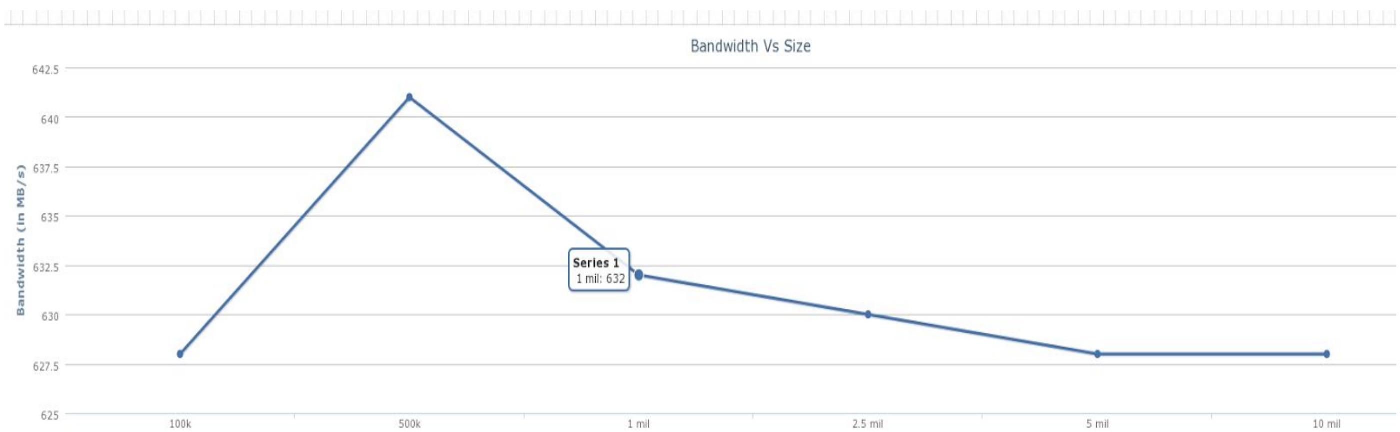
CS525 - Parallel Computing - Homework 1

[AYUSH JAIN - jain207@purdue.edu]

All plots in Plot Folder
Programs:

Q1. - membw.c
Q2. memspacing.c
Q3. loopic.c
Loopic2.c
Q4. matrixmul.c
Minor additions for measurements in given code

Q1.



X- Axis - Vector size 100,000 upto 10 million

Detailed Plot as separate png : **1-Plot.png**

Measurements

size = 100000

Elapsed time = 0.060672 seconds
Bytes Accessed 40000000
Bandwidth 628.740515

size = 500000

Elapsed time = 0.002973 seconds
Bytes Accessed 2000000
Bandwidth 641.539695

size = 1 mil

Elapsed time = 0.006027 seconds
Bytes Accessed 4000000
Bandwidth 632.911392

2.5 mil

Elapsed time = 0.015134 seconds
Bytes Accessed 10000000
Bandwidth 630.159430

5 mil

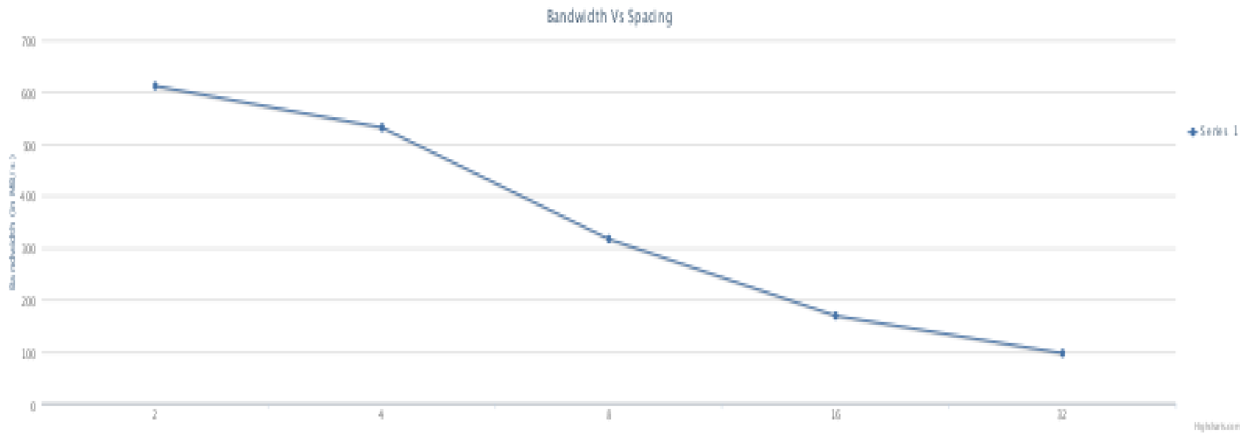
Elapsed time = 0.030354 seconds
Bytes Accessed 20000000
Bandwidth 628.367658

