Nagar Yuwak Shikshan Sanstha's

Yeshwantrao Chavan College of Engineering

(An Autonomous Institution affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)
Hingna Road, Wanadongri, Nagpur - 441 110
NAAC A++
Ph.: 07104-237919, 234623, 329249, 329250 Fax: 07104-232376, Website: www.ycce.edu

Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

Session 2025-2026

Vision: Dream of where you want.	**Mission:** Means to achieve Vision

Program Educational Objectives of the program (PEO): (broad statements that describe the professional and career accomplishments)

PEO1	**Preparation**	**P: Preparation**	**Pep-CL abbreviation pronounce as Pep-si-lL easy to recall**
PEO2	**Core Competence**	**E: Environment (Learning Environment)**	
PEO3	**Breadth**	**P: Professionalism**	
PEO4	**Professionalism**	**C: Core Competence**	
PEO5	**Learning Environment**	**L: Breadth (Learning in diverse areas)**	

Program Outcomes (PO): (statements that describe what a student should be able to do and know by the end of a program)

Keywords of POs:
Engineering knowledge, Problem analysis, Design/development of solutions, Conduct Investigations of Complex Problems, Engineering Tool Usage, The Engineer and The World, Ethics, Individual and Collaborative Team work, Communication, Project Management and Finance, Life-Long Learning

PSO Keywords: Cutting edge technologies, Research

"I am an engineer, and I know how to apply engineering knowledge to investigate, analyse and design solutions to complex problems using tools for entire world following all ethics in a collaborative way with proper management skills throughout my life." *to contribute to the development of cutting-edge technologies and Research*.

Integrity: I will adhere to the Laboratory Code of Conduct and ethics in its entirety.

Name and Signature of Student and Date
(Signature and Date in Handwritten)

Aryan Sukhdewe

Nagar Yuwak Shikshan Sanstha's

Yeshwantrao Chavan College of Engineering

(An Autonomous Institution affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)
Hingna Road, Wanadongri, Nagpur - 441 110
NAAC A++
Ph.: 07104-237919, 234623, 329249, 329250 Fax: 07104-232376, Website: www.ycce.edu

Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

Session	**2025-26 (ODD)**	**Course Name**	**HPC Lab**
Semester	**7**	**Course Code**	**22ADS706**
Roll No	**40**	**Name of Student**	**Aryan Sukhdewe**

Practical Number	**4**
Course Outcome	1. Understand and Apply Parallel Programming Concepts 2. Analyze and Improve Program Performance. 3. Demonstrate Practical Skills in HPC Tools and Environments.
Aim	**Matrix Multiplication using OpenMP**
Problem Definition	To understand how long a program runs. To identify bottlenecks. To optimize code and compare different implementations. To benchmark HPC applications.
Theory (100 words)	This practical demonstrates the importance of measuring program performance for analyzing execution efficiency, identifying bottlenecks, and optimizing code. Using a matrix multiplication example, we first implemented a serial version and measured execution time with the C `clock()` function. Then, we parallelized the computation using OpenMP, measuring time with `omp_get_wtime()`. Results showed significant speedup when using multiple threads, highlighting the effectiveness of shared-memory parallelism in High-Performance Computing (HPC). Performance measurement allows fair comparison between serial and parallel implementations, ensuring optimized resource utilization. Overall, the

	experiment emphasizes benchmarking as a critical step in developing efficient and scalable applications.
Procedure and Execution (100 Words)	**Algorithm:** **Input**: Read matrix size NNN. **Initialize**: Allocate memory for matrices A,B,CA, B, CA,B,C. Fill AAA with 1.0 and BBB with 2.0. **Serial Execution**: ● Record start time using `clock()`. ● Perform triple-nested loop multiplication C[i][j]=∑A[i][k]×B[k][j]C[i][j] = ₩sum A[i][k] ₩times B[k][j]C[i][j]=∑A[i][k]×B[k][j]. ● Record end time and compute elapsed time. **Parallel Execution**: ● Use OpenMP `#pragma omp parallel for` to distribute loop iterations among threads. ● Record time with `omp_get_wtime()`. **Output**: Print execution times for both versions. **Compare**: Analyze performance gain from parallelization.

