Title : Write a Parallel program (using OpenMP) for matrix multiplication.

1)OpenMP:

OpenMP (Open Multi-Processing) is an [application programming interface](https://en.wikipedia.org/wiki/Application_programming_interface) (API) that supports multi-platform [shared memory](https://en.wikipedia.org/wiki/Shared_memory_architecture) [multiprocessing](https://en.wikipedia.org/wiki/Multiprocessing) programming in [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C++), and [Fortran](https://en.wikipedia.org/wiki/Fortran" \o "Fortran),on most platforms, [instruction set architectures](https://en.wikipedia.org/wiki/Instruction_set_architecture) and [operating systems](https://en.wikipedia.org/wiki/Operating_system), including [Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system)), [AIX](https://en.wikipedia.org/wiki/IBM_AIX), [HP-UX](https://en.wikipedia.org/wiki/HP-UX), [Linux](https://en.wikipedia.org/wiki/Linux), [macOS](https://en.wikipedia.org/wiki/MacOS), and [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows). It consists of a set of [compiler directives](https://en.wikipedia.org/wiki/Compiler_directive), [library routines](https://en.wikipedia.org/wiki/Library_(computing)), and [environment variables](https://en.wikipedia.org/wiki/Environment_variable) that influence run-time behavior.

OpenMP is managed by the [nonprofit](https://en.wikipedia.org/wiki/Nonprofit_organization) technology [consortium](https://en.wikipedia.org/wiki/Consortium) OpenMP Architecture Review Board (or OpenMP ARB), jointly defined by a group of major computer hardware and software vendors, including [AMD](https://en.wikipedia.org/wiki/AMD), [IBM](https://en.wikipedia.org/wiki/IBM), [Intel](https://en.wikipedia.org/wiki/Intel), [Cray](https://en.wikipedia.org/wiki/Cray), [HP](https://en.wikipedia.org/wiki/Hewlett-Packard), [Fujitsu](https://en.wikipedia.org/wiki/Fujitsu), [Nvidia](https://en.wikipedia.org/wiki/Nvidia), [NEC](https://en.wikipedia.org/wiki/NEC), [Red Hat](https://en.wikipedia.org/wiki/Red_Hat), [Texas Instruments](https://en.wikipedia.org/wiki/Texas_Instruments), [Oracle Corporation](https://en.wikipedia.org/wiki/Oracle_Corporation), and more.

OpenMP uses a [portable](https://en.wikipedia.org/wiki/Software_portability), scalable model that gives [programmers](https://en.wikipedia.org/wiki/Programmer) a simple and flexible interface for developing parallel applications for platforms ranging from the standard [desktop computer](https://en.wikipedia.org/wiki/Desktop_computer) to the [supercomputer](https://en.wikipedia.org/wiki/Supercomputer).

Comprised of three primary API components:

* Compiler Directives
* Runtime Library Routines
* Environment Variables

**Goals of OpenMP:**

* **Standardization:**
  + Provide a standard among a variety of shared memory architectures/platforms
  + Jointly defined and endorsed by a group of major computer hardware and software vendors
* **Lean and Mean:**
  + Establish a simple and limited set of directives for programming shared memory machines.
  + Significant parallelism can be implemented by using just 3 or 4 directives.
  + This goal is becoming less meaningful with each new release, apparently.
* **Ease of Use:**
  + Provide capability to incrementally parallelize a serial program, unlike message-passing libraries which typically require an all or nothing approach
  + Provide the capability to implement both coarse-grain and fine-grain parallelism
* **Portability:**
  + The API is specified for C/C++ and Fortran
  + Public forum for API and membership
  + Most major platforms have been implemented including Unix/Linux platforms and Windows

**OpenMP Programming Model :**

**Shared Memory Model:**

* OpenMP is designed for multi-processor/core, shared memory machines. The underlying architecture can be shared memory UMA or NUMA.
* Because OpenMP is designed for shared memory parallel programming, it largely limited to **single node** parallelism. Typically, the number of processing elements (cores) on a node determine how much parallelism can be implemented.

