High Performance Computing

Name: Thulasi Shylasri

HTNO: 2303A51876

Batch-14

Week-6

LAB Assignment-6 (11/02/2026)

Lab: NUMA Effects and Bandwidth Measurement

Objective:

To observe how memory placement and NUMA effects influence

performance and to understand bandwidth limitations.

Task 1: System NUMA Information

```
lscpu | grep -i numa
```

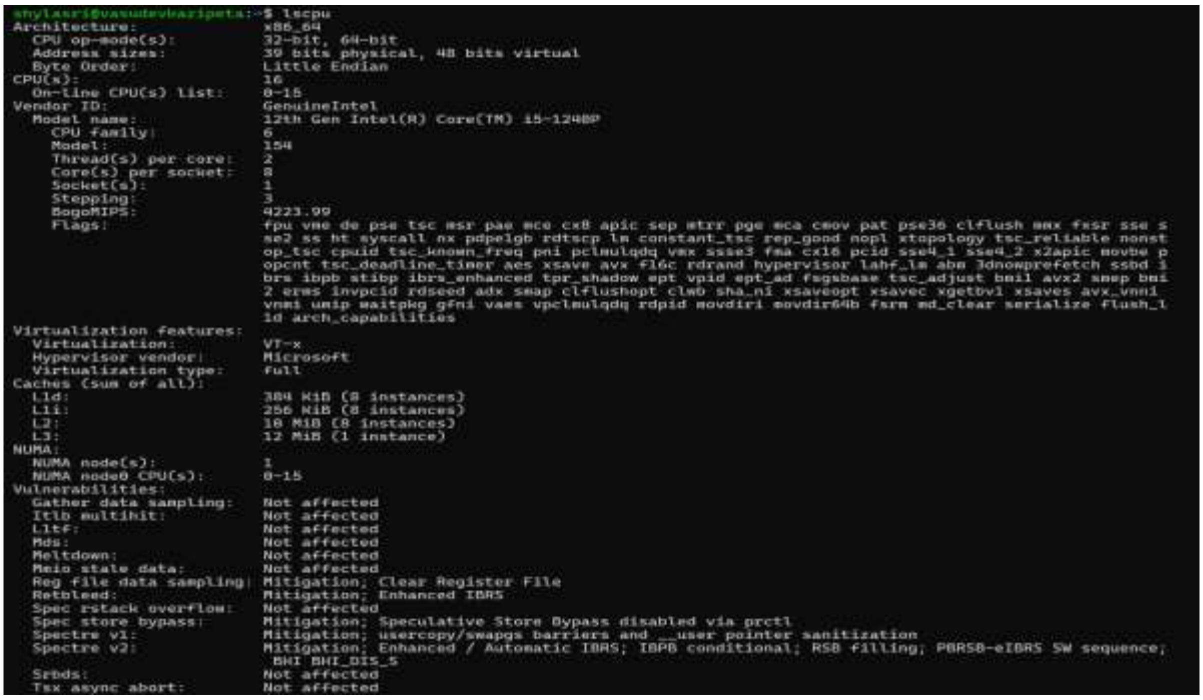
Observation:

