Last time

Simple model for message exchange time:

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

ping-pong.

non-dimensionalize relative to unit of work t_c:

$$t^{nd}(m) = \frac{\alpha}{t_c} + \frac{\beta m}{t_c}$$

This time

Why can we do parallel at all?

→ Sparsity

How should we design parallel data structures for FD?

Sparsity

Time stepping:

$$\partial_t u - Au = 0$$

Explicit timestepping

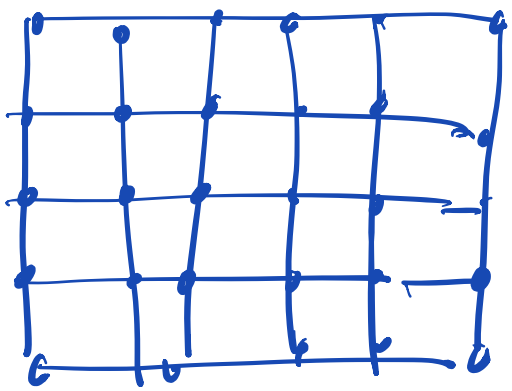
$$\frac{u_{n+1} - u_n}{\Delta t} - A u_n = 0$$ Explicit Euler.

$$u_{n+1} = u_n + \Delta t A u_n$$

Implicit timestepping.

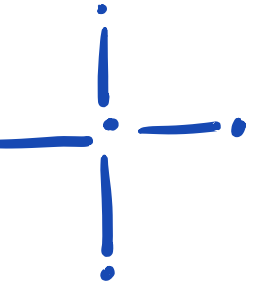
$$u_{n+1} - \Delta t A u_{n+1} = u_n$$ Implicit Euler.

$$u_{n+1} = (I - \Delta t A)^{-1} u_n$$

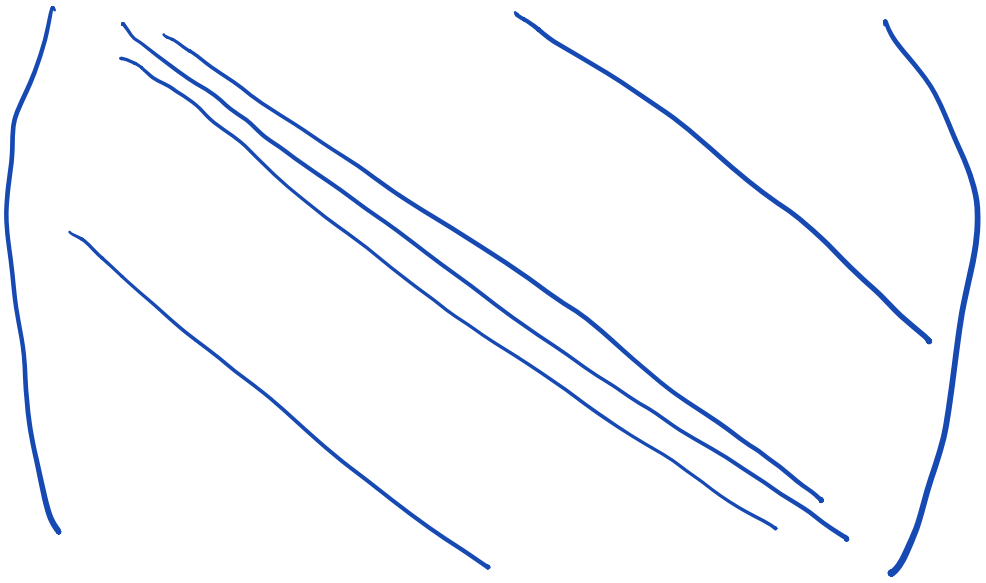


A is some matrix. Remarkably, A is very sparse.

Suppose $A = \nabla^2$

Then, A couples 

Sparsity pattern



Only 5 diagonals are non zero.

$\Rightarrow$ Consequence:

Explicit schemes we can implement purely with local operations.

$u_{n+1} = u_n + \Delta t A u_n$

$\Leftarrow$

Implicit

$u_{n+1} = (I - \Delta t A)^{-1} u_n$

Inverse of a sparse matrix is not necessarily sparse.

But we can often find algorithms that compute what A inverse on a vector, using sparse operations.

So: we can't make A^{-1}

But fortunately we only need

$A^{-1}u$

↳ We can make this in a sparse manner.

→ Multigrid.

Why is this sparse?

