

Walchand College of Engineering, Sangli
Department of Computer Science and Engineering
Class: Final Year (Computer Science and Engineering)
Year: 2022-23

Semester 1

Course: High Performance Computing Lab

Name : Sidharth Mukkanawar

PRN :2019BTECS00100

Batch: B4

Q1: Analyse and implement a Parallel code for below program using OpenMP. // C Program to find the minimum scalar product of two vectors (dot product)

Code

```
#include<omp.h>

#include<stdio.h>

#include<iostream>

#include<time.h>

#define n 1000

using namespace std;

void sort(int nums[])

{

    int i,j;

    for(int i=0;i <n-1;i++)
```

```

{
    int turn = i%2;

    #pragma omp parallel for
    for(int j=turn; j<n-1; j++)
    {
        if(nums[j]>nums[j+1])
        {
            int t=nums[j];
            nums[j]=nums[j+1];
            nums[j+1]=t;
        }
    }
}

void sort_desc(int nums[])
{
    int i,j;
    for(int i=0; i<n-1; i++)
    {
        int turn=i%2;

        #pragma omp parallel for
        for(int j=i; j<n-1; j++)
        {
            if(nums[j]<nums[j+1])
            {
                int t=nums[j];

```

```

        nums[j]=nums[j+1];
        nums[j+1]=t;
    }
}
}
}

int main()
{
    int nums1[n],nums2[2];
    for(int i=0;i <n;i++)
    {
        if(i%2)
        {
            nums1[i]=i+2;
        }
        else{
            nums1[i]=i*2;
        }

    }

    for(int i=0;i <n;i++)
    {
        if(i%2)
        {
            nums2[i]=i*2;
        }
    }
}

```

```

        else{

            nums2[i]=i+1;

        }

        cout << nums2[i] << endl;

    }

    clock_t t;

    t=clock();

    sort(nums1);

    sort_desc(nums2);

    t=clock()-t;

    cout << endl << "hello";

double time_taken = ((double)t) / CLOCKS_PER_SEC;

printf("Time taken (para): %f\n", time_taken);

int sum = 0;

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

    sum = sum + (nums1[i] * nums2[i]);

}

printf("%d\n", sum);

return 0;

}

```

Output

```
7 C:\Users\Lenovo\OneDrive\Documents\HPCLab\Assg3Q1.exe
7 131
7 262
7 -----
7 Process exited after 2.004 seconds with return value 3221225477
7 Press any key to continue . . .
7
7
```

- Q2) Write OpenMP code for two 2D Matrix addition, vary the size of your matrices from 250, 500, 750, 1000, and 2000 and measure the runtime with one thread (Use functions in C in calculate the execution time or use GPROF)
- For each matrix size, change the number of threads from 2,4,8., and plot the speedup versus the number of threads.
 - Explain whether or not the scaling behaviour is as expected.

```
#include<iostream>

#include<stdio.h>

#include<omp.h>

using namespace std;

int main()

{

int size=0;

printf("Size of matrix\n");

scanf("%d", &size);

// omp_set_num_threads(10);

int* matrix = new int[size*size];

int* vector = new int[size];

int* result = new int[size];
```

```

#pragma omp for
for(int i=0; i<size; i++)
{
    for(int j=0; j<size; j++)
        *(matrix+i*size+j) = 2;
}

#pragma omp for
for(int i=0; i<size; i++)
{
    *(vector+i)= 3;
}

int temp=0,threads;

double time = omp_get_wtime();

#pragma omp parallel for num_threads(8)
for(int i=0; i<size; i++)
{
    for(int j=0; j<size; j++)
    {
        temp=0;
        for(int k=0; k<size; k++)
        {
            temp += *(matrix+j*size+k) * *(vector+k);
        }
    }

    threads = omp_get_num_threads();

    *(result+i) = temp;
}

