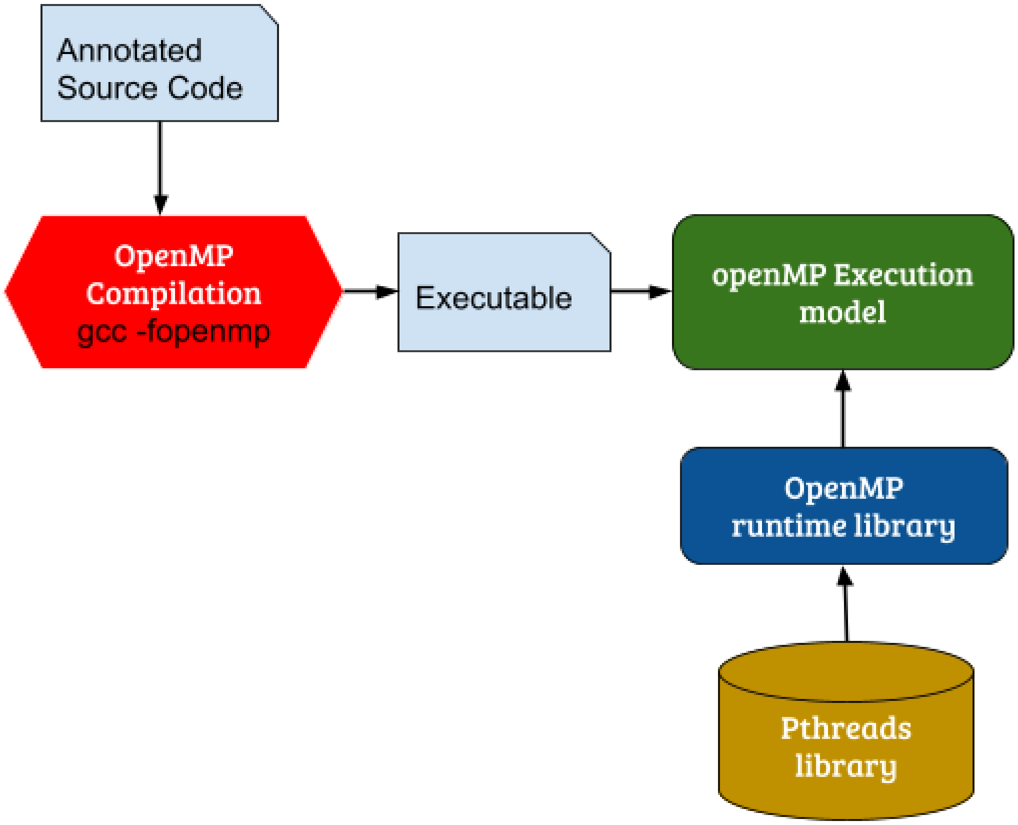
Web Links

1. https://www.glennklockwood.com/hpc-howtos/process-affinity.html
2. https://redirect.cs.umbc.edu/~tsimo1/CMSC483/cs220/code.html
3. https://hps.vi4io.org/teaching/summer_term_2022/pchpc
4. https://github.com/PawanKL/Pthread-vs-OpenMP
5. **https://en.algorithmica.org/hpc/algorithms/matmul/**
6. **https://www.mgaillard.fr/2020/08/29/matrix-multiplication-optimizing.html**
7.

OpenMP Compilation and Execution Structure



OpenMP vs Pthreads

Managing Process Affinity in Linux

(Thread Binding to core)

What is Affinity(Thread binding to core)?
Ans : It schedule the thread to specific core

How to check the thread running on which core?
Ans: Assuming your executable is called application.x, you can easily see what cores each thread is using by issuing the following command in bash:

```
$ for i in $(pgrep application.x); do ps -mo pid,tid,fname,user,psr -p $i;done
```

The PSR field is the OS identifier for the core each TID (thread id) is utilizing. It return thread number.

+++

OpenMP Runtime Extensions

Set and export the KMP_AFFINITY env variable to express binding preferences. KMP_AFFINITY has three principal binding strategies:

- *compact* fills up one socket before allocating to other sockets
  - $export KMP_AFFINITY = compact
- *scatter* evenly spreads threads across all sockets and cores
  - $export KMP_AFFINITY = scatter
- *explicit* allows you define exactly which cores/sockets to use
  - $export KMP_AFFINITY='proclist=[0,2,4,6],explicit'

GNU's implementation of OpenMP has a environment variable similar to KMP_AFFINITY called GOMP_CPU_AFFINITY.

$export GOMP_CPU_AFFINITY='0,2,4,6'

+++

Matrix Multiplication

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

#define N 5000

int main()
{
long A[N][N],B[N][N],mul[N][N],r,C,i,j,k,cnt = 1;

    /*omp_set_num_threads(1);*/
    printf("enter the first matrix element\n");
        for(i=0;i<N;i++)
            {
                for(j=0;j<N;j++)
                    {
                        A[i][j] = 1;
                    }
            }
    cnt = 1;
    printf("enter the second matrix element\n");
        for(i=0;i<N;i++)
            {
                for(j=0;j<N;j++)
                    {
                        B[i][j] = 1;
                    }
            }
    printf("enter the mul matrix element\n");
        for(i=0;i<N;i++)
            {
                for(j=0;j<N;j++)
                    {
                        mul[i][j] = 0;
                    }
            }

