

Practical No 9

PRN: 22520005

Name: Aftab Imtiyaj Bhadgaonkar

Batch: B6

Course: High Performance Computing Lab

Title of practical:

Matrix-Matrix and Matrix-Vector Multiplication

1) Implement Matrix-Matrix Multiplication using MPI. Use different number of processes and analyze the performance.

Code:

```
#include <mpi.h>

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

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);

    int world_rank, world_size;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    int n;
    if (world_rank == 0) {
        scanf("%d", &n);
    }
    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
    if (n % world_size != 0 && world_rank == 0) {
        printf("Error: Matrix size must be divisible by the number of processes.\n");
        MPI_Abort(MPI_COMM_WORLD, 1);
    }

    int (*A) = malloc(n * n * sizeof(int));
    int (*B) = malloc(n * n * sizeof(int));
    int (*C) = malloc(n * n * sizeof(int));
    if (world_rank == 0) {
```

```
for (int i = 0; i < n; i++) {  
    for (int j = 0; j < n; j++) {  
        A[i * n + j] = i * n + j + 1;  
        B[i * n + j] = i * n + j + 1;  
    }  
}  
  
double start_time = MPI_Wtime();  
int local_rows = n / world_size;  
int *local_A = malloc(local_rows * n * sizeof(int));  
int *local_C = malloc(local_rows * n * sizeof(int));  
MPI_Scatter(A, local_rows * n, MPI_INT, local_A, local_rows * n, MPI_INT, 0,  
MPI_COMM_WORLD);  
MPI_Bcast(B, n * n, MPI_INT, 0, MPI_COMM_WORLD);  
for (int i = 0; i < local_rows; i++) {  
    for (int j = 0; j < n; j++) {  
        local_C[i * n + j] = 0;  
        for (int k = 0; k < n; k++) {  
            local_C[i * n + j] += local_A[i * n + k] * B[k * n + j];  
        }  
    }  
}  
  
MPI_Gather(local_C, local_rows * n, MPI_INT, C, local_rows * n, MPI_INT, 0,  
MPI_COMM_WORLD);  
double end_time = MPI_Wtime();  
double elapsed_time = end_time - start_time;  
if (world_rank == 0) {  
    printf("Resulting matrix C:\n");  
    for (int i = 0; i < n; i++) {  
        for (int j = 0; j < n; j++) {  
            printf("%d ", C[i * n + j]);  
        }  
        printf("\n");  
    }  
    printf("Elapsed time: %f seconds\n", elapsed_time);  
}  
  
free(A);  
free(B);  
free(C);  
free(local_A);  
free(local_C);  
  
MPI_Finalize();  
return 0;  
}
```

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

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);

    int world_rank, world_size;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    int n;
    if (world_rank == 0) {
        scanf("%d", &n);
    }

    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (n % world_size != 0 && world_rank == 0) {
        printf("Error: Matrix size must be divisible by the number of processes.\n");
        MPI_Abort(MPI_COMM_WORLD, 1);
    }

    int (*A) = malloc(n * n * sizeof(int));
    int (*B) = malloc(n * n * sizeof(int));
    int (*C) = malloc(n * n * sizeof(int));
    if (world_rank == 0) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                A[i * n + j] = i * n + j + 1;
                B[i * n + j] = i * n + j + 1;
            }
        }
    }

    double start_time = MPI_Wtime();

    int local_rows = n / world_size;
    int *local_A = malloc(local_rows * n * sizeof(int));
    int *local_C = malloc(local_rows * n * sizeof(int));

    MPI_Scatter(A, local_rows * n, MPI_INT, local_A, local_rows * n, MPI_INT, 0,
        MPI_COMM_WORLD);

    MPI_Bcast(B, n * n, MPI_INT, 0, MPI_COMM_WORLD);

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

```
local_C[i * n + j] = 0;
for (int k = 0; k < n; k++) {
    local_C[i * n + j] += local_A[i * n + k] * B[k * n + j];
}
}
```

```
MPI_Gather(local_C, local_rows * n, MPI_INT, C, local_rows * n, MPI_INT, 0,
MPI_COMM_WORLD);
```

```
double end_time = MPI_Wtime();
double elapsed_time = end_time - start_time;
```

```
if (world_rank == 0) {
    printf("Resulting matrix C:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", C[i * n + j]);
        }
        printf("\n");
    }
    printf("Elapsed time: %f seconds\n", elapsed_time);
}
```

```
free(A);
free(B);
free(C);
free(local_A);
free(local_C);
```

```
MPI_Finalize();
return 0;
}
```

Walchand College of Engineering, Sangli
Department of Computer Science and Engineering

