PERFORMANCE EVALUATION OF MPI PROGRAMS

- The run-times of serial programs and the run-times of corresponding parallel programs.

- We denote the serial run-time by Tserial. Since it typically depends on the size of the input, n, we'll frequently denote it as Tserial(n).

- We denote the parallel run-time by Tparallel. Since it depends on both the input size, n, and the number of processes, comm._sz = p, we'll frequently denote it as Tparallel.(n,p).

- The parallel program will divide the work of the serial program among the processes, and add in some overhead time, which we denoted Toverhead:

$$T_{\text{parallel}}(n,p) = T_{\text{serial}}(n)/p + T_{\text{overhead}}.$$

- In MPI programs, the parallel overhead typically comes from communication, and it can depend on both the problem size and the number of processes.

- If the serial program multiplies an ***nxn*** matrix by an n-dimensional vector, then each process in the parallel program multiplies an ***n/pxn*** matrix by an n-dimensional vector. The local matrix-vector multiplication part of the parallel program therefore executes $\boldsymbol{n^2/p}$ floating point operations. Thus, it appears that this local matrix-vector multiplication reduces the work per process by a factor of ***p***.
- However, the parallel program also needs to complete a call to MPI_Allgather before it can carry out the local matrix-vector multiplication. In our example, it appears that

$$T_{\text{parallel}}(n,p) = T_{\text{serial}}(n)/p + T_{\text{allgather}}.$$

Speedup and efficiency

- The relation between the serial and the parallel run-times is the speedup. It's just the ratio of the serial run-time to the parallel run-time:

$$S(n,p) = \frac{T_{\text{serial}}(n)}{T_{\text{parallel}}(n,p)}.$$

- The ideal value for S(n,p) is p. If S(n,p) = p, then our parallel program with comm_ sz = p processes is running p times faster than the serial program. In practice, this speedup, sometimes called linear speedup, is rarely achieved.

- Another widely used measure of parallel performance is parallel efficiency. This is “per process” speedup:

$$E(n,p) = \frac{S(n,p)}{p} = \frac{T_{\text{serial}}(n)}{p \times T_{\text{parallel}}(n,p)}.$$

- Linear speedup corresponds to a parallel efficiency of p/p =1.0, and, in general, we expect that our efficiencies will be less than 1.

Scalability

- A program is scalable if the problem size can be increased at a rate so that the efficiency doesn’t decrease as the number of processes increase.
  Consider two parallel programs: program A and program B. Suppose that if $p \geq 2$, the efficiency of program A is 0.75, regardless of problem size.

- Also suppose that the efficiency of program B is $n/(625p)$ provided $p \geq 2$ and $1000 \leq n \leq 625p$. Then according to our “definition,” both programs are scalable.

- For program A, the rate of increase needed to maintain constant efficiency is 0, while for program B if we increase n at the same rate as we increase p, we’ll maintain a constant efficiency.
- For example, if $n = 1000$ and $p = 2$, the efficiency of B is 0.80. If we then double p to 4 and we leave the problem size at $n = 1000$, the efficiency will drop to 0.40, but if we also double the problem size to $n = 2000$, the efficiency will remain constant at 0.80. Program A is thus more scalable than B, but both satisfy our definition of scalability.

- Programs that can maintain a constant efficiency without increasing the problem size are sometimes said to be ***strongly scalable***. Programs that can maintain a constant efficiency if the problem size increases at the same rate as the number of processes are sometimes said to be ***weakly scalable***.