THE UNIVERSITY OF WARWICK

High Performance Computing
Course Notes

Performance II

Dr Ligang He

Four approaches to modelling the performance of a parallel application

Speedup

Amdahl's law

Asymptotic analysis

Model execution time (Construct the performance model)

Modelling execution time – an example

Atmosphere model

- The atmosphere model is used in weather forecast
- Capture the relation among atmospheric attributes, such as velocity of air, temperature, pressure, moisture, etc.
- The model is established by physical laws and represented by a set of partial differential equations

Partial differential equations in the atmosphere model

Conservation of momentum:

$$\frac{du}{dt} - \left(f + u\frac{\tan\phi}{a}\right)v = -\frac{1}{a\cos\phi}\frac{1}{\rho}\frac{\partial p}{\partial \lambda} + F_{\lambda}$$

$$\frac{dv}{dt} + \left(f + u\frac{\tan\phi}{a}\right)u = -\frac{1}{\rho a}\frac{\partial p}{\partial \phi} + F_{\phi}$$

Hydrostatic approximation:

$$g = -\frac{1}{\rho}\frac{\partial p}{\partial z}$$

Conservation of mass:

$$\frac{\partial \rho}{\partial t} = -\frac{1}{a\cos\phi}\left(\frac{\partial}{\partial \lambda}(\rho u) + \frac{\partial}{\partial \phi}(\rho v\cos\phi)\right) - \frac{\partial}{\partial z}(\rho w)$$

Conservation of energy:

$$C_p\frac{dT}{dt} - \frac{1}{\rho}\frac{dp}{dt} = Q$$

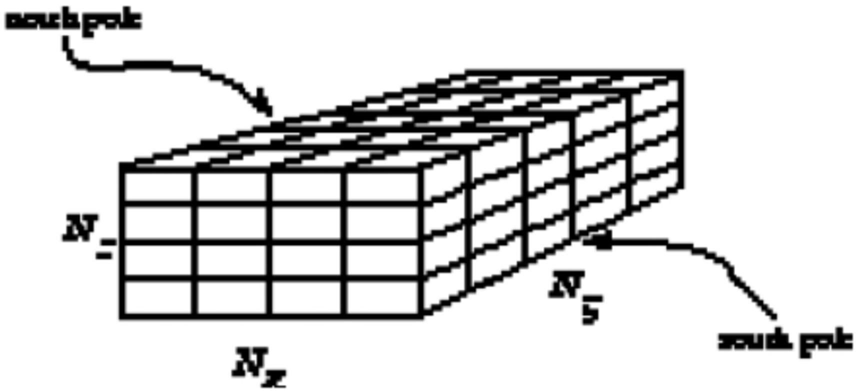
State equation (atmosphere):

$$p = \rho RT$$

Modelling execution time – an example

Atmosphere model

- too complicated to be solved by mathematical derivation
- We have to resort to the numerical method
- Cannot generate the solution for every physical point
- discretize the space, i.e., partition the space into a set of cells and use one point in a cell to represent all points in the cell.
- a continuous space is approximated by a finite set of regularly spaced points in that space
- Calculate the solution at these discrete points



Communication pattern

Each point uses the nine-point stencil to calculate its horizontal motion and uses the three-point stencil to calculate its vertical motion

