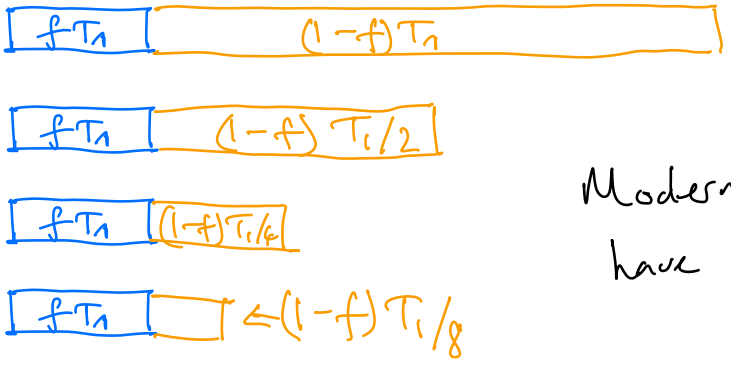
PSC II : More computational scaling

Last time saw

Strong scaling (Amdahl) 1960s

$$T_p = f T_1^{global} + \frac{(1-f) T_1^{global}}{p}$$

Fixed global problem size, add more resource → get answer faster.

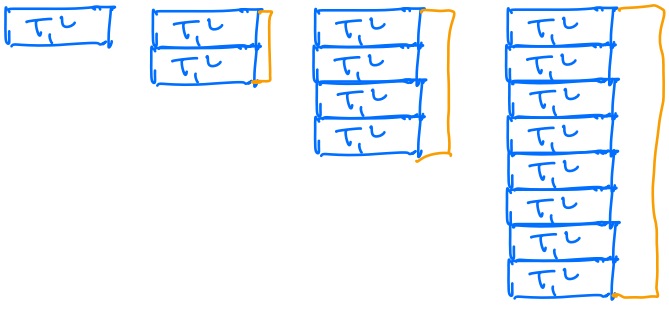


Modern supercomputers have $O(10^6)$ cores.

Weak scaling (Gustafson) 1990s

$$T_p = T_1^{local} + \underbrace{O(p) T_1^{local}}_{\text{overhead}}$$

Fixed local problem size, add more resource → solve bigger problem in same time.



Putative time on one process is pT_1^{local}

Speedup (Amdahl)

$$S_p = \frac{T_1}{T_p} = \frac{T_1}{fT_1 - \frac{(1-f)T_1}{p}} = \frac{1}{f - \frac{1-f}{p}}$$

$$= \frac{p}{(p+1)f - 1}$$

$f = 0.1 \quad p = 100$

$$S_p = \frac{100}{10 \cdot 1 - 1} \simeq 11$$

Speedup (Gustafson)

$$S_p = \frac{p T_1^{local}}{T_1^{local} + o(p) T_1^{local}} = \frac{p}{1 + o(p)}$$

Efficiency

cost on one process

Amdahl $\eta_p = \frac{T_1}{p T_p} = \frac{1}{(p+1)f - 1}$

cost on p processes

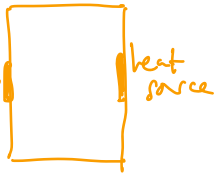
Gustafson $\eta_p = \frac{p T_1^{local}}{p(T_1^{local} + o(p) T_1^{local})} = \frac{1}{1 + o(p)}$

Why is timestepping a strong scaling problem?

Parabolic problems (eg heat equation)

$$\frac{\partial u}{\partial t} - \nabla^2 u = f$$

$-\nabla^2 u$: diffuses out; f: external forcing



Stable explicit timestep is $O(h^2)$ for grid spacing h.

implicit

$$\frac{u^{n+1} - u^n}{\Delta t} - \nabla^2 u^{n+1} = f$$

explicit scheme

$$\Delta t = O(h^2)$$

↳ This is why explicit schemes are terrible for parabolic problems.

Hyperbolic problems (eg advection equation)

concentration

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi \underline{u}) = f$$

$\underline{u}$: fluid velocity

Stable explicit timestep is $O(h)$.

What's the consequence for parallel computing?

Discretize heat equation
⇒ Grid spacing h ⇒ answer in 1 hr. On one process.

But now want spacing $\frac{h}{2}$

⇒ In 2D this is 4x work per timestep.

OK: sounds good. 4x work / timestep.
But problem is 4x bigger
so ⇒ use 4x processes.

Problem for stability I also need 4x timesteps.

So to get the answer in 1 hr, need to do 4x as many time steps.
⇒ add more compute
→ local problem gets smaller
(strong scaling)

What are sources of serial fraction and parallel overhead?

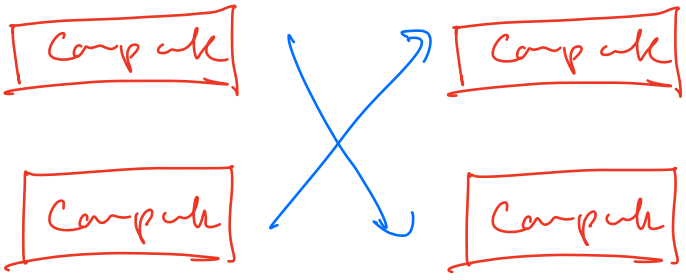
- Communication latency

Time to send b-byte message

$$t(b) = \alpha + \beta b$$

α: latency

β: inverse bandwidth.

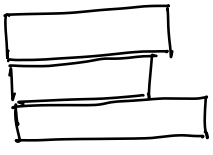


message exchange phase

Rich hardware designers.

→ we'll revisit this

- Load imbalance



Each process takes a diff amount of time to do local work.

⇒ perhaps # pairs it sees is different.

This creates a "serial fraction".

⇒ For multigrid this can happen early

Grid-based PDEs: design considerations

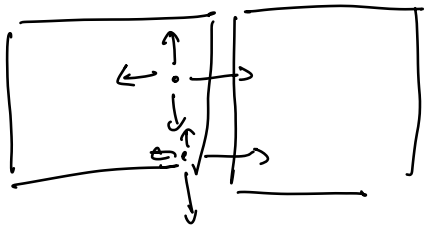
- Nothing that is $O(P)$.

  Typically you might have a lookup array that tells you which process a dof belongs to.

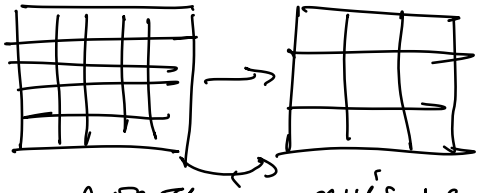
  $\Rightarrow$ we can never "gather" all the data to a single process.

- At worst $O(\log P)$ communication complexity.
  - → Nearest neighbour
  - → Multilevel

Stencil app



Stencils only access a neighborhood ✓



coarsening gives us log P behaviour for long range interactions

- At worst $O(N \log N)$ compute complexity.

Multigrid gives us this, assuming $2^d \sum_{i=1}^{\infty} \frac{1}{2^i} < 2$.

Machine characteristics

Messaging:

$$t(b) = \alpha + \beta b$$

α: latency

β: inverse bandwidth

$\Rightarrow$ write ping pong code.