13,763      branch-misses             #    4.03% of all branches

       0.003446904 seconds time elapsed

       0.000000000 seconds user
       0.003473000 seconds sys


saurav@ubuntu-18:~$
```



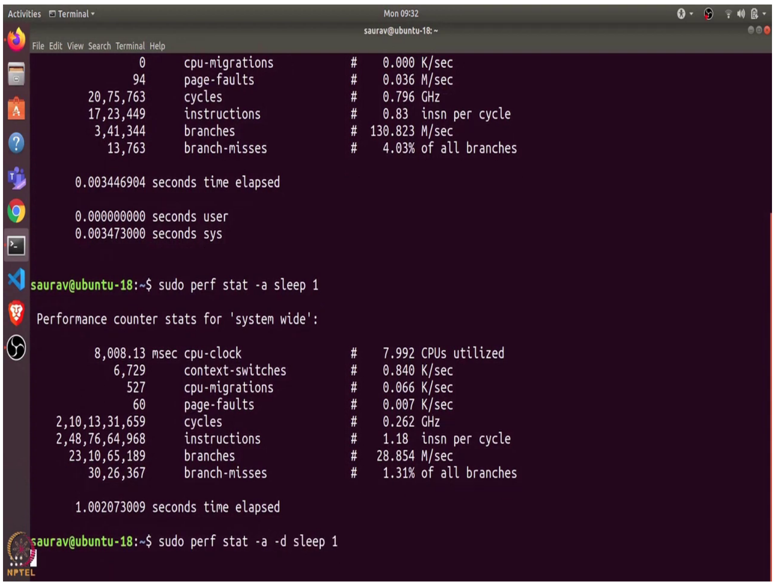
```
saurav@ubuntu-18:~$ sudo perf stat ls
[sudo] password for saurav:
'2022-01-10 09-27-48.mp4'   Downloads           snap
'2022-01-10 09-28-40.mp4'   HW3                 subtitledpart2.mkv
'2022-01-10 09-30-02.mp4'   javasharedresources  Templates
 CS765                      Music               Truffle
 Desktop                    Pictures            Videos
 Documents                  Public             'VirtualBox VMs'

 Performance counter stats for 'ls':

              2.61 msec task-clock                #    0.757 CPUs utilized
                 0      context-switches          #    0.000 K/sec
                 0      cpu-migrations            #    0.000 K/sec
                94      page-faults               #    0.036 M/sec
         20,75,763      cycles                    #    0.796 GHz
         17,23,449      instructions              #    0.83  insn per cycle
          3,41,344      branches                  #  130.823 M/sec
            13,763      branch-misses             #    4.03% of all branches

       0.003446904 seconds time elapsed

       0.000000000 seconds user
       0.003473000 seconds sys


saurav@ubuntu-18:~$ sudo perf stat -a sleep 1
```

First, it executed ls and then it shows the counter statistics for ls, gives us how many context switches are there, how many CPU migrations page fault cycles, instructions et cetera. This is very basic use of perf to collect statistics about any program. Let us say we want to collect stats for the complete system. So, we want to collect statistics about every process that is running on any processor, we can use the -a flag to collect the statistics. And let us say we want to collect the statistics for one second, perf stat -a and then we can give it command sleep 1.

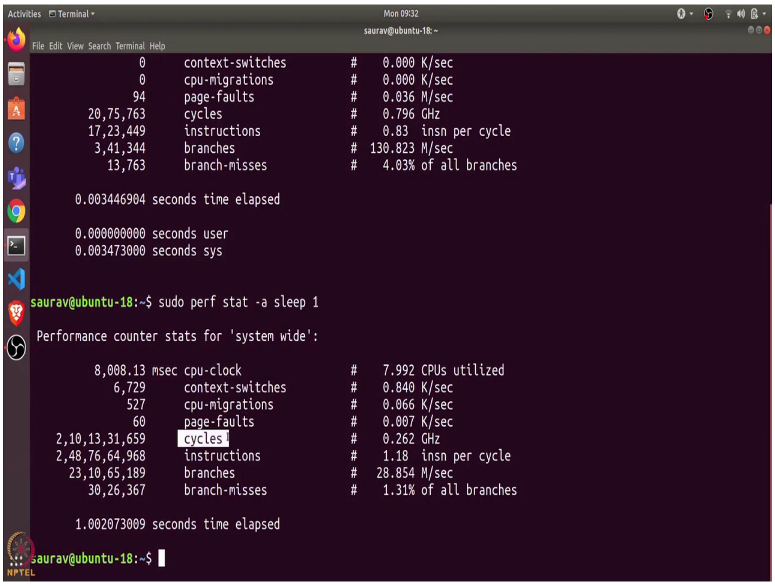
(Refer Slide Time: 1:36)



```
                 0      context-switches          #    0.000 K/sec
                 0      cpu-migrations            #    0.000 K/sec
                94      page-faults               #    0.036 M/sec
         20,75,763      cycles                    #    0.796 GHz
         17,23,449      instructions              #    0.83  insn per cycle
          3,41,344      branches                  #  130.823 M/sec
            13,763      branch-misses             #    4.03% of all branches

       0.003446904 seconds time elapsed

       0.000000000 seconds user
       0.003473000 seconds sys


saurav@ubuntu-18:~$ sudo perf stat -a sleep 1

 Performance counter stats for 'system wide':

          8,008.13 msec cpu-clock                 #    7.992 CPUs utilized
             6,729      context-switches          #    0.840 K/sec
               527      cpu-migrations            #    0.066 K/sec
                60      page-faults               #    0.007 K/sec
    2,10,13,31,659      cycles                    #    0.262 GHz
    2,48,76,64,968      instructions              #    1.18  insn per cycle
      23,10,65,189      branches                  #   28.854 M/sec
         30,26,367      branch-misses             #    1.31% of all branches

       1.002073009 seconds time elapsed

saurav@ubuntu-18:~$
```



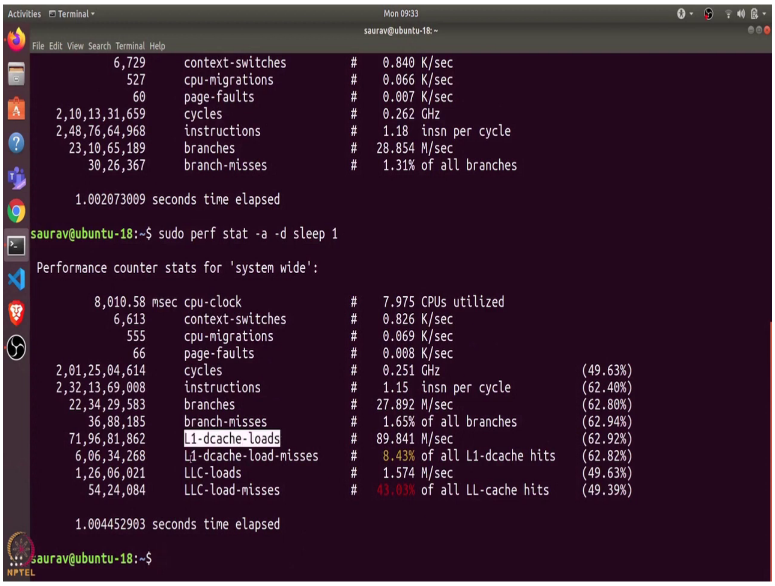
```
                 0      cpu-migrations            #    0.000 K/sec
                94      page-faults               #    0.036 M/sec
         20,75,763      cycles                    #    0.796 GHz
         17,23,449      instructions              #    0.83  insn per cycle
          3,41,344      branches                  #  130.823 M/sec
            13,763      branch-misses             #    4.03% of all branches

       0.003446904 seconds time elapsed

       0.000000000 seconds user
       0.003473000 seconds sys


saurav@ubuntu-18:~$ sudo perf stat -a sleep 1

 Performance counter stats for 'system wide':

          8,008.13 msec cpu-clock                 #    7.992 CPUs utilized
             6,729      context-switches          #    0.840 K/sec
               527      cpu-migrations            #    0.066 K/sec
                60      page-faults               #    0.007 K/sec
      2,10,13,31,659    cycles                    #    0.262 GHz
      2,48,76,64,968    instructions              #    1.18  insn per cycle
        23,10,65,189    branches                  #   28.854 M/sec
         30,26,367      branch-misses             #    1.31% of all branches

       1.002073009 seconds time elapsed

saurav@ubuntu-18:~$ sudo perf stat -a -d sleep 1
```



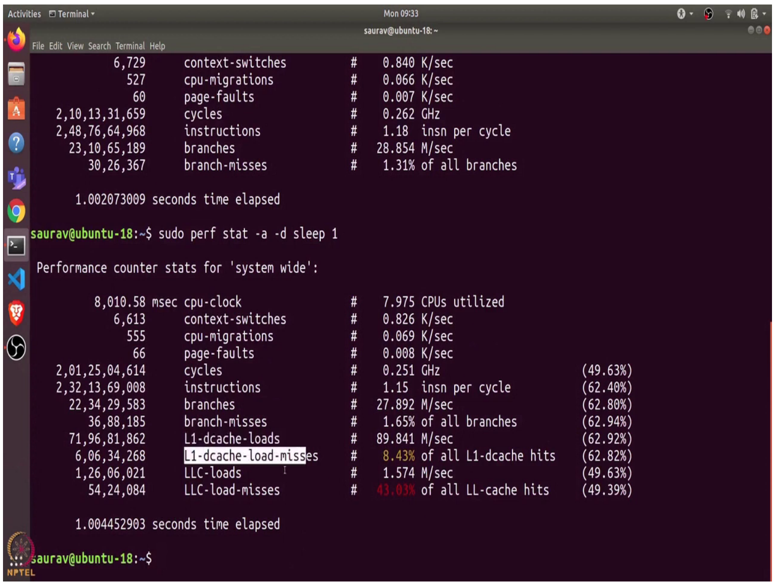
```
                 0      context-switches          #    0.000 K/sec
                 0      cpu-migrations            #    0.000 K/sec
                94      page-faults               #    0.036 M/sec
         20,75,763      cycles                    #    0.796 GHz
         17,23,449      instructions              #    0.83  insn per cycle
          3,41,344      branches                  #  130.823 M/sec
            13,763      branch-misses             #    4.03% of all branches

       0.003446904 seconds time elapsed

       0.000000000 seconds user
       0.003473000 seconds sys


saurav@ubuntu-18:~$ sudo perf stat -a sleep 1

 Performance counter stats for 'system wide':