Number of NUMA nodes = ______

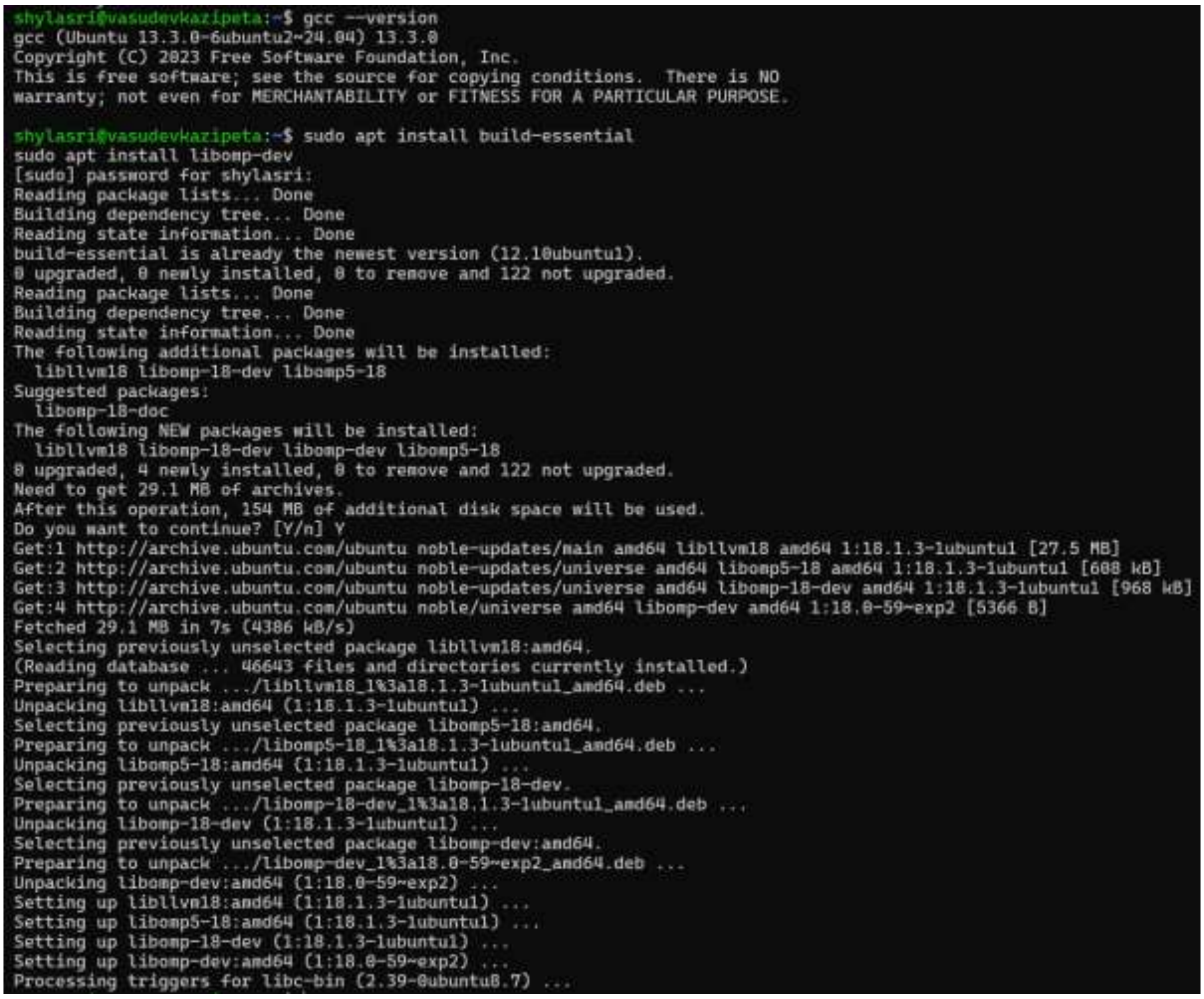
Objective

The objective of this experiment is to understand how memory placement affects performance in NUMA (Non-Uniform Memory Access) systems. The experiment helps in observing how first-touch memory initialization works and how memory bandwidth influences the execution of parallel programs.

