OMP performance

This is the result of openMP version of matrix multiplication
All time units are second.

Matrix Size	Sequential	Parallel	Speed Up
1024(2 threads)	7.805551	4.759242	1.640083
1024(4 threads)	7.805551	2.447019	3.189820
1024(8 threads)	7.805551	1.256499	6.212143
1024(16 threads)	7.805551	0.633837	12.314752

Matrix Size	Sequential	Parallel	Speed Up
2048(2 threads)	179.425026	98.992216	1.812517
2048(4 threads)	179.425026	48.245036	3.719036
2048(8 threads)	179.425026	24.113031	7.440998
2048(16 threads)	179.425026	12.101211	14.827031

Matrix Size	Sequential	Parallel	Speed Up
4096(2 threads)	924.223986	410.879718	2.249378
4096(4 threads)	924.223986	203.997306	4.530570
4096(8 threads)	924.223986	101.451543	9.110004
4096(16 threads)	924.223986	50.064780	18.460562

MPI performance and analysis

Karp-Flatt Metric calculated by e = $\frac{1/\psi - 1/p}{1 - 1/p}$

This is the result of block strip algorithm.

All time units are second.

Matrix Size	Sequential	Parallel	Speed Up	e
1024(4 cores)	11.903415	1.963964	6.060913	0.0233495
1024(8 cores)	11.903415	1.927598	6.175258	0.0422132
1024(16 cores)	11.903415	2.109852	5.641824	0.1223975

Matrix Size	Sequential	Parallel	Speed Up	e
2048(4 cores)	179.425026	20.721067	8.659063	0.0138421
2048(16 cores)	179.425026	20.254976	8.858318	0.0537475
2048(32 cores)	179.425026	32.049514	5.598370	0.1521274

Matrix Size	Sequential	Parallel	Speed Up	e
4096(4 cores)	924.223986	195.475477	4.728081	0.0051330
4096(16 cores)	924.223986	163.554053	12.246455	0.0204334
4096(32 cores)	924.223986	250.603269	3.687997	0.2476386

Result of Cannon's algorithm distribute using checkerboard.

All time units are second. Since Cannon's algorithm require processor number be a complete square of an integer, in our case, using 4 cores and 16 cores.

Matrix Size	Sequential	Parallel	Speed Up	e
1024(4 cores)	11.903415	0.679364	17.521411	-0.257236
1024(16 cores)	11.903415	0.195235	60.969673	-0.049172

Matrix Size	Sequential	Parallel	Speed Up	e
2048(4 cores)	179.425026	6.021917	29.795333	-0.2885836
2048(16 cores)	179.425026	2.145920	83.612177	-0.0539094

Matrix Size	Sequential	Parallel	Speed Up	e
4096(4 cores)	924.223986	61.927173	14.924369	-0.243994
4096(16 cores)	924.223986	15.441184	59.854476	-0.048846

Iso-Efficiency Analysis:

T_0 = total overhead

T_1 = single processor time

= W(units of work) * t_c(time per unit of work)

T_p = parallel time

Total time = P T_p = T_1 + T_0

$S = \frac{T1}{Tp}$

For A * B = C

```
for (i=0; i<m; i++){
      for (j=0; j<n; j++){
         for (k=0; k<p; k++){
            mat_3[i][j] += mat_1[i][k]*mat_2[k][j];
         }
      }
   }
```

T_1 is n^3 for sequential computing.

For parallel computing:
Assume A and B are pregenerated.
Initial alignment step, max distance over which block shifts is $\sqrt{p} - 1$
Circular shit operations in row and column take time: $t_{comm} = 2(t_{start} + (n^2 * t_c)/p)$
Each of $\sqrt{p}$ step compute and shift take time: $t_s + (n^2 * t_c)/p$
Multiplying $\sqrt{p}$ sub-matrices of size $(n/\sqrt{p}) * (n/\sqrt{p})$ take time: n^3/p
$T_p = n^3/p + 2\sqrt{p}\ (t_{start} + (n^2 * t_c)/p) + 2\ (t_{start} + (n^2 * t_c)/p)$
$T_0 = T_p - T_1$
$= (p*t_{start} + n^2 * t_c) * (2\sqrt{p} + 2)$
$= \sqrt{p} * (p*t_{start} + n^2 * t_c)$
if we assume $t_{sart} = t_c = 1$
Iso-efficiency will be $\sqrt{p} * (p + n^2)$
For example, if processor number goes from 4 to 16, T_0 goes from $8+2n^2$ to $64+4n^2$.
When n is large enough, 8 and 64 can be neglect, which means workload has to be increase twice to maintain the efficiency.
In the table above, efficiency always approximately 4 when the matrix size increase 2 times than before.