Output:

```
EXPLORER  C mm.c  x
> vscode
22520005_HPC_Assg8.docx
C mm.c
C mvc
Practical No 9.docx
F q1

C mm.c  x
1 #include <mpi.h>
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
• aftab@Aftab:~/Desktop/AB/HPC/Assignment9$ mpicc mm.c -o q1
• aftab@Aftab:~/Desktop/AB/HPC/Assignment9$ mpirun -np 2 ./q1
20
Resulting matrix C:
53410 53620 53830 54040 54250 54460 54670 54880 55090 55300 55510 55720 55930 56140 56350 56560 56770 56980 57190 57400
129810 130420 131030 131640 132250 132860 133470 134080 134690 135300 135910 136520 137130 137740 138350 138960 139570 140180 140790 141400
206210 207220 208230 209240 210250 211260 212270 213280 214290 215300 216310 217320 218330 219340 220350 221360 222370 223380 224390 225400
282610 284020 285430 286840 288250 289660 291070 292480 293890 295300 296710 298120 299530 300940 302350 303760 305170 306580 307990 309400
359010 360820 362630 364440 366250 368060 369870 371680 373490 375300 377110 378920 380730 382540 384350 386160 387970 389780 391590 393400
435410 437620 439830 442040 444250 446460 448670 450880 453090 455300 457510 459720 461930 464140 466350 468560 470770 472980 475190 477400
511810 514420 517030 519640 522250 524860 527470 530080 532690 535300 537910 540520 543130 545740 548350 550960 553570 556180 558790 561400
588210 591220 594230 597240 600250 603260 606270 609280 612290 615300 618310 621320 624330 627340 630350 633360 636370 639380 642390 645400
664610 668020 671430 674840 678250 681660 685070 688480 691890 695300 698710 702120 705530 708940 712350 715760 719170 722580 725990 729400
741010 744820 748630 752440 756250 760060 763870 767680 771490 775300 779110 782920 786730 790540 794350 798160 801970 805780 809590 813400
839810 844020 848230 852440 856650 860860 865070 869280 873490 877700 881910 886120 890330 894540 898750 902960 907170 911380 915590 919800
893810 898420 903030 907640 912250 916860 921470 926080 930690 935300 939910 944520 949130 953740 958350 962960 967570 972180 976790 981400
970210 975220 980230 985240 990250 995260 1000270 1005280 1010290 1015300 1020310 1025320 1030330 1035340 1040350 1045360 1050370 1055380 1060390 1065400
1046610 1052020 1057430 1062840 1068250 1073660 1079070 1084480 1089890 1095300 1100710 1106120 1111530 1116940 1122350 1127760 1133170 1138580 1143990 1149400
1123010 1128820 1134630 1140440 1146250 1152060 1157870 1163680 1169490 1175300 1181110 1186920 1192730 1198540 1204350 1210160 1215970 1221780 1227590 1233400
1159410 1205620 1211830 1218040 1224250 1230460 1236670 1242880 1249090 1255300 1261510 1267720 1273930 1280140 1286350 1292560 1298770 1304980 1311190 1317400
1275810 1282420 1289030 1295640 1302250 1308860 1315470 1322080 1328690 1335300 1341910 1348520 1355130 1361740 1368350 1374960 1381570 1388180 1394790 1401400
1352210 1359220 1366230 1373240 1380250 1387260 1394270 1401280 1408290 1415300 1422310 1429320 1436330 1443340 1450350 1457360 1464370 1471380 1478390 1485400
1428610 1436020 1443430 1450840 1458250 1465660 1473070 1480480 1487890 1495300 1502710 1510120 1517530 1524940 1532350 1539760 1547170 1554580 1561990 1569400
1505010 1512820 1520630 1528440 1536250 1544060 1551870 1559680 1567490 1575300 1583110 1590920 1598730 1606540 1614350 1622160 1629970 1637780 1645590 1653400
Elapsed time: 0.000111 seconds
• aftab@Aftab:~/Desktop/AB/HPC/Assignment9$ mpirun -np 4 ./q1
20
Resulting matrix C:
53410 53620 53830 54040 54250 54460 54670 54880 55090 55300 55510 55720 55930 56140 56350 56560 56770 56980 57190 57400
129810 130420 131030 131640 132250 132860 133470 134080 134690 135300 135910 136520 137130 137740 138350 138960 139570 140180 140790 141400
206210 207220 208230 209240 210250 211260 212270 213280 214290 215300 216310 217320 218330 219340 220350 221360 222370 223380 224390 225400
282610 284020 285430 286840 288250 289660 291070 292480 293890 295300 296710 298120 299530 300940 302350 303760 305170 306580 307990 309400
359010 360820 362630 364440 366250 368060 369870 371680 373490 375300 377110 378920 380730 382540 384350 386160 387970 389780 391590 393400
435410 437620 439830 442040 444250 446460 448670 450880 453090 455300 457510 459720 461930 464140 466350 468560 470770 472980 475190 477400
511810 514420 517030 519640 522250 524860 527470 530080 532690 535300 537910 540520 543130 545740 548350 550960 553570 556180 558790 561400
588210 591220 594230 597240 600250 603260 606270 609280 612290 615300 618310 621320 624330 627340 630350 633360 636370 639380 642390 645400
664610 668020 671430 674840 678250 681660 685070 688480 691890 695300 698710 702120 705530 708940 712350 715760 719170 722580 725990 729400
741010 744820 748630 752440 756250 760060 763870 767680 771490 775300 779110 782920 786730 790540 794350 798160 801970 805780 809590 813400
839810 844020 848230 852440 856650 860860 865070 869280 873490 877700 881910 886120 890330 894540 898750 902960 907170 911380 915590 919800
893810 898420 903030 907640 912250 916860 921470 926080 930690 935300 939910 944520 949130 953740 958350 962960 967570 972180 976790 981400
970210 975220 980230 985240 990250 995260 1000270 1005280 1010290 1015300 1020310 1025320 1030330 1035340 1040350 1045360 1050370 1055380 1060390 1065400
1046610 1052020 1057430 1062840 1068250 1073660 1079070 1084480 1089890 1095300 1100710 1106120 1111530 1116940 1122350 1127760 1133170 1138580 1143990 1149400
1123010 1128820 1134630 1140440 1146250 1152060 1157870 1163680 1169490 1175300 1181110 1186920 1192730 1198540 1204350 1210160 1215970 1221780 1227590 1233400
1159410 1205620 1211830 1218040 1224250 1230460 1236670 1242880 1249090 1255300 1261510 1267720 1273930 1280140 1286350 1292560 1298770 1304980 1311190 1317400
1275810 1282420 1289030 1295640 1302250 1308860 1315470 1322080 1328690 1335300 1341910 1348520 1355130 1361740 1368350 1374960 1381570 1388180 1394790 1401400
1352210 1359220 1366230 1373240 1380250 1387260 1394270 1401280 1408290 1415300 1422310 1429320 1436330 1443340 1450350 1457360 1464370 1471380 1478390 1485400
1428610 1436020 1443430 1450840 1458250 1465660 1473070 1480480 1487890 1495300 1502710 1510120 1517530 1524940 1532350 1539760 1547170 1554580 1561990 1569400
1505010 1512820 1520630 1528440 1536250 1544060 1551870 1559680 1567490 1575300 1583110 1590920 1598730 1606540 1614350 1622160 1629970 1637780 1645590 1653400
Elapsed time: 0.000071 seconds
• aftab@Aftab:~/Desktop/AB/HPC/Assignment9$
```