10 mil

Elapsed time = 0.060727 seconds
Bytes Accessed 40000000
Bandwidth 628.172763

Average Value of Bandwidth = 631.643 MB/s

Q2.



X-Axis - Spacing from 2-32

Detailed Plot as separate png : **2-Plot.png**

Measurements

s= 2

Elapsed time = 0.062309 seconds
Bytes Accessed 40000000
Bandwidth 612.222252

s= 4

Elapsed time = 0.071659 seconds
Bytes Accessed 40000000
Bandwidth 532.341404

s = 8
Elapsed time = 0.120177 seconds
Bytes Accessed 40000000
Bandwidth 317.423789

s = 16
Elapsed time = 0.224580 seconds
Bytes Accessed 40000000
Bandwidth 169.859134

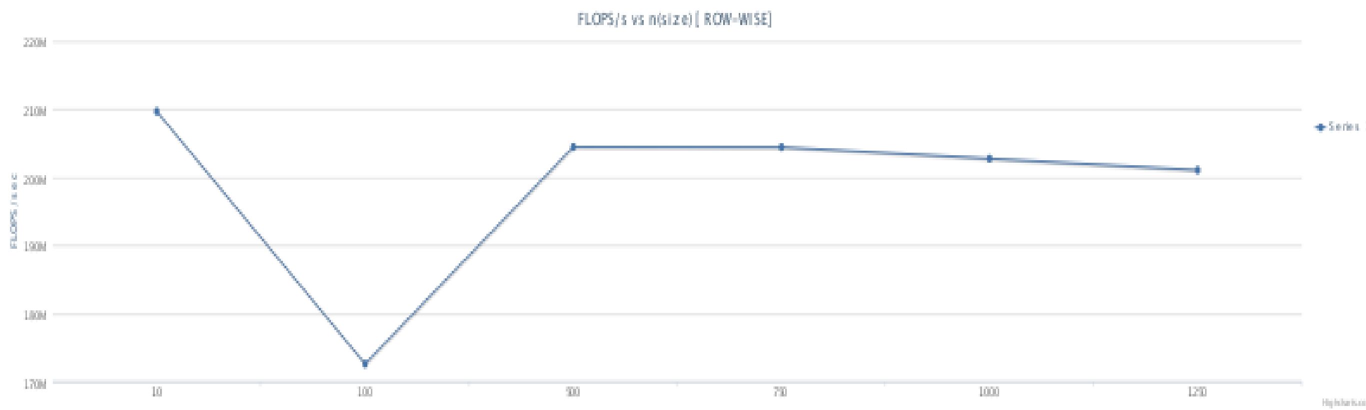
s = 32

Elapsed time = 0.390289 seconds
Bytes Accessed 40000000
Bandwidth 97.740305

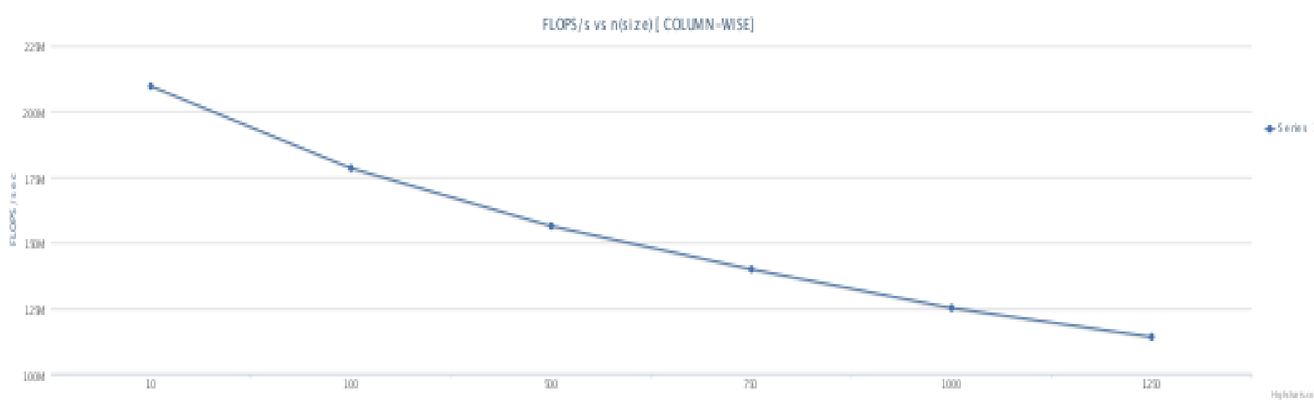
Comparing 1-Plot and 2-Plot

We see that Bandwidth decreases as spacing increases. With spacing = 0, bandwidth is maximum. This is because strided access does not access continuous locations of memory

Q3.



Detailed Plot as separate png : **3-1-Plot.png**