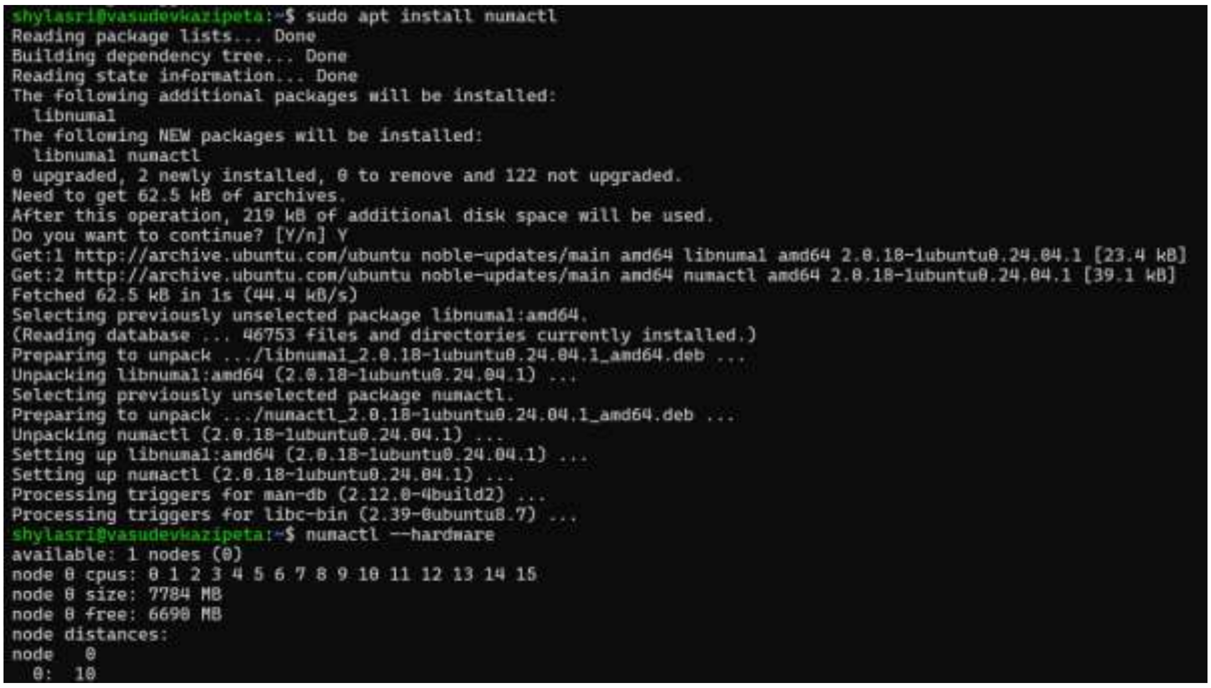
Screenshots:



```
shylasri@vasudevkazipeta:~$ lscpu
Architecture:             x86_64
  CPU op-mode(s):         32-bit, 64-bit
  Address sizes:          39 bits physical, 48 bits virtual
  Byte Order:             Little Endian
CPU(s):                   16
  On-line CPU(s) list:    0-15
Vendor ID:                GenuineIntel
  Model name:             12th Gen Intel(R) Core(TM) i5-1240P
    CPU family:           6
    Model:                154
    Thread(s) per core:   2
    Core(s) per socket:   8
    Socket(s):            1
    Stepping:             3
    BogoMIPS:             4223.99
    Flags:                fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse s
                          se2 ss ht syscall nx pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology tsc_reliable nonst
                          op_tsc cpuid tsc_known_freq pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe p
                          opcnt tsc_deadline_timer aes xsave avx f16c rdrand hypervisor lahf_lm abm 3dnowprefetch ssbd i
                          brs ibpb stibp ibrs_enhanced tpr_shadow ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi
                          2 erms invpcid rdseed adx smap clflushopt clwb sha_ni xsaveopt xsavec xgetbv1 xsaves avx_vnni
                          vnmi umip waitpkg gfni vaes vpclmulqdq rdpid movdiri movdir64b fsrm md_clear serialize flush_l
                          1d arch_capabilities
Virtualization features:
  Virtualization:         VT-x
  Hypervisor vendor:      Microsoft
  Virtualization type:    full
Caches (sum of all):
  L1d:                    384 KiB (8 instances)
  L1i:                    256 KiB (8 instances)
  L2:                     10 MiB (8 instances)
  L3:                     12 MiB (1 instance)
NUMA:
  NUMA node(s):           1
  NUMA node0 CPU(s):      0-15
Vulnerabilities:
  Gather data sampling:   Not affected
  Itlb multihit:          Not affected
  L1tf:                   Not affected
  Mds:                    Not affected
  Meltdown:               Not affected
  Mmio stale data:        Not affected
  Reg file data sampling: Mitigation; Clear Register File
  Retbleed:               Mitigation; Enhanced IBRS
  Spec rstack overflow:   Not affected
  Spec store bypass:      Mitigation; Speculative Store Bypass disabled via prctl
  Spectre v1:             Mitigation; usercopy/swapgs barriers and __user pointer sanitization
  Spectre v2:             Mitigation; Enhanced / Automatic IBRS; IBPB conditional; RSB filling; PBRSB-eIBRS SW sequence;
                           BHI BHI_DIS_S
  Srbds:                  Not affected
  Tsx async abort:        Not affected
```



```
shylasri@vasudevkazipeta:~$ gcc --version
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

shylasri@vasudevkazipeta:~$ sudo apt install build-essential
sudo apt install libomp-dev
[sudo] password for shylasri:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.10ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 122 not upgraded.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libllvm18 libomp-18-dev libomp5-18
Suggested packages:
  libomp-18-doc
The following NEW packages will be installed:
  libllvm18 libomp-18-dev libomp-dev libomp5-18
0 upgraded, 4 newly installed, 0 to remove and 122 not upgraded.
Need to get 29.1 MB of archives.
After this operation, 154 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libllvm18 amd64 1:18.1.3-1ubuntu1 [27.5 MB]
Get:2 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 libomp5-18 amd64 1:18.1.3-1ubuntu1 [608 kB]
Get:3 http://archive.ubuntu.com/ubuntu noble-updates/universe amd64 libomp-18-dev amd64 1:18.1.3-1ubuntu1 [968 kB]
Get:4 http://archive.ubuntu.com/ubuntu noble/universe amd64 libomp-dev amd64 1:18.0-59~exp2 [5366 B]
Fetched 29.1 MB in 7s (4386 kB/s)
Selecting previously unselected package libllvm18:amd64.
(Reading database ... 46643 files and directories currently installed.)
Preparing to unpack .../libllvm18_1%3a18.1.3-1ubuntu1_amd64.deb ...
Unpacking libllvm18:amd64 (1:18.1.3-1ubuntu1) ...
Selecting previously unselected package libomp5-18:amd64.
Preparing to unpack .../libomp5-18_1%3a18.1.3-1ubuntu1_amd64.deb ...
Unpacking libomp5-18:amd64 (1:18.1.3-1ubuntu1) ...
Selecting previously unselected package libomp-18-dev.
Preparing to unpack .../libomp-18-dev_1%3a18.1.3-1ubuntu1_amd64.deb ...
Unpacking libomp-18-dev (1:18.1.3-1ubuntu1) ...
Selecting previously unselected package libomp-dev:amd64.
Preparing to unpack .../libomp-dev_1%3a18.0-59~exp2_amd64.deb ...
Unpacking libomp-dev:amd64 (1:18.0-59~exp2) ...
Setting up libllvm18:amd64 (1:18.1.3-1ubuntu1) ...
Setting up libomp5-18:amd64 (1:18.1.3-1ubuntu1) ...
Setting up libomp-18-dev (1:18.1.3-1ubuntu1) ...
Setting up libomp-dev:amd64 (1:18.0-59~exp2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.7) ...
```