          8,008.13 msec cpu-clock                 #    7.992 CPUs utilized
             6,729      context-switches          #    0.840 K/sec
               527      cpu-migrations            #    0.066 K/sec
                60      page-faults               #    0.007 K/sec
      2,10,13,31,659    cycles                    #    0.262 GHz
      2,48,76,64,968    instructions              #    1.18  insn per cycle
        23,10,65,189    branches                  #   28.854 M/sec
         30,26,367      branch-misses             #    1.31% of all branches

       1.002073009 seconds time elapsed

saurav@ubuntu-18:~$
```

So, this will count the statistics for 1 second. So, it says statistics for system wide. So, we have these many contracts, which is in one second, these many page faults, etc. And we can use the -d flag, get some detailed statistics.

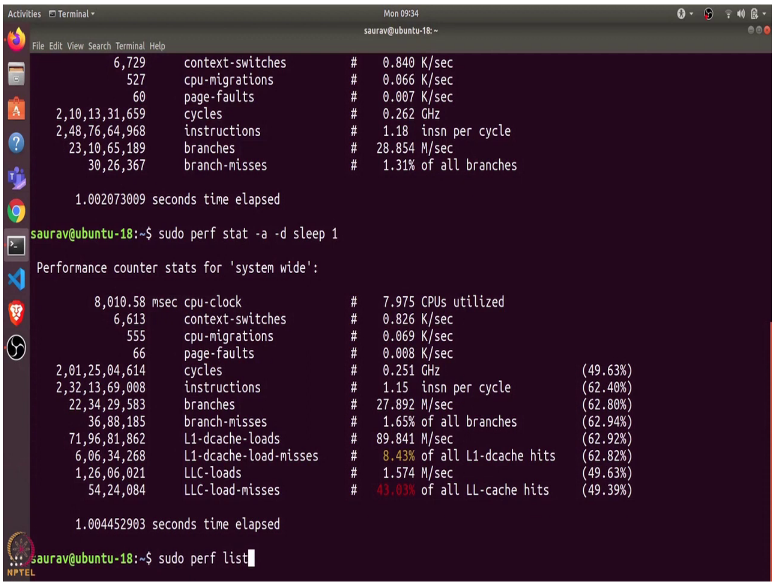
(Refer Slide Time: 1:50)



```
               6,729      context-switches          #    0.840 K/sec
                 527      cpu-migrations            #    0.066 K/sec
                  60      page-faults               #    0.007 K/sec
      2,10,13,31,659      cycles                    #    0.262 GHz
      2,48,76,64,968      instructions              #    1.18  insn per cycle
        23,10,65,189      branches                  #   28.854 M/sec
           30,26,367      branch-misses             #    1.31% of all branches

       1.002073009 seconds time elapsed

saurav@ubuntu-18:~$ sudo perf stat -a -d sleep 1

 Performance counter stats for 'system wide':

          8,010.58 msec cpu-clock                 #    7.975 CPUs utilized
               6,613      context-switches          #    0.826 K/sec
                 555      cpu-migrations            #    0.069 K/sec
                  66      page-faults               #    0.008 K/sec
      2,01,25,04,614      cycles                    #    0.251 GHz                      (49.63%)
      2,32,13,69,008      instructions              #    1.15  insn per cycle           (62.40%)
        22,34,29,583      branches                  #   27.892 M/sec                    (62.80%)
           36,88,185      branch-misses             #    1.65% of all branches          (62.94%)
        71,96,81,862      L1-dcache-loads           #   89.841 M/sec                    (62.92%)
         6,06,34,268      L1-dcache-load-misses     #    8.43% of all L1-dcache hits    (62.82%)
         1,26,06,021      LLC-loads                 #    1.574 M/sec                    (49.63%)
           54,24,084      LLC-load-misses           #   43.03% of all LL-cache hits     (49.39%)

       1.004452903 seconds time elapsed

saurav@ubuntu-18:~$
```



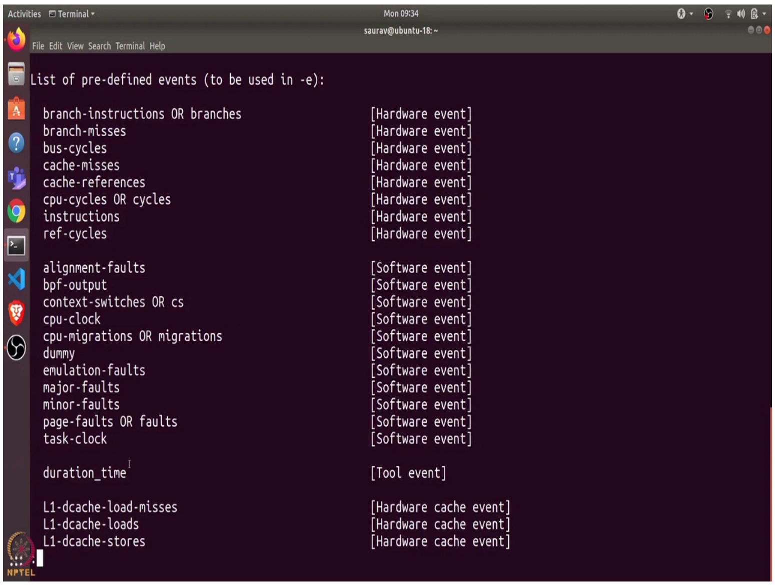
```
               6,729      context-switches          #    0.840 K/sec
                 527      cpu-migrations            #    0.066 K/sec
                  60      page-faults               #    0.007 K/sec
      2,10,13,31,659      cycles                    #    0.262 GHz
      2,48,76,64,968      instructions              #    1.18  insn per cycle
        23,10,65,189      branches                  #   28.854 M/sec
           30,26,367      branch-misses             #    1.31% of all branches

       1.002073009 seconds time elapsed

saurav@ubuntu-18:~$ sudo perf stat -a -d sleep 1

 Performance counter stats for 'system wide':

          8,010.58 msec cpu-clock                 #    7.975 CPUs utilized
               6,613      context-switches          #    0.826 K/sec
                 555      cpu-migrations            #    0.069 K/sec
                  66      page-faults               #    0.008 K/sec
      2,01,25,04,614      cycles                    #    0.251 GHz                      (49.63%)
      2,32,13,69,008      instructions              #    1.15  insn per cycle           (62.40%)
        22,34,29,583      branches                  #   27.892 M/sec                    (62.80%)
           36,88,185      branch-misses             #    1.65% of all branches          (62.94%)
        71,96,81,862      L1-dcache-loads           #   89.841 M/sec                    (62.92%)
         6,06,34,268      L1-dcache-load-misses     #    8.43% of all L1-dcache hits    (62.82%)
         1,26,06,021      LLC-loads                 #    1.574 M/sec                    (49.63%)
           54,24,084      LLC-load-misses           #   43.03% of all LL-cache hits     (49.39%)

       1.004452903 seconds time elapsed

saurav@ubuntu-18:~$
```



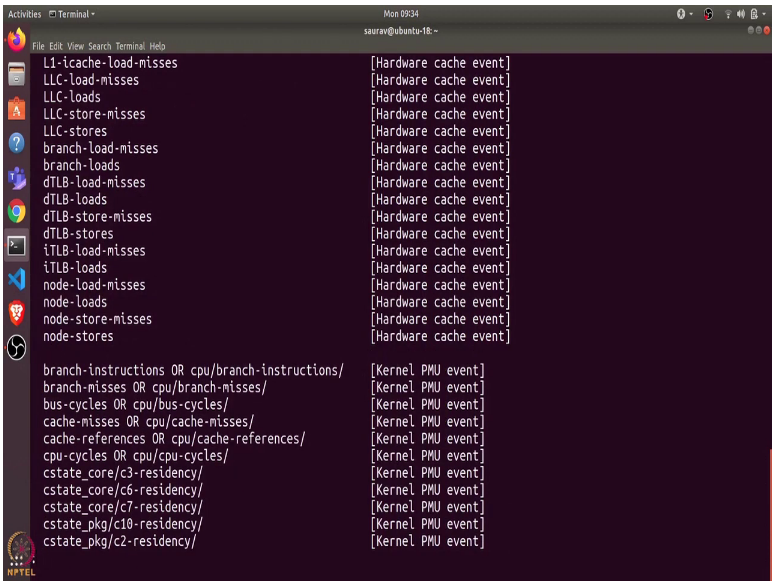
```
               6,729      context-switches          #    0.840 K/sec
                 527      cpu-migrations            #    0.066 K/sec
                  60      page-faults               #    0.007 K/sec
      2,10,13,31,659      cycles                    #    0.262 GHz
      2,48,76,64,968      instructions              #    1.18  insn per cycle
        23,10,65,189      branches                  #   28.854 M/sec
           30,26,367      branch-misses             #    1.31% of all branches

       1.002073009 seconds time elapsed

saurav@ubuntu-18:~$ sudo perf stat -a -d sleep 1

 Performance counter stats for 'system wide':

          8,010.58 msec cpu-clock                 #    7.975 CPUs utilized
               6,613      context-switches          #    0.826 K/sec
                 555      cpu-migrations            #    0.069 K/sec
                  66      page-faults               #    0.008 K/sec
      2,01,25,04,614      cycles                    #    0.251 GHz                      (49.63%)
      2,32,13,69,008      instructions              #    1.15  insn per cycle           (62.40%)
        22,34,29,583      branches                  #   27.892 M/sec                    (62.80%)
           36,88,185      branch-misses             #    1.65% of all branches          (62.94%)
        71,96,81,862      L1-dcache-loads           #   89.841 M/sec                    (62.92%)
         6,06,34,268      L1-dcache-load-misses     #    8.43% of all L1-dcache hits    (62.82%)
         1,26,06,021      LLC-loads                 #    1.574 M/sec                    (49.63%)
           54,24,084      LLC-load-misses           #   43.03% of all LL-cache hits     (49.39%)