Detailed Plot as separate png : **3-2-Plot.png**

Measurements

Row - wise

Elapsed time = 0.000001 seconds
Size 10
FLOPS/s 209715200.000000

Elapsed time = 0.000116 seconds
Size 100
FLOPS/s 172605102.880658

Elapsed time = 0.002445 seconds
Size 500
FLOPS/s 204500438.810336

Elapsed time = 0.005502 seconds
Size 750
FLOPS/s 204471638.427872

Elapsed time = 0.009864 seconds
Size 1000
FLOPS/s 202760514.357536

Elapsed time = 0.015540 seconds
Size 1250
FLOPS/s 201095444.851869

209715200.00, 172605102.88, 204500438.81, 204471638.42, 202760514.35, 201095444.85

Column-Wise - 2nd Version

Elapsed time = 0.000001 seconds
Size 10
FLOPS/s 209715200.000000

Elapsed time = 0.000112 seconds
Size 100
FLOPS/s 178481021.276596

Elapsed time = 0.003196 seconds
Size 500
FLOPS/s 156445505.408430

Elapsed time = 0.008038 seconds
Size 750

FLOPS/s 139959423.384944

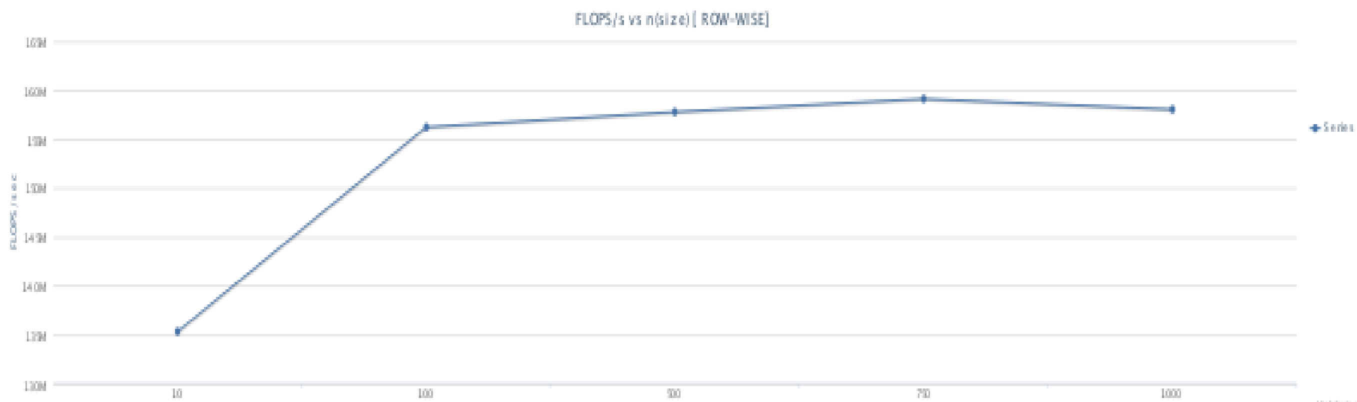
Elapsed time = 0.015970 seconds
Size 1000
FLOPS/s 125234880.492065