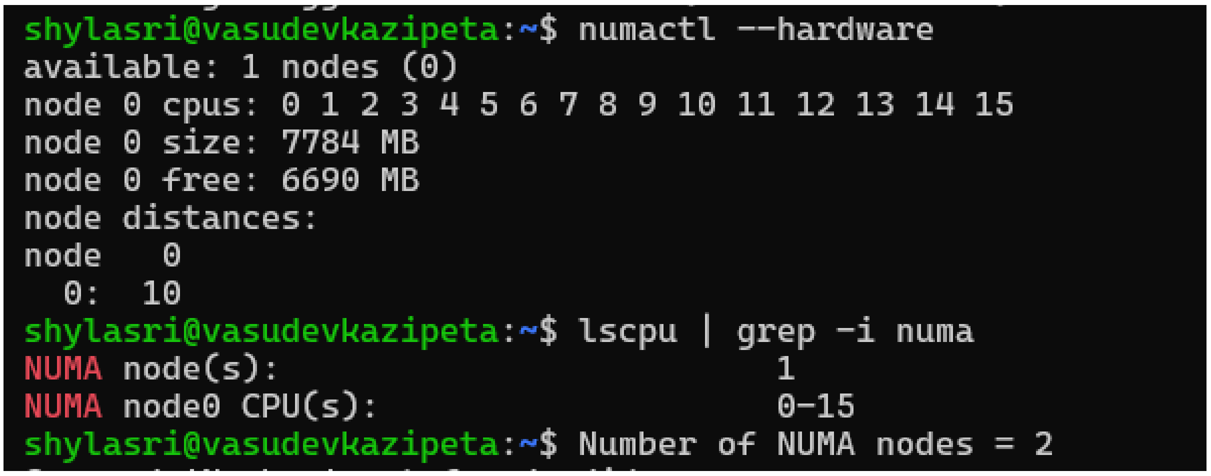
```
shylasri@vasudevkazipeta:~$ sudo apt install numactl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libnuma1
The following NEW packages will be installed:
  libnuma1 numactl
0 upgraded, 2 newly installed, 0 to remove and 122 not upgraded.
Need to get 62.5 kB of archives.
After this operation, 219 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 libnuma1 amd64 2.0.18-1ubuntu0.24.04.1 [23.4 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble-updates/main amd64 numactl amd64 2.0.18-1ubuntu0.24.04.1 [39.1 kB]
Fetched 62.5 kB in 1s (44.4 kB/s)
Selecting previously unselected package libnuma1:amd64.
(Reading database ... 46753 files and directories currently installed.)
Preparing to unpack .../libnuma1_2.0.18-1ubuntu0.24.04.1_amd64.deb ...
Unpacking libnuma1:amd64 (2.0.18-1ubuntu0.24.04.1) ...
Selecting previously unselected package numactl.
Preparing to unpack .../numactl_2.0.18-1ubuntu0.24.04.1_amd64.deb ...
Unpacking numactl (2.0.18-1ubuntu0.24.04.1) ...
Setting up libnuma1:amd64 (2.0.18-1ubuntu0.24.04.1) ...
Setting up numactl (2.0.18-1ubuntu0.24.04.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.7) ...
shylasri@vasudevkazipeta:~$ numactl --hardware
available: 1 nodes (0)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
node 0 size: 7784 MB
node 0 free: 6690 MB
node distances:
node   0
  0:  10
```