       1.004452903 seconds time elapsed

saurav@ubuntu-18:~$ sudo perf list
```

If I run it again, there, it shows us some other strategies such as L1 cache loads, L1 cache, load misses et cetera. Another important command that you can use with perf is perf list. So, if I run sudo perf list.

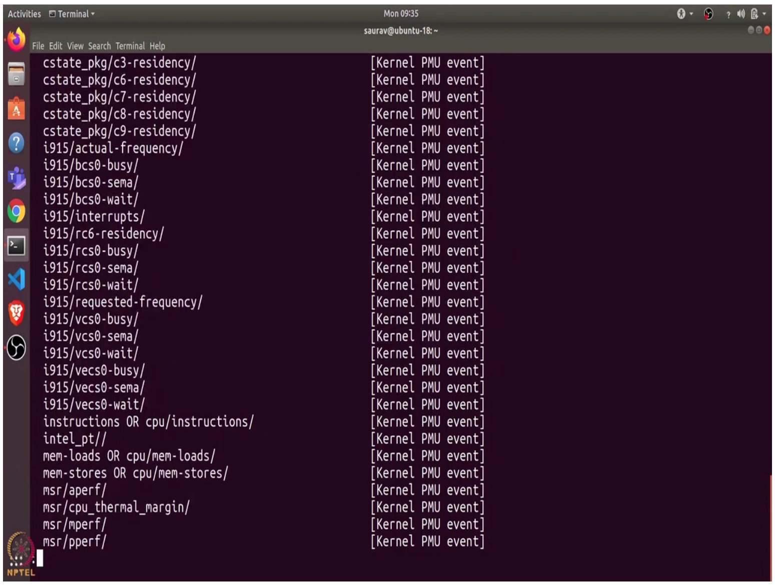
(Refer Slide Time: 2:03)



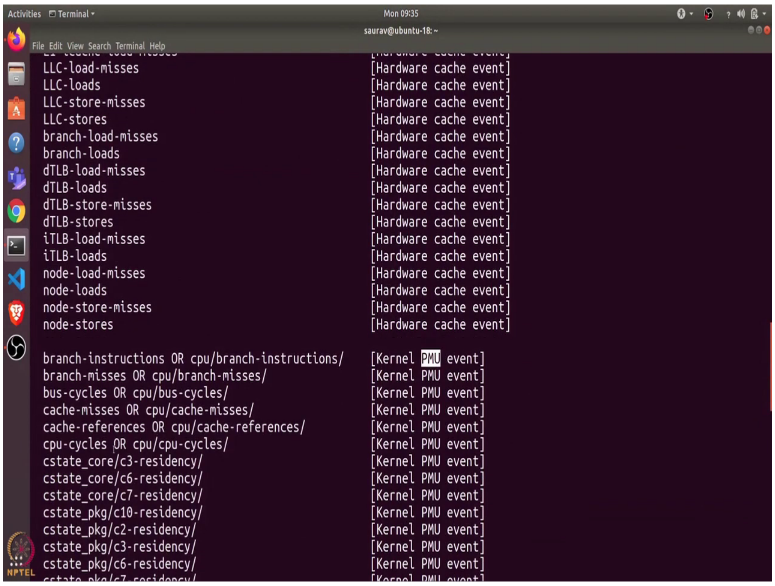
```
List of pre-defined events (to be used in -e):

  branch-instructions OR branches                    [Hardware event]
  branch-misses                                      [Hardware event]
  bus-cycles                                         [Hardware event]
  cache-misses                                       [Hardware event]
  cache-references                                   [Hardware event]
  cpu-cycles OR cycles                               [Hardware event]
  instructions                                       [Hardware event]
  ref-cycles                                         [Hardware event]

  alignment-faults                                   [Software event]
  bpf-output                                         [Software event]
  context-switches OR cs                             [Software event]
  cpu-clock                                          [Software event]
  cpu-migrations OR migrations                       [Software event]
  dummy                                              [Software event]
  emulation-faults                                   [Software event]
  major-faults                                       [Software event]
  minor-faults                                       [Software event]
  page-faults OR faults                              [Software event]
  task-clock                                         [Software event]

  duration_time                                      [Tool event]

  L1-dcache-load-misses                              [Hardware cache event]
  L1-dcache-loads                                    [Hardware cache event]
  L1-dcache-stores                                   [Hardware cache event]
```



```
  L1-icache-load-misses                              [Hardware cache event]
  LLC-load-misses                                    [Hardware cache event]
  LLC-loads                                          [Hardware cache event]
  LLC-store-misses                                   [Hardware cache event]
  LLC-stores                                         [Hardware cache event]
  branch-load-misses                                 [Hardware cache event]
  branch-loads                                       [Hardware cache event]
  dTLB-load-misses                                   [Hardware cache event]
  dTLB-loads                                         [Hardware cache event]
  dTLB-store-misses                                  [Hardware cache event]
  dTLB-stores                                        [Hardware cache event]
  iTLB-load-misses                                   [Hardware cache event]
  iTLB-loads                                         [Hardware cache event]
  node-load-misses                                   [Hardware cache event]
  node-loads                                         [Hardware cache event]
  node-store-misses                                  [Hardware cache event]
  node-stores                                        [Hardware cache event]

  branch-instructions OR cpu/branch-instructions/    [Kernel PMU event]
  branch-misses OR cpu/branch-misses/                [Kernel PMU event]
  bus-cycles OR cpu/bus-cycles/                      [Kernel PMU event]
  cache-misses OR cpu/cache-misses/                  [Kernel PMU event]
  cache-references OR cpu/cache-references/          [Kernel PMU event]
  cpu-cycles OR cpu/cpu-cycles/                      [Kernel PMU event]
  cstate_core/c3-residency/                          [Kernel PMU event]
  cstate_core/c6-residency/                          [Kernel PMU event]
  cstate_core/c7-residency/                          [Kernel PMU event]
  cstate_pkg/c10-residency/                          [Kernel PMU event]
  cstate_pkg/c2-residency/                           [Kernel PMU event]
```



```
cstate_pkg/c3-residency/                           [Kernel PMU event]
cstate_pkg/c6-residency/                           [Kernel PMU event]
cstate_pkg/c7-residency/                           [Kernel PMU event]
cstate_pkg/c8-residency/                           [Kernel PMU event]
cstate_pkg/c9-residency/                           [Kernel PMU event]
i915/actual-frequency/                             [Kernel PMU event]
i915/bcs0-busy/                                    [Kernel PMU event]
i915/bcs0-sema/                                    [Kernel PMU event]
i915/bcs0-wait/                                    [Kernel PMU event]
i915/interrupts/                                   [Kernel PMU event]
i915/rc6-residency/                                [Kernel PMU event]
i915/rcs0-busy/                                    [Kernel PMU event]
i915/rcs0-sema/                                    [Kernel PMU event]
i915/rcs0-wait/                                    [Kernel PMU event]
i915/requested-frequency/                          [Kernel PMU event]
i915/vcs0-busy/                                    [Kernel PMU event]
i915/vcs0-sema/                                    [Kernel PMU event]
i915/vcs0-wait/                                    [Kernel PMU event]
i915/vecs0-busy/                                   [Kernel PMU event]
i915/vecs0-sema/                                   [Kernel PMU event]
i915/vecs0-wait/                                   [Kernel PMU event]
instructions OR cpu/instructions/                  [Kernel PMU event]
intel_pt//                                         [Kernel PMU event]
mem-loads OR cpu/mem-loads/                        [Kernel PMU event]
mem-stores OR cpu/mem-stores/                      [Kernel PMU event]
msr/aperf/                                         [Kernel PMU event]
msr/cpu_thermal_margin/                            [Kernel PMU event]
msr/mperf/                                         [Kernel PMU event]
msr/pperf/                                         [Kernel PMU event]
```

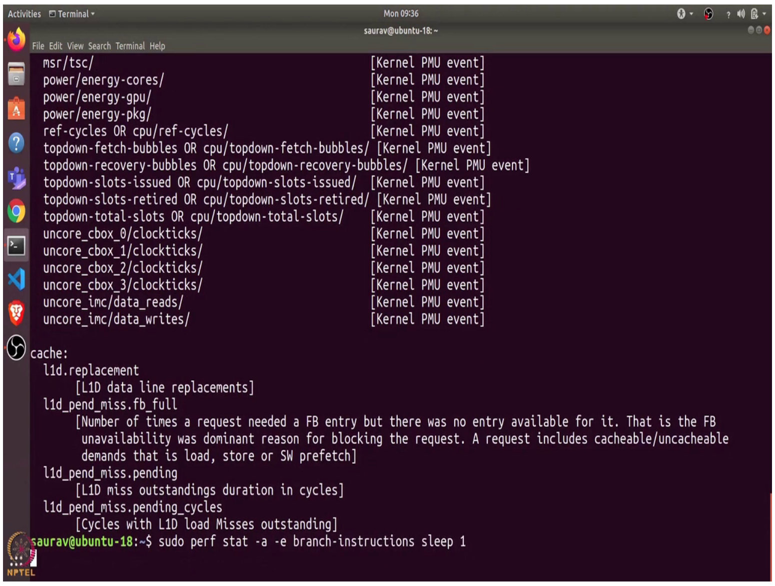


```
LLC-load-misses                                    [Hardware cache event]
LLC-loads                                          [Hardware cache event]
LLC-store-misses                                   [Hardware cache event]
LLC-stores                                         [Hardware cache event]
branch-load-misses                                 [Hardware cache event]
branch-loads                                       [Hardware cache event]
dTLB-load-misses                                   [Hardware cache event]
dTLB-loads                                         [Hardware cache event]
dTLB-store-misses                                  [Hardware cache event]
dTLB-stores                                        [Hardware cache event]
iTLB-load-misses                                   [Hardware cache event]
iTLB-loads                                         [Hardware cache event]
node-load-misses                                   [Hardware cache event]
node-loads                                         [Hardware cache event]
node-store-misses                                  [Hardware cache event]
node-stores                                        [Hardware cache event]