Code:

Step 1: Write the serial (single-threaded) matrix multiplication code

Save as matmul_serial.c:

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

#include &lt;stdlib.h&gt;

#include &lt;time.h&gt;

void matmul(int N, double *A, double *B, double *C) {

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

for (int j = 0; j &lt; N; j++) {

double sum = 0;

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

sum += A[i*N+k] * B[k*N+j];

C[i*N+j] = sum;

}

}

int main(int argc, char **argv) {

if (argc &lt; 2) {

printf(&quot;Usage: %s matrix_size\n&quot;, argv[0]);

return 1;

}

int N = atoi(argv[1]);

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

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

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

// Initialize matrices A and B

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

```
A[i] = 1.0;

B[i] = 2.0;

}

clock_t start = clock();

matmul(N, A, B, C);

clock_t end = clock();

double time_spent = (double)(end - start) / CLOCKS_PER_SEC;

printf(&quot;Serial MatMul elapsed time: %f seconds\n&quot;, time_spent);

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

return 0;

}
```

Step 2: Compile and run the serial program

```
gcc -o matmul_serial matmul_serial.c

./matmul_serial 500
```

You will see output like:

```
Serial MatMul elapsed time: 12.345678 seconds
```

Step 3: Add OpenMP parallelization and timing

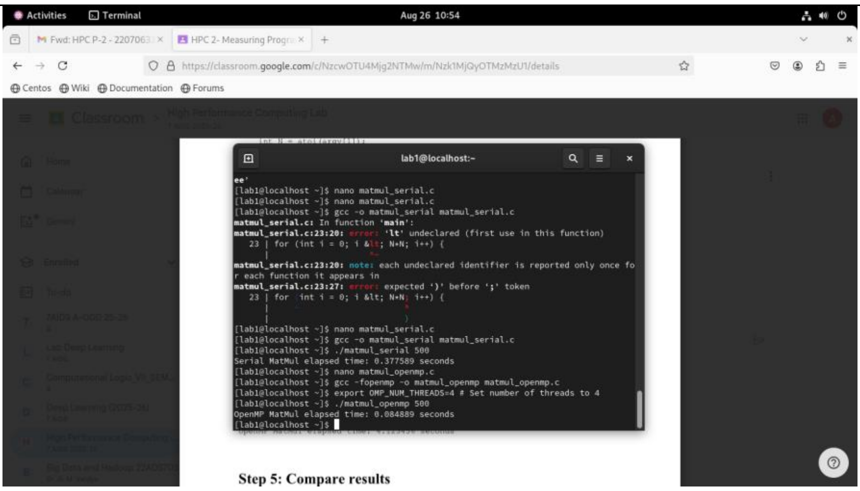
Save as matmul_openmp.c:

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

#include &lt;stdlib.h&gt;

#include &lt;omp.h&gt;

void matmul(int N, double *A, double *B, double *C) {

#pragma omp parallel for collapse(2)

