Last time
- First steps with MPI

Today
- Computational scaling
  → how is my code performing in parallel?

60s ← Strong, weak, machine scaling. (weak → 90s, machine → 2010s thing)
- Data decomposition choices. → Vectors.

Strong scaling

$$S_p = \frac{T_1}{T_p}$$

S_p ← speedup

T_1 ← time on one process

T_p ← on p processes.

Problem size is fixed.

Hope $S_p > 1$ when $p > 1$.

1 person can build a wall in 4 weeks. How long will 4 people take? $T_4 = \frac{T_1}{4}$ → ideal linear scaling

Gene Amdahl

parallel part.

$$T_p = f\,T_1 + \boxed{(1-f)\frac{T_1}{p}}$$

$$\underbrace{\boxed{\text{serial fraction}}}_{f\,T_1}\underbrace{\boxed{\text{"parallel fraction"}}}_{(1-f)\,T_1}$$

Amdahl's law:

$$\lim_{p\to\infty} T_p = f\,T_1$$

$$\Longleftrightarrow \quad \lim_{p\to\infty} S_p = \frac{1}{f}$$

So, if 1% of the code is serial, best speedup is?

$$S_\infty = \frac{1}{0.01} = 100.$$

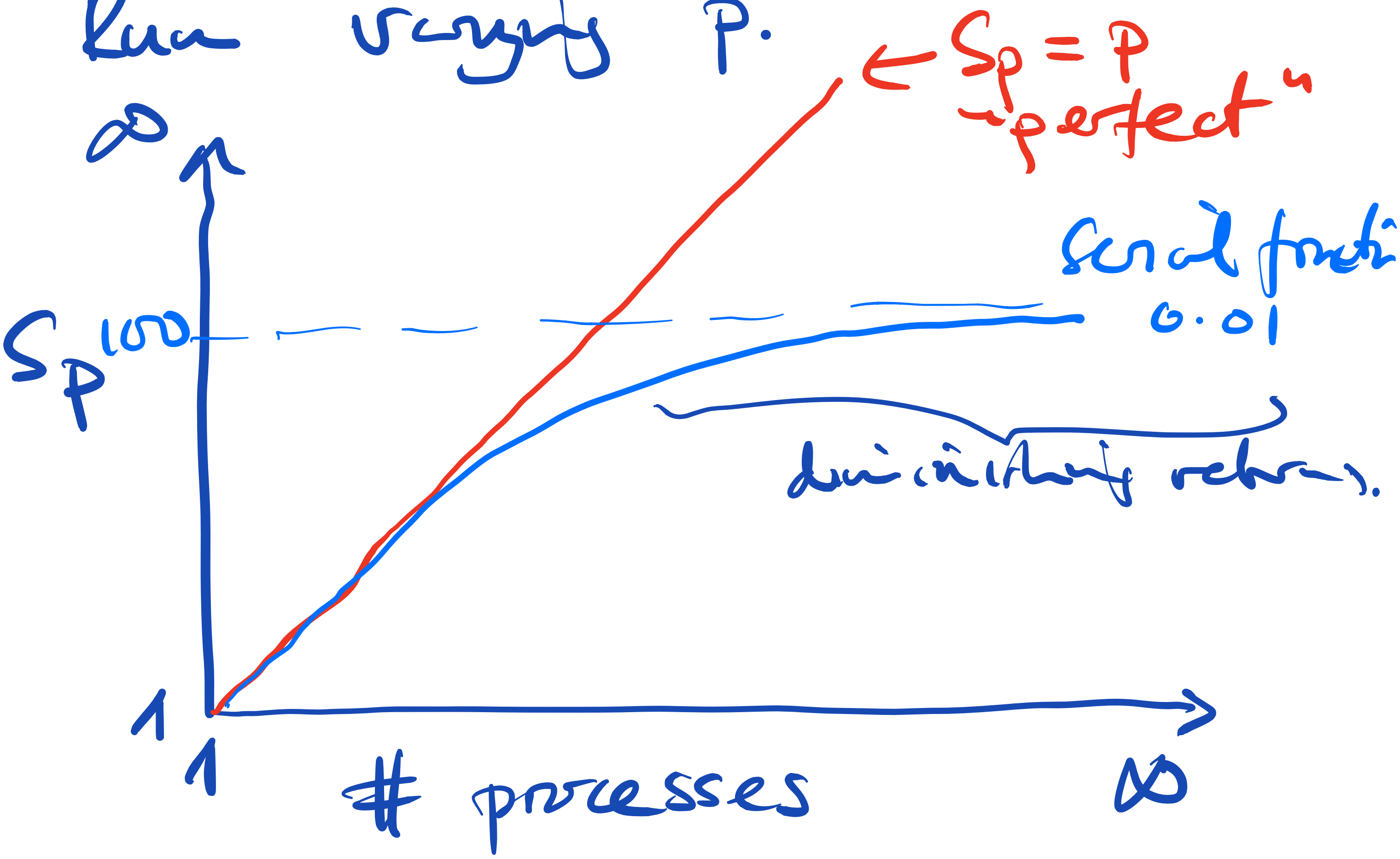
Consequence

→ Need to strip out as much of the serial code as possible.

Presenting speedup data:

Problem size N: ← fix.

Run varying P.



Note: dividing out by T_1, so can hide inefficient implementation.

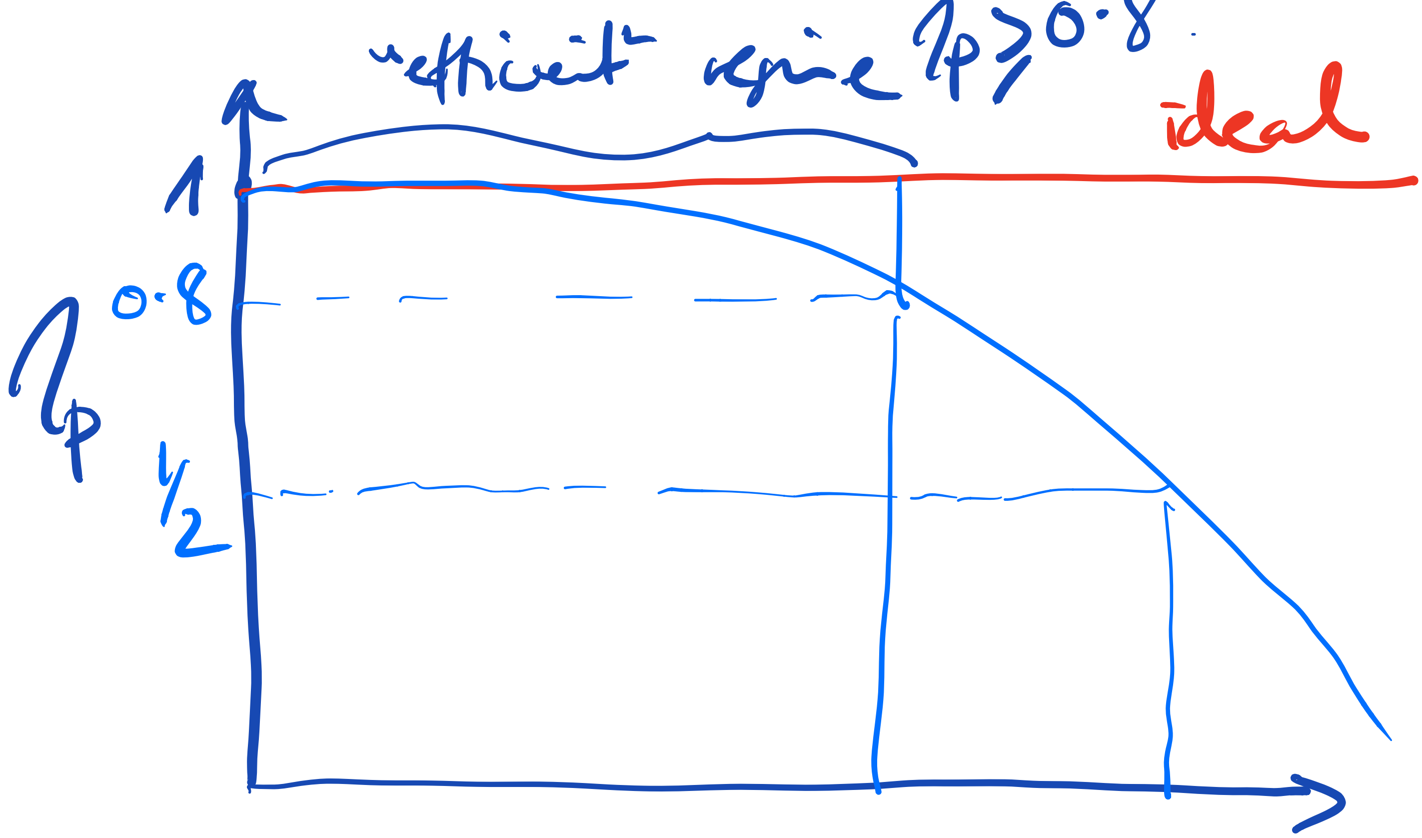
Can also plot efficiency.