branch-instructions OR cpu/branch-instructions/    [Kernel PMU event]
branch-misses OR cpu/branch-misses/                [Kernel PMU event]
bus-cycles OR cpu/bus-cycles/                      [Kernel PMU event]
cache-misses OR cpu/cache-misses/                  [Kernel PMU event]
cache-references OR cpu/cache-references/          [Kernel PMU event]
cpu-cycles OR cpu/cpu-cycles/                      [Kernel PMU event]
cstate_core/c3-residency/                          [Kernel PMU event]
cstate_core/c6-residency/                          [Kernel PMU event]
cstate_core/c7-residency/                          [Kernel PMU event]
cstate_pkg/c10-residency/                          [Kernel PMU event]
cstate_pkg/c2-residency/                           [Kernel PMU event]
cstate_pkg/c3-residency/                           [Kernel PMU event]
cstate_pkg/c6-residency/                           [Kernel PMU event]
cstate_pkg/c7-residency/                           [Kernel PMU event]
```

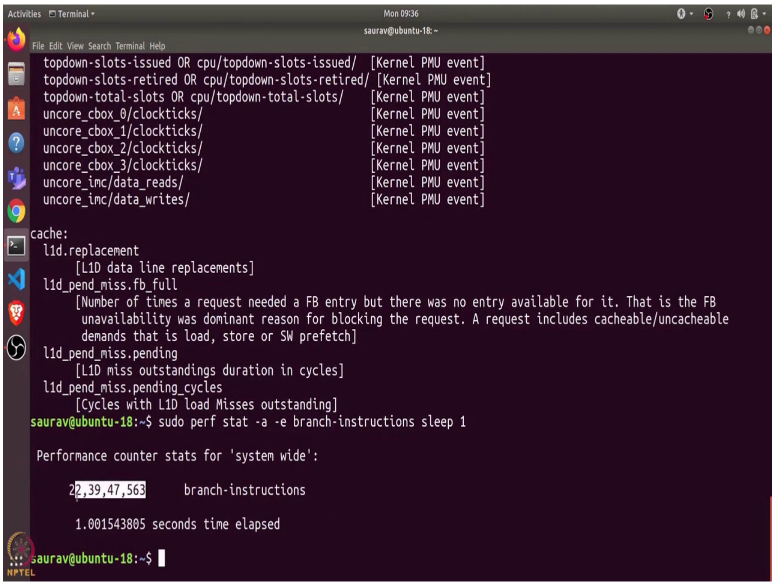
It will list all the events that we can count using perf. So, it shows us some hardware events, such as branch instruction, branch misses, then there are some software events for which kernel maintains a counter, such as CPU migration, context switches. And then we have some cache events. This PMU refers to performance monitoring unit. So, there is a separate unit with the processor, which counts various hardware events, which include these branch instructions, or cache misses, et cetera.

So, there are a lot of events, let us say want to measure certain event. For instance, let us try to measure these branch instructions.

(Refer Slide Time: 2:42)



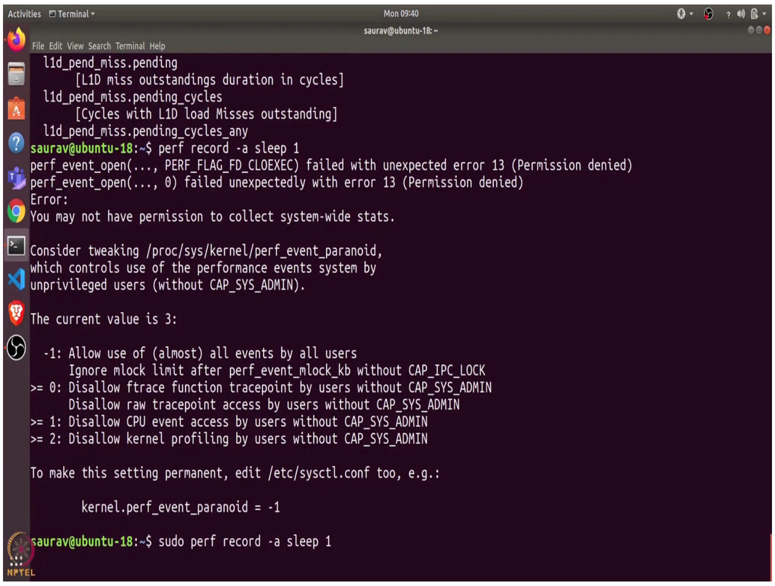
```
  msr/tsc/                                          [Kernel PMU event]
  power/energy-cores/                               [Kernel PMU event]
  power/energy-gpu/                                 [Kernel PMU event]
  power/energy-pkg/                                 [Kernel PMU event]
  ref-cycles OR cpu/ref-cycles/                     [Kernel PMU event]
  topdown-fetch-bubbles OR cpu/topdown-fetch-bubbles/ [Kernel PMU event]
  topdown-recovery-bubbles OR cpu/topdown-recovery-bubbles/ [Kernel PMU event]
  topdown-slots-issued OR cpu/topdown-slots-issued/  [Kernel PMU event]
  topdown-slots-retired OR cpu/topdown-slots-retired/ [Kernel PMU event]
  topdown-total-slots OR cpu/topdown-total-slots/    [Kernel PMU event]
  uncore_cbox_0/clockticks/                         [Kernel PMU event]
  uncore_cbox_1/clockticks/                         [Kernel PMU event]
  uncore_cbox_2/clockticks/                         [Kernel PMU event]
  uncore_cbox_3/clockticks/                         [Kernel PMU event]
  uncore_imc/data_reads/                            [Kernel PMU event]
  uncore_imc/data_writes/                           [Kernel PMU event]

cache:
  l1d.replacement
       [L1D data line replacements]
  l1d_pend_miss.fb_full
       [Number of times a request needed a FB entry but there was no entry available for it. That is the FB
        unavailability was dominant reason for blocking the request. A request includes cacheable/uncacheable
        demands that is load, store or SW prefetch]
  l1d_pend_miss.pending
       [L1D miss outstandings duration in cycles]
  l1d_pend_miss.pending_cycles
       [Cycles with L1D load Misses outstanding]
saurav@ubuntu-18:~$ sudo perf stat -a -e branch-instructions sleep 1
```



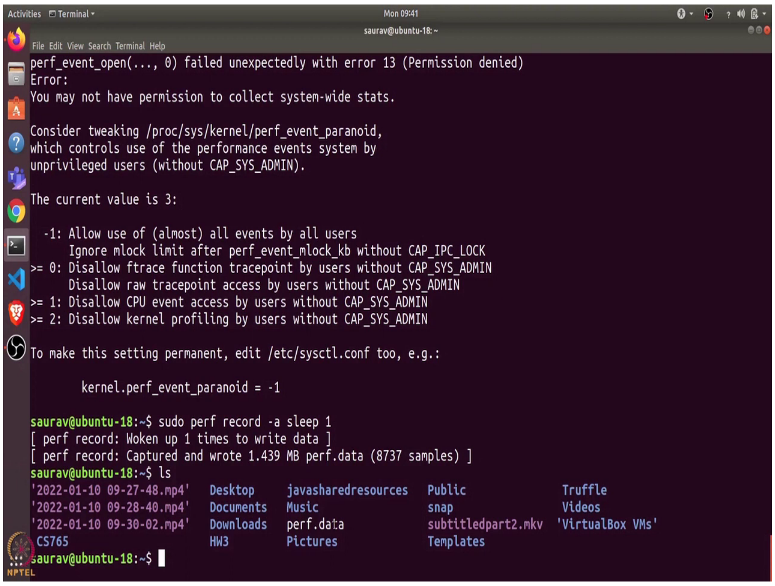
```
  topdown-slots-issued OR cpu/topdown-slots-issued/  [Kernel PMU event]
  topdown-slots-retired OR cpu/topdown-slots-retired/ [Kernel PMU event]
  topdown-total-slots OR cpu/topdown-total-slots/    [Kernel PMU event]
  uncore_cbox_0/clockticks/                         [Kernel PMU event]
  uncore_cbox_1/clockticks/                         [Kernel PMU event]
  uncore_cbox_2/clockticks/                         [Kernel PMU event]
  uncore_cbox_3/clockticks/                         [Kernel PMU event]
  uncore_imc/data_reads/                            [Kernel PMU event]
  uncore_imc/data_writes/                           [Kernel PMU event]

cache:
  l1d.replacement
       [L1D data line replacements]
  l1d_pend_miss.fb_full
       [Number of times a request needed a FB entry but there was no entry available for it. That is the FB
        unavailability was dominant reason for blocking the request. A request includes cacheable/uncacheable
        demands that is load, store or SW prefetch]
  l1d_pend_miss.pending
       [L1D miss outstandings duration in cycles]
  l1d_pend_miss.pending_cycles
       [Cycles with L1D load Misses outstanding]
saurav@ubuntu-18:~$ sudo perf stat -a -e branch-instructions sleep 1

 Performance counter stats for 'system wide':

       22,39,47,563      branch-instructions

       1.001543805 seconds time elapsed

saurav@ubuntu-18:~$ 
```

So, what we can do is we can use the -e flag. Let us say you want to measure branch instructions for 1 second, across all the processes, so I can use -e branch instructions. And then it will give me the count for the branch instructions for 1 second.

(Refer Slide Time: 3:04)



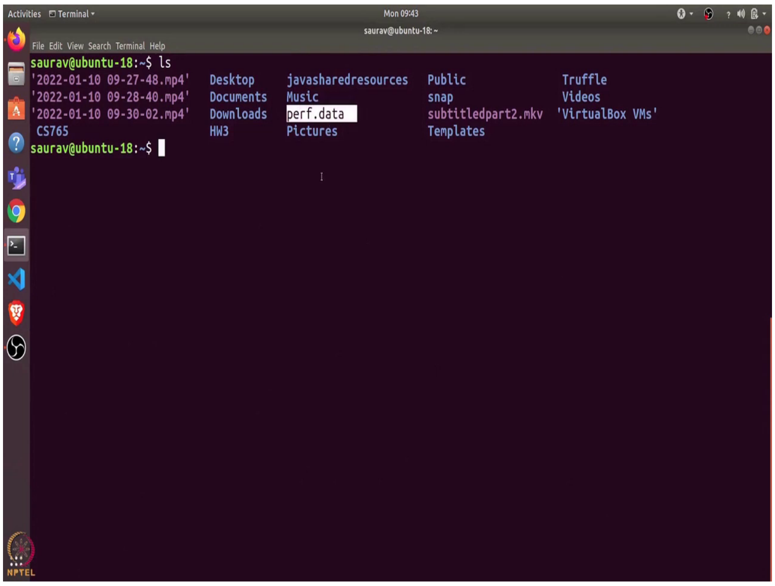
```
  l1d_pend_miss.pending
       [L1D miss outstandings duration in cycles]
  l1d_pend_miss.pending_cycles
       [Cycles with L1D load Misses outstanding]
  l1d_pend_miss.pending_cycles_any
saurav@ubuntu-18:~$ perf record -a sleep 1
perf_event_open(..., PERF_FLAG_FD_CLOEXEC) failed with unexpected error 13 (Permission denied)
perf_event_open(..., 0) failed unexpectedly with error 13 (Permission denied)
Error:
You may not have permission to collect system-wide stats.

Consider tweaking /proc/sys/kernel/perf_event_paranoid,
which controls use of the performance events system by
unprivileged users (without CAP_SYS_ADMIN).