  #pragma omp parallel for private(i,j,k) shared(A,B,C)
  for (i = 0; i < N; ++i) {
    for (j = 0; j < N; ++j) {
      for (k = 0; k < N; ++k) {
        mul[i][j] += A[i][k] * B[k][j];
      }
    }
    }
}
```

How to compile: $gcc -fopenmp matrix.c -o matrix

Set number of threads: $export OMP_NUM_THREADS=<num>

How to run: $./matrix

+++

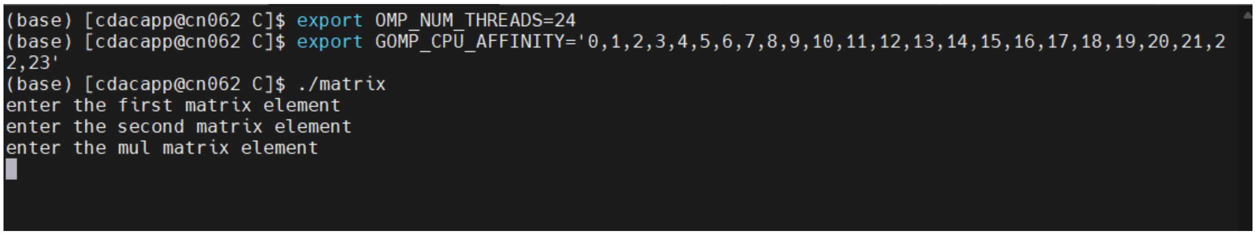
Test 1 : All cores of one socket.

Note : In my one socket there are 24 cores are available.

Open Three terminal

Terminal 1:

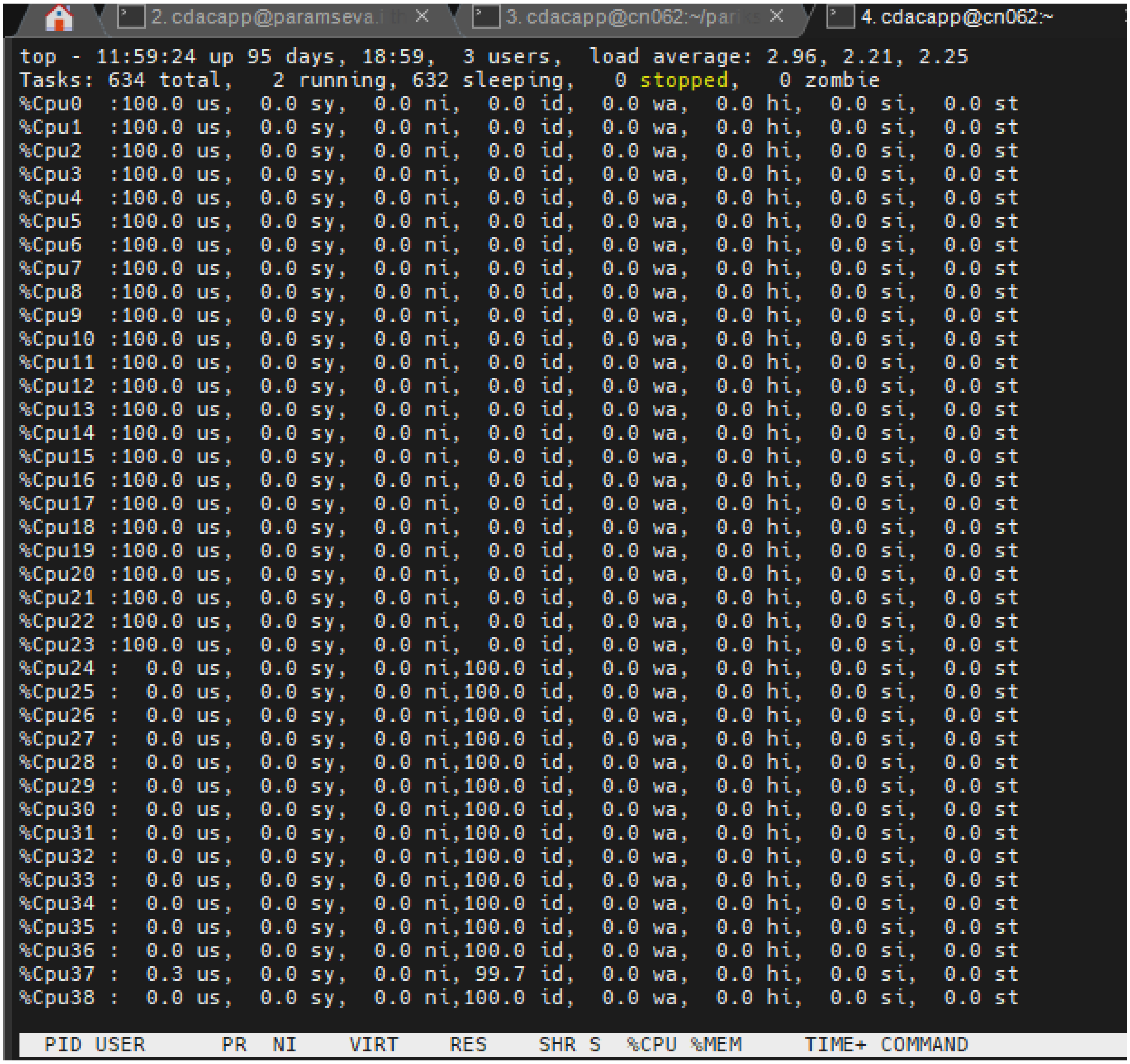
- Set the number of threads:
  - $export OMP_NUM_THREADS=24
- Set the AFFINITY (which core to use)
  - $export GOMP_CPU_AFFINITY='0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23'
- Run the program
  - $./matrix



```
(base) [cdacapp@cn062 C]$ export OMP_NUM_THREADS=24
(base) [cdacapp@cn062 C]$ export GOMP_CPU_AFFINITY='0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,2
2,23'
(base) [cdacapp@cn062 C]$ ./matrix
enter the first matrix element
enter the second matrix element
enter the mul matrix element
```

Terminal 2:

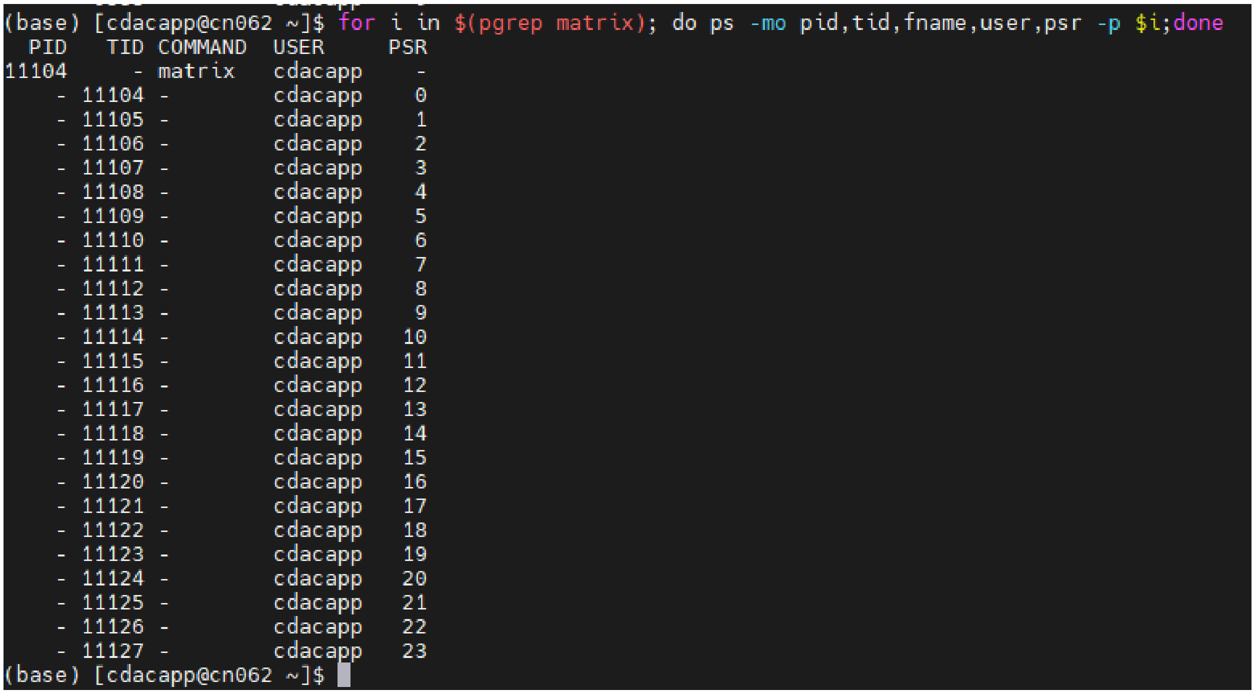
- Check the cpu utilization
  - $top
  - Press 1 after that