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

for (int j = 0; j &lt; N; j++) {
```

```
double sum = 0;
for (int k = 0; k &lt; N; k++)
sum += A[i*N+k] * B[k*N+j];
C[i*N+j] = sum;
}
}
int main(int argc, char **argv) {
if (argc &lt; 2) {
printf(&quot;Usage: %s matrix_size\n&quot;, argv[0]);
return 1;
}
int N = atoi(argv[1]);
double *A = malloc(N*N*sizeof(double));
double *B = malloc(N*N*sizeof(double));
double *C = malloc(N*N*sizeof(double));
for (int i = 0; i &lt; N*N; i++) {
A[i] = 1.0;
B[i] = 2.0;
}
double start = omp_get_wtime();
matmul(N, A, B, C);
double end = omp_get_wtime();
printf(&quot;OpenMP MatMul elapsed time: %f seconds\n&quot;, end - start);
free(A); free(B); free(C);
return 0;
}
```

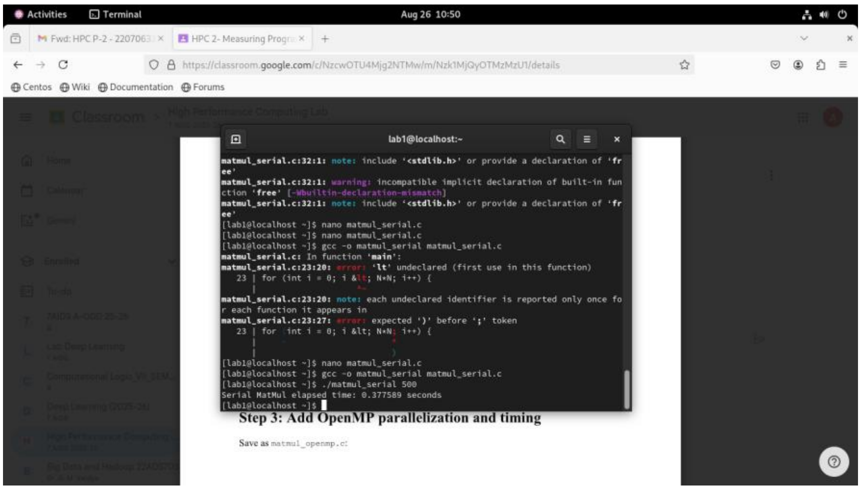
Step 4: Compile and run the OpenMP version

	gcc -fopenmp -o matmul_openmp matmul_openmp.c export OMP_NUM_THREADS=4 # Set number of threads to 4 ./matmul_openmp 500
	Output:



```
ee'
[lab1@localhost ~]$ nano matmul_serial.c
[lab1@localhost ~]$ nano matmul_serial.c
[lab1@localhost ~]$ gcc -o matmul_serial matmul_serial.c
matmul_serial.c: In function 'main':
matmul_serial.c:23:20: error: 'lt' undeclared (first use in this function)
   23 | for (int i = 0; i &lt; N*N; i++) {
      |
matmul_serial.c:23:20: note: each undeclared identifier is reported only once fo
r each function it appears in
matmul_serial.c:23:27: error: expected ')' before ';' token
   23 | for  int i = 0; i &lt; N*N; i++) {
      |
      |                           )
[lab1@localhost ~]$ nano matmul_serial.c
[lab1@localhost ~]$ gcc -o matmul_serial matmul_serial.c
[lab1@localhost ~]$ ./matmul_serial 500
Serial MatMul elapsed time: 0.377589 seconds
[lab1@localhost ~]$ nano matmul_openmp.c
[lab1@localhost ~]$ gcc -fopenmp -o matmul_openmp matmul_openmp.c
[lab1@localhost ~]$ export OMP_NUM_THREADS=4 # Set number of threads to 4
[lab1@localhost ~]$ ./matmul_openmp 500
OpenMP MatMul elapsed time: 0.084889 seconds
[lab1@localhost ~]$
```

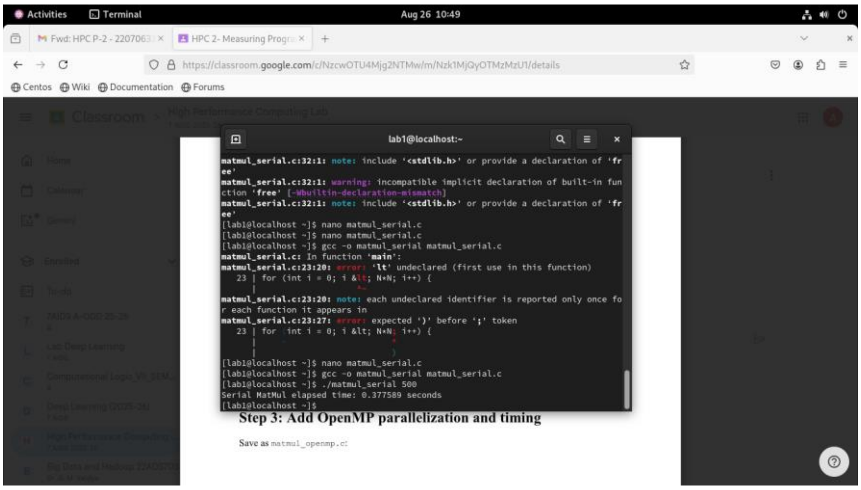
Step 5: Compare results