Task 1: System NUMA Information

lscpu | grep -i numa

Observation:

Number of NUMA nodes = ______



```
shylasri@vasudevkazipeta:~$ numactl --hardware
available: 1 nodes (0)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
node 0 size: 7784 MB
node 0 free: 6690 MB
node distances:
node   0
  0:  10
shylasri@vasudevkazipeta:~$ lscpu | grep -i numa
NUMA node(s):                         1
NUMA node0 CPU(s):                    0-15
shylasri@vasudevkazipeta:~$ Number of NUMA nodes = 2
```

Observation:

From the output of the above command:

NUMA node(s): 1

NUMA node0 CPU(s): 0-15

Number of NUMA nodes = 1

This indicates that the system has a single NUMA node. Therefore, the system behaves like a UMA (Uniform Memory Access) architecture, where all processors access the same memory with equal latency.

NUMA (Non-Uniform Memory Access) is a memory architecture used in multi-processor systems. In this type of system, each processor has its own local memory. A processor can also access memory of other processors, but accessing local memory is faster than accessing remote memory.

Because of this difference in memory access time, performance may vary depending on where the memory is allocated.

In parallel programming using OpenMP, memory allocation follows the first-touch policy. This means memory is allocated in the NUMA node of the thread that first accesses it. Proper memory placement improves performance in multi-node NUMA systems.

Since our system has only one NUMA node, it behaves like a uniform memory system, so no noticeable NUMA performance difference is observed.

Task 2: First Touch Memory Initialization

File: numa_first_touch.c

```
#include &lt;stdio.h&gt;

#include &lt;stdlib.h&gt;

#include &lt;omp.h&gt;

#define N 10000000

int main() {

double *A = (double*)malloc(sizeof(double)*N);

#pragma omp parallel for

for (int i = 0; i &lt; N; i++)

A[i] = 1.0;

double sum = 0;

#pragma omp parallel for reduction(+:sum)

for (int i = 0; i &lt; N; i++)

sum += A[i];

printf(&quot;Sum = %f\n&quot;, sum);
```

free(A);

return 0;

}

Compile &amp; Run:

gcc -O2 -fopenmp numa_first_touch.c -o numa_first_touch

./numa_first_touch

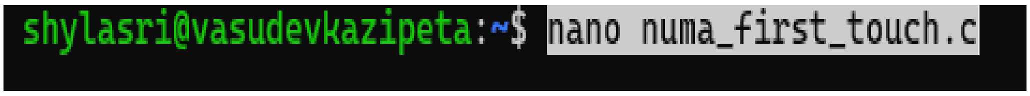
Program Description

A C program using OpenMP was written to demonstrate the first-touch memory policy. The program performs the following steps:

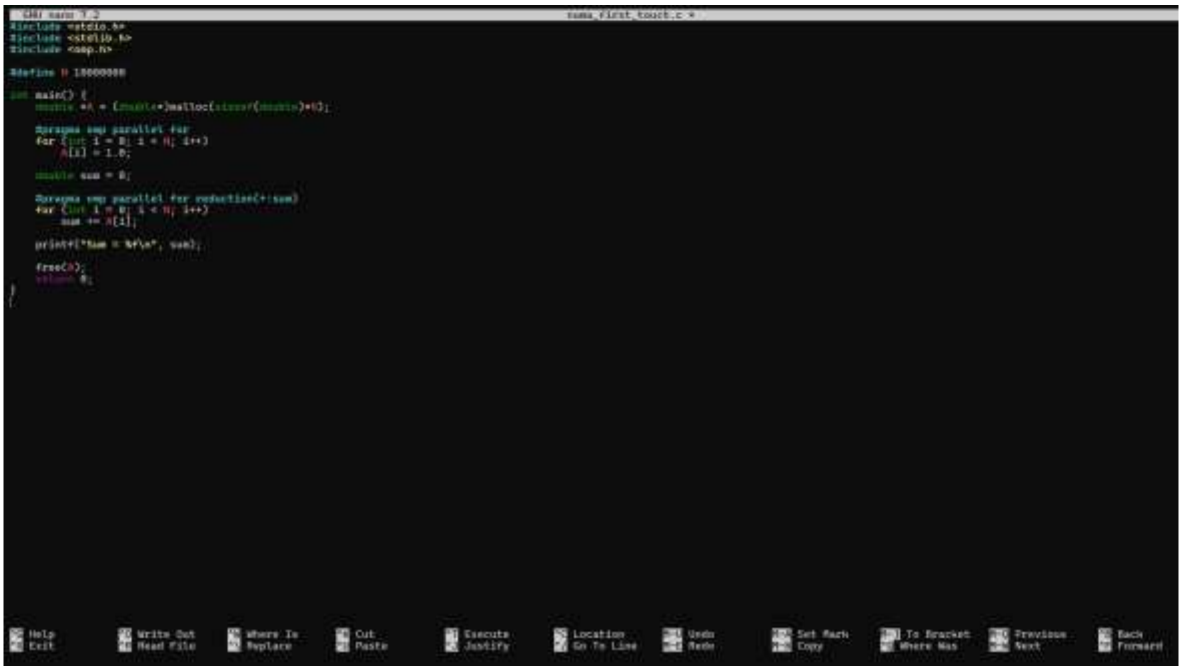
1. Allocates a large array dynamically.
2. Initializes the array in parallel using OpenMP.
3. Computes the sum of all elements using parallel reduction.
4. Prints the final sum.