```
top - 11:59:24 up 95 days, 18:59,  3 users,  load average: 2.96, 2.21, 2.25
Tasks: 634 total,   2 running, 632 sleeping,   0 stopped,   0 zombie
%Cpu0  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu4  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu5  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu6  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu7  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu8  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu9  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu10 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu11 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu12 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu13 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu14 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu15 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu16 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu17 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu18 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu19 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu20 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu21 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu22 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu23 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu24 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu25 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu26 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu27 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu28 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu29 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu30 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu31 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu32 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu33 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu34 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu35 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu36 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu37 :  0.3 us,  0.0 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu38 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
```

Terminal 3:

- Check the process and thread id
  - **$ for i in $(pgrep matrix); do ps -mo pid,tid,fname,user,psr -p $i;done**



```
(base) [cdacapp@cn062 ~]$ for i in $(pgrep matrix); do ps -mo pid,tid,fname,user,psr -p $i;done
  PID   TID COMMAND  USER     PSR
11104     - matrix   cdacapp    -
    - 11104 -        cdacapp    0
    - 11105 -        cdacapp    1
    - 11106 -        cdacapp    2
    - 11107 -        cdacapp    3
    - 11108 -        cdacapp    4
    - 11109 -        cdacapp    5
    - 11110 -        cdacapp    6
    - 11111 -        cdacapp    7
    - 11112 -        cdacapp    8
    - 11113 -        cdacapp    9
    - 11114 -        cdacapp   10
    - 11115 -        cdacapp   11
    - 11116 -        cdacapp   12
    - 11117 -        cdacapp   13
    - 11118 -        cdacapp   14
    - 11119 -        cdacapp   15
    - 11120 -        cdacapp   16
    - 11121 -        cdacapp   17
    - 11122 -        cdacapp   18
    - 11123 -        cdacapp   19
    - 11124 -        cdacapp   20
    - 11125 -        cdacapp   21
    - 11126 -        cdacapp   22
    - 11127 -        cdacapp   23
(base) [cdacapp@cn062 ~]$
```

++

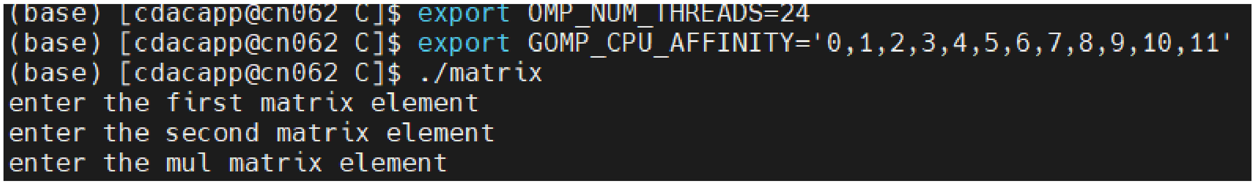
Test 2 : Two threads on one core.

Note : In my one socket there are 24 cores available.

Open Three terminal

Terminal 1:

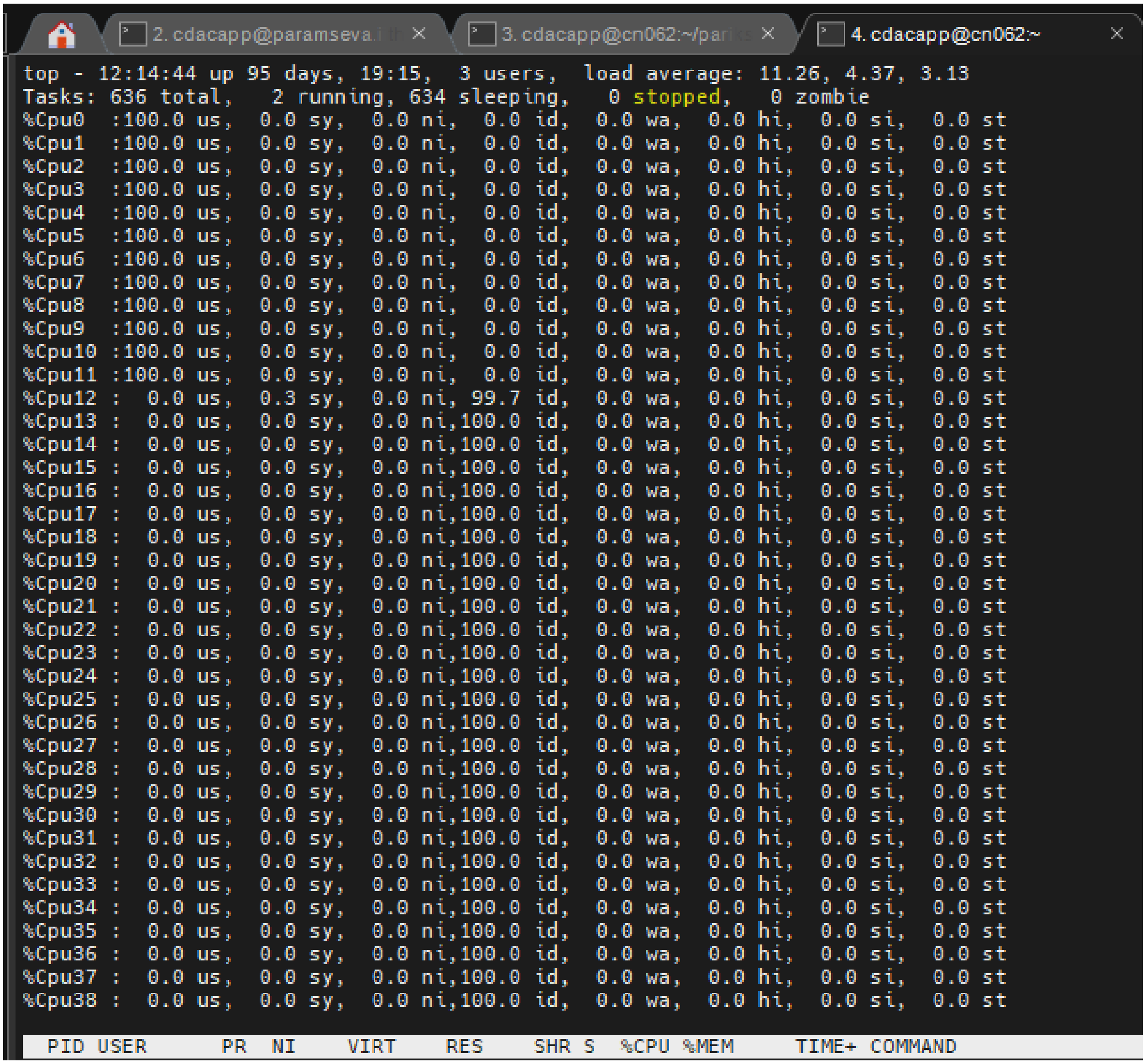
- Set the number of threads:
  - $export OMP_NUM_THREADS=24
- Set the AFFINITY (which core to use)
  - $export GOMP_CPU_AFFINITY='0,1,2,3,4,5,6,7,8,9,10,11'
- Run the program
  - $./matrix



