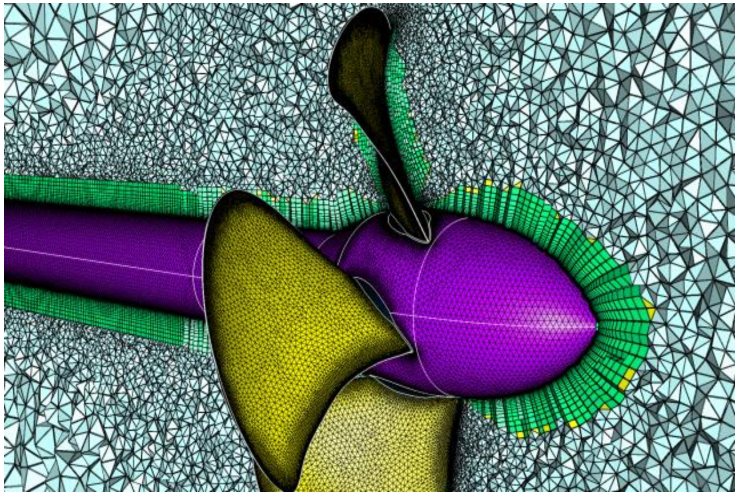
Parallel Design Patterns

Geometric Decomposition

Reusing this material

Geometric Decomposition – Problem

- A problem domain can often be subdivided (or partitioned) into many smaller spaces that can be operated on concurrently.
  - How can an algorithm be organised so as to exploit this potential parallelism?



Courtesy of Another Fine Mesh, Pointwise

- This is *very* common in computer simulation where you're simulating what goes on in time and space.
  - Operate on different parts of space concurrently
  - Also known as domain decomposition and as coarse-grained data parallelism.