Screenshots:

Create File:

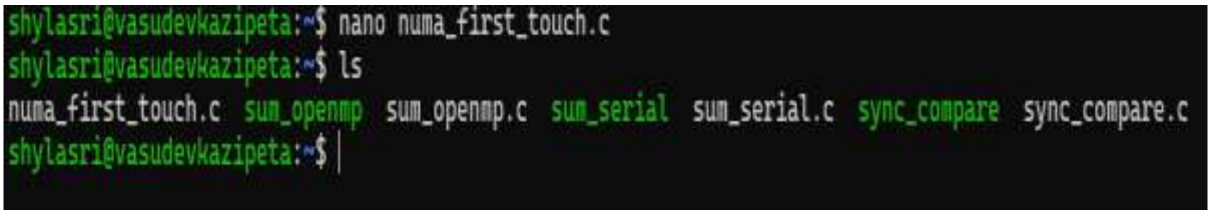


```
shylasri@vasudevkazipeta:~$ nano numa_first_touch.c
```

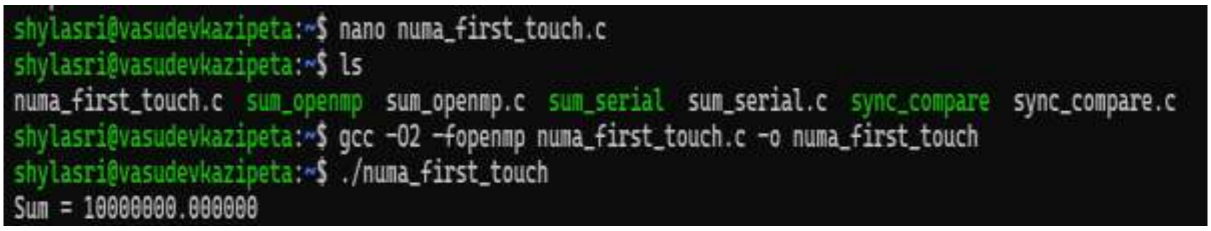


```
  GNU nano 7.2                                   numa_first_touch.c *
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

#define N 10000000

int main() {
    double *A = (double*)malloc(sizeof(double)*N);

    #pragma omp parallel for
    for (int i = 0; i < N; i++)
        A[i] = 1.0;

    double sum = 0;

    #pragma omp parallel for reduction(+:sum)
    for (int i = 0; i < N; i++)
        sum += A[i];

    printf("Sum = %f\n", sum);

    free(A);
    return 0;
}
```