The current value is 3:

  -1: Allow use of (almost) all events by all users
      Ignore mlock limit after perf_event_mlock_kb without CAP_IPC_LOCK
>= 0: Disallow ftrace function tracepoint by users without CAP_SYS_ADMIN
      Disallow raw tracepoint access by users without CAP_SYS_ADMIN
>= 1: Disallow CPU event access by users without CAP_SYS_ADMIN
>= 2: Disallow kernel profiling by users without CAP_SYS_ADMIN

To make this setting permanent, edit /etc/sysctl.conf too, e.g.:

        kernel.perf_event_paranoid = -1

saurav@ubuntu-18:~$ sudo perf record -a sleep 1
```



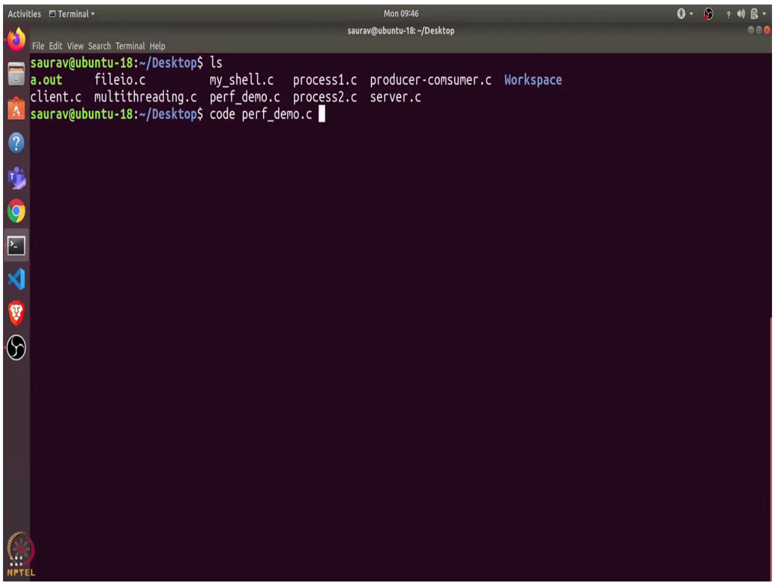
```
perf_event_open(..., 0) failed unexpectedly with error 13 (Permission denied)
Error:
You may not have permission to collect system-wide stats.

Consider tweaking /proc/sys/kernel/perf_event_paranoid,
which controls use of the performance events system by
unprivileged users (without CAP_SYS_ADMIN).

The current value is 3:

  -1: Allow use of (almost) all events by all users
      Ignore mlock limit after perf_event_mlock_kb without CAP_IPC_LOCK
>= 0: Disallow ftrace function tracepoint by users without CAP_SYS_ADMIN
      Disallow raw tracepoint access by users without CAP_SYS_ADMIN
>= 1: Disallow CPU event access by users without CAP_SYS_ADMIN
>= 2: Disallow kernel profiling by users without CAP_SYS_ADMIN

To make this setting permanent, edit /etc/sysctl.conf too, e.g.:

        kernel.perf_event_paranoid = -1

saurav@ubuntu-18:~$ sudo perf record -a sleep 1
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 1.439 MB perf.data (8737 samples) ]
saurav@ubuntu-18:~$ ls
'2022-01-10 09-27-48.mp4'   Desktop     javasharedresources   Public               Truffle
'2022-01-10 09-28-40.mp4'   Documents   Music                 snap                 Videos
'2022-01-10 09-30-02.mp4'   Downloads   perf.data             subtitledpart2.mkv  'VirtualBox VMs'
 CS765                      HW3         Pictures              Templates
saurav@ubuntu-18:~$ 
```

The next command that we will see is perf report. Perf report is used to collect the profile data about a certain program in a separate file. So, let us say I want to record the data for complete system for 1 second, I need to use sudo. If we do ls, then we can see there is this perf.data file. This is a binary file, which stores the profiling data for our complete system.

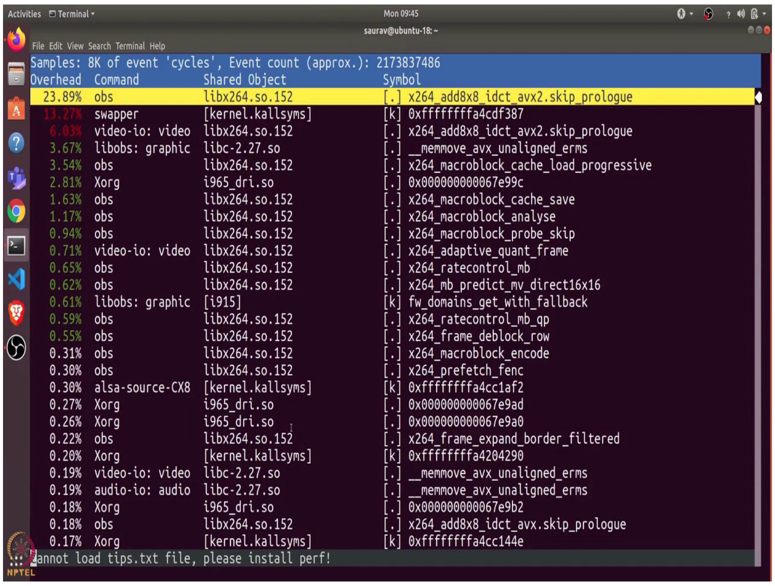
(Refer Slide Time: 3:44)



```
saurav@ubuntu-18:~$ ls
'2022-01-10 09-27-48.mp4'  Desktop    javasharedresources  Public              Truffle
'2022-01-10 09-28-40.mp4'  Documents  Music                snap                Videos
'2022-01-10 09-30-02.mp4'  Downloads  perf.data            subtitledpart2.mkv  'VirtualBox VMs'
 CS765                     HW3        Pictures             Templates
saurav@ubuntu-18:~$
```


Activities
Terminal
Mon 09:46
saurav@ubuntu-18: ~/Desktop
File Edit View Search Terminal Help
NPTEL

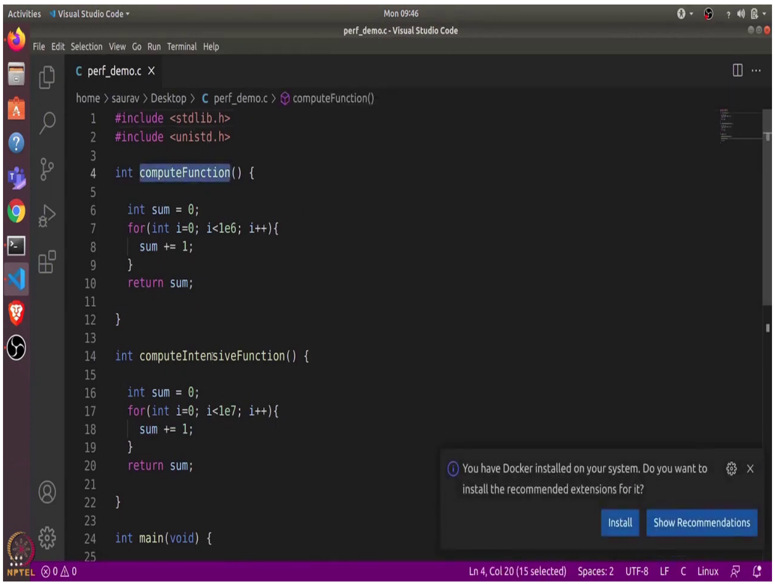
```
saurav@ubuntu-18:~/Desktop$ ls
a.out     fileio.c          my_shell.c   process1.c  producer-consumer.c  Workspace
client.c  multithreading.c  perf_demo.c  process2.c  server.c
saurav@ubuntu-18:~/Desktop$ code perf_demo.c
```



```
Samples: 8K of event 'cycles', Event count (approx.): 2173837486
Overhead  Command          Shared Object        Symbol
  23.89%  obs              libx264.so.152       [.] x264_add8x8_idct_avx2.skip_prologue
  13.27%  swapper          [kernel.kallsyms]    [k] 0xffffffffa4cdf387
   6.03%  video-io: video  libx264.so.152       [.] x264_add8x8_idct_avx2.skip_prologue
   3.67%  libobs: graphic  libc-2.27.so         [.] __memmove_avx_unaligned_erms
   3.54%  obs              libx264.so.152       [.] x264_macroblock_cache_load_progressive
   2.81%  Xorg             i965_dri.so          [.] 0x000000000067e99c
   1.63%  obs              libx264.so.152       [.] x264_macroblock_cache_save
   1.17%  obs              libx264.so.152       [.] x264_macroblock_analyse
   0.94%  obs              libx264.so.152       [.] x264_macroblock_probe_skip
   0.71%  video-io: video  libx264.so.152       [.] x264_adaptive_quant_frame
   0.65%  obs              libx264.so.152       [.] x264_ratecontrol_mb
   0.62%  obs              libx264.so.152       [.] x264_mb_predict_mv_direct16x16
   0.61%  libobs: graphic  [i915]               [k] fw_domains_get_with_fallback
   0.59%  obs              libx264.so.152       [.] x264_ratecontrol_mb_qp
   0.55%  obs              libx264.so.152       [.] x264_frame_deblock_row
   0.31%  obs              libx264.so.152       [.] x264_macroblock_encode
   0.30%  obs              libx264.so.152       [.] x264_prefetch_fenc
   0.30%  alsa-source-CX8  [kernel.kallsyms]    [k] 0xffffffffa4cc1af2
   0.27%  Xorg             i965_dri.so          [.] 0x000000000067e9ad
   0.26%  Xorg             i965_dri.so          [.] 0x000000000067e9a0
   0.22%  obs              libx264.so.152       [.] x264_frame_expand_border_filtered
   0.20%  Xorg             [kernel.kallsyms]    [k] 0xffffffffa4204290
   0.19%  video-io: video  libc-2.27.so         [.] __memmove_avx_unaligned_erms
   0.19%  audio-io: audio  libc-2.27.so         [.] __memmove_avx_unaligned_erms
   0.18%  Xorg             i965_dri.so          [.] 0x000000000067e9b2
   0.18%  obs              libx264.so.152       [.] x264_add8x8_idct_avx.skip_prologue
   0.17%  Xorg             [kernel.kallsyms]    [k] 0xffffffffa4cc144e
Cannot load tips.txt file, please install perf!
```

So, how can we do this perf dot data file? We need to use perf report to read this binary file. Let us use perf report where input file perf.data. So, this shows us what is the percentage of CPU cycles that were used for this particular command and the shared object and so on. So, let us quit this using Q and let us try to profile some C program. So, I have written this perf demo dot c file. Let us first have a look at its contents.

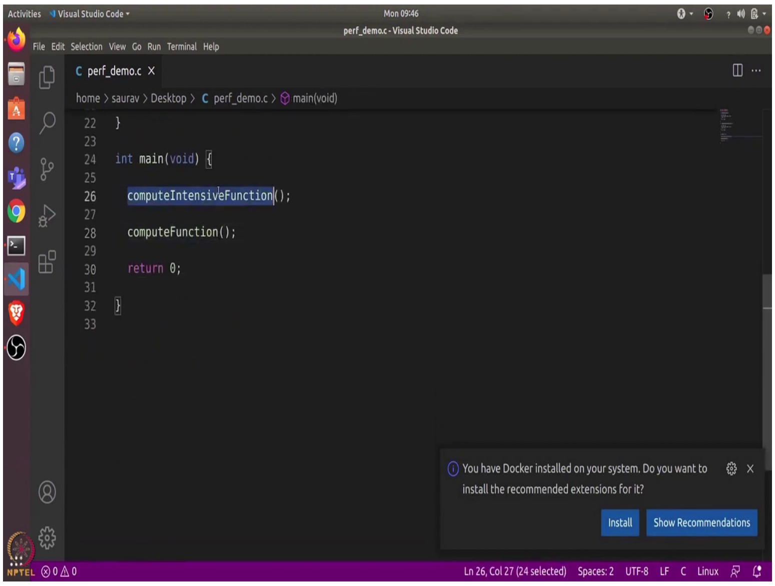
(Refer Slide Time: 4:18)



perf_demo.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help
perf_demo.c
home > saurav > Desktop > perf_demo.c > computeFunction()
Install
Show Recommendations
Ln 4, Col 20 (15 selected) Spaces: 2 UTF-8 LF C Linux