→ sparse Restriction operator R local stencil

→ sparse Smoother: Jacobi smoother → pointwise

→ sparse Prolongation operator $P = R^T$ local stencil

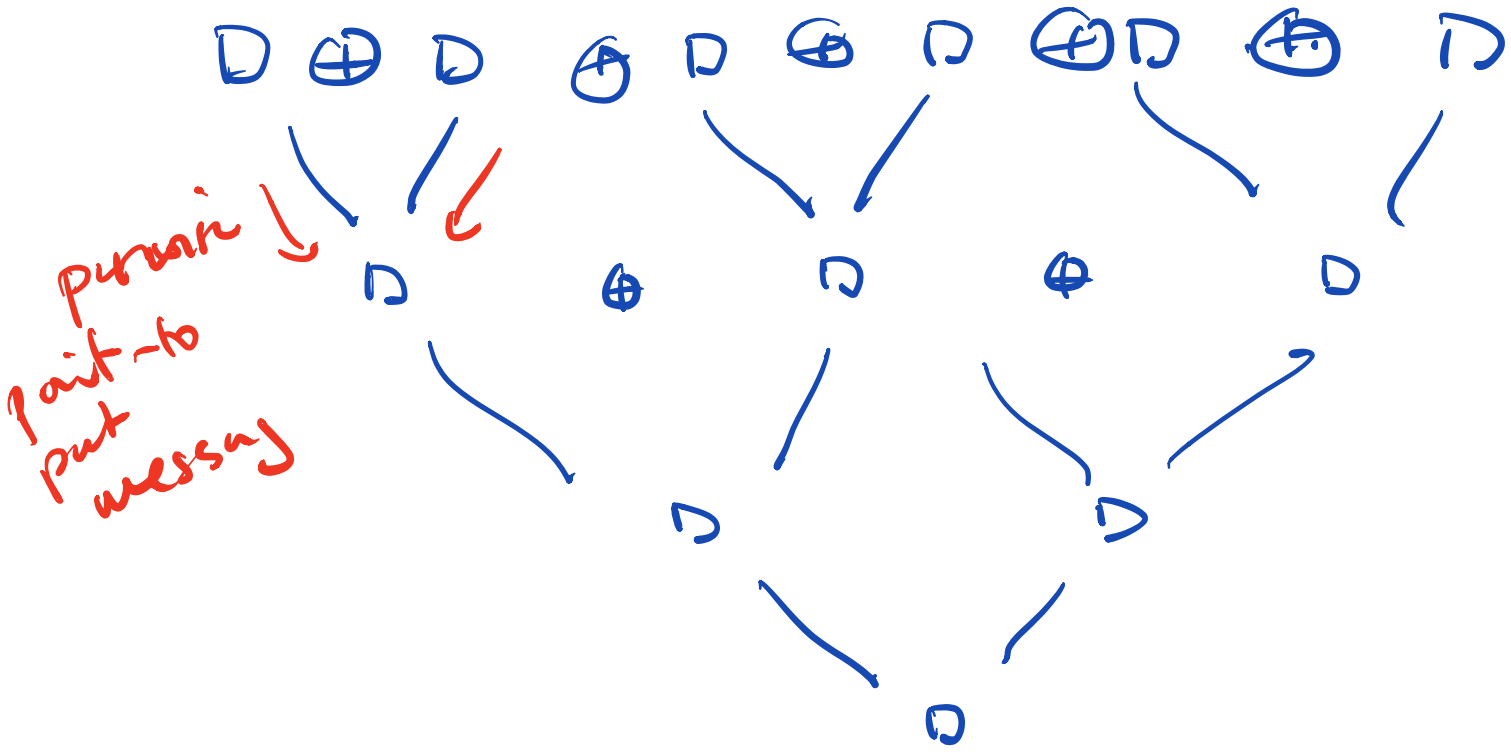
What does this mean for implementation.

Goal: total work is N

Work goes like $\frac{N}{P}$ on P processes.

Message exchange is P-independent.
we'll be happy with
$\log P$ dependence.