Elapsed time = 0.027373 seconds
Size 1250
FLOPS/s 114164271.404930

As expected, row-wise has more FlOPS/s than COLUMN wise version because values are stored in row-wise manner. Thus, their access from cache is more.

Although the maximum FLOPS/s for access is same for both versions when whole matrix and vector can be accomodated in the cache.
Max. FLOPS/s : 209715200.000000

Q4.



Detailed Plot as separate png : **4-Plot.png**

As matrix size increases, the FLOPS/s reaches a saturation. This is because the access occur in row wise manner which increases till whole matrices can be accomodated in the cache.

Elapsed time = 0.000015 seconds
Size (n) 10
FLOPS/s 135300129.032258

Elapsed time = 0.012799 seconds
Size 100
FLOPS/s 156261907.866550

Elapsed time = 1.584321 seconds

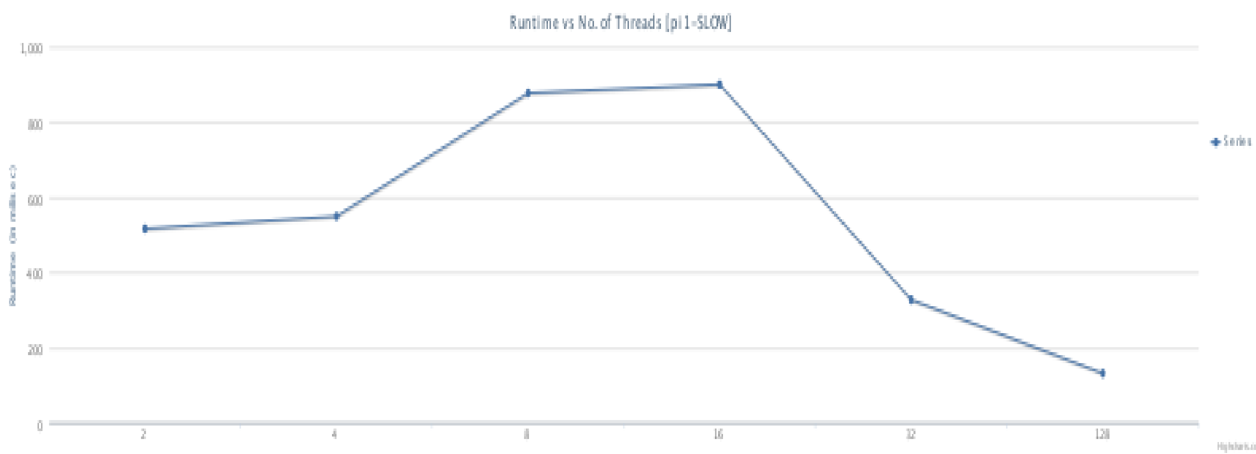
Size 500
FLOPS/s 157796302.973428

Elapsed time = 5.302780 seconds
Size 750
FLOPS/s 159114655.680823

Elapsed time = 12.653145 seconds
Size 1000
FLOPS/s 158063468.660941

Q5.