```

```

// for(int i=0; i<size; i++)

// {

// printf(" %d ", result[i]);

// }

printf("\nExecuted when size = %d and threads =%d \nDone in %f seconds\n",

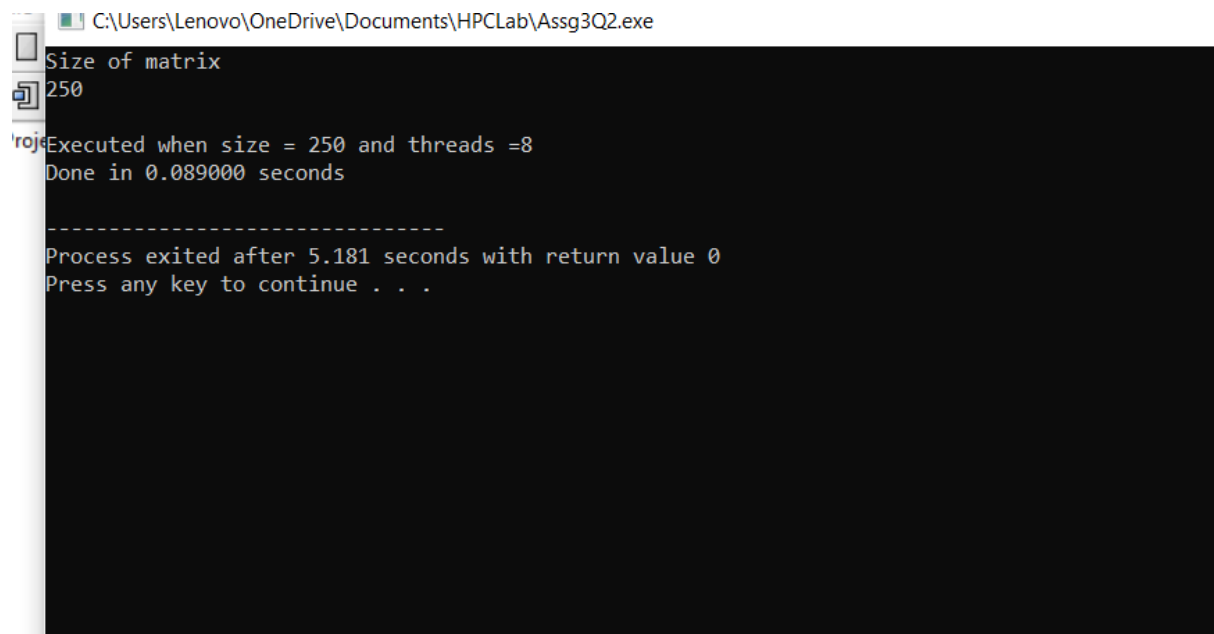
size,threads, omp_get_wtime()-time);

return 0;

}

```

Output



```

C:\Users\Lenovo\OneDrive\Documents\HPCLab\Assg3Q2.exe
Size of matrix
250
Executed when size = 250 and threads =8
Done in 0.089000 seconds
-----
Process exited after 5.181 seconds with return value 0
Press any key to continue . . .

```

Q3. For 1D Vector (size=200) and scalar addition, Write a OpenMP code with the following: i. ii. iii. Use STATIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup. Use DYNAMIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup. Demonstrate the use of nowait clause.

.static code

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

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

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

```
#define CHUNK 500
```

```
void static_fun(){
```

```
int a[1000];
```

```
int b[1000];
```

```
int c[1000];
```

```
double time = omp_get_wtime();
```

```
#pragma omp parallel
```

```
{
```

```
#pragma omp for nowait
```

```
for(int i=0;i<1000;i++)
```

```
a[i] = i;
```

```
#pragma omp for nowait
```

```
for(int i=0;i<1000;i++)
```

```
b[i] = i+1;
```

```
}
```

```
int threads;
```

```
#pragma omp parallel for num_threads(4) schedule(static,CHUNK)
```

```
for(int i=0;i<1000;i++) {
```

```
c[i] = a[i] + b[i];
```

```
threads = omp_get_num_threads();
```



```
}
```

```
for(int i=0;i<1000;i++)
```

```
printf("%d ",c[i]);
```

```
printf("\n Using %d no of threads with chunk size=%d and execution time=%f\
```

```
n",threads,CHUNK,omp_get_wtime() - time);
```

```
}
```

```
int main() {
```

```
static_fun();
```

```
return(0);
```

```
}
```