Allreduce → $\log P$

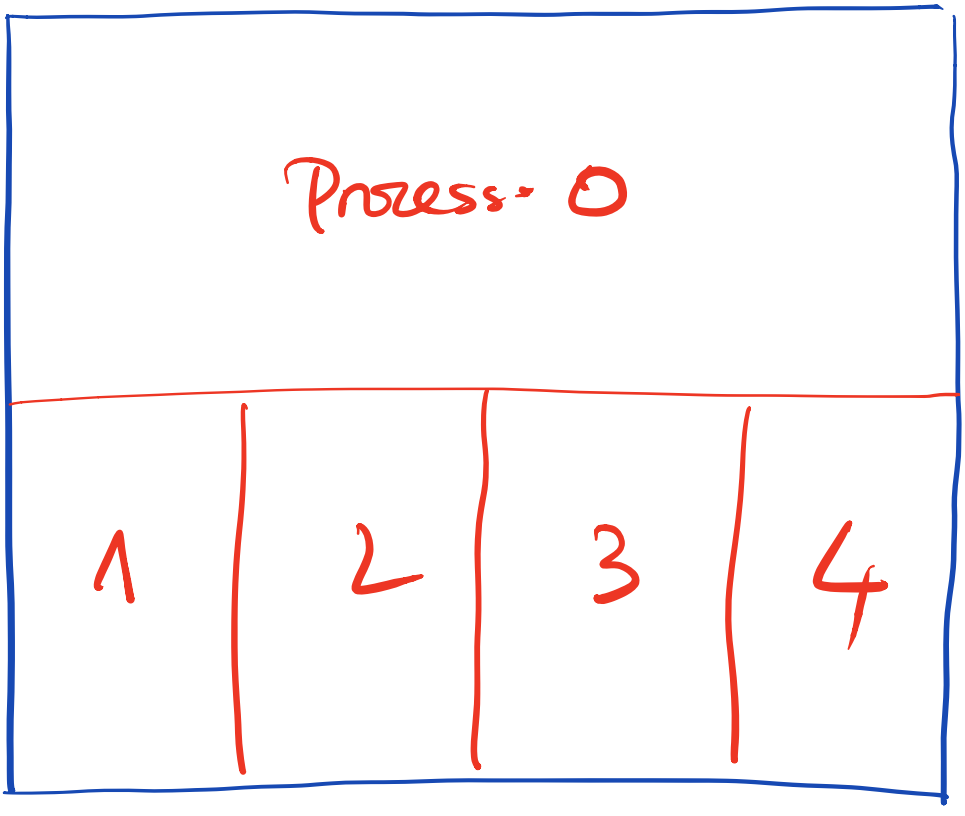


Binary tree with P leaves has $\log_2 P$ depth.

So. $\lg P$ pair-to-pair messages do a reduction.

$\Rightarrow$ Don't want any data structures that scale with P
We're happy with $\log P$.

FD grid 5 processes.

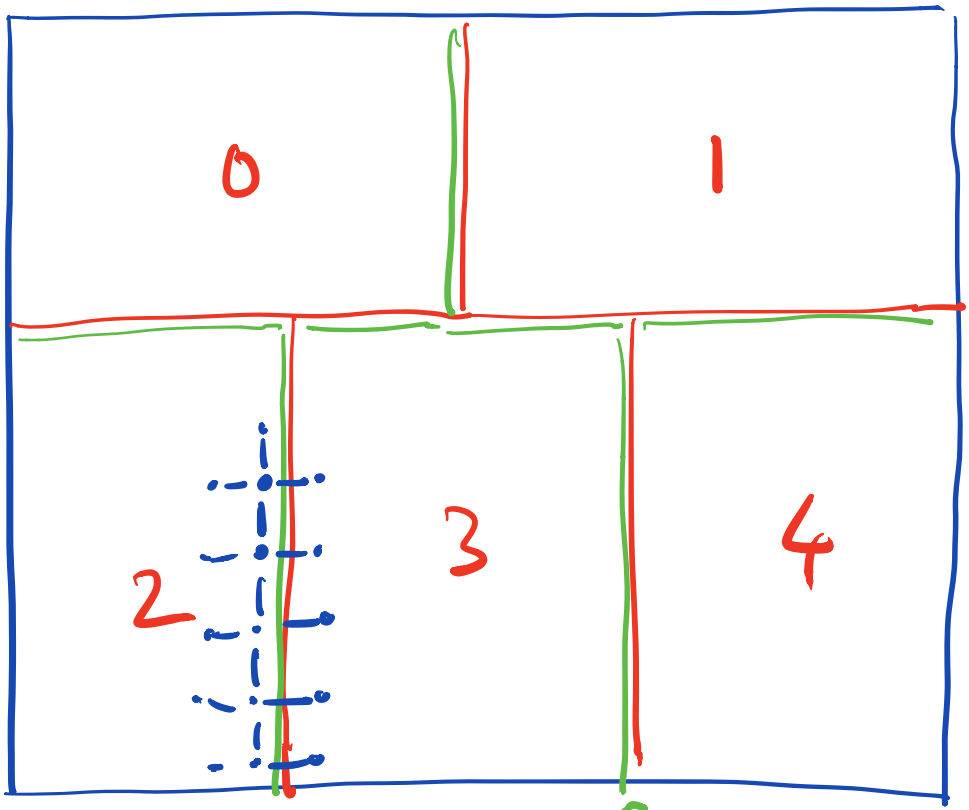


Divide grid between processes. ideally at equal size.

Bad decision at 4th.

Process 0 does $\frac{N}{2}$ work, everyone else does $\frac{N}{8}$.

$\Rightarrow$ Not scalable.



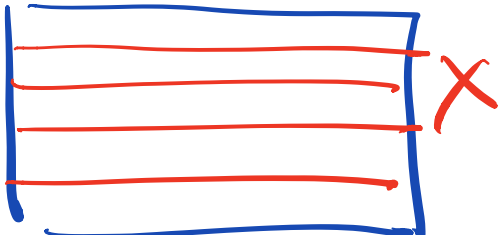
This is better.

Now everyone has equal work.

Caveats: rectangular patches

: Minimise interface regions

Want surface-to-volume ratio of our subdomains to be as small as possible



Bad: high surface to volume.

Can phrase this problem in terms of graphs.

Each dot is a vertex

edges between coupled dots.

Best k-partitioning.

is: Partition graph into k pieces s.t. each has the same # dots,

Minimizing the graph cut.

→ Minimizing # edges that go between partitions.

↳ for regular grids we don't use this setting.

→ we do for irregular grids

Parallel decomposition.

Split domain (cube)
into equal (approx) subcubes.
per process.

→ Attempting to balance work.

Minimize communication volume
(size interface).

Next time: actually computing.