```
(base) [cdacapp@cn062 C]$ export OMP_NUM_THREADS=24
(base) [cdacapp@cn062 C]$ export GOMP_CPU_AFFINITY='0,1,2,3,4,5,6,7,8,9,10,11'
(base) [cdacapp@cn062 C]$ ./matrix
enter the first matrix element
enter the second matrix element
enter the mul matrix element
```

Terminal 2:

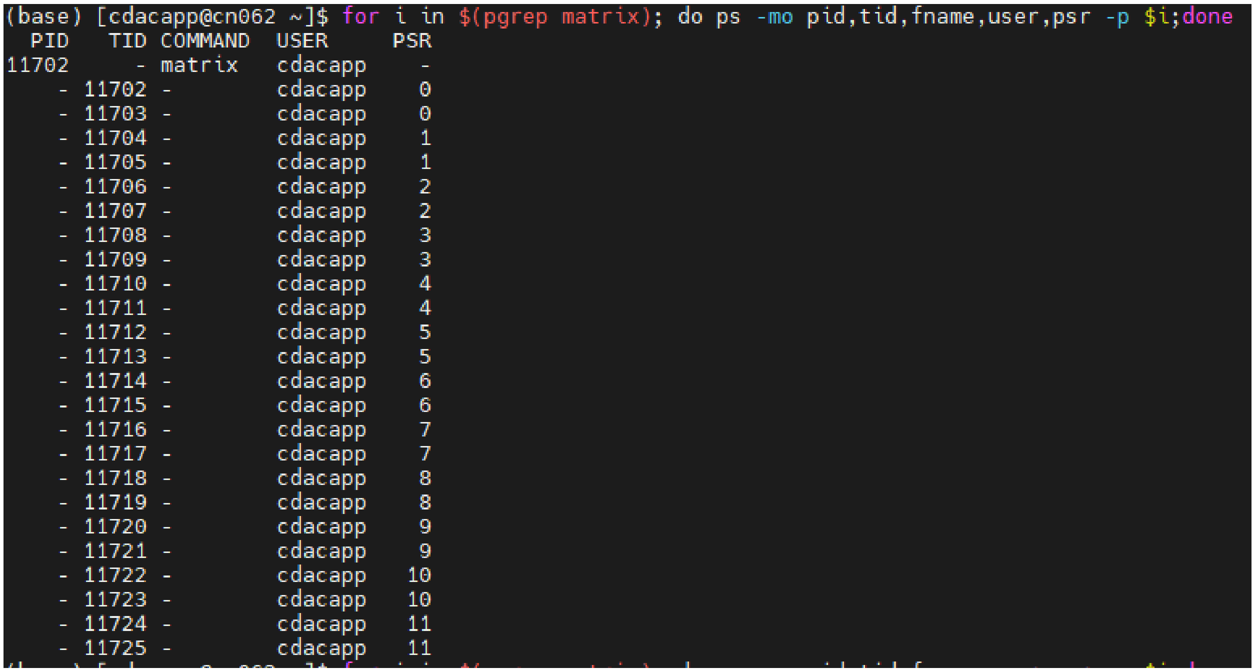
- Check the cpu utilization
  - $top
  - Press 1 after that



```
top - 12:14:44 up 95 days, 19:15,  3 users,  load average: 11.26, 4.37, 3.13
Tasks: 636 total,   2 running, 634 sleeping,   0 stopped,   0 zombie
%Cpu0  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu2  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu3  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu4  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu5  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu6  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu7  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu8  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu9  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu10 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu11 :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu12 :  0.0 us,  0.3 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu13 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu14 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu15 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu16 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu17 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu18 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu19 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu20 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu21 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu22 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu23 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu24 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu25 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu26 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu27 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu28 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu29 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu30 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu31 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu32 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu33 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu34 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu35 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu36 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu37 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu38 :  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
```

Terminal 3:

- Check the process and thread id
  - **$ for i in $(pgrep matrix); do ps -mo pid,tid,fname,user,psr -p $i;done**