Output

```
C:\Users\Lenovo\OneDrive\Documents\HPCLab\Assg3Q3(Static code).exe
749 751 753 755 757 759 761 763 765 767 769 771 773 775 777 779 781 783 785 787 789 791 793 795 797 799 801 803 805 807
809 811 813 815 817 819 821 823 825 827 829 831 833 835 837 839 841 843 845 847 849 851 853 855 857 859 861 863 865 867
869 871 873 875 877 879 881 883 885 887 889 891 893 895 897 899 901 903 905 907 909 911 913 915 917 919 921 923 925 927
929 931 933 935 937 939 941 943 945 947 949 951 953 955 957 959 961 963 965 967 969 971 973 975 977 979 981 983 985 987
989 991 993 995 997 999 1001 1003 1005 1007 1009 1011 1013 1015 1017 1019 1021 1023 1025 1027 1029 1031 1033 1035 1037
1039 1041 1043 1045 1047 1049 1051 1053 1055 1057 1059 1061 1063 1065 1067 1069 1071 1073 1075 1077 1079 1081 1083 1085
1087 1089 1091 1093 1095 1097 1099 1101 1103 1105 1107 1109 1111 1113 1115 1117 1119 1121 1123 1125 1127 1129 1131 1133
1135 1137 1139 1141 1143 1145 1147 1149 1151 1153 1155 1157 1159 1161 1163 1165 1167 1169 1171 1173 1175 1177 1179 1181
1183 1185 1187 1189 1191 1193 1195 1197 1199 1201 1203 1205 1207 1209 1211 1213 1215 1217 1219 1221 1223 1225 1227 1229
1231 1233 1235 1237 1239 1241 1243 1245 1247 1249 1251 1253 1255 1257 1259 1261 1263 1265 1267 1269 1271 1273 1275 1277
1279 1281 1283 1285 1287 1289 1291 1293 1295 1297 1299 1301 1303 1305 1307 1309 1311 1313 1315 1317 1319 1321 1323 1325
1327 1329 1331 1333 1335 1337 1339 1341 1343 1345 1347 1349 1351 1353 1355 1357 1359 1361 1363 1365 1367 1369 1371 1373
1375 1377 1379 1381 1383 1385 1387 1389 1391 1393 1395 1397 1399 1401 1403 1405 1407 1409 1411 1413 1415 1417 1419 1421
1423 1425 1427 1429 1431 1433 1435 1437 1439 1441 1443 1445 1447 1449 1451 1453 1455 1457 1459 1461 1463 1465 1467 1469
1471 1473 1475 1477 1479 1481 1483 1485 1487 1489 1491 1493 1495 1497 1499 1501 1503 1505 1507 1509 1511 1513 1515 1517
1519 1521 1523 1525 1527 1529 1531 1533 1535 1537 1539 1541 1543 1545 1547 1549 1551 1553 1555 1557 1559 1561 1563 1565
1567 1569 1571 1573 1575 1577 1579 1581 1583 1585 1587 1589 1591 1593 1595 1597 1599 1601 1603 1605 1607 1609 1611 1613
1615 1617 1619 1621 1623 1625 1627 1629 1631 1633 1635 1637 1639 1641 1643 1645 1647 1649 1651 1653 1655 1657 1659 1661
1663 1665 1667 1669 1671 1673 1675 1677 1679 1681 1683 1685 1687 1689 1691 1693 1695 1697 1699 1701 1703 1705 1707 1709
1711 1713 1715 1717 1719 1721 1723 1725 1727 1729 1731 1733 1735 1737 1739 1741 1743 1745 1747 1749 1751 1753 1755 1757
1759 1761 1763 1765 1767 1769 1771 1773 1775 1777 1779 1781 1783 1785 1787 1789 1791 1793 1795 1797 1799 1801 1803 1805
1807 1809 1811 1813 1815 1817 1819 1821 1823 1825 1827 1829 1831 1833 1835 1837 1839 1841 1843 1845 1847 1849 1851 1853
1855 1857 1859 1861 1863 1865 1867 1869 1871 1873 1875 1877 1879 1881 1883 1885 1887 1889 1891 1893 1895 1897 1899 1901
1903 1905 1907 1909 1911 1913 1915 1917 1919 1921 1923 1925 1927 1929 1931 1933 1935 1937 1939 1941 1943 1945 1947 1949
1951 1953 1955 1957 1959 1961 1963 1965 1967 1969 1971 1973 1975 1977 1979 1981 1983 1985 1987 1989 1991 1993 1995 1997
1999
Using 4 no of threads with chunk size=500 and execution time=0.044000n
-----
Process exited after 0.1669 seconds with return value 0
Press any key to continue . . .
```