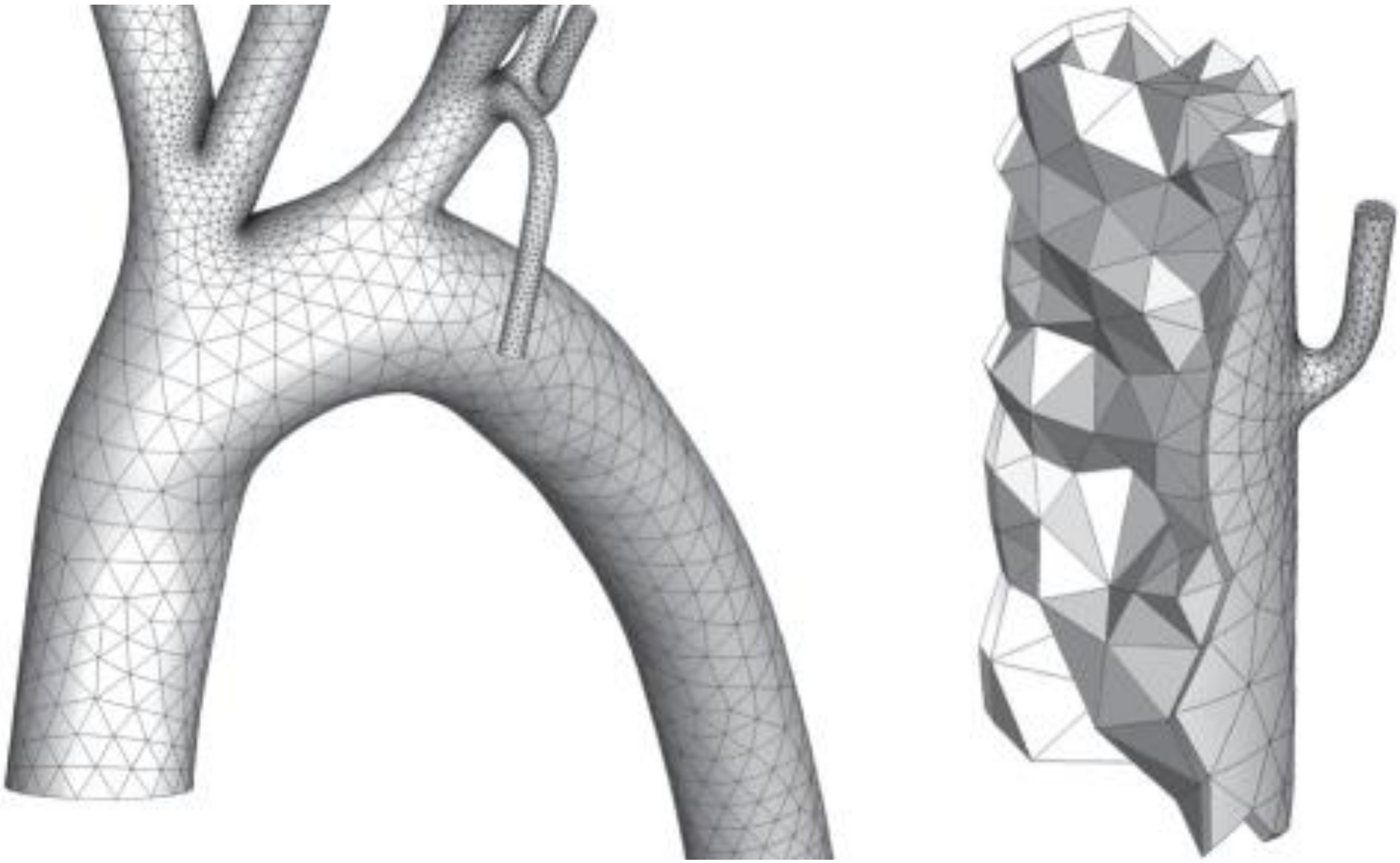
Geometric Decomposition - Context

- The algorithm will probably involve one key data structure whose elements can be operated on concurrently.
  - Typically, the data structure is an array, but it could also be a graph.
  - Data structures with inherent hierarchy (e.g., trees) are often better dealt with by the recursive data pattern.

- Operations on an element usually involve the element itself and some neighbouring elements.
  - "...domain decomposition methods solve a boundary value problem by splitting it into smaller boundary value problems on subdomains and iterating to coordinate the solution between adjacent subdomains." http://en.wikipedia.org/wiki/Domain_decomposition_methods