Analysis:

Input Size (n)	Cores	Time
20	2	0.000111
	4	0.000071
40	2	0.000230
	4	0.000149

2) Implement Matrix-Matrix Multiplication using MPI. Use different number of processes and analyze the performance.

Code:

```
#include <mpi.h>

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

int main(int argc, char** argv) {
    MPI_Init(&argc, &argv);

    int world_rank, world_size;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    int n;
    if (world_rank == 0) {
        printf("Enter the size of the matrix (n x n): ");
        scanf("%d", &n);
    }

    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (n % world_size != 0 && world_rank == 0) {
        printf("Error: Matrix size must be divisible by the number of processes.\n");
        MPI_Abort(MPI_COMM_WORLD, 1);
    }

    int* matrix = malloc(n * n * sizeof(int));
    int* vector = malloc(n * sizeof(int));
    int* result = malloc(n * sizeof(int));
    if (world_rank == 0) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i * n + j] = i * n + j + 1;
            }
            vector[i] = 1;
        }
    }

    double start_time = MPI_Wtime();

    int rows_per_process = n / world_size;
    int* local_matrix = malloc(rows_per_process * n * sizeof(int));
    int* local_result = malloc(rows_per_process * sizeof(int));
```

```
MPI_Scatter(matrix, rows_per_process * n, MPI_INT, local_matrix, rows_per_process * n,  
MPI_INT, 0, MPI_COMM_WORLD);
```

```
MPI_Bcast(vector, n, MPI_INT, 0, MPI_COMM_WORLD);
```

```
for (int i = 0; i < rows_per_process; i++) {  
    local_result[i] = 0;  
    for (int j = 0; j < n; j++) {  
        local_result[i] += local_matrix[i * n + j] * vector[j];  
    }  
}
```

```
MPI_Gather(local_result, rows_per_process, MPI_INT, result, rows_per_process, MPI_INT, 0,  
MPI_COMM_WORLD);
```

```
double end_time = MPI_Wtime();  
double elapsed_time = end_time - start_time;
```