$$\eta_p = \frac{T_1}{p\,T_p} \in [0,1)$$

total resource we used on p processes.

exercise

What is η_p as a function of the serial fraction, f?



processes

Often on log scales

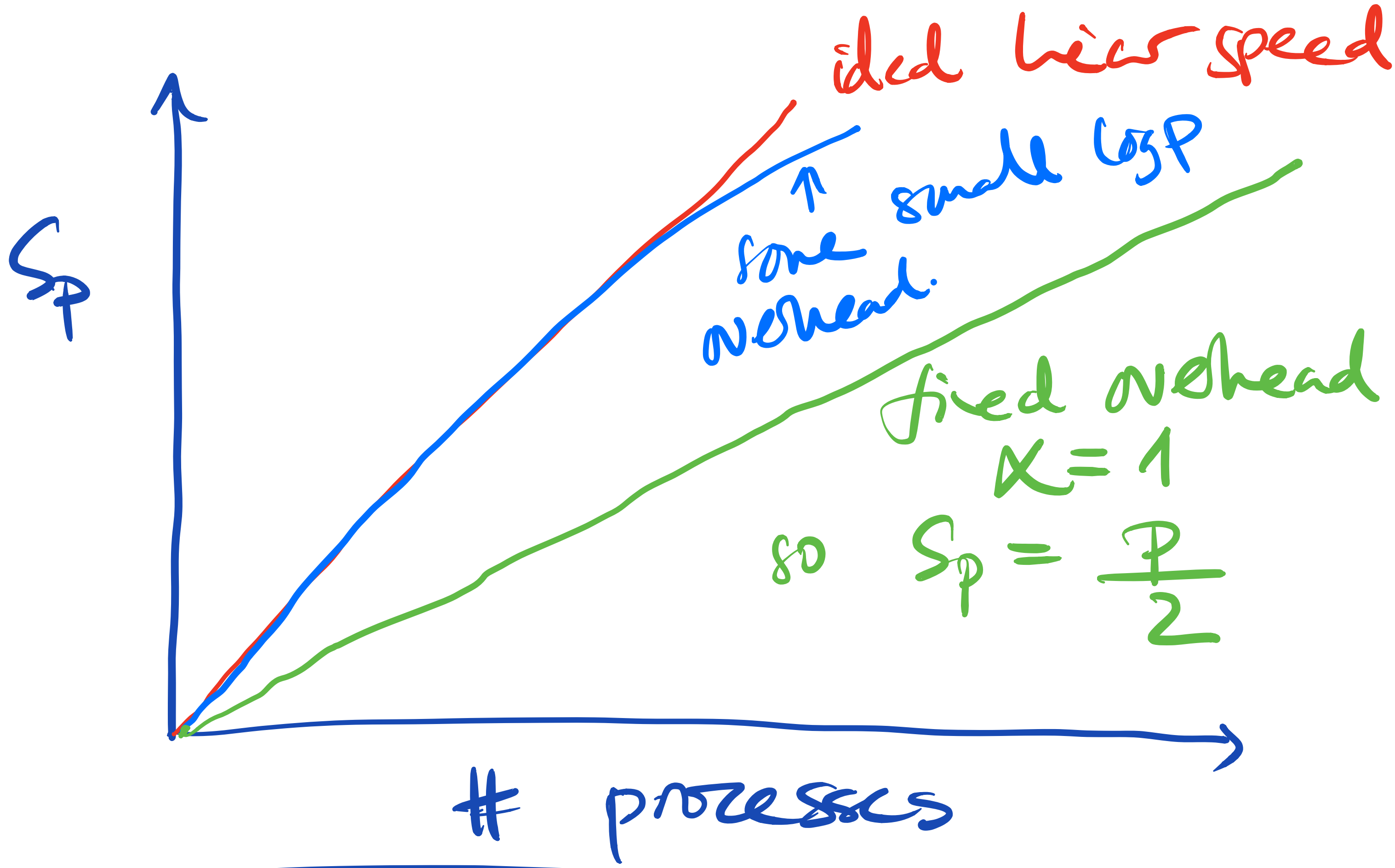
Weak scaling

$$T_p = \underbrace{T_1}_{\text{time to solve local problem.}} + \overbrace{\alpha(p)}^{\text{small, grows slowly.}}\, T_1$$

"Gustafson's law".

$$S_p = \frac{p\,T_1}{T_p}$$

Local (per process) problem size fixed.



Real applications scale problems up with weak scaling & then at fixed size do strong scaling.

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

overhead fixed

$$\eta_p^{fix} = \frac{1}{1+\alpha} \; ; \; S_p^{fix} = p\,\frac{1}{1+\alpha}$$

$$\eta_p^{log} = \frac{1}{1+O(\log p)} \; ; \; S_p^{log} = \frac{p}{1+O(\log p)}$$

Typical setup for PDEs.

overhead : $\alpha + O(\log p)$

message latency

reductions (combining data)

e.g. dot product $a \cdot b$ → has to scale "problem size fixed".

Dense matrix-matrix.

$\mathbb{R}^{N \times N}$

$O(N^3)$ algorithm.

weak scale.

If I double N.

& double P

→ is the time to solution fixed?

$N \to 2N$. $P \to 2P$

