ECSE 420 - Parallel Computing
Student: Daniel Macario
ID: 260503662

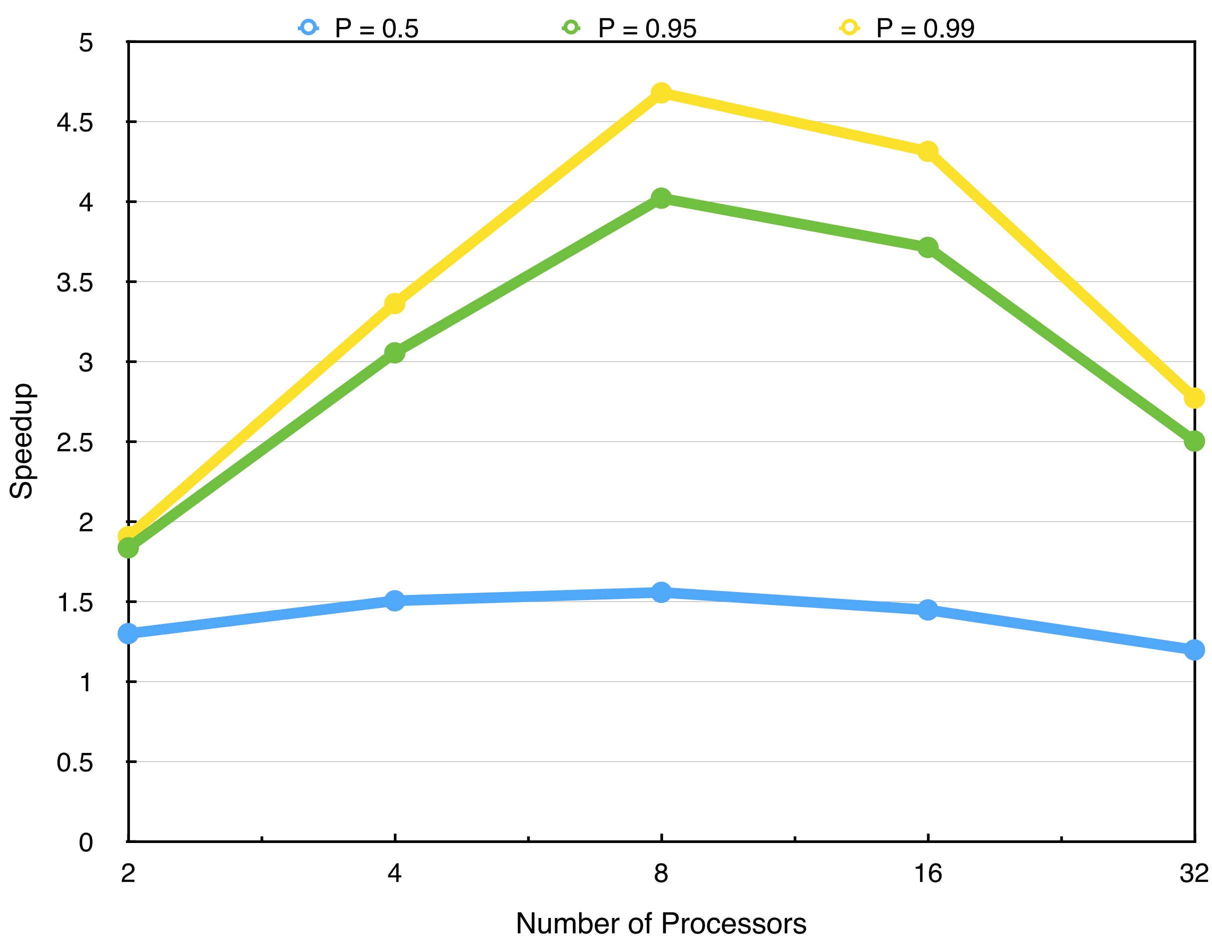
Theoretical Assignment 2

1. Amdahl's Law (25%)

a. Write a more restrictive version of Equation 1 which takes the overhead OV(n) into account.

$$S(n)\ \ with\ \ overhead = \frac{T_1}{T_n} = \frac{T_1}{T_1(1-P) + T_1(\frac{P}{n}) + OV(n)}$$

b. Assume that OV(n) = n. Plot the maximum possible speedup for P = 0.5, 0.95, 0.99, N = 2,4,8,16,32, and T_1 = 100 seconds.



2. (%)

a. How much data must be communicated per step?