```
(base) [cdacapp@cn062 ~]$ for i in $(pgrep matrix); do ps -mo pid,tid,fname,user,psr -p $i;done
  PID   TID COMMAND  USER     PSR
11702     - matrix   cdacapp    -
    - 11702 -        cdacapp    0
    - 11703 -        cdacapp    0
    - 11704 -        cdacapp    1
    - 11705 -        cdacapp    1
    - 11706 -        cdacapp    2
    - 11707 -        cdacapp    2
    - 11708 -        cdacapp    3
    - 11709 -        cdacapp    3
    - 11710 -        cdacapp    4
    - 11711 -        cdacapp    4
    - 11712 -        cdacapp    5
    - 11713 -        cdacapp    5
    - 11714 -        cdacapp    6
    - 11715 -        cdacapp    6
    - 11716 -        cdacapp    7
    - 11717 -        cdacapp    7
    - 11718 -        cdacapp    8
    - 11719 -        cdacapp    8
    - 11720 -        cdacapp    9
    - 11721 -        cdacapp    9
    - 11722 -        cdacapp   10
    - 11723 -        cdacapp   10
    - 11724 -        cdacapp   11
    - 11725 -        cdacapp   11
```

+++

Note: Get the mapping of core and socket

$cat /proc/cpuinfo | grep -e processor -e physical | grep -vi address

```
processor	: 0
physical id	: 0
processor	: 1
physical id	: 0
processor	: 2
physical id	: 0
processor	: 3
physical id	: 0
processor	: 4
physical id	: 0
processor	: 5
physical id	: 0
processor	: 6
physical id	: 0
processor	: 7
physical id	: 0
processor	: 8
physical id	: 0
processor	: 9
physical id	: 0
processor	: 10
physical id	: 0
```

```
processor       : 11
physical id     : 0
processor       : 12
physical id     : 0
processor       : 13
physical id     : 0
processor       : 14
physical id     : 0
processor       : 15
physical id     : 0
processor       : 16
physical id     : 0
processor       : 17
physical id     : 0
processor       : 18
physical id     : 0
processor       : 19
physical id     : 0
processor       : 20
physical id     : 0
processor       : 21
physical id     : 0
processor       : 22
physical id     : 0
processor       : 23
physical id     : 0
processor       : 24
physical id     : 1
processor       : 25
physical id     : 1
processor       : 26
physical id     : 1
processor       : 27
physical id     : 1
processor       : 28
physical id     : 1
processor       : 29
physical id     : 1
processor       : 30
physical id     : 1
processor       : 31
physical id     : 1
processor       : 32
physical id     : 1
```

```
processor	: 33
physical id	: 1
processor	: 34
physical id	: 1
processor	: 35
physical id	: 1
processor	: 36
physical id	: 1
processor	: 37
physical id	: 1
processor	: 38
physical id	: 1
processor	: 39
physical id	: 1
processor	: 40
physical id	: 1
processor	: 41
physical id	: 1
processor	: 42
physical id	: 1
processor	: 43
physical id	: 1
processor	: 44
physical id	: 1
processor	: 45
physical id	: 1
processor	: 46
physical id	: 1
processor	: 47
physical id	: 1
```

+++

+++

Matrix multiplication program for Uniprocessor optimization

===

```
/*Navie Matrix Matrix Multiplication */
#include <iostream>
#include <bits/stdc++.h>
#include <sys/time.h>
using namespace std;
#define n 4096
double A[n][n], B[n][n], C[n][n];
```

```
int main() {
    // Initialize Matrices
        for(int i = 0; i < n; ++i)
        for(int j = 0; j < n; ++j)
        {
            A[i][j] = (double)rand()/ (double)RAND_MAX;
                B[i][j] = (double)rand()/ (double)RAND_MAX;
                C[i][j] = 0;
        }

    struct timeval start, end;
      // start timer.
    gettimeofday(&start, NULL);
      // unsync the I/O of C and C++.
    ios_base::sync_with_stdio(false);

    // Matrix multiplication
    for(int i = 0; i < n; ++i)
        for(int j = 0; j < n; ++j)
            for(int k = 0; k < n; ++k)
            {
                C[i][j] += A[i][k] * B[k][j];
            }

    gettimeofday(&end, NULL);

    // Calculating total time taken by the program.
    double time_taken;

    time_taken = (end.tv_sec - start.tv_sec) * 1e6;
    time_taken = (time_taken + (end.tv_usec -
                              start.tv_usec)) * 1e-6;

    cout << "Time taken by program is : " << fixed
         << time_taken << setprecision(6);
    cout << " sec" << endl;
    return 0;
}
```

==

Execution command

```
==============================================================
==============================================================
// Loop Order naive matrix-matrix multiplication //
#include <iostream>
#include <bits/stdc++.h>
#include <sys/time.h>
using namespace std;
#define n 4096
double A[n][n], B[n][n], C[n][n];

int main() {

   // Initialize Matrices
   for(int i = 0; i < n; ++i)
      for(int j = 0; j < n; ++j)
      {
         A[i][j] = (double)rand()/ (double)RAND_MAX;
            B[i][j] = (double)rand()/ (double)RAND_MAX;
            C[i][j] = 0;
      }

   struct timeval start, end;
   // start timer.
   gettimeofday(&start, NULL);

   // unsync the I/O of C and C++.
   ios_base::sync_with_stdio(false);

   // Matrix multiplication
   for(int i = 0; i < n; ++i) {
      for(int k = 0; k < n; ++k) {
    for(int j = 0; j < n; ++j)
            {
            C[i][j] += A[i][k] * B[k][j];
         }
         }
   }

   gettimeofday(&end, NULL);

   // Calculating total time taken by the program.
   double time_taken;

   time_taken = (end.tv_sec - start.tv_sec) * 1e6;
   time_taken = (time_taken + (end.tv_usec -
                             start.tv_usec)) * 1e-6;

   cout << "Time taken by program is : " << fixed
        << time_taken << setprecision(6);
   cout << " sec" << endl;

   return 0;
}
==============================================================================
```

Execution :

```
==============================================================================
==============================================================================
//Matrix Matrix Multiplication with OpenMP//
#include <iostream>
#include <omp.h>
#include <bits/stdc++.h>
#include <sys/time.h>

using namespace std;

#define n 4096

double A[n][n], B[n][n], C[n][n];

int main() {


    // Initialize Matrices

    for(int i = 0; i < n; ++i)
```

```
        for(int j = 0; j < n; ++j)
        {
            A[i][j] = (double)rand()/ (double)RAND_MAX;
                B[i][j] = (double)rand()/ (double)RAND_MAX;
                C[i][j] = 0;
        }