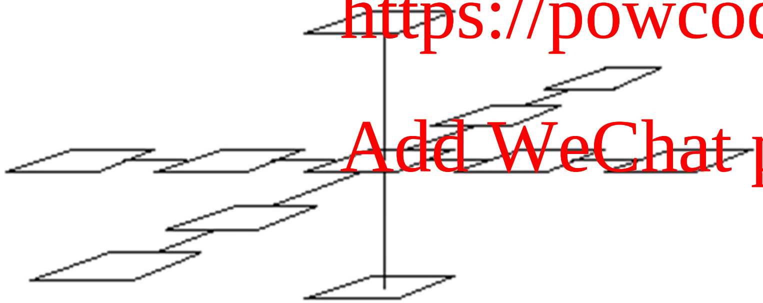


Fig.1. The 3-point stencil for calculating the vertical velocity

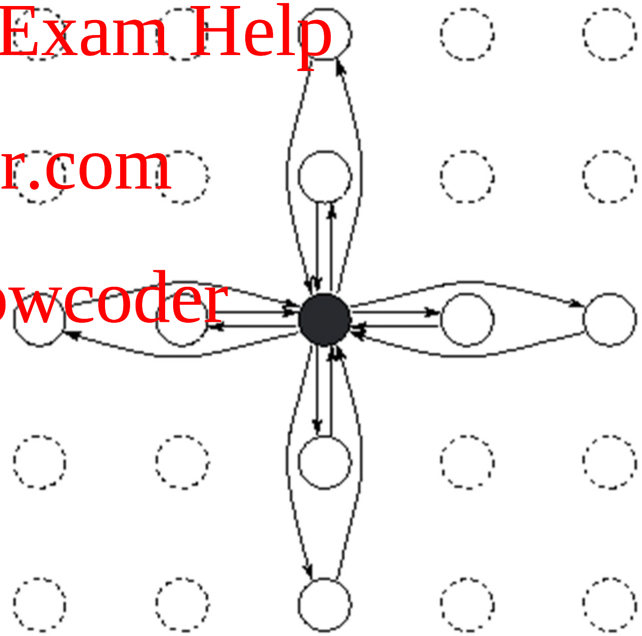
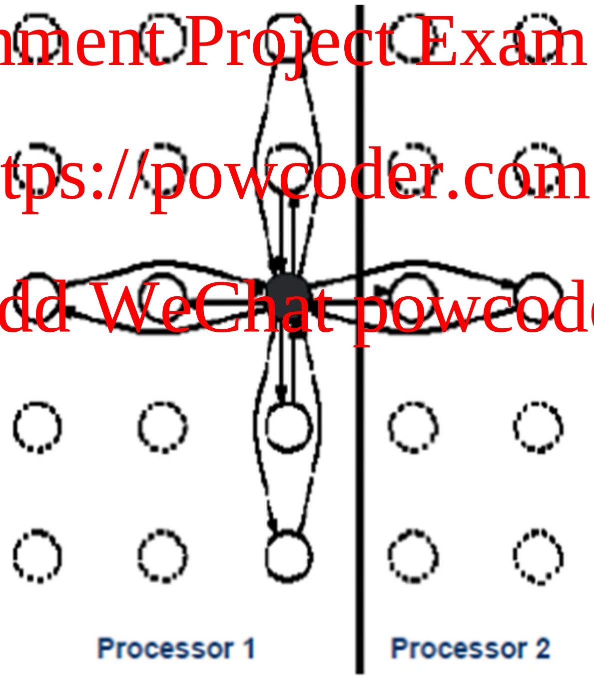


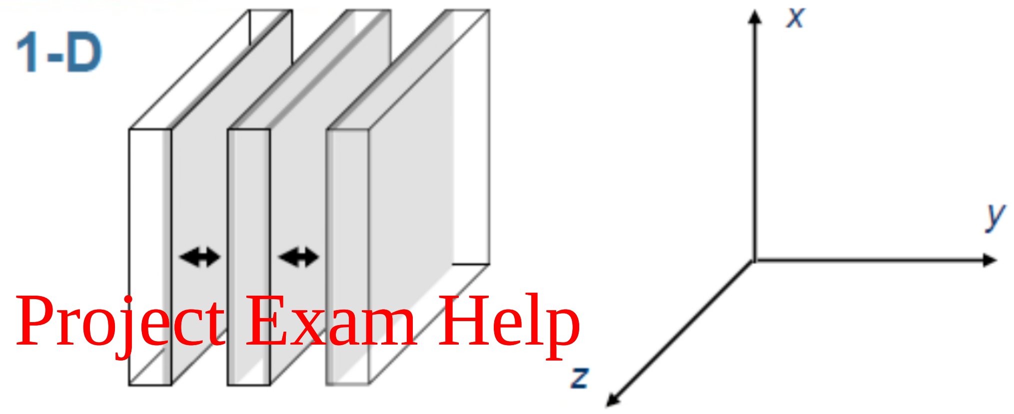
Fig. 2. The 9-point stencil for calculating the horizontal velocity

Question

When processor 1 wants to calculate the data points allocated to it, how many data items does it have to obtain from processor 2?