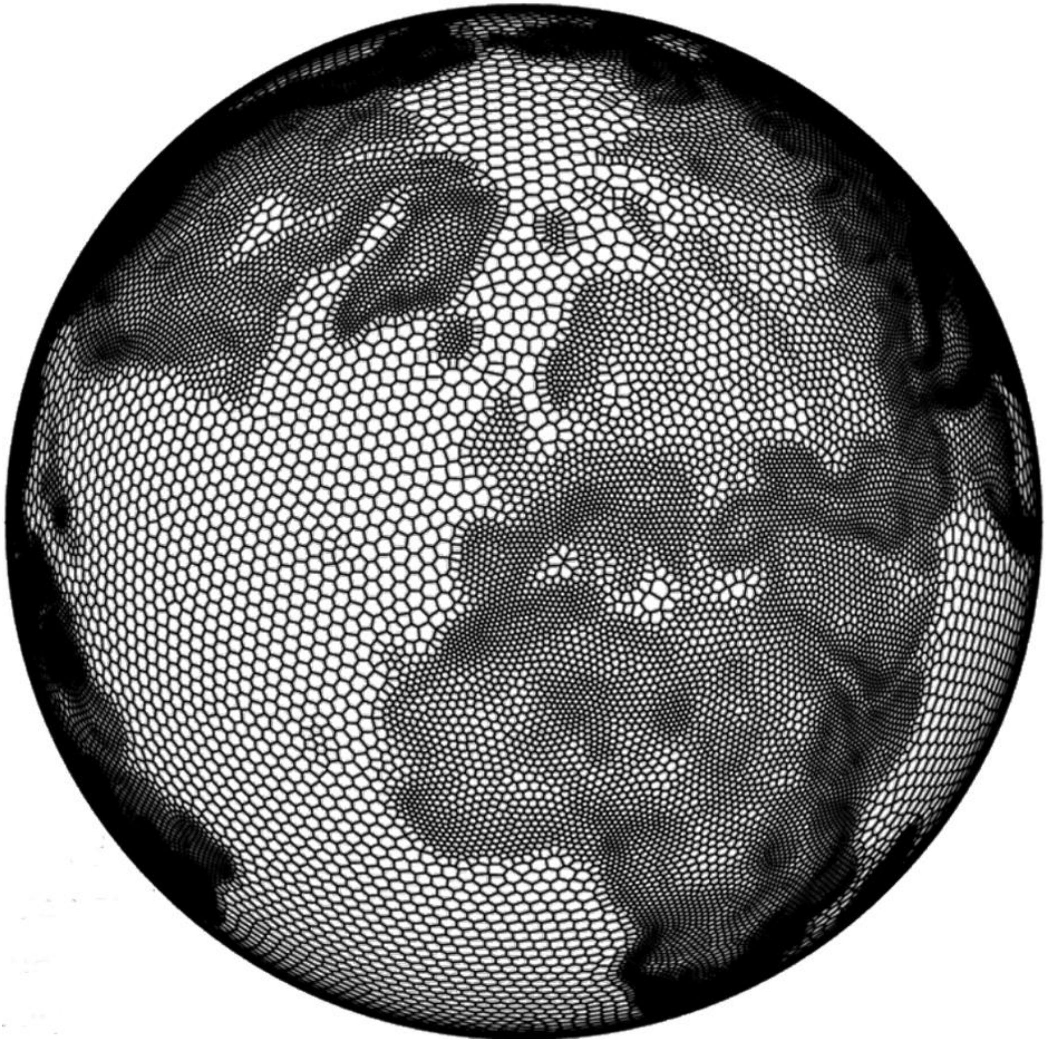
Vincent, Plata, Hunt, et al.,
Journal of the Royal Society Interface, 2011

