The basic code block that does the computation can be written in two ways as shown in the table below. The code on right side is faster than that on the left column of the table as it is cache-friendly. It might not make much difference for small values of NC but for larger values the speedup is upto 2. We are going to use this code for matrix multiplication in all files.

```
for( i=0; i<NR; i++ ){
  for( j=0; j<NC; j++ ){
    for( k=0; k<NC; k++){
          C[i][j] += A[i][k]*B[k][j];
}
}
}
```

```
for( i=0; i<NR; i++ ){
  for( k=0; k<NC; k++){
    for( j=0; j<NC; j++ ){
          C[i][j] += A[i][k]*B[k][j];
}
}
}
```

Table 1: Matrix Multiplication; right is cache-friendly

| N | a | b |
|---|---|---|
| 100 | 0.008075 | 0.008205 |
| 200 | 0.063293 | 0.064621 |
| 400 | 0.594145 | 0.610361 |
| 800 | 5.386019 | 4.080133 |
| 1000 | 9.487167 | 5.573937 |
| 2000 | 87.209641 | 45.544228 |
| 5000 | 1908.666260 | 703.574890 |
| 8000 | 8362.995117 | 2878.790771 |
| 10000 | 17790.554688 | 6591.408691 |

**Results**

| N | serial | omp n=2 | omp n=4 | omp n=8 | omp n= 24 | mpi n=2 | mpi n=4 | mpi n=8 | mpi n= |
|---|---|---|---|---|---|---|---|---|---|
| 100 | 0.008205 | 0.009146 | 0.002916 | 0.001719 | 0.006868 | 0.004521 | 0.004978 | 0.002792 | 0.00507 |
| 200 | 0.064621 | 0.006868 | 0.041666 | 0.021512 | 0.010956 | 0.058968 | 0.033004 | 0.018016 | 0.02257 |
| 400 | 0.610361 | 0.329870 | 0.186483 | 0.105709 | 0.080594 | 0.345524 | 0.250767 | 0.183066 | 0.09222 |
| 800 | 4.080133 | 2.195364 | 1.172719 | 0.573523 | 0.486476 | 2.328038 | 1.368429 | 0.696170 | 0.58859 |
| 1000 | 5.573937 | 2.841520 | 2.338277 | 0.804072 | 0.593531 | 2.898817 | 2.535994 | 1.165923 | 0.71774 |
| 2000 | 45.544228 | 22.061188 | 11.366979 | 5.903165 | 4.064920 | 22.534018 | 11.835779 | 10.047218 | 4.58103 |
| 5000 | 703.574890 | 342.611450 | 175.706894 | 90.065071 | 62.474705 | 346.948578 | 179.307938 | 91.601669 | 57.7127 |
| 8000 | 2878.790771 | 1400.647705 | 697.816223 | 363.321991 | 234.618332 | 1434.890625 | 715.344727 | 636.908081 | 233.626 |
| 10000 | 6591.408691 | 2737.574951 | 1798.366455 | 740.366760 | 499.029694 | 2812.980469 | 1680.102051 | 877.236450 | 557.603 |

Table 2: Time taken. N is size of matrix, n is no. of threads/procesess