In order two answer this question, we make the following assumptions: Each element is a 64 bit floating point number (8 bytes), each processor gets its own element (there are 1200 processors), and elements across the diagonal or that don't have an opposite element (because it is not a square matrix) are not swapped.

Number of elements communicated every step = (30 * 29) * 8 Bytes = 6960 Bytes

b. For a machine with the message start-up time T0 of 500 ns, what is the required asymptotic peak bandwidth for reaching the half of the peak bandwidth(in Byte)?

The same assumptions discussed above apply here. From the linear model of data transfer:

$$Bandwidth = \frac{n_{1/2}}{T_0} = \frac{6960 \;\; Bytes}{5 \times 10^{-7} \;\; seconds} = 13920 \;\; MB/s$$

3. (20%)

a. For a machine with the communication overhead and network delay (message start-up time) of 200 ns, the biggest assist occupancy and the asymptotic peak bandwidth of 5 GB/s, calculate the communication time.

$$Communication \;\; Time = Overhead + Network \;\; Delay + Occupancy$$

$$Communication \;\; Time = 200 \times 10^{-9} + \frac{1}{5 \times 10^{-9}} = 200.2 \;\; ns$$

b. **Considering a program that run 100 times an operation, assuming that 10% of operation communication time is during other useful work of processor, calculate the the communication cost.**

$$Communication\ \ Cost\ =\ Frequency\ \times\ (Communication\ \ Time\ -\ Overlap)$$

$$Communication\ \ Cost\ =\ 100\ \times\ (200.2\times10^{-9}\ -\ 0.1\ \times\ 200.2\times10^{-9})\ =\ 18018\ \ ns$$

4. MPI - Function Parallelism (35%)

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

int main(int argc, char *argv[]) {
  int myrank;
  int npes;
  MPI_Status status;
  MPI_Init(&argc, &argv);
  MPI_Comm_size(MPI_COMM_WORLD, &npes);
  MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

  if (myrank == 0) {
    int A, D, AmultB, BdivCplusA, secondMult;
    A = 1;

    // broadcast A to all processes
    MPI_Bcast(&A, 1, MPI_INT, 0, MPI_COMM_WORLD);

    // receive result from (A * B)
    MPI_Recv(&AmultB, 1, MPI_INT, 1, 7, MPI_COMM_WORLD, &status);

    // receive result from B/(C + A)
    MPI_Recv(&BdivCplusA, 1, MPI_INT, 2, 7, MPI_COMM_WORLD, &status);

    // receive result from (A - 1)*(A - 2)
    MPI_Recv(&secondMult, 1, MPI_INT, 3, 7, MPI_COMM_WORLD, &status);

    // calculate final result
    D = AmultB + BdivCplusA + secondMult;

    printf("Result form expression is: %d\n", D);
  }
```

```
if (myrank == 1) {
  // first multipication
  int B, receivedA, AmultB;
  B = 2;

  // receive A
  MPI_Bcast(&receivedA, 1, MPI_INT, 0, MPI_COMM_WORLD);
  // Send B
  MPI_Send(&B, 1, MPI_INT, 2, 7, MPI_COMM_WORLD);

  AmultB = B * receivedA;
  // send result of A * B
  MPI_Send(&AmultB, 1, MPI_INT, 0, 7, MPI_COMM_WORLD);

}

if (myrank == 2) {
  //adition-division
    int C, receivedA, receivedB, CplusA, BdivCplusA, Aminus1, Aminus2, secondMult,
valueD, AmultB;
  C = 3;

  // recieve A
  MPI_Bcast(&receivedA, 1, MPI_INT, 0, MPI_COMM_WORLD);
  // receive B
  MPI_Recv(&receivedB, 1, MPI_INT, 1, 7, MPI_COMM_WORLD, &status);

  CplusA = C + receivedA;

  // result from B/(C + A)
  BdivCplusA = receivedB / CplusA;

  MPI_Send(&BdivCplusA, 1, MPI_INT, 0, 7, MPI_COMM_WORLD);
}

if (myrank == 3) {
  int receivedA, secondMult;

  // receive A
  MPI_Bcast(&receivedA, 1 , MPI_INT, 0 , MPI_COMM_WORLD);

  secondMult = (receivedA - 1) * (receivedA - 2);
  // send result from second multiplication
  MPI_Send(&secondMult, 1, MPI_INT, 0, 7, MPI_COMM_WORLD);
}
```

```
  MPI_Finalize();
  return 0;
}
```

Sample execution:

```
$ mpirun -np 4 ./a.out
$ Result form expression is: 11
```