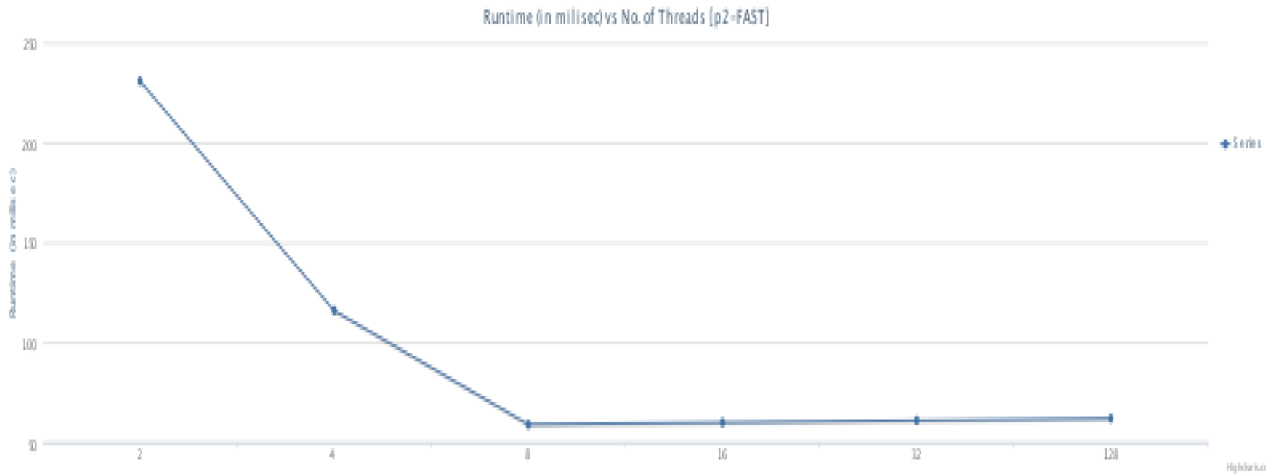
Pi1-Program



<u>Runtime (in milisec) vs No. of Threads [pi1-SLOW]</u>

Detailed Plot as separate png : **5-1-Plot.png**

Pi2- Program



<u>Runtime (in milisec) vs No. of Threads [p2-FAST]</u>

Detailed Plot as separate png : **5-1-Plot.png**

2nd Pi2 is much faster as in first version global variable is accessed and as the number of threads > number of processors, the threads get swapped and cache access time also comes into picture.

While in second version, local variable is incremented and final output is written to global variable. Thus, it is much faster

Measurements

10 million darts

Slow method- Pi1

Enter number of threads (use a number between 1 and 512): 2
2 Computed PI = 3.141602
Elapsed time = 0.517995 seconds

Enter number of threads (use a number between 1 and 512): 4
4 Computed PI = 3.141234
Elapsed time = 0.550369 seconds

Enter number of threads (use a number between 1 and 512): 8
8 Computed PI = 3.142049
Elapsed time = 0.878424 seconds

Enter number of threads (use a number between 1 and 512): 16

16 Computed PI = 3.141364
Elapsed time = 0.901104 seconds

Enter number of threads (use a number between 1 and 512): 32
32 Computed PI = 3.142031
Elapsed time = 0.328439 seconds

Enter number of threads (use a number between 1 and 512): 128
128 Computed PI = 3.145512
Elapsed time = 0.132442 seconds

0.517, 0.550, 0.878, 0.901, 0.328, 0.132

Fast Method- Pi2

Enter number of threads (use a number between 1 and 512): 2
2 Computed PI = 3.141602
Elapsed time = 0.231605 seconds

Enter number of threads (use a number between 1 and 512): 4
4 Computed PI = 3.141231
Elapsed time = 0.116828 seconds

Enter number of threads (use a number between 1 and 512): 8
8 Computed PI = 3.142038
Elapsed time = 0.059487 seconds

Enter number of threads (use a number between 1 and 512): 16
16 Computed PI = 3.141316
Elapsed time = 0.060054 seconds

Enter number of threads (use a number between 1 and 512): 32
32 Computed PI = 3.141832
Elapsed time = 0.061274 seconds

Enter number of threads (use a number between 1 and 512): 128
128 Computed PI = 3.142261
Elapsed time = 0.062470 seconds

Q6.