```c
#include <stdlib.h>
#include <unistd.h>

int computeFunction() {

  int sum = 0;
  for(int i=0; i<1e6; i++){
    sum += 1;
  }
  return sum;

}

int computeIntensiveFunction() {

  int sum = 0;
  for(int i=0; i<1e7; i++){
    sum += 1;
  }
  return sum;

}

int main(void) {
```

You have Docker installed on your system. Do you want to install the recommended extensions for it?



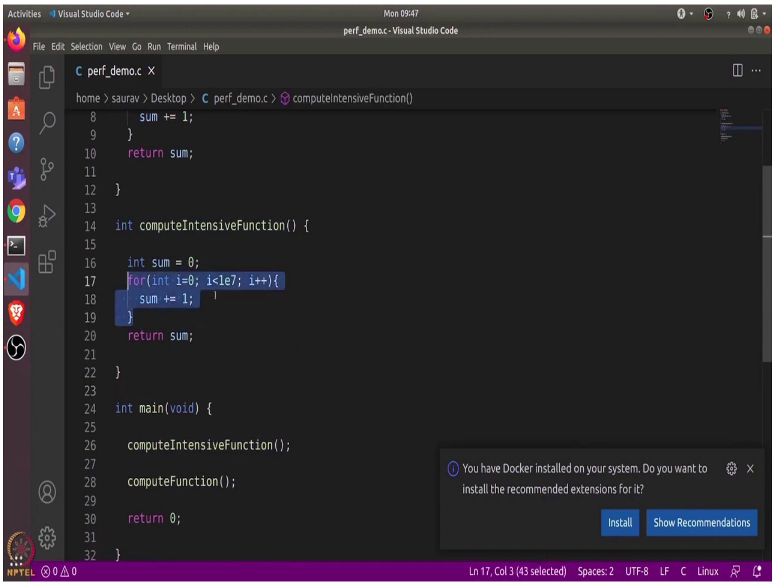
perf_demo.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help
perf_demo.c
home > saurav > Desktop > perf_demo.c > main(void)
Install
Show Recommendations
Ln 26, Col 27 (24 selected) Spaces: 2 UTF-8 LF C Linux

```c
}

int main(void) {

  computeIntensiveFunction();

  computeFunction();

  return 0;

}
```

You have Docker installed on your system. Do you want to install the recommended extensions for it?

So, this is a very simple program. It has two functions. One is computeFunction, another is computeIntensiveFunction. And in the main function, we just call these two functions.

(Refer Slide Time: 4:31)

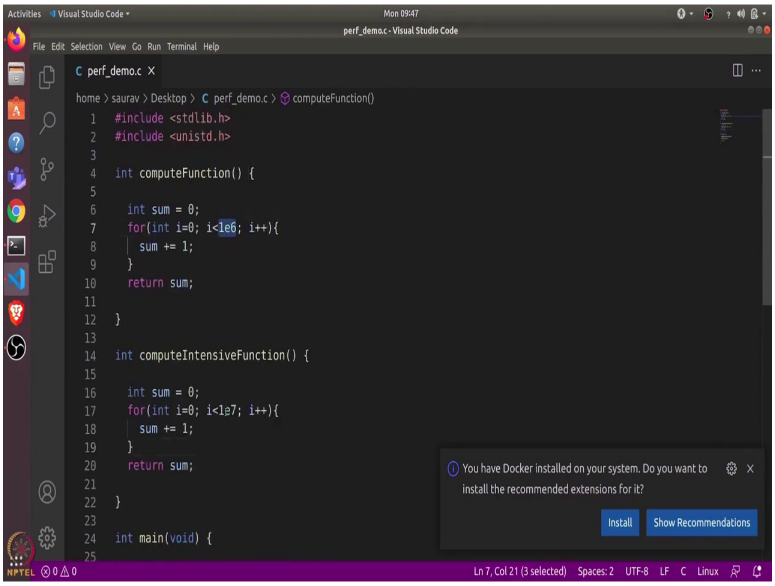

Activities
Visual Studio Code
Mon 09:47
perf_demo.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help
perf_demo.c
home > saurav > Desktop > perf_demo.c > computeIntensiveFunction()
Install
Show Recommendations
Ln 17, Col 3 (43 selected) Spaces: 2 UTF-8 LF C Linux
NPTEL

```
    sum += 1;
  }
  return sum;

}

int computeIntensiveFunction() {

  int sum = 0;
  for(int i=0; i<1e7; i++){
    sum += 1;
  }
  return sum;

}

int main(void) {

  computeIntensiveFunction();

  computeFunction();

  return 0;

}
```

You have Docker installed on your system. Do you want to install the recommended extensions for it?

In the computeIntensiveFunction, we iterate 10 raise to 7 times, and we just add one to the sum a dummy variable and return that sum.

(Refer Slide Time: 4:38)

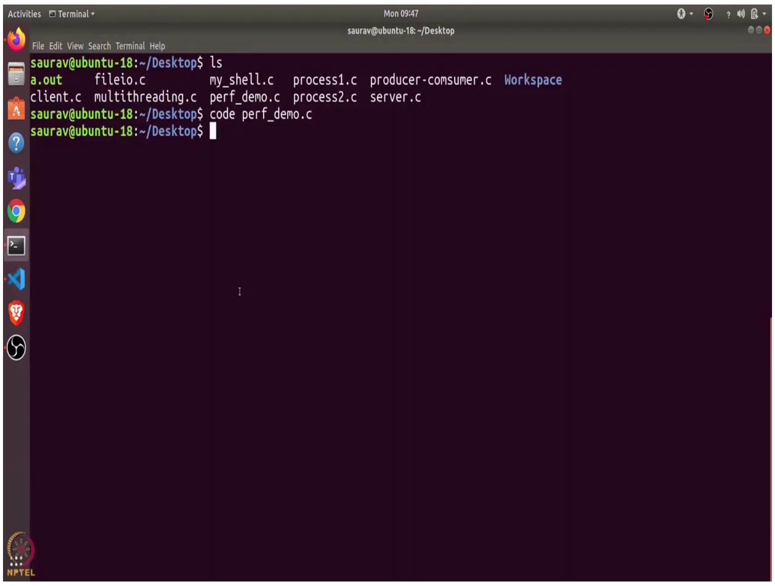

Activities
Visual Studio Code
Mon 09:47
perf_demo.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help
perf_demo.c
home > saurav > Desktop > perf_demo.c > computeFunction()
Install
Show Recommendations
Ln 7, Col 21 (3 selected) Spaces: 2 UTF-8 LF C Linux
NPTEL

```
#include <stdlib.h>
#include <unistd.h>

int computeFunction() {

  int sum = 0;
  for(int i=0; i<1e6; i++){
    sum += 1;
  }
  return sum;

}

int computeIntensiveFunction() {

  int sum = 0;
  for(int i=0; i<1e7; i++){
    sum += 1;
  }
  return sum;