Some more examples

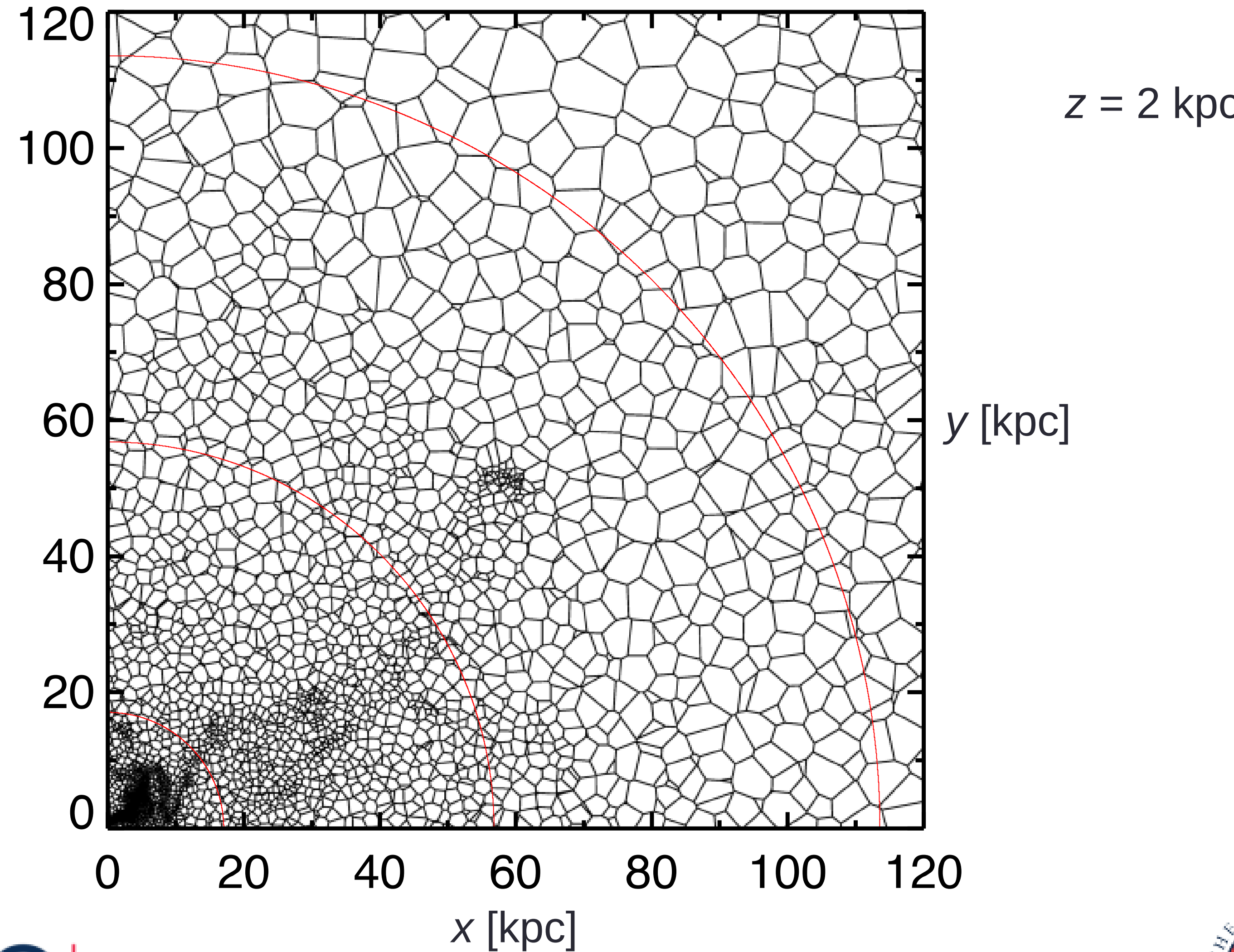


Courtesy of Michael Duda and Bill Skamarock,
National Centre for Atmospheric Research (NCAR), US

Some more examples



Some more examples



Geometric Decomposition - Forces

- How do we define subdomains and assign these to units of execution (UEs)?
- We need to consider the usual qualities,…
  - efficiency, simplicity, portability and scalability
  - and load balancing too.

- We need to ensure that data is available to perform the operation on the subdomain.
- All decomposition approaches introduce parallelisation overheads.

Geometric Decomposition – Solution

- Data decomposition
  - How to split up the domain into subdomains.

- Exchange operation
  - How neighbouring subdomains influence each other.