Modelling computation time



If we assume a grid of size N*N*Z points, and using 1-D decomposition (along y-axis) to partition the grid among P processors, then

- each task is responsible for a subgrid of size N*(N/P)*Z
- then, T_{comp} for each subgrid can be calculated as follows, where t_c is the average time of calculating a single grid point

$$T_{comp}=t_c*N*(N/P)*z$$

Modelling the time of sending one message

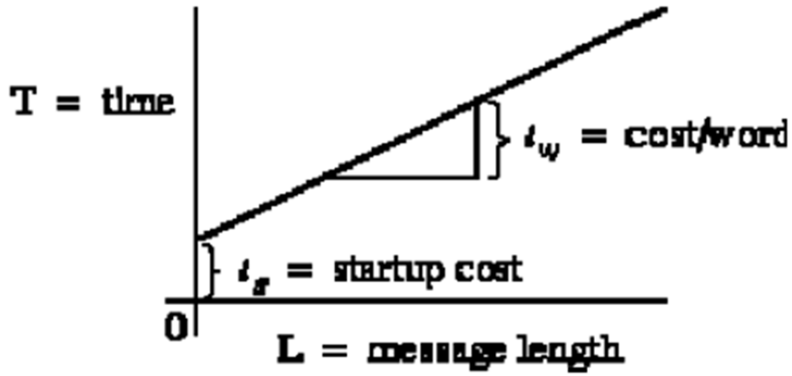
- T_{msg} (the time spent in sending one message) can be calculated as follows,

$$T_{msg}=t_s+t_wL$$

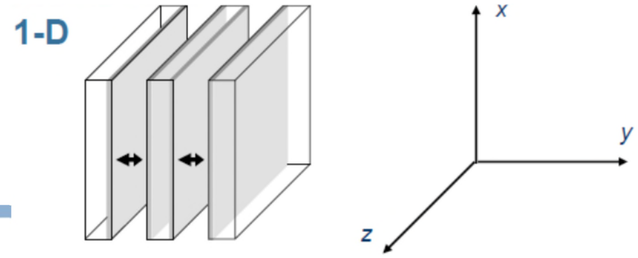
where t_s is the message startup time, t_w is the transfer time per byte, L is the size of the message

T_{msg} is a function of L; t_s and t_w are constants given a computing platform

Question: if I plot the function of T_{msg} over L, what does the plot look like? How are ts and tw represented in the line?



Modelling communication time

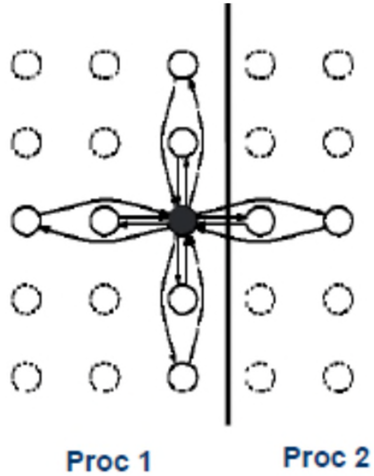


- Communication time for calculating a subgrid can be computed as

$$T_{comm}=2(t_s+t_w 2NZ)$$

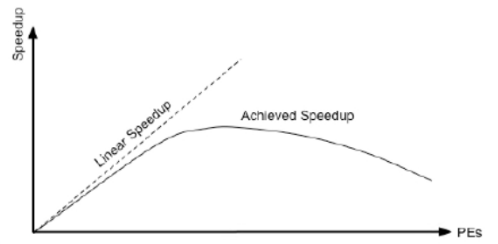
- Hence, the performance model for the execution time of calculating the velocity of the grid of points is

$$T_p=T_{comp}+T_{comm}=t_c*N*(N/p)*Z+2(t_s+t_w 2NZ)$$



- From the performance model of the execution time, we can know a lot of information.
  - What will happen when we increase p?