[matrix multiplication](https://en.wikipedia.org/wiki/Matrix_multiplication):

[matrix multiplication](https://en.wikipedia.org/wiki/Matrix_multiplication) is such a central operation in many [numerical algorithms](https://en.wikipedia.org/wiki/Numerical_algorithm), much work has been invested in making **matrix multiplication algorithms** efficient. Applications of matrix multiplication in computational problems are found in many fields including [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing) and [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition) and in seemingly unrelated problems such as counting the paths through a [graph](https://en.wikipedia.org/wiki/Graph_(graph_theory)).[]](https://en.wikipedia.org/wiki/Matrix_multiplication_algorithm#cite_note-skiena-1) Many different algorithms have been designed for multiplying matrices on different types of hardware, including [parallel](https://en.wikipedia.org/wiki/Parallel_computing) and [distributed](https://en.wikipedia.org/wiki/Distributed_computing) systems, where the computational work is spread over multiple processors (perhaps over a network).

Directly applying the mathematical definition of matrix multiplication gives an algorithm that [takes time](https://en.wikipedia.org/wiki/Analysis_of_algorithms) on the order of *n*3 to multiply two *n* × *n* matrices (Θ(*n*3) in [big O notation](https://en.wikipedia.org/wiki/Big_O_notation)). Better asymptotic bounds on the time required to multiply matrices have been known since the work of Strassen in the 1960s, but it is still unknown what the optimal time is .

**Code:**

**Parallel programming for matrix multiplication:**

#include <omp.h>

#include <stdio.h>

#include <stdlib.h>

#define NRA 62 /\* number of rows in matrix A \*/

#define NCA 15 /\* number of columns in matrix A \*/

#define NCB 7 /\* number of columns in matrix B \*/

int main (int argc, char \*argv[])

{

int tid, nthreads, i, j, k, chunk;

double a[NRA][NCA], /\* matrix A to be multiplied \*/

b[NCA][NCB], /\* matrix B to be multiplied \*/

c[NRA][NCB]; /\* result matrix C \*/

chunk = 10; /\* set loop iteration chunk size \*/

/\*\*\* Spawn a parallel region explicitly scoping all variables \*\*\*/

#pragma omp parallel shared(a,b,c,nthreads,chunk) private(tid,i,j,k)

{

tid = omp\_get\_thread\_num();

if (tid == 0)

{

nthreads = omp\_get\_num\_threads();

printf("Starting matrix multiple example with %d threads\n",nthreads);

printf("Initializing matrices...\n");

}

/\*\*\* Initialize matrices \*\*\*/

#pragma omp for schedule (static, chunk)

for (i=0; i<NRA; i++)

for (j=0; j<NCA; j++)

a[i][j]= i+j;

#pragma omp for schedule (static, chunk)

for (i=0; i<NCA; i++)

for (j=0; j<NCB; j++)

b[i][j]= i\*j;

#pragma omp for schedule (static, chunk)

for (i=0; i<NRA; i++)

for (j=0; j<NCB; j++)

c[i][j]= 0;

/\*\*\* Do matrix multiply sharing iterations on outer loop \*\*\*/

/\*\*\* Display who does which iterations for demonstration purposes \*\*\*/

printf("Thread %d starting matrix multiply...\n",tid);

#pragma omp for schedule (static, chunk)

for (i=0; i<NRA; i++)

{

printf("Thread=%d did row=%d\n",tid,i);

for(j=0; j<NCB; j++)

for (k=0; k<NCA; k++)

c[i][j] += a[i][k] \* b[k][j];

}

} /\*\*\* End of parallel region \*\*\*/

/\*\*\* Print results \*\*\*/

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("Result Matrix:\n");

for (i=0; i<NRA; i++)

{

for (j=0; j<NCB; j++)

printf("%6.2f ", c[i][j]);

printf("\n");

}

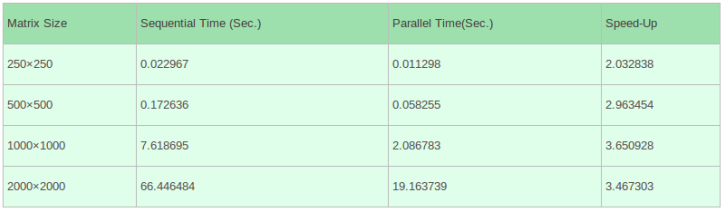
printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf ("Done.\n");

}

### Experimental results

We have described a simple parallel algorithm for the matrix-matrix product using OpenMP. But what about its performance? Lets see them now. All executions, times and speed-ups are measured using the following machine:

[](https://i1.wp.com/www.appentra.com/wp-content/uploads/2014/06/table_mat_mul.png)