Dynamic Code

```
#include <omp.h>

#include <stdio.h>

#include <stdlib.h>

#define CHUNK 500

void static_fun(){

int a[1000];

int b[1000];

int c[1000];

double time = omp_get_wtime();

#pragma omp parallel

{

#pragma omp for nowait

for(int i=0;i<1000;i++)

a[i] = i;

#pragma omp for nowait

for(int i=0;i<1000;i++)

b[i] = i+1;

}

int threads;

#pragma omp parallel for num_threads(4) schedule(dynamic,CHUNK)

for(int i=0;i<1000;i++) {

c[i] = a[i] + b[i];

threads = omp_get_num_threads();

}
```

```

for(int i=0;i<1000;i++)

printf("%d ",c[i]);

printf("\n Using %d no of threads with chunk size=%d and execution time=%f\n",threads,CHUNK,omp_get_wtime() - time);

}

int main() {

static_fun();

return 0;

}

```

Output

```

C:\Users\Lenovo\OneDrive\Documents\HPLab\Assg3\3(Dynamic Code).exe
749 751 753 755 757 759 761 763 765 767 769 771 773 775 777 779 781 783 785 787 789 791 793 795 797 799 801 803 805 807
809 811 813 815 817 819 821 823 825 827 829 831 833 835 837 839 841 843 845 847 849 851 853 855 857 859 861 863 865 867
869 871 873 875 877 879 881 883 885 887 889 891 893 895 897 899 901 903 905 907 909 911 913 915 917 919 921 923 925 927
929 931 933 935 937 939 941 943 945 947 949 951 953 955 957 959 961 963 965 967 969 971 973 975 977 979 981 983 985 987
989 991 993 995 997 999 1001 1003 1005 1007 1009 1011 1013 1015 1017 1019 1021 1023 1025 1027 1029 1031 1033 1035 1037
1039 1041 1043 1045 1047 1049 1051 1053 1055 1057 1059 1061 1063 1065 1067 1069 1071 1073 1075 1077 1079 1081 1083 1085
1087 1089 1091 1093 1095 1097 1099 1101 1103 1105 1107 1109 1111 1113 1115 1117 1119 1121 1123 1125 1127 1129 1131 1133
1135 1137 1139 1141 1143 1145 1147 1149 1151 1153 1155 1157 1159 1161 1163 1165 1167 1169 1171 1173 1175 1177 1179 1181
1183 1185 1187 1189 1191 1193 1195 1197 1199 1201 1203 1205 1207 1209 1211 1213 1215 1217 1219 1221 1223 1225 1227 1229
1231 1233 1235 1237 1239 1241 1243 1245 1247 1249 1251 1253 1255 1257 1259 1261 1263 1265 1267 1269 1271 1273 1275 1277
1279 1281 1283 1285 1287 1289 1291 1293 1295 1297 1299 1301 1303 1305 1307 1309 1311 1313 1315 1317 1319 1321 1323 1325
1327 1329 1331 1333 1335 1337 1339 1341 1343 1345 1347 1349 1351 1353 1355 1357 1359 1361 1363 1365 1367 1369 1371 1373
1375 1377 1379 1381 1383 1385 1387 1389 1391 1393 1395 1397 1399 1401 1403 1405 1407 1409 1411 1413 1415 1417 1419 1421
1423 1425 1427 1429 1431 1433 1435 1437 1439 1441 1443 1445 1447 1449 1451 1453 1455 1457 1459 1461 1463 1465 1467 1469
1471 1473 1475 1477 1479 1481 1483 1485 1487 1489 1491 1493 1495 1497 1499 1501 1503 1505 1507 1509 1511 1513 1515 1517
1519 1521 1523 1525 1527 1529 1531 1533 1535 1537 1539 1541 1543 1545 1547 1549 1551 1553 1555 1557 1559 1561 1563 1565
1567 1569 1571 1573 1575 1577 1579 1581 1583 1585 1587 1589 1591 1593 1595 1597 1599 1601 1603 1605 1607 1609 1611 1613
1615 1617 1619 1621 1623 1625 1627 1629 1631 1633 1635 1637 1639 1641 1643 1645 1647 1649 1651 1653 1655 1657 1659 1661
1663 1665 1667 1669 1671 1673 1675 1677 1679 1681 1683 1685 1687 1689 1691 1693 1695 1697 1699 1701 1703 1705 1707 1709
1711 1713 1715 1717 1719 1721 1723 1725 1727 1729 1731 1733 1735 1737 1739 1741 1743 1745 1747 1749 1751 1753 1755 1757
1759 1761 1763 1765 1767 1769 1771 1773 1775 1777 1779 1781 1783 1785 1787 1789 1791 1793 1795 1797 1799 1801 1803 1805
1807 1809 1811 1813 1815 1817 1819 1821 1823 1825 1827 1829 1831 1833 1835 1837 1839 1841 1843 1845 1847 1849 1851 1853
1855 1857 1859 1861 1863 1865 1867 1869 1871 1873 1875 1877 1879 1881 1883 1885 1887 1889 1891 1893 1895 1897 1899 1901
1903 1905 1907 1909 1911 1913 1915 1917 1919 1921 1923 1925 1927 1929 1931 1933 1935 1937 1939 1941 1943 1945 1947 1949
1951 1953 1955 1957 1959 1961 1963 1965 1967 1969 1971 1973 1975 1977 1979 1981 1983 1985 1987 1989 1991 1993 1995 1997
1999
Using 4 no of threads with chunk size=500 and execution time=0.134000n
-----
Process exited after 0.3035 seconds with return value 0
Press any key to continue . . .

