am250-hw5

Yanwen Xu

May 2020

1 Latency.f90

For the Ping-Pong program I used *MPI_Sendrecv* to send message (array of size N) back and forth. I Put a 10000 times loop around my ping-pong code, and I put a timer (*MPI_WTIME*) around my loop. I have run a bunch of cases with different data amounts ($N = 1, 10, 100, 1000, 10000$). I have got the following values respectively (in seconds):

- $9.76610183715820313E-003$
- $1.05371475219726563E-002$
- $2.04608440399169922E-002$
- $8.52351188659667969E-002$
- $0.541198968887329 10$

Thus we can estimate that $t_s \approx 0.01637$ seconds and $t_w \approx 0.00005$ seconds. (Used regression calculator)

2 Atmospheric Model

Grid N_x, N_y, N_z

Assume 2-D domain decomposition: partitioned into columns/pencils ($1 \times 1 \times N_z$).

P tasks for subgrids $\frac{N}{P} \times \frac{N}{P} \times N_z$

No replicated computation $\Rightarrow T_{comp} = t_c N^2 N_z$

Using a 9 point stencil $\Rightarrow$ each task need to communicate with its 8 neighbors. Each neighbor is a column of $1 \times 1 \times N_z$. Thus we have $T_{comm} = 8P(t_s + t_w N_z)$.

If P divides N exactly, then assume load-balanced and no idle time.

Therefore, the new execution time is:

$$T_{2D\_finite\_diff} = \frac{T_{comp} + T_{comm}}{P}$$

$$T_{2D\_finite\_diff} = \frac{t_c N^2 N_z}{P} + 8t_s + 8t_w N_z$$

And hence, the efficiency would be:

$$E = \frac{t_c N^2 N_z}{t_c N^2 N_z + 8Pt_s + 8Pt_w N_z}$$

Therefore, for constant E, require

$$t_c N^2 N_z = E(t_c N^2 N_z + 8t_s P + 8t_w N_z P)$$

Notice the dominant term is $N^2 N_z$, which is **almost** like a cubic term, but $N_z P$ is like a squared term. So I think the Isoefficiency of this algorithm $O(P)$.