N^3 $8N^3$, $\frac{N^3}{P} \to \frac{8N^3}{2P} = 4N^3$

So to get constant T_p need to add 8x processes

$$N^3 \longrightarrow 8N^3$$

$$p \longrightarrow 8p$$

$$\frac{N^3}{p} \longrightarrow \frac{N^3}{p} \checkmark$$

But local matrix size.

$$N \times N \longrightarrow \frac{N}{\sqrt{8}} \times \frac{N}{\sqrt{8}} \; ?$$

so local problem is <u>smaller</u>

$\Rightarrow$ strong scaling.

Machine / algorithmic scaling

Range of sizes, Time to sol

plot T vs $\frac{N}{T}$ → want flat.

→ Add link to paper in notes.

Vectors in parallel.

$x \in \mathbb{R}^N$

pointwise

$y \leftarrow \alpha x + y$ $\qquad y \in \mathbb{R}^N$

$\qquad\qquad\qquad\qquad \alpha \in \mathbb{R}$

$y \leftarrow 1/y$

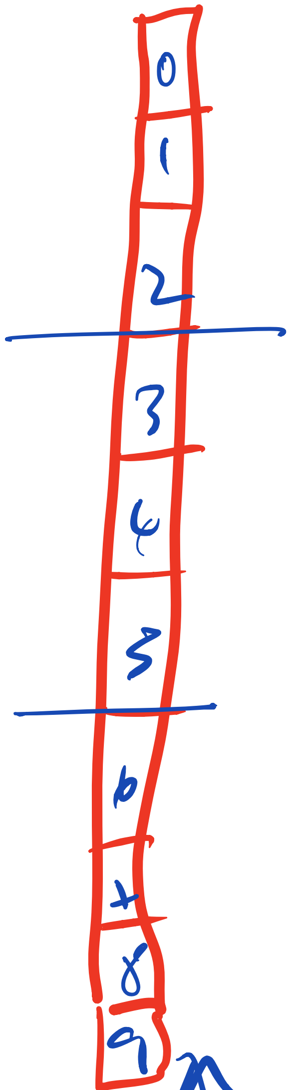
⋮

collective. computing norms / dot products.

$\beta \leftarrow a \cdot b$ $\qquad \beta \in \mathbb{R}$

$\qquad\qquad\qquad a, b \in \mathbb{R}^N$

pointwise updates.
Imagine we have
3 processes.



Aim to evenly distribute
total length N
across the p processes.
Scalability limit for
pointwise comes from
uneven distr^n of
local work.

⇒ "load imbalance"

3, 3, 4

Dot products & grids
next time.