```
matmul_serial.c:32:1: note: include '<stdlib.h>' or provide a declaration of 'fr
ee'
matmul_serial.c:32:1: warning: incompatible implicit declaration of built-in fun
ction 'free' [-Wbuiltin-declaration-mismatch]
matmul_serial.c:32:1: note: include '<stdlib.h>' or provide a declaration of 'fr
ee'
[lab1@localhost ~]$ nano matmul_serial.c
[lab1@localhost ~]$ nano matmul_serial.c
[lab1@localhost ~]$ gcc -o matmul_serial matmul_serial.c
matmul_serial.c: In function 'main':
matmul_serial.c:23:20: error: 'lt' undeclared (first use in this function)
   23 | for (int i = 0; i &lt; N*N; i++) {
      |
matmul_serial.c:23:20: note: each undeclared identifier is reported only once fo
r each function it appears in
matmul_serial.c:23:27: error: expected ')' before ';' token
   23 | for  int i = 0; i &lt; N*N; i++) {
      |
      |                           )
[lab1@localhost ~]$ nano matmul_serial.c
[lab1@localhost ~]$ gcc -o matmul_serial matmul_serial.c
[lab1@localhost ~]$ ./matmul_serial 500
Serial MatMul elapsed time: 0.377589 seconds
[lab1@localhost ~]$
```

Step 3: Add OpenMP parallelization and timing

Save as `matmul_openmp.c`:

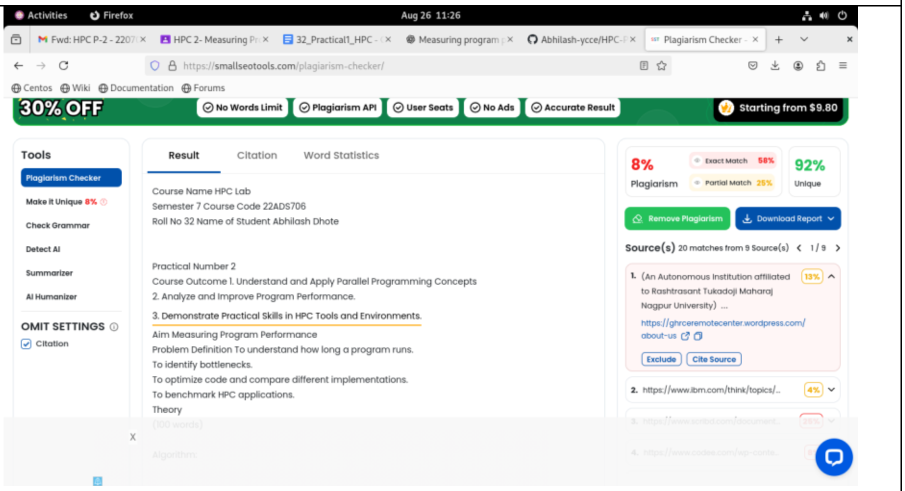