    // Matrix multiplication

    int i,j,k;

    struct timeval start, end;

    // start timer.
    gettimeofday(&start, NULL);

    // unsync the I/O of C and C++.
    ios_base::sync_with_stdio(false);

    #pragma omp parallel for private(i,j,k) shared(A,B,C)
    for(i = 0; i < n; ++i) {
        for(int k = 0; k < n; ++k) {
      for(j = 0; j < n; ++j) {
                C[i][j] += A[i][k] * B[k][j];
                }
            }
    }

    gettimeofday(&end, NULL);

    double time_taken;

    time_taken = (end.tv_sec - start.tv_sec) * 1e6;
    time_taken = (time_taken + (end.tv_usec -
                               start.tv_usec)) * 1e-6;

    cout << "Time taken by program is : " << fixed
         << time_taken << setprecision(6);
    cout << " sec" << endl;

    return 0;
}
```

Matrix Multiplication in C : Loops interchange

References

https://sites.cs.ucsb.edu/~tyang/class/240a16w/hw1/cache/matrixMultiply.c

Combined (O0, O1, O2, O3 and Ofast)

1000 x 1000	O0	O1	O2	O3	O3 mavx	O3 mavx 2	Ofast	Ofast mavx	Ofast mavx 2
k,i,j	4.47s	1.29s	1.28s	1.29s			1.28s		
i,k,j	4.44s	1.40s	1.28s	1.28s			1.49s		
j,i,k	4.04s	2.53s	1.13s	1.134			1.18s		
k,j,i	3.39s	0.48s	0.48s	0.17s			0.17s		
i,j,k	3.85s	2.76s	1.22s	1.17s	1.17s	1.31s	1.25s	1.31s	1.33s
i,j,k	With -ftree-vectorize				1.33s	1.24s		1.21s	1.31s
j,k,i	3.31s	0.57s	0.48s	0.16s			0.17s		

5000x5000	O0	O1	O2	O3	O3 mavx	O3 mavx 2	Ofast	Ofast mavx	Ofast mavx 2
i,j,k	474.22s	329.81s	149.98s	149.94s	150.07s	150.03s	150.48s	149.67s	149.88s

Default optimization level : O0

Loop (1000x1000 matrix)	Time	GFlops
k,i,j	4.476s	0.447 Gflop/s
i,k,j	4.449s	0.450 Gflop/s
j,i,k	4.049s	0.494 Gflop/s
k,j,i	3.395s	0.589 Gflop/s
i,j,k	3.857s	0.517 Gflop/s
j,k,i	3.310s	0.604 Gflop/s

// gcc -O1 <file_name> -o <output_file_name>

Optimization level : O1

Loop (1000x1000 matrix)	Time	GFlops
k,i,j	1.290s	1.550 Gflop/s
i,k,j	1.404s	1.425 Gflop/s
j,i,k	2.536s	0.789 Gflop/s
k,j,i	0.488s	4.102 Gflop/s
i,j,k	2.767s	0.723 Gflop/s
j,k,i	0.579s	3.454 Gflop/s

Optimization level : O2

Loop (1000x1000 matrix)	Time	GFlops
k,i,j	1.284s	1.557 Gflop/s
i,k,j	1.286s	1.555 Gflop/s

j,i,k	1.134s	1.763 Gflop/s
k,j,i	0.489s	4.088 Gflop/s
i,j,k	1.226s	1.632 Gflop/s
j,k,i	0.481s	4.159 Gflop/s

Optimization level : O3

Loop (1000x1000 matrix)	Time	GFlops
k,i,j	1.290s	1.551 Gflop/s
i,k,j	1.287s	1.554 Gflop/s
j,i,k	1.134s	1.763 Gflop/s
k,j,i	0.172s	11.631 Gflop/s
i,j,k	1.175s	1.702 Gflop/s
j,k,i	0.169s	11.852 Gflop/s

Optimization level : Ofast

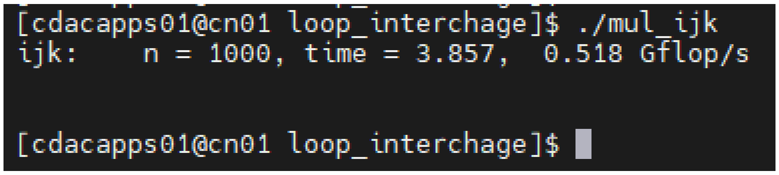
Loop (1000x1000 matrix)	Time	GFlops
k,i,j	1.287s	1.554 Gflop/s
i,k,j	1.497s	1.336 Gflop/s
j,i,k	1.188s	1.684 Gflop/s
k,j,i	0.175s	11.459 Gflop/s
i,j,k	1.258s	1.590 Gflop/s
j,k,i	0.174s	11.487 Gflop/s

Programs

1. **i,j,k loop**

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>

void multMat1( int n, float *A, float *B, float *C ) {
    int i,j,k;
    /* This is ijk loop order. */
    for( i = 0; i < n; i++ )
        for( j = 0; j < n; j++ )
            for( k = 0; k < n; k++ )
                C[i+j*n] += A[i+k*n]*B[k+j*n];
}

int main( int argc, char **argv ) {
    int nmax = 1000,i;

    void (*orderings[])(int,float *,float *,float *) = {&multMat1};
    char *names[] = {"ijk"};

    float *A = (float *)malloc( nmax*nmax * sizeof(float));
    float *B = (float *)malloc( nmax*nmax * sizeof(float));
    float *C = (float *)malloc( nmax*nmax * sizeof(float));

    struct timeval start, end;

    /* fill matrices with random numbers */
    for( i = 0; i < nmax*nmax; i++ ) A[i] = drand48()*2-1;
    for( i = 0; i < nmax*nmax; i++ ) B[i] = drand48()*2-1;
    for( i = 0; i < nmax*nmax; i++ ) C[i] = drand48()*2-1;

        gettimeofday( &start, NULL );
        (*orderings[0])( nmax, A, B, C );
        gettimeofday( &end, NULL );

        /* convert time to Gflop/s */
        double seconds = (end.tv_sec - start.tv_sec) +
            1.0e-6 * (end.tv_usec - start.tv_usec);
        double Gflops = 2e-9*nmax*nmax*nmax/seconds;
        printf( "%s:\tn = %d, time = %.3f,  %.3f Gflop/s\n",
names[0], nmax, seconds, Gflops );

    free( A );
    free( B );
    free( C );

    printf("\n\n");

    return 0;
}
```

```
gcc mul_ijk.c -o mul_ijk
./mul_ijk
```