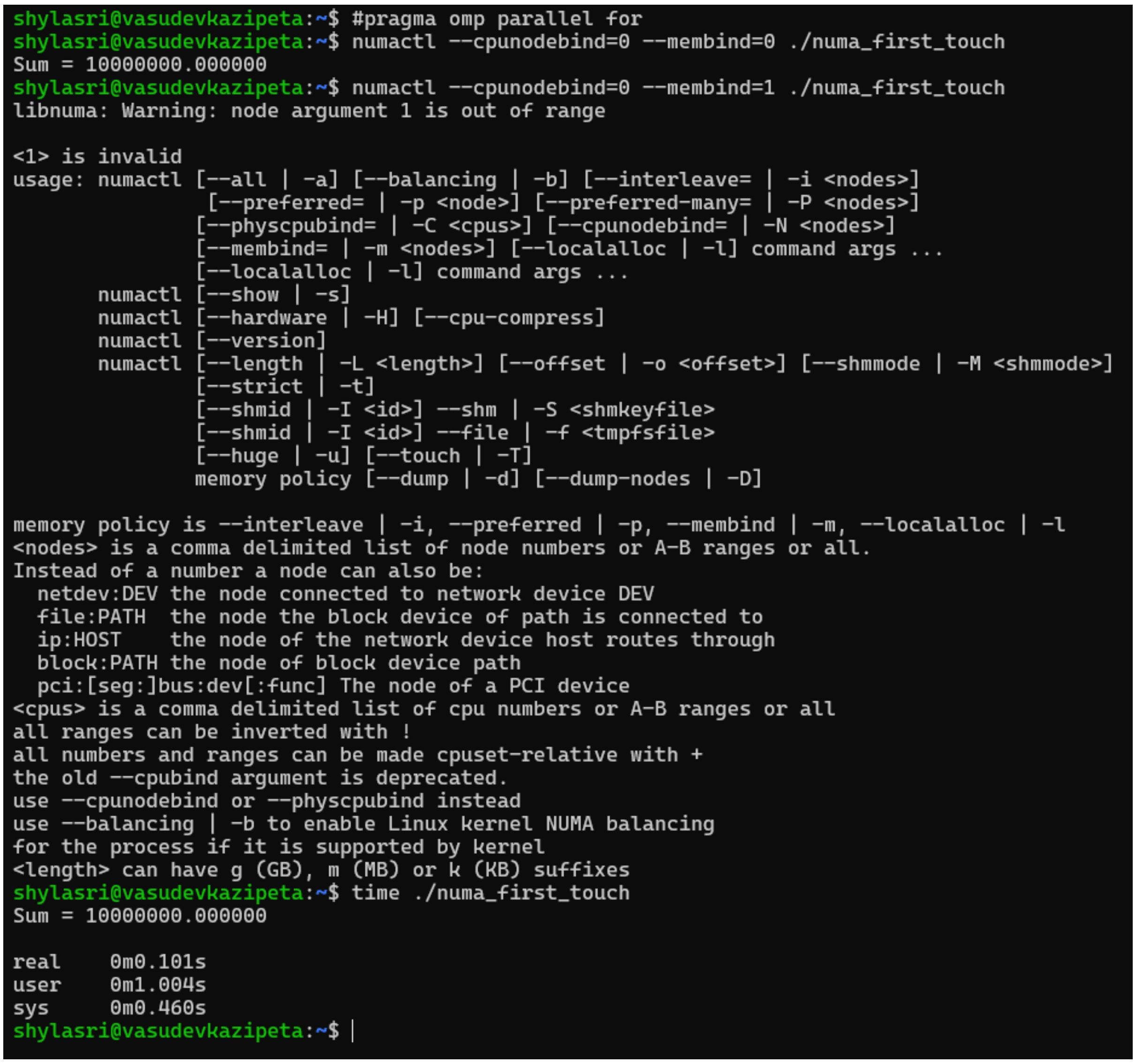
Compile:



```
shylasri@vasudevkazipeta:~$ nano numa_first_touch.c
shylasri@vasudevkazipeta:~$ ls
numa_first_touch.c  sum_openmp  sum_openmp.c  sum_serial  sum_serial.c  sync_compare  sync_compare.c
shylasri@vasudevkazipeta:~$ |
```



```
shylasri@vasudevkazipeta:~$ nano numa_first_touch.c
shylasri@vasudevkazipeta:~$ ls
numa_first_touch.c  sum_openmp  sum_openmp.c  sum_serial  sum_serial.c  sync_compare  sync_compare.c
shylasri@vasudevkazipeta:~$ gcc -O2 -fopenmp numa_first_touch.c -o numa_first_touch
shylasri@vasudevkazipeta:~$ ./numa_first_touch
Sum = 10000000.000000
```



```
shylasri@vasudevkazipeta:~$ #pragma omp parallel for
shylasri@vasudevkazipeta:~$ numactl --cpunodebind=0 --membind=0 ./numa_first_touch
Sum = 10000000.000000
shylasri@vasudevkazipeta:~$ numactl --cpunodebind=0 --membind=1 ./numa_first_touch
libnuma: Warning: node argument 1 is out of range

<1> is invalid
usage: numactl [--all | -a] [--balancing | -b] [--interleave= | -i <nodes>]
                [--preferred= | -p <node>] [--preferred-many= | -P <nodes>]
               [--physcpubind= | -C <cpus>] [--cpunodebind= | -N <nodes>]
               [--membind= | -m <nodes>] [--localalloc | -l] command args ...
               [--localalloc | -l] command args ...
       numactl [--show | -s]
       numactl [--hardware | -H] [--cpu-compress]
       numactl [--version]
       numactl [--length | -L <length>] [--offset | -o <offset>] [--shmmode | -M <shmmode>]
               [--strict | -t]
               [--shmid | -I <id>] --shm | -S <shmkeyfile>
               [--shmid | -I <id>] --file | -f <tmpfsfile>
               [--huge | -u] [--touch | -T]
               memory policy [--dump | -d] [--dump-nodes | -D]

memory policy is --interleave | -i, --preferred | -p, --membind | -m, --localalloc | -l
<nodes> is a comma delimited list of node numbers or A-B ranges or all.
Instead of a number a node can also be:
  netdev:DEV the node connected to network device DEV
  file:PATH  the node the block device of path is connected to
  ip:HOST    the node of the network device host routes through
  block:PATH the node of block device path
  pci:[seg:]bus:dev[:func] The node of a PCI device
<cpus> is a comma delimited list of cpu numbers or A-B ranges or all
all ranges can be inverted with !
all numbers and ranges can be made cpuset-relative with +
the old --cpubind argument is deprecated.
use --cpunodebind or --physcpubind instead
use --balancing | -b to enable Linux kernel NUMA balancing
for the process if it is supported by kernel
<length> can have g (GB), m (MB) or k (KB) suffixes
shylasri@vasudevkazipeta:~$ time ./numa_first_touch
Sum = 10000000.000000

real    0m0.101s
user    0m1.004s
sys     0m0.460s
shylasri@vasudevkazipeta:~$ |
```