    Execution time decreases with increasing P; the proportion of communication cost increases
  - Execution time increases with increasing N, Z, t_c, t_s, t_w



Speedup and parallel efficiency

- The execution time on one processor is

$$T_1 = t_c N^2 Z$$

- Speedup is

$$S(P) = \frac{t_c N^2 Z P}{t_c N^2 Z + 2t_s P + t_w 4NZP}$$

- Parallel efficiency can be calculated as

$$E = \frac{t_c N^2 Z}{t_c N^2 Z + 2t_s P + t_w 4NZP}$$

Iso-efficiency

$$E = \frac{t_c N^2 Z}{t_c N^2 Z + 2 t_s P + t_w 4 N Z P}$$

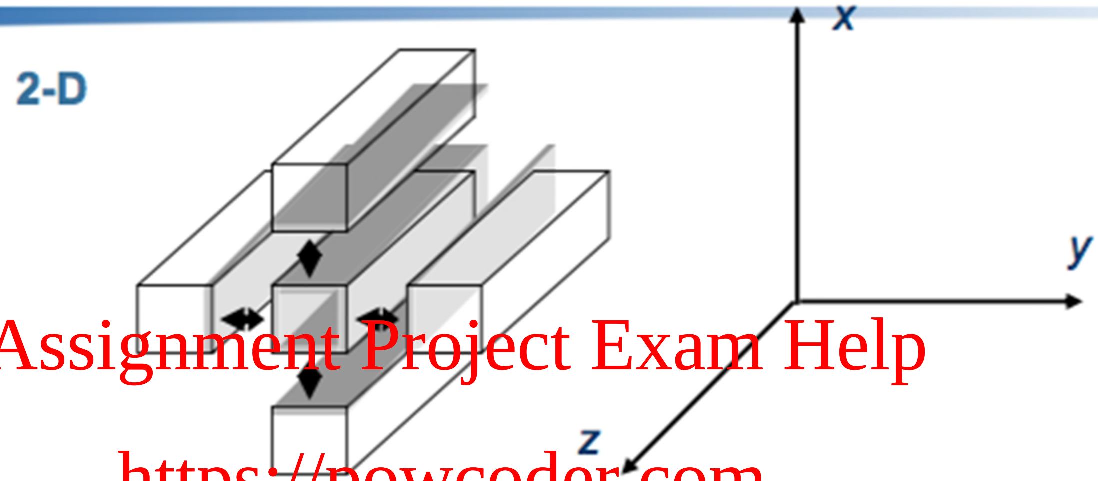
Question: What is the iso-efficiency function?

E can be reduced to

$$E = \frac{t_c}{t_c + \frac{t_s 2P}{Z N^2} + \frac{t_w 4P}{N}}$$

When N=P, E remains approximately constant as P changes(except when P is small)