}

int main(void) {
```

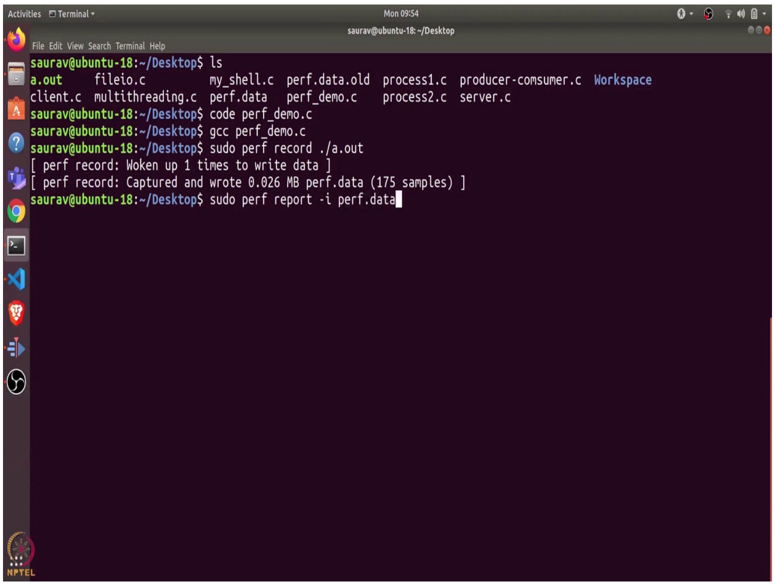
You have Docker installed on your system. Do you want to install the recommended extensions for it?

In computeFunction, we iterate 10 raise to 6 times. So, computeIntensiveFunction uses 10 times more number of iterations then computeFunction.

(Refer Slide Time: 4:46)



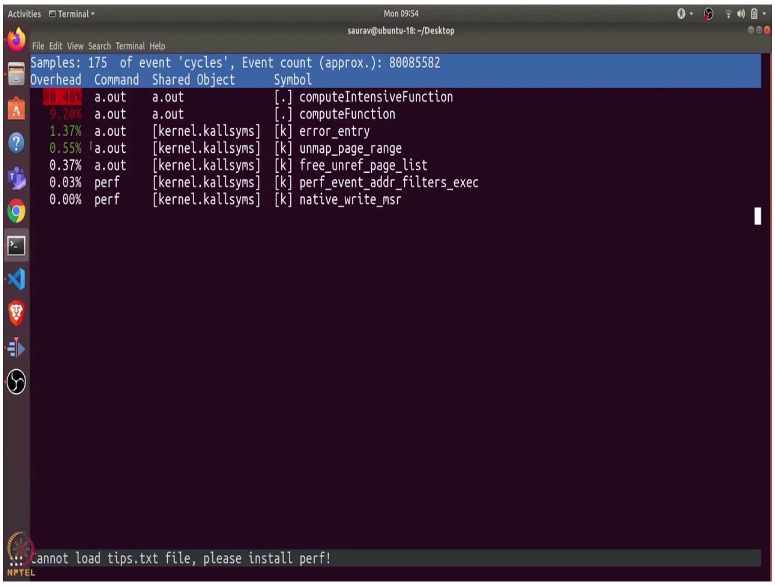
```
saurav@ubuntu-18:~/Desktop$ ls
a.out     fileio.c          my_shell.c   process1.c  producer-consumer.c  Workspace
client.c  multithreading.c  perf_demo.c  process2.c  server.c
saurav@ubuntu-18:~/Desktop$ code perf_demo.c
saurav@ubuntu-18:~/Desktop$ 
```



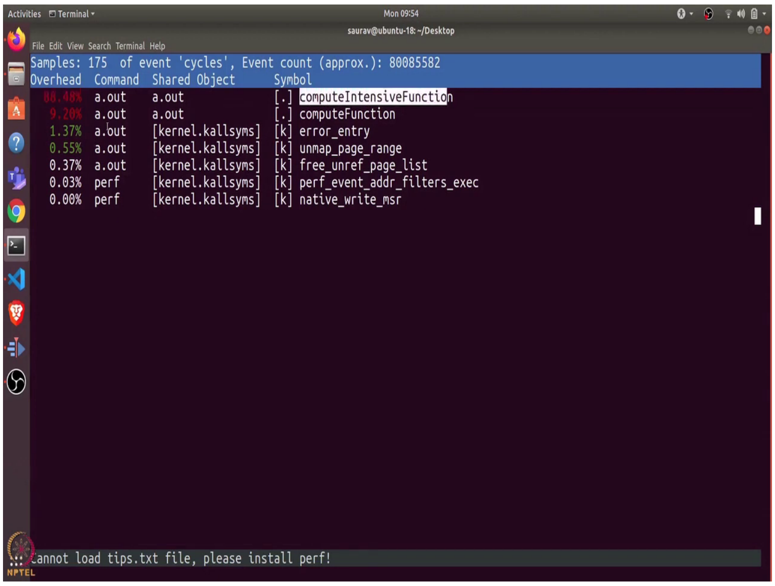
```
saurav@ubuntu-18:~/Desktop$ ls
a.out     fileio.c          my_shell.c  perf.data.old  process1.c  producer-consumer.c  Workspace
client.c  multithreading.c  perf.data   perf_demo.c    process2.c  server.c
saurav@ubuntu-18:~/Desktop$ code perf_demo.c
saurav@ubuntu-18:~/Desktop$ gcc perf_demo.c
saurav@ubuntu-18:~/Desktop$ sudo perf record ./a.out
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.026 MB perf.data (175 samples) ]
saurav@ubuntu-18:~/Desktop$ sudo perf report -i perf.data
```

Let us try to compile this gcc perf_demo.c. And let us run the executable and try to profile it using perf. So, I will use sudo perf record ./a.out and this will generate the perf.data file. Now let us try to see the profile using sudo perf report -i perf.data.

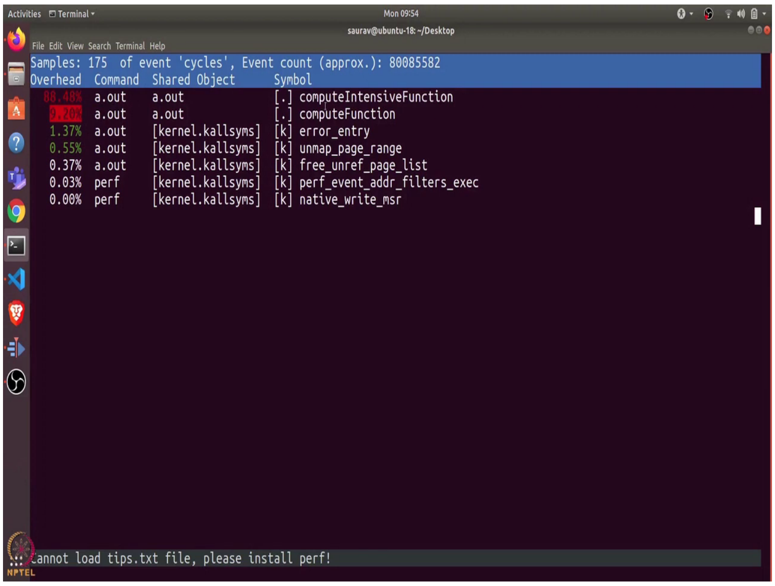
(Refer Slide Time: 5:22)



```
Samples: 175  of event 'cycles', Event count (approx.): 80085582
Overhead  Command  Shared Object      Symbol
  88.40%  a.out    a.out              [.] computeIntensiveFunction
   9.20%  a.out    a.out              [.] computeFunction
   1.37%  a.out    [kernel.kallsyms]  [k] error_entry
   0.55%  a.out    [kernel.kallsyms]  [k] unmap_page_range
   0.37%  a.out    [kernel.kallsyms]  [k] free_unref_page_list
   0.03%  perf     [kernel.kallsyms]  [k] perf_event_addr_filters_exec
   0.00%  perf     [kernel.kallsyms]  [k] native_write_msr

Cannot load tips.txt file, please install perf!
```

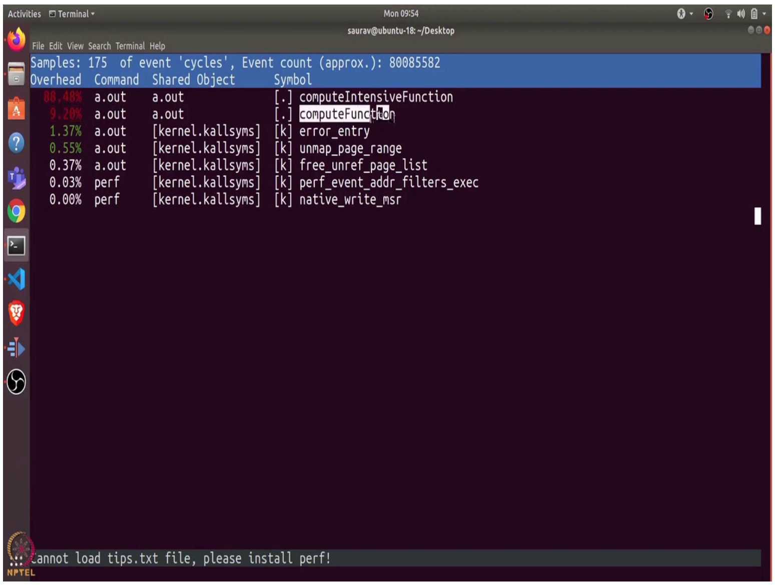


```
Samples: 175  of event 'cycles', Event count (approx.): 80085582
Overhead  Command  Shared Object      Symbol
  88.40%  a.out    a.out              [.] computeIntensiveFunction
   9.20%  a.out    a.out              [.] computeFunction
   1.37%  a.out    [kernel.kallsyms]  [k] error_entry
   0.55%  a.out    [kernel.kallsyms]  [k] unmap_page_range
   0.37%  a.out    [kernel.kallsyms]  [k] free_unref_page_list
   0.03%  perf     [kernel.kallsyms]  [k] perf_event_addr_filters_exec
   0.00%  perf     [kernel.kallsyms]  [k] native_write_msr

Cannot load tips.txt file, please install perf!
```



Activities
Terminal
Mon 09:54
saurav@ubuntu-18: ~/Desktop
File Edit View Search Terminal Help

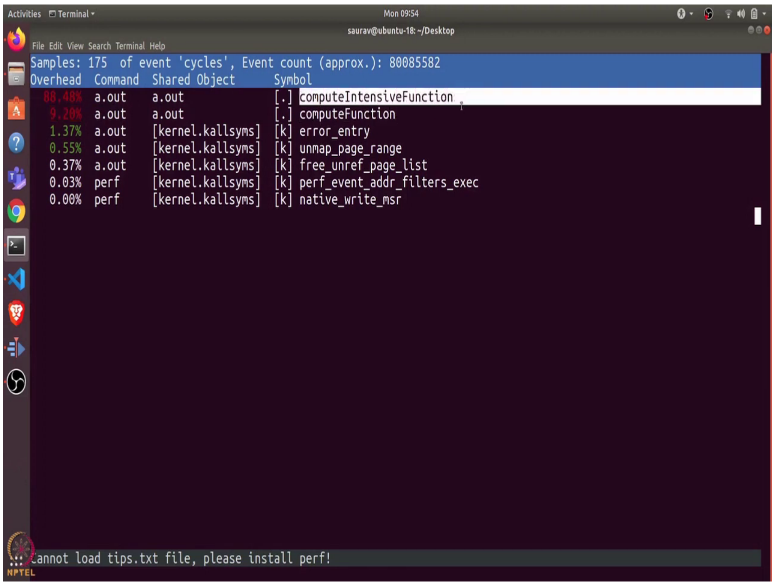
```
Samples: 175  of event 'cycles', Event count (approx.): 80085582
Overhead  Command  Shared Object      Symbol
  88.48%  a.out    a.out              [.] computeIntensiveFunction
   9.20%  a.out    a.out              [.] computeFunction
   1.37%  a.out    [kernel.kallsyms]  [k] error_entry
   0.55%  a.out    [kernel.kallsyms]  [k] unmap_page_range
   0.37%  a.out    [kernel.kallsyms]  [k] free_unref_page_list
   0.03%  perf     [kernel.kallsyms]  [k] perf_event_addr_filters_exec
   0.00%  perf     [kernel.kallsyms]  [k] native_write_msr




Cannot load tips.txt file, please install perf!
```



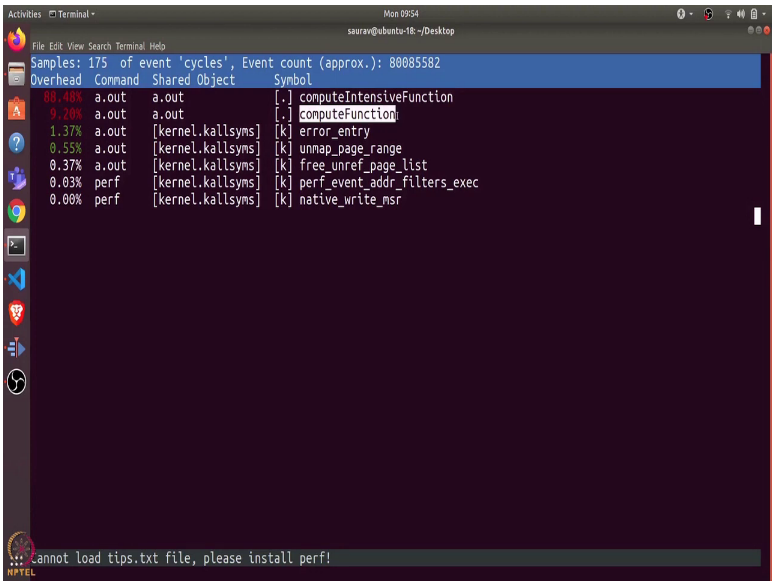
```
Samples: 175  of event 'cycles', Event count (approx.): 80085582
Overhead  Command  Shared Object      Symbol
  88.48%  a.out    a.out              [.] computeIntensiveFunction
   9.20%  a.out    a.out              [.] computeFunction
   1.37%  a.out    [kernel.kallsyms]  [k] error_entry
   0.55%  a.out    [kernel.kallsyms]  [k] unmap_page_range
   0.37%  a.out    [kernel.kallsyms]  [k] free_unref_page_list
   0.03%  perf     [kernel.kallsyms]  [k] perf_event_addr_filters_exec
   0.00%  perf     [kernel.kallsyms]  [k] native_write_msr