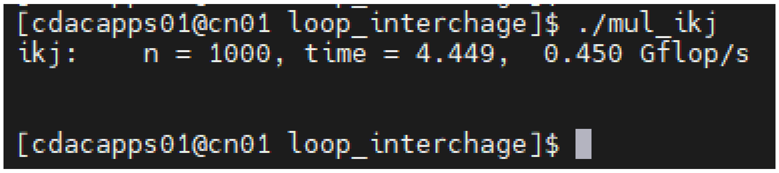


```
[cdacapps01@cn01 loop_interchage]$ ./mul_ijk
ijk:    n = 1000, time = 3.857,  0.518 Gflop/s


[cdacapps01@cn01 loop_interchage]$ 
```

2. i,k,j loop

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>

void multMat1( int n, float *A, float *B, float *C ) {
    int i,j,k;
    /* This is ikj loop order. */
    for( i = 0; i < n; i++ )
        for( k = 0; k < n; k++ )
            for( j = 0; j < n; j++ )
                C[i+j*n] += A[i+k*n]*B[k+j*n];
}


int main( int argc, char **argv ) {
    int nmax = 1000,i;

    void (*orderings[])(int,float *,float *,float *) = {&multMat1};
    char *names[] = {"ijk"};

    float *A = (float *)malloc( nmax*nmax * sizeof(float));
    float *B = (float *)malloc( nmax*nmax * sizeof(float));
    float *C = (float *)malloc( nmax*nmax * sizeof(float));

    struct timeval start, end;
    /* fill matrices with random numbers */
    for( i = 0; i < nmax*nmax; i++ ) A[i] = drand48()*2-1;
    for( i = 0; i < nmax*nmax; i++ ) B[i] = drand48()*2-1;
    for( i = 0; i < nmax*nmax; i++ ) C[i] = drand48()*2-1;

        gettimeofday( &start, NULL );
        (*orderings[0])( nmax, A, B, C );
        gettimeofday( &end, NULL );

        /* convert time to Gflop/s */
        double seconds = (end.tv_sec - start.tv_sec) +
            1.0e-6 * (end.tv_usec - start.tv_usec);
        double Gflops = 2e-9*nmax*nmax*nmax/seconds;
        printf( "%s:\tn = %d, time = %.3f,  %.3f Gflop/s\n",
names[0], nmax, seconds, Gflops );

    free( A );
    free( B );
    free( C );

    printf("\n\n");

    return 0;
}
```

gcc mul_ikj.c -o mul_ikj
./mul_ikj



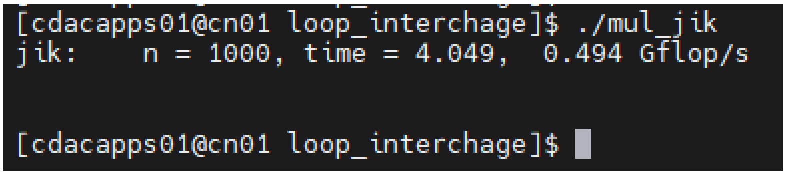
```
[cdacapps01@cn01 loop_interchage]$ ./mul_ikj
ikj:    n = 1000, time = 4.449,  0.450 Gflop/s


[cdacapps01@cn01 loop_interchage]$ 
```

3. j,i,k look

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>

void multMat1( int n, float *A, float *B, float *C ) {
    int i,j,k;
    /* This is jik loop order. */
    for( j = 0; j < n; j++ )
        for( i = 0; i < n; i++ )
            for( k = 0; k < n; k++ )
                C[i+j*n] += A[i+k*n]*B[k+j*n];
}

int main( int argc, char **argv ) {
    int nmax = 1000,i;

    void (*orderings[])(int,float *,float *,float *) = {&multMat1};
    char *names[] = {"jik"};

    float *A = (float *)malloc( nmax*nmax * sizeof(float));
    float *B = (float *)malloc( nmax*nmax * sizeof(float));
    float *C = (float *)malloc( nmax*nmax * sizeof(float));

    struct timeval start, end;

    /* fill matrices with random numbers */
    for( i = 0; i < nmax*nmax; i++ ) A[i] = drand48()*2-1;
    for( i = 0; i < nmax*nmax; i++ ) B[i] = drand48()*2-1;
    for( i = 0; i < nmax*nmax; i++ ) C[i] = drand48()*2-1;

        gettimeofday( &start, NULL );
        (*orderings[0])( nmax, A, B, C );
        gettimeofday( &end, NULL );

        /* convert time to Gflop/s */
        double seconds = (end.tv_sec - start.tv_sec) +
            1.0e-6 * (end.tv_usec - start.tv_usec);
        double Gflops = 2e-9*nmax*nmax*nmax/seconds;
        printf( "%s:\tn = %d, time = %.3f,  %.3f Gflop/s\n",
names[0], nmax, seconds, Gflops );

    free( A );
    free( B );
    free( C );

    printf("\n\n");

    return 0;
}
```

```
gcc mul_jik.c -o mul_jik
```

./mul_jik

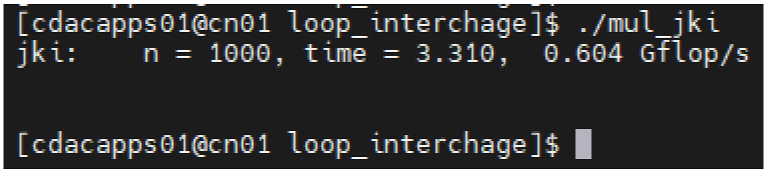


```
[cdacapps01@cn01 loop_interchage]$ ./mul_jik
jik:    n = 1000, time = 4.049,  0.494 Gflop/s

[cdacapps01@cn01 loop_interchage]$ 
```

4. j,k,i loop

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>

void multMat1( int n, float *A, float *B, float *C ) {
    int i,j,k;
    /* This is jki loop order. */
    for( j = 0; j < n; j++ )
        for( k = 0; k < n; k++ )
            for( i = 0; i < n; i++ )
                C[i+j*n] += A[i+k*n]*B[k+j*n];
}

int main( int argc, char **argv ) {
    int nmax = 1000,i;

    void (*orderings[])(int,float *,float *,float *) = {&multMat1};
    char *names[] = {"jki"};

    float *A = (float *)malloc( nmax*nmax * sizeof(float));
    float *B = (float *)malloc( nmax*nmax * sizeof(float));
    float *C = (float *)malloc( nmax*nmax * sizeof(float));

    struct timeval start, end;