```

Nowwait code

```

#include <omp.h>

#include <stdio.h>

#include <stdlib.h>

```

```

#define CHUNK 100

void static_fun(){
    int a[1000];

    int b[1000];

    int c[1000];

    int count = 0;

    double time = omp_get_wtime();

    #pragma omp parallel
    {
        #pragma omp for nowait
        for(int i=0;i<1000;i++)
            a[i] = i;

        #pragma omp for nowait
        for(int i=0;i<1000;i++)
            b[i] = i+1;
    }

    int threads;

    #pragma omp parallel for num_threads(4) schedule(dynamic,CHUNK)
    for(int i=0;i<1000;i++) {
        c[i] = a[i] + b[i];

        threads = omp_get_num_threads();
    }

    for(int i=0;i<1000;i++)

```

```

printf("%d ",c[i]);

// printf("\n\n %d Times loop executed",count);

printf("\n Using %d no of threads with chunk size=%d and %f execution time\
n",threads,CHUNK,omp_get_wtime() - time);
}

int main() {

static_fun();

return(0);

}

```

Output

```

C:\Users\Lenovo\OneDrive\Documents\HPCLab\Assg3Q3(nowwait code).exe
749 751 753 755 757 759 761 763 765 767 769 771 773 775 777 779 781 783 785 787 789 791 793 795 797 799 801 803 805 807
809 811 813 815 817 819 821 823 825 827 829 831 833 835 837 839 841 843 845 847 849 851 853 855 857 859 861 863 865 867
869 871 873 875 877 879 881 883 885 887 889 891 893 895 897 899 901 903 905 907 909 911 913 915 917 919 921 923 925 927
929 931 933 935 937 939 941 943 945 947 949 951 953 955 957 959 961 963 965 967 969 971 973 975 977 979 981 983 985 987
989 991 993 995 997 999 1001 1003 1005 1007 1009 1011 1013 1015 1017 1019 1021 1023 1025 1027 1029 1031 1033 1035 1037
1039 1041 1043 1045 1047 1049 1051 1053 1055 1057 1059 1061 1063 1065 1067 1069 1071 1073 1075 1077 1079 1081 1083 1085
1087 1089 1091 1093 1095 1097 1099 1101 1103 1105 1107 1109 1111 1113 1115 1117 1119 1121 1123 1125 1127 1129 1131 1133
1135 1137 1139 1141 1143 1145 1147 1149 1151 1153 1155 1157 1159 1161 1163 1165 1167 1169 1171 1173 1175 1177 1179 1181
1183 1185 1187 1189 1191 1193 1195 1197 1199 1201 1203 1205 1207 1209 1211 1213 1215 1217 1219 1221 1223 1225 1227 1229