Cannot load tips.txt file, please install perf!
```

So, here it shows us the percentage of CPU that is used for various functions and the command that is a.out, and then the function name. Here we can see that almost 88 percent of CPU is used only in computeIntensiveFunction, and almost 9 percent is used in the computeFunction.

(Refer Slide Time: 5:46)


Activities
Terminal
Mon 09:54
saurav@ubuntu-18: ~/Desktop
File Edit View Search Terminal Help
NPTEL

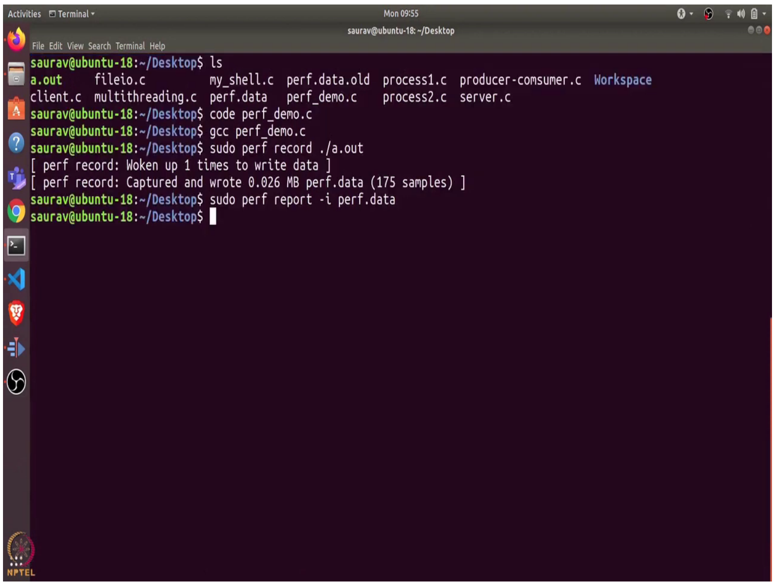
```
Samples: 175  of event 'cycles', Event count (approx.): 80085582
Overhead  Command  Shared Object      Symbol
  88.48%  a.out    a.out              [.] computeIntensiveFunction
   9.20%  a.out    a.out              [.] computeFunction
   1.37%  a.out    [kernel.kallsyms]  [k] error_entry
   0.55%  a.out    [kernel.kallsyms]  [k] unmap_page_range
   0.37%  a.out    [kernel.kallsyms]  [k] free_unref_page_list
   0.03%  perf     [kernel.kallsyms]  [k] perf_event_addr_filters_exec
   0.00%  perf     [kernel.kallsyms]  [k] native_write_msr

Cannot load tips.txt file, please install perf!
```


Activities
Terminal
Mon 09:54
saurav@ubuntu-18: ~/Desktop
File Edit View Search Terminal Help
NPTEL

```
Samples: 175  of event 'cycles', Event count (approx.): 80085582
Overhead  Command  Shared Object      Symbol
  88.48%  a.out    a.out              [.] computeIntensiveFunction
   9.20%  a.out    a.out              [.] computeFunction
   1.37%  a.out    [kernel.kallsyms]  [k] error_entry
   0.55%  a.out    [kernel.kallsyms]  [k] unmap_page_range
   0.37%  a.out    [kernel.kallsyms]  [k] free_unref_page_list
   0.03%  perf     [kernel.kallsyms]  [k] perf_event_addr_filters_exec
   0.00%  perf     [kernel.kallsyms]  [k] native_write_msr

Cannot load tips.txt file, please install perf!
```

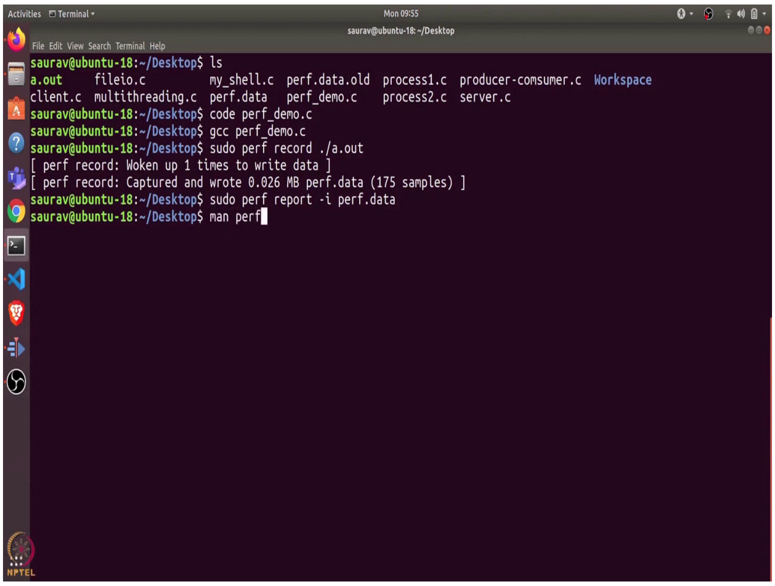
So, computeIntensiFunction is almost 10 times more expensive than compute function. And this also shows us what is the bottleneck of our program that is the computeIntensiveFunction.

(Refer Slide Time: 5:55)



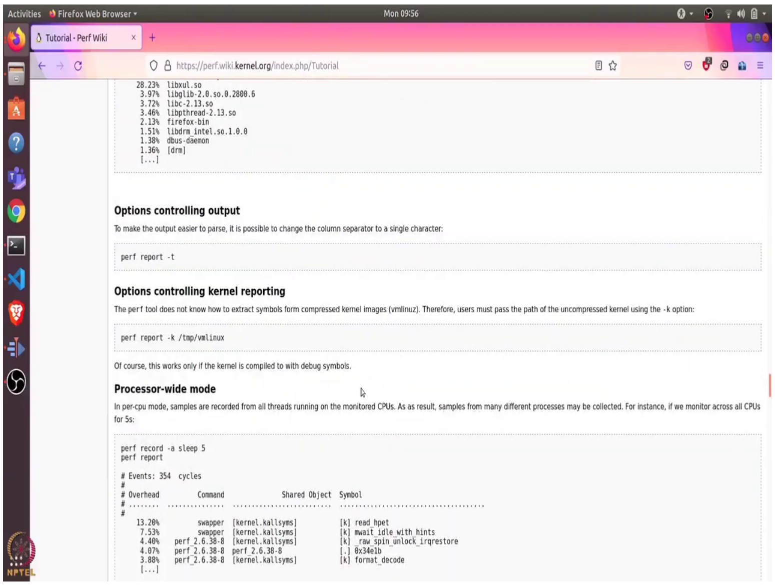
Activities
Terminal
Mon 09:55
saurav@ubuntu-18: ~/Desktop
File Edit View Search Terminal Help
NPTEL

```
saurav@ubuntu-18:~/Desktop$ ls
a.out     fileio.c         my_shell.c  perf.data.old  process1.c  producer-comsumer.c  Workspace
client.c  multithreading.c  perf.data   perf_demo.c    process2.c  server.c
saurav@ubuntu-18:~/Desktop$ code perf_demo.c
saurav@ubuntu-18:~/Desktop$ gcc perf_demo.c
saurav@ubuntu-18:~/Desktop$ sudo perf record ./a.out
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.026 MB perf.data (175 samples) ]
saurav@ubuntu-18:~/Desktop$ sudo perf report -i perf.data
saurav@ubuntu-18:~/Desktop$ 
```



Activities
Terminal
Mon 09:55
saurav@ubuntu-18: ~/Desktop
File Edit View Search Terminal Help
NPTEL

```
saurav@ubuntu-18:~/Desktop$ ls
a.out     fileio.c         my_shell.c  perf.data.old  process1.c  producer-comsumer.c  Workspace
client.c  multithreading.c  perf.data   perf_demo.c    process2.c  server.c
saurav@ubuntu-18:~/Desktop$ code perf_demo.c
saurav@ubuntu-18:~/Desktop$ gcc perf_demo.c
saurav@ubuntu-18:~/Desktop$ sudo perf record ./a.out
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.026 MB perf.data (175 samples) ]
saurav@ubuntu-18:~/Desktop$ sudo perf report -i perf.data
saurav@ubuntu-18:~/Desktop$ man perf
```



```
    28.23%  libxul.so
     3.97%  libglib-2.0.so.0.2800.6
     3.72%  libc-2.13.so
     3.46%  libpthread-2.13.so
     2.13%  firefox-bin
     1.51%  libdrm_intel.so.1.0.0
     1.38%  dbus-daemon
     1.36%  [drm]
     [...]
```

Options controlling output

To make the output easier to parse, it is possible to change the column separator to a single character:

```
perf report -t
```

Options controlling kernel reporting

The perf tool does not know how to extract symbols form compressed kernel images (vmlinuz). Therefore, users must pass the path of the uncompressed kernel using the -k option:

```
perf report -k /tmp/vmlinux
```

Of course, this works only if the kernel is compiled to with debug symbols.

Processor-wide mode

In per-cpu mode, samples are recorded from all threads running on the monitored CPUs. As as result, samples from many different processes may be collected. For instance, if we monitor across all CPUs for 5s:

```
perf record -a sleep 5
perf report

# Events: 354  cycles
#
# Overhead          Command             Shared Object  Symbol
# ........  ...............  ..........................  ......................................
#
    13.20%          swapper  [kernel.kallsyms]          [k] read_hpet
     7.53%          swapper  [kernel.kallsyms]          [k] mwait_idle_with_hints
     4.40%    perf_2.6.38-8  [kernel.kallsyms]          [k] _raw_spin_unlock_irqrestore
     4.07%    perf_2.6.38-8  perf_2.6.38-8              [.] 0x34e1b
     3.88%    perf_2.6.38-8  [kernel.kallsyms]          [k] format_decode
     [...]
```

So, that is how we can profile various C programs and try to optimize their bottlenecks. If you want to see more details about perf then you can have a look at the man page of perf. And also there is a very nice tutorial on perf which you can find online. So, this explains various commands of perf, and what else you can do using perf. So, that is it for this video. Thanks and have a nice day.