2D decomposition



- If applying 2-D decomposition, then

- Execution time can be modelled as

$$T_{comp} = t_c N^2 Z / P$$

$$T_{comm} = 4(t_s + t_w 2 \frac{N}{\sqrt{P}} Z)$$

$$T_P = T_{comp} + T_{comm} = \frac{t_c N^2 Z + t_s 4P + t_w 8NZ\sqrt{P}}{P}$$

1D decomposition

$$T_{comp} = t_c * N * (N/P) * z$$

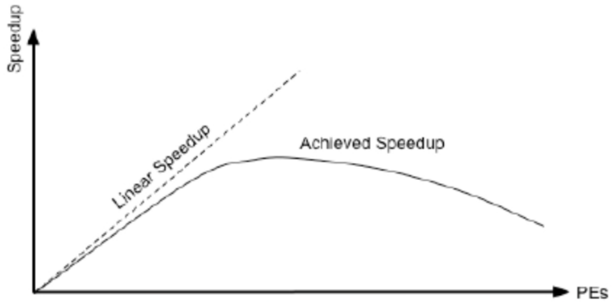
$$T_{comm} = 2(t_s + t_w 2NZ)$$

Iso-efficiency

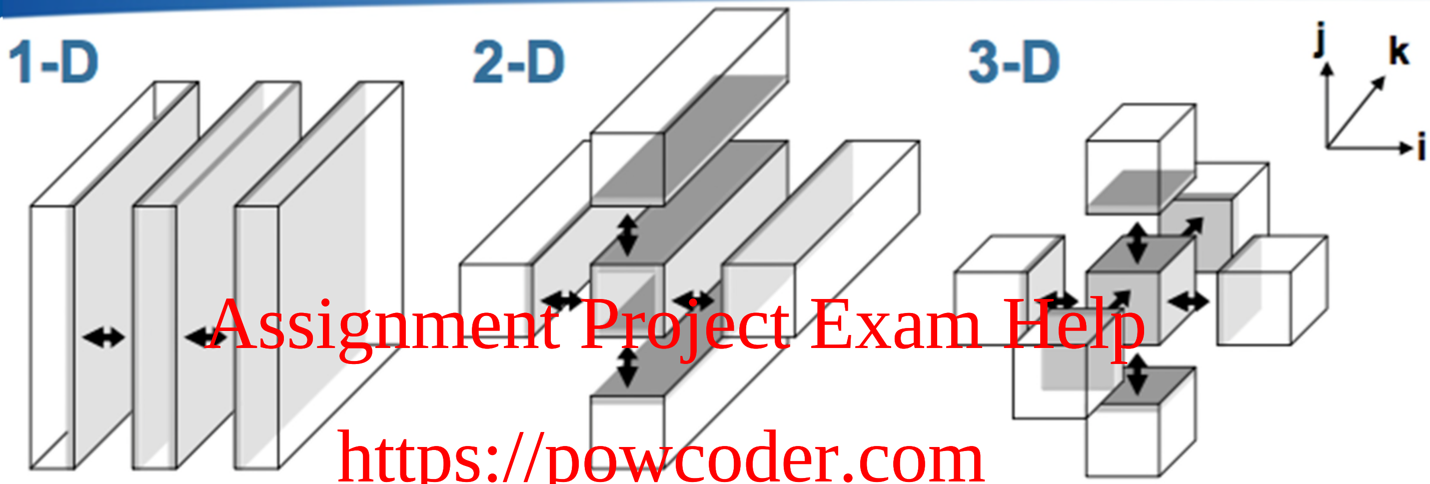
Parallel efficiency can be modelled as

$$\frac{t_c N^2 Z}{t_c N^2 Z + t_s 4P + t_w 8NZ\sqrt{P}}$$

- When N=$\sqrt{P}$, E remains constant as P increases
- Iso-efficiency in 1D is N=P
- Therefore, for this particular communication pattern, applying 2D decomposition will achieve better scalability than 1D decomposition



Decomposition analysis



- Boundary surfaces between sub-grids are shaded.
- Data on boundary surfaces need to be communicated.
- The lower surface-to-volume ratio, the better:
  - Surface = communication
  - Volume = computation

Decomposition analysis

Consider a 3-D grid and assume the grid is a cube

- Volume V=c*n, where c = number of cells per PE, n is the number of processors

- The length of the grid in each dimension is $V^{1/3}$

Decomposition analysis

		Sub-grid Length				
	Sub-grid Volume	I	J	K	Sub-grid Surfaces	Surface to Volume
1-D	c	$V^{1/3} / n$	$V^{1/3}$	$V^{1/3}$	$2c^{2/3} .n^{2/3}$	$2n^{2/3} / c^{1/3}$
2-D	c	$V^{1/3} / n^{1/2}$	$V^{1/3} / n^{1/2}$	$V^{1/3}$	$4c^{2/3} .n^{1/6}$	$4n^{1/6} / c^{1/3}$
3-D	c	$V^{1/3} / n^{1/3}$	$V^{1/3} / n^{1/3}$	$V^{1/3} / n^{1/3}$	$6c^{2/3}$	$6 / c^{1/3}$