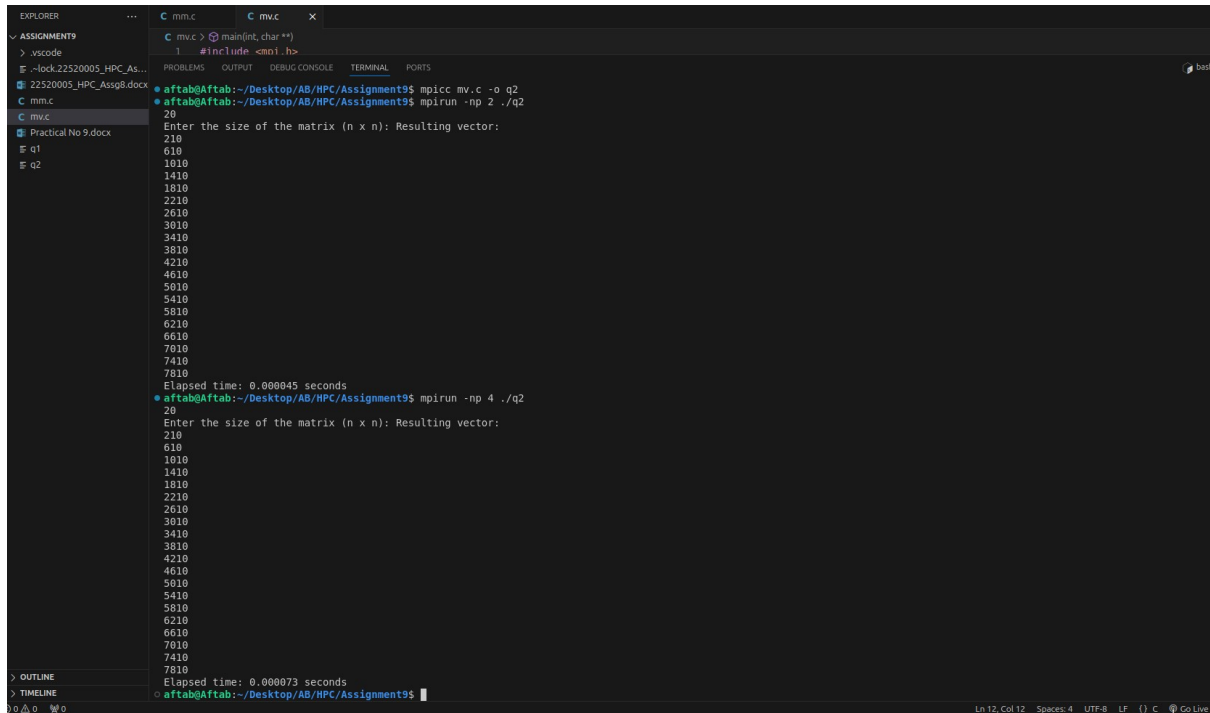
```
if (world_rank == 0) {  
    printf("Resulting vector:\n");  
    for (int i = 0; i < n; i++) {  
        printf("%d\n", result[i]);  
    }  
    printf("Elapsed time: %f seconds\n", elapsed_time);  
}
```

```
free(matrix);  
free(vector);  
free(result);  
free(local_matrix);  
free(local_result);
```

```
MPI_Finalize();  
return 0;  
}
```

Walchand College of Engineering, Sangli
Department of Computer Science and Engineering

Output:



```
EXPLORER  ...  C mm.c  C mm.c  X
ASSIGNMENT9
> .vscode
- lock-22520005_HPC_Assg8.docx
22520005_HPC_Assg8.docx
C mm.c
C mm.c
Practical No 9.docx
q1
q2

C mm.c > main(int, char **)
1 #include <mpi.h>

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

aftab@Aftab:~/Desktop/AB/HPC/Assignment9$ mpicc mv.c -o q2
aftab@Aftab:~/Desktop/AB/HPC/Assignment9$ mpirun -np 2 ./q2
20
Enter the size of the matrix (n x n): Resulting vector:
210
610
1010
1410
1810
2210
2610
3010
3410
3810
4210
4610
5010
5410
5810
6210
6610
7010
7410
7810
Elapsed time: 0.000045 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assignment9$ mpirun -np 4 ./q2
20
Enter the size of the matrix (n x n): Resulting vector:
210
610
1010
1410
1810
2210
2610
3010
3410
3810
4210
4610
5010
5410
5810
6210
6610
7010
7410
7810
Elapsed time: 0.000073 seconds
aftab@Aftab:~/Desktop/AB/HPC/Assignment9$
```

Analysis:

Input Size (n)	Cores	Time
20	2	0.000045
	4	0.000073
40	2	0.000039
	4	0.000073