    /* fill matrices with random numbers */
    for( i = 0; i < nmax*nmax; i++ ) A[i] = drand48()*2-1;
    for( i = 0; i < nmax*nmax; i++ ) B[i] = drand48()*2-1;
    for( i = 0; i < nmax*nmax; i++ ) C[i] = drand48()*2-1;

        gettimeofday( &start, NULL );
        (*orderings[0])( nmax, A, B, C );
        gettimeofday( &end, NULL );
        /* convert time to Gflop/s */
        double seconds = (end.tv_sec - start.tv_sec) +
            1.0e-6 * (end.tv_usec - start.tv_usec);
        double Gflops = 2e-9*nmax*nmax*nmax/seconds;
        printf( "%s:\tn = %d, time = %.3f,  %.3f Gflop/s\n",
names[0], nmax, seconds, Gflops );

    free( A );
    free( B );
    free( C );

    printf("\n\n");

    return 0;
}
```

```
gcc mul_jki.c -o mul_jki
./mul_jki
```

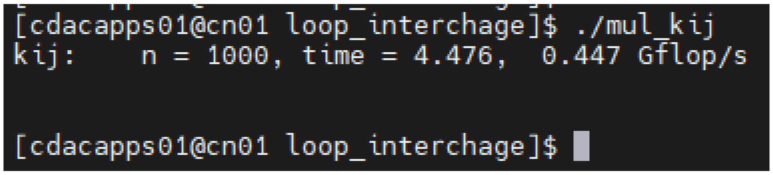


```
[cdacapps01@cn01 loop_interchage]$ ./mul_jki
jki:    n = 1000, time = 3.310,  0.604 Gflop/s

[cdacapps01@cn01 loop_interchage]$ 
```

5. k,i,j loop

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>

void multMat1( int n, float *A, float *B, float *C ) {
    int i,j,k;
    /* This is kij loop order. */
    for( k = 0; k < n; k++ )
        for( i = 0; i < n; i++ )
            for( j = 0; j < n; j++ )
                C[i+j*n] += A[i+k*n]*B[k+j*n];
}

int main( int argc, char **argv ) {
    int nmax = 1000,i;
    void (*orderings[])(int,float *,float *,float *) = {&multMat1};
    char *names[] = {"kij"};

    float *A = (float *)malloc( nmax*nmax * sizeof(float));
    float *B = (float *)malloc( nmax*nmax * sizeof(float));
    float *C = (float *)malloc( nmax*nmax * sizeof(float));

    struct timeval start, end;

    /* fill matrices with random numbers */
    for( i = 0; i < nmax*nmax; i++ ) A[i] = drand48()*2-1;
    for( i = 0; i < nmax*nmax; i++ ) B[i] = drand48()*2-1;
    for( i = 0; i < nmax*nmax; i++ ) C[i] = drand48()*2-1;

        gettimeofday( &start, NULL );
        (*orderings[0])( nmax, A, B, C );
        gettimeofday( &end, NULL );

        /* convert time to Gflop/s */
        double seconds = (end.tv_sec - start.tv_sec) +
            1.0e-6 * (end.tv_usec - start.tv_usec);
        double Gflops = 2e-9*nmax*nmax*nmax/seconds;
        printf( "%s:\tn = %d, time = %.3f,  %.3f Gflop/s\n",
names[0], nmax, seconds, Gflops );

    free( A );
    free( B );
    free( C );

    printf("\n\n");

    return 0;
}
```

gcc mul_kij.c -o mul_kij
./mul_kij

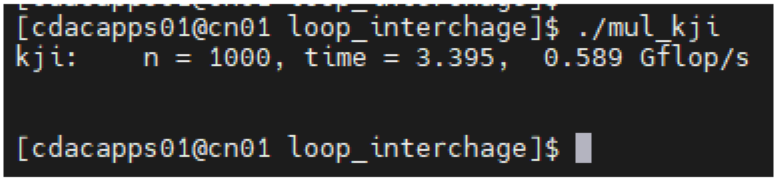


```
[cdacapps01@cn01 loop_interchage]$ ./mul_kij
kij:    n = 1000, time = 4.476,  0.447 Gflop/s


[cdacapps01@cn01 loop_interchage]$ 
```

6. k,j,i loop

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>
#include <time.h>

void multMat1( int n, float *A, float *B, float *C ) {
    int i,j,k;
    /* This is kji loop order. */
    for( k = 0; k < n; k++ )
        for( j = 0; j < n; j++ )
            for( i = 0; i < n; i++ )
                C[i+j*n] += A[i+k*n]*B[k+j*n];
}
int main( int argc, char **argv ) {
    int nmax = 1000,i;

    void (*orderings[])(int,float *,float *,float *) = {&multMat1};
    char *names[] = {"kji"};

    float *A = (float *)malloc( nmax*nmax * sizeof(float));
    float *B = (float *)malloc( nmax*nmax * sizeof(float));
    float *C = (float *)malloc( nmax*nmax * sizeof(float));

    struct timeval start, end;

    /* fill matrices with random numbers */
    for( i = 0; i < nmax*nmax; i++ ) A[i] = drand48()*2-1;
    for( i = 0; i < nmax*nmax; i++ ) B[i] = drand48()*2-1;
    for( i = 0; i < nmax*nmax; i++ ) C[i] = drand48()*2-1;

        gettimeofday( &start, NULL );
        (*orderings[0])( nmax, A, B, C );
        gettimeofday( &end, NULL );

        /* convert time to Gflop/s */
        double seconds = (end.tv_sec - start.tv_sec) +
            1.0e-6 * (end.tv_usec - start.tv_usec);
        double Gflops = 2e-9*nmax*nmax*nmax/seconds;
        printf( "%s:\tn = %d, time = %.3f,  %.3f Gflop/s\n",
names[0], nmax, seconds, Gflops );

    free( A );
    free( B );
    free( C );

    printf("\n\n");

    return 0;
}
```

gcc mul_kji.c -o mul_kji
./mul_kji



```
[cdacapps01@cn01 loop_interchage]$ ./mul_kji
kji:    n = 1000, time = 3.395,  0.589 Gflop/s


[cdacapps01@cn01 loop_interchage]$ 
```