- Update operation
  - Computational work

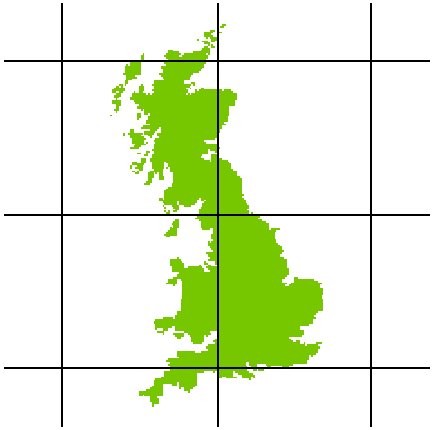
- Task scheduling
- Program structure

Data Decomposition

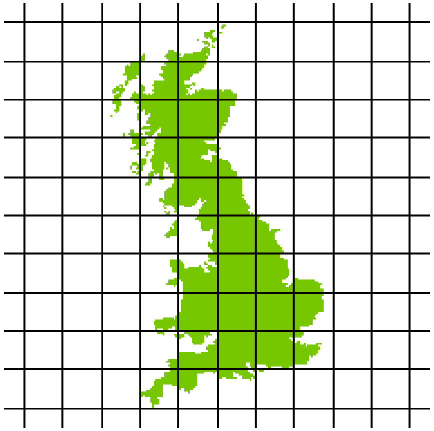
- How do we decompose the domain?
  - Do we decompose in all dimensions?
  - Subdomain shapes can be structured (regular connectivity) or unstructured.
- How do we read the data?
  - Does the format mirror domain decomposition?
  - Is initial state read by one UE and then broadcast?
  - Or is data read in parallel?
- Will the workload (data per UE) be balanced?

- Efficiency depends on the granularity of data decomposition.
  - the balance between communication and computation

Granularity



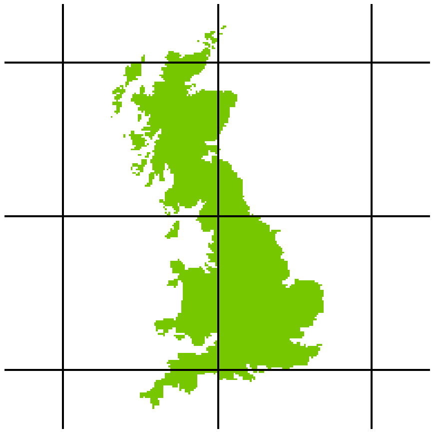
Coarse-grained

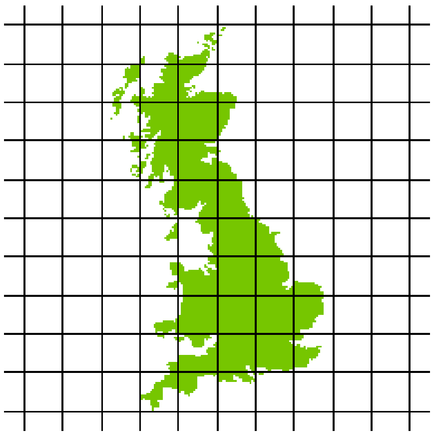


Fine-grained

- How much work should we assign to each UE?
- The finer the mesh the greater the communication required.
- Splitting a problem has time cost, but this can be recouped through parallelism

Granularity

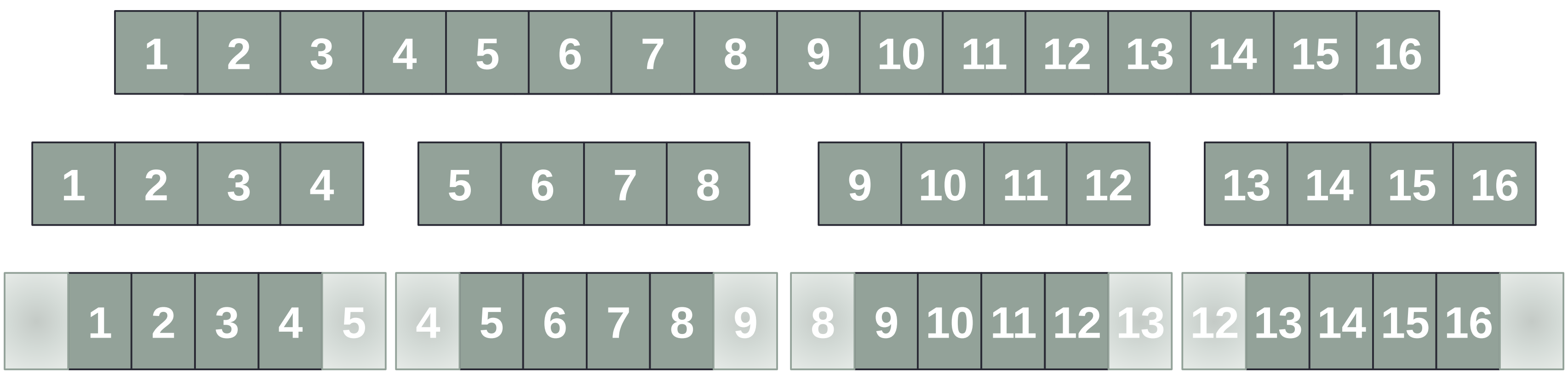

compute dominates


comms dominate

- Usually optimum granularity can only be determine experimentally
  - depends on problem size and target architecture (especially the relative strengths of processing and communications network)
  - granularity can be fixed during compilation or at runtime