Startup time is calculated using message size = 0 and dividing the elapsed time by (2*(ping_pong_limit).

Time per word transfer (in nanoseconds) is calculated by having a large message size so that startup time does not dominate the measurement of elapsed time.

It is calculated by formula (1/Bandwidth(in word/sec) * 10^9)
Bandwidth = (Message_size * 4)/Time per message transfer in sec

Measurements

The difference between 3rd , 4th and 5th elapsed time is as follows:
3rd - Total Elapsed time for ping_pong_limit iterations
4th - Time per message transfer in milliseconds (3rd Divided by 2*ping_pong_limit)*1000
5th - Time per message transfer in microseconds (3rd Divided by 2*ping_pong_limit)*1000000

Enter ping pong message size: 0
Enter ping pong limit: 1000
Elapsed time = 0.000639 seconds
Elapsed time = 0.000319 ms
Elapsed time = 0.319481 micros
Bandwidth in bytes/second 0.000000

Enter ping pong message size: 100000
Enter ping pong limit: 1000
Elapsed time = 0.440746 seconds
Elapsed time = 0.220373 ms
Elapsed time = 220.373034 micros
Bandwidth in bytes/second 1815104107.219265
Per word transfer rate 2.203730 in nanoseconds

Enter ping pong message size: 250000
Enter ping pong limit: 1000
Elapsed time = 0.968907 seconds
Elapsed time = 0.484454 ms
Elapsed time = 484.453559 micros
Bandwidth in bytes/second 2064181347.383579
Per word transfer rate 1.937814 in nanoseconds

Enter ping pong message size: 500000
Enter ping pong limit: 1000

Elapsed time = 2.068677 seconds
Elapsed time = 1.034338 ms
Elapsed time = 1034.338474 micros
Bandwidth in bytes/second 1933603022.361139
Per word transfer rate 2.068677 in nanoseconds

Enter ping pong message size: 750000
Enter ping pong limit: 1000
Elapsed time = 3.360457 seconds
Elapsed time = 1.680228 ms
Elapsed time = 1680.228472 micros
Bandwidth in bytes/second 1785471470.355901
Per word transfer rate 2.240305 in nanoseconds

Enter ping pong message size: 1000000
Enter ping pong limit: 1000
Elapsed time = 4.621758 seconds
Elapsed time = 2.310879 ms
Elapsed time = 2310.878992 micros
Bandwidth in bytes/second 1730943079.974277
Per word transfer rate 2.310879 in nanoseconds

Taking average of Per-word Transfer rate (of message size >=100000) = 2.2 nanoseconds

Detailed Plot as 6.png

Q7. For multiple ping pong, measurements are as follows:

N= 2

Enter ping pong message size: 0
Enter ping pong limit: 1000
Elapsed time = 0.000629 seconds
Elapsed time = 0.000314 ms
Elapsed time = 0.314474 micros
Bandwidth in bytes/second 0.000000
Per word transfer rate inf in nanoseconds

Enter ping pong message size: 100000
Enter ping pong limit: 1000

Elapsed time = 0.438839 seconds
Elapsed time = 0.219419 ms
Elapsed time = 219.419479 micros
Bandwidth in bytes/second 1822992202.644320
Per word transfer rate 2.194195 in nanoseconds

Enter ping pong message size: 500000
Enter ping pong limit: 1000
Elapsed time = 1.992963 seconds
Elapsed time = 0.996481 ms
Elapsed time = 996.481419 micros
Bandwidth in bytes/second 2007062011.041390
Per word transfer rate 1.992963 in nanoseconds

Enter ping pong message size: 1000000
Enter ping pong limit: 1000
Elapsed time = 4.613734 seconds
Elapsed time = 2.306867 ms
Elapsed time = 2306.867003 micros
Bandwidth in bytes/second 1733953450.300219
Per word transfer rate 2.306867 in nanoseconds