1231 1233 1235 1237 1239 1241 1243 1245 1247 1249 1251 1253 1255 1257 1259 1261 1263 1265 1267 1269 1271 1273 1275 1277
1279 1281 1283 1285 1287 1289 1291 1293 1295 1297 1299 1301 1303 1305 1307 1309 1311 1313 1315 1317 1319 1321 1323 1325
1327 1329 1331 1333 1335 1337 1339 1341 1343 1345 1347 1349 1351 1353 1355 1357 1359 1361 1363 1365 1367 1369 1371 1373
1375 1377 1379 1381 1383 1385 1387 1389 1391 1393 1395 1397 1399 1401 1403 1405 1407 1409 1411 1413 1415 1417 1419 1421
1423 1425 1427 1429 1431 1433 1435 1437 1439 1441 1443 1445 1447 1449 1451 1453 1455 1457 1459 1461 1463 1465 1467 1469
1471 1473 1475 1477 1479 1481 1483 1485 1487 1489 1491 1493 1495 1497 1499 1501 1503 1505 1507 1509 1511 1513 1515 1517
1519 1521 1523 1525 1527 1529 1531 1533 1535 1537 1539 1541 1543 1545 1547 1549 1551 1553 1555 1557 1559 1561 1563 1565
1567 1569 1571 1573 1575 1577 1579 1581 1583 1585 1587 1589 1591 1593 1595 1597 1599 1601 1603 1605 1607 1609 1611 1613
1615 1617 1619 1621 1623 1625 1627 1629 1631 1633 1635 1637 1639 1641 1643 1645 1647 1649 1651 1653 1655 1657 1659 1661
1663 1665 1667 1669 1671 1673 1675 1677 1679 1681 1683 1685 1687 1689 1691 1693 1695 1697 1699 1701 1703 1705 1707 1709
1711 1713 1715 1717 1719 1721 1723 1725 1727 1729 1731 1733 1735 1737 1739 1741 1743 1745 1747 1749 1751 1753 1755 1757
1759 1761 1763 1765 1767 1769 1771 1773 1775 1777 1779 1781 1783 1785 1787 1789 1791 1793 1795 1797 1799 1801 1803 1805
1807 1809 1811 1813 1815 1817 1819 1821 1823 1825 1827 1829 1831 1833 1835 1837 1839 1841 1843 1845 1847 1849 1851 1853
1855 1857 1859 1861 1863 1865 1867 1869 1871 1873 1875 1877 1879 1881 1883 1885 1887 1889 1891 1893 1895 1897 1899 1901
1903 1905 1907 1909 1911 1913 1915 1917 1919 1921 1923 1925 1927 1929 1931 1933 1935 1937 1939 1941 1943 1945 1947 1949
1951 1953 1955 1957 1959 1961 1963 1965 1967 1969 1971 1973 1975 1977 1979 1981 1983 1985 1987 1989 1991 1993 1995 1997
1999
Using 4 no of threads with chunk size=100 and 0.134000 execution timen
-----
Process exited after 0.1748 seconds with return value 0
Press any key to continue . . .

```

Github link: https://github.com/SiddharthM29/HPC_lab/tree/main/Assignment%203