The Exchange Interaction: Halo Swapping

- Sub-domains need to know who their neighbours are in order to exchange data
- Non-local data must be present before work can begin.
- Common approach is to use halo-swapping.
  - a.k.a. ghost or shadow boundaries



2D Geometric Decomposition

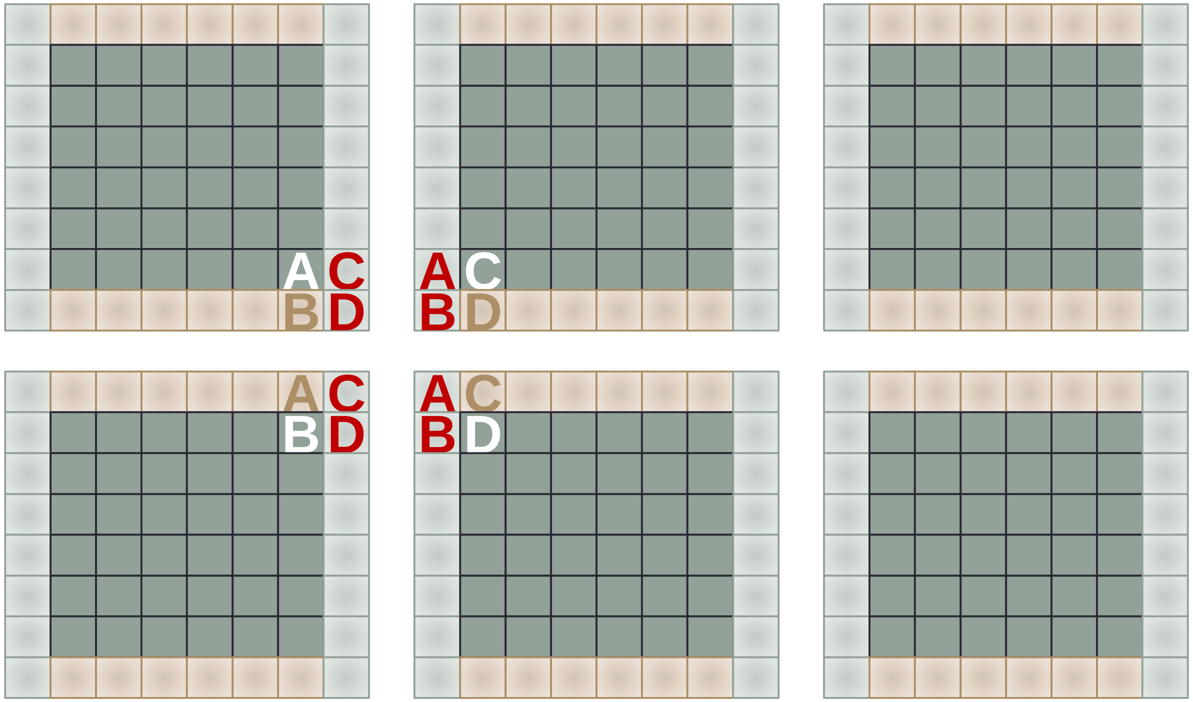
- For each time step...
  - update halos
  - perform calculation
- Improve efficiency...
  - group together the comms associated with swapping each side of a halo

2D Geometric Decomposition

- All decomposition approaches introduce overheads...
  - transferring data on the boundaries
  - synchronisation
  - calculating global quantities
  - volume gives us