N = 4

Enter ping pong message size: 0
Enter ping pong limit: 1000
Elapsed time = 0.001159 seconds
Elapsed time = 0.000579 ms
Elapsed time = 0.579476 micros
Bandwidth in bytes/second 0.000000
Per word transfer rate inf in nanoseconds

Enter ping pong message size: 0
Enter ping pong limit: 1000
Elapsed time = 0.000625 seconds
Elapsed time = 0.000312 ms
Elapsed time = 0.312448 micros
Bandwidth in bytes/second 0.000000
Per word transfer rate inf in nanoseconds

Enter ping pong message size: 0

Enter ping pong limit: 1000
Elapsed time = 0.001213 seconds
Elapsed time = 0.000607 ms
Elapsed time = 0.606537 micros
Bandwidth in bytes/second 0.000000
Per word transfer rate inf in nanoseconds

Enter ping pong message size: 0
Enter ping pong limit: 1000
Elapsed time = 0.001324 seconds
Elapsed time = 0.000662 ms
Elapsed time = 0.662088 micros
Bandwidth in bytes/second 0.000000
Per word transfer rate inf in nanoseconds

Enter ping pong message size: 100000
Enter ping pong limit: 1000
Elapsed time = 0.114932 seconds
Elapsed time = 0.057466 ms
Elapsed time = 57.466030 micros
Bandwidth in bytes/second 6960633945.981828
Per word transfer rate 0.574660 in nanoseconds

Enter ping pong message size: 500000
Enter ping pong limit: 1000
Elapsed time = 2.919983 seconds
Elapsed time = 1.459992 ms
Elapsed time = 1459.991574 micros
Bandwidth in bytes/second 1369870919.273044
Per word transfer rate 2.919983 in nanoseconds

Enter ping pong message size: 750000
Enter ping pong limit: 1000
Elapsed time = 3.949444 seconds
Elapsed time = 1.974722 ms
Elapsed time = 1974.722028 micros
Bandwidth in bytes/second 1519201162.390797
Per word transfer rate 2.632963 in nanoseconds

Enter ping pong message size: 1000000
Enter ping pong limit: 1000
Elapsed time = 6.384290 seconds

Elapsed time = 3.192145 ms
Elapsed time = 3192.144990 micros
Bandwidth in bytes/second 1253075913.710586
Per word transfer rate 3.192145 in nanoseconds

N = 8

Enter ping pong message size: 0
Enter ping pong limit: 1000
Elapsed time = 0.000701 seconds
Elapsed time = 0.000350 ms
Elapsed time = 0.350475 micros
Bandwidth in bytes/second 0.000000
Per word transfer rate inf in nanoseconds

Enter ping pong message size: 100000
Enter ping pong limit: 1000
Elapsed time = 0.097661 seconds
Elapsed time = 0.048830 ms
Elapsed time = 48.830390 micros
Bandwidth in bytes/second 8191620017.626135
Per word transfer rate 0.488304 in nanoseconds

Enter ping pong message size: 500000
Enter ping pong limit: 1000
Elapsed time = 5.907197 seconds
Elapsed time = 2.953598 ms
Elapsed time = 2953.598499 micros
Bandwidth in bytes/second 677140105.696589
Per word transfer rate 5.907197 in nanoseconds

Enter ping pong message size: 750000
Enter ping pong limit: 1000
Elapsed time = 4.758958 seconds
Elapsed time = 2.379479 ms
Elapsed time = 2379.479051 micros
Bandwidth in bytes/second 1260780169.170969
Per word transfer rate 3.172639 in nanoseconds

Enter ping pong message size: 1000000
Enter ping pong limit: 1000
Elapsed time = 12.862611 seconds
Elapsed time = 6.431305 ms
Elapsed time = 6431.305408 micros
Bandwidth in bytes/second 621957712.461793
Per word transfer rate 6.431305 in nanoseconds

N = 16

Enter ping pong message size: 0
Enter ping pong limit: 1000
Elapsed time = 0.004965 seconds
Elapsed time = 0.002482 ms
Elapsed time = 2.482414 micros
Bandwidth in bytes/second 0.000000