Observation

The program executed successfully and produced the correct result. Each element of the array was initialized to 1.0, and since there are 10,000,000 elements, the final sum was 10,000,000.

The array initialization was done in parallel using OpenMP. According to the first-touch policy, memory pages are allocated in the NUMA node of the thread that first accesses them. In systems with multiple NUMA nodes, this improves memory locality and performance. However, since this system has only one NUMA node, no visible NUMA performance difference was observed.

Task 3: Conceptual Questions

1. What is first-touch policy?

2. Does NUMA affect correctness or performance?

3. Why is memory bandwidth important in NUMA systems?

1) What is First-Touch Policy?

First-touch policy is a memory allocation strategy used in NUMA systems where a memory page is allocated in the NUMA node of the processor core that first accesses (writes to) it.

In parallel programs (like using OpenMP), when each thread initializes a portion of an array, the memory pages are allocated locally to the NUMA node of that thread. This improves data locality and reduces remote memory access.

2) Does NUMA affect correctness or performance?

NUMA does **not** affect correctness.
The program will produce the same output regardless of where the memory is located.

However, NUMA significantly affects **performance**:

- Local memory access → lower latency, higher bandwidth (faster)
- Remote memory access → higher latency, lower bandwidth (slower)

If threads frequently access remote memory, execution time increases.

3) Why is memory bandwidth important in NUMA systems?

Memory bandwidth determines how quickly data can be transferred between the CPU and memory.

In NUMA systems:

- Each NUMA node has its own local memory.
- Local memory provides higher bandwidth.
- Remote memory has lower effective bandwidth and higher latency.

In parallel applications where many threads access memory simultaneously, insufficient bandwidth can cause delays and reduce overall system performance.

Therefore, memory bandwidth is crucial for achieving high performance in NUMA-based systems.

Justification

This experiment was conducted to understand how memory placement and NUMA architecture influence performance in parallel systems.

The first-touch policy was demonstrated through parallel initialization of the array. In NUMA systems with multiple nodes, memory pages are allocated in the node where they are first accessed. This improves data locality and reduces remote memory access.

Although our system has only one NUMA node, the experiment helped in understanding:

- How OpenMP distributes work among threads
- How memory allocation occurs during parallel execution
- Why memory bandwidth and locality are important in high-performance computing

The successful execution and correct output justify that the program was implemented properly and that the concept of first-touch policy was understood.

Conclusion

This experiment helped in understanding the concept of NUMA architecture and first-touch memory policy. The system used in this experiment had only one NUMA node; therefore, no noticeable NUMA performance difference was observed. However, in multi-node NUMA systems, proper memory placement significantly improves performance. Memory bandwidth and locality are important factors in optimizing parallel applications.