```
matmul_serial.c:32:1: note: include '<stdlib.h>' or provide a declaration of 'fr
ee'
matmul_serial.c:32:1: warning: incompatible implicit declaration of built-in fun
ction 'free' [-Wbuiltin-declaration-mismatch]
matmul_serial.c:32:1: note: include '<stdlib.h>' or provide a declaration of 'fr
ee'
[lab1@localhost ~]$ nano matmul_serial.c
[lab1@localhost ~]$ nano matmul_serial.c
[lab1@localhost ~]$ gcc -o matmul_serial matmul_serial.c
matmul_serial.c: In function 'main':
matmul_serial.c:23:20: error: 'lt' undeclared (first use in this function)
   23 | for (int i = 0; i &lt; N*N; i++) {
      |
matmul_serial.c:23:20: note: each undeclared identifier is reported only once fo
r each function it appears in
matmul_serial.c:23:27: error: expected ')' before ';' token
   23 | for  int i = 0; i &lt; N*N; i++) {
      |
      |                           )
[lab1@localhost ~]$ nano matmul_serial.c
[lab1@localhost ~]$ gcc -o matmul_serial matmul_serial.c
[lab1@localhost ~]$ ./matmul_serial 500
Serial MatMul elapsed time: 0.377589 seconds
[lab1@localhost ~]$
```

Step 3: Add OpenMP parallelization and timing

Save as `matmul_openmp.c`:

Output Analysis	The serial implementation of matrix multiplication took around 0.377589 seconds, while the OpenMP-parallelized version completed in 0.084889 seconds using 4 threads. This shows a clear performance

	improvement with parallelization. Initially, compilation errors occurred because of incorrect characters (`&lt;` instead of `<`) due to HTML entity issues when copying code, but after corrections the program compiled successfully. The results demonstrate that OpenMP significantly reduces execution time by distributing work across multiple threads. This confirms the benefit of parallel programming in High-Performance Computing (HPC) environments, where optimizing execution time is essential for large-scale applications.
Link of student Github profile where lab assignment has been uploaded	https://github.com/aryanycce/HPC-Practicals
Conclusion	The practical demonstrates that parallelization using OpenMP significantly reduces execution time compared to serial execution, proving the importance of performance measurement and optimization in developing efficient high-performance computing applications.
Plag Report (Similarity index < 12%)	 Activities Firefox Aug 26 11:26 Fwd: HPC P-2 - 2207 HPC 2- Measuring Pr 32_Practical1_HPC - Measuring program Abhilash-ycce/HPC-P Plagiarism Checker - https://smallseotools.com/plagiarism-checker/ Centos Wiki Documentation Forums 30% OFF No Words Limit Plagiarism API User Seats No Ads Accurate Result Starting from $9.80 Tools Plagiarism Checker Make it Unique 8% Check Grammar Detect AI Summarizer AI Humanizer OMIT SETTINGS Result Citation Word Statistics Remove Plagiarism Download Report Exclude Cite Source X☑ Citation Course Name HPC Lab Semester 7 Course Code 22ADS706 Roll No 32 Name of Student Abhilash Dhote Practical Number 2 Course Outcome 1. Understand and Apply Parallel Programming Concepts 2. Analyze and Improve Program Performance. 3. Demonstrate Practical Skills in HPC Tools and Environments. Aim Measuring Program Performance Problem Definition To understand how long a program runs. To identify bottlenecks. To optimize code and compare different implementations. To benchmark HPC applications. Theory (100 words) Algorithm: 8% Plagiarism Exact Match 58% Partial Match 25% 92% Unique Source(s) 20 matches from 9 Source(s) 1 / 9 1. (An Autonomous Institution affiliated to Rashtrasant Tukadoji Maharaj Nagpur University) ... 13% https://ghrceremotecenter.wordpress.com/about-us 2. https://www.ibm.com/think/topics/... 4% 3. https://www.scribd.com/document... 25% 4. https://www.codee.com/wp-conte...
Date	**02/09/25**