Enter ping pong message size: 100000
Enter ping pong limit: 1000
Elapsed time = 1.474517 seconds
Elapsed time = 0.737258 ms
Elapsed time = 737.258434 micros
Bandwidth in bytes/second 542550592.021566
Per word transfer rate 7.372584 in nanoseconds

Enter ping pong message size: 500000
Enter ping pong limit: 1000
Elapsed time = 8.325813 seconds
Elapsed time = 4.162907 ms
Elapsed time = 4162.906528 micros
Bandwidth in bytes/second 480433559.288166
Per word transfer rate 8.325813 in nanoseconds

Enter ping pong message size: 750000
Enter ping pong limit: 1000
Elapsed time = 21.406239 seconds
Elapsed time = 10.703120 ms
Elapsed time = 10703.119516 micros
Bandwidth in bytes/second 280292114.407474
Per word transfer rate 14.270826 in nanoseconds

Enter ping pong message size: 1000000
Enter ping pong limit: 1000
Elapsed time = 25.027862 seconds
Elapsed time = 12.513931 ms
Elapsed time = 12513.931036 micros
Bandwidth in bytes/second 319643762.499111
Per word transfer rate 12.513931 in nanoseconds

Startup time shown in black which remains constant with increase in number of processors

Per-word transfer rate also increase with increase in number of processors

Theoretical Questions

2.6.
Mean access time = $0.8 \times 1 + 0.1 \times 100 + 0.8 \times 400$ = 50ns (approx)
Assuming 1 FLOP/word

Computation rate = 20 MFLOPS

Mean access time for serial computation = $0.7 \times 1 + 0.3 \times 100$ = 30ns (approx)

Computation rate = 33 MFLOPS

Fractional CPU rate = 20/33 = 0.60 (approx.)

2.12

Let us consider a cycle S(1) , S(2) , . . . , S (k)in a hypercube.
As we travel from node S(i) to S(i +1), the number of ones in the processor label (parity) must change.
Since S(1) = S(k) , the number of parity changes must be even. Therefore, there can be no cycles of odd length in a hypercube. Hence Proved

2.13

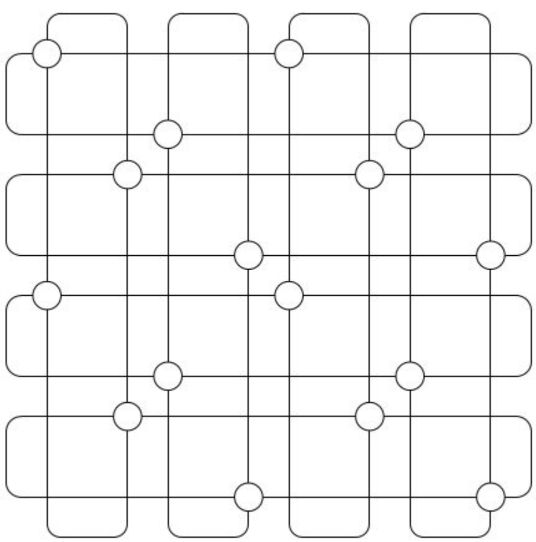
If we consider a 2-D processor hypercube by fixing k of the d bits in the processor label, we can change the remaining d − k bits. There are 2^(d−k) distinct processors that have identical values at the remaining k bit positions.

We will first prove that each processor in a group has d − k communication links going to other processors in the same group in order to prove that the 2^(d−k) processors are connected in a hypercube topology

A p-processor hypercube has the property that every processor has log p communication links, one each to a processor whose label differs in one bit position.
Since the selected d bits are fixed for each processor in the group, no communication link corresponding to these bit positions exists between processors within a group.

Also, since all possible combinations of the d − k bits are allowed for any processor, all d − k processors that differ along any of these bit positions are also in the same group. Since the processor will be connected to each of these processors, each processor within a group is connected to d − k other processors. Thus, the processors in the group are connected in a hypercube topology

2.20



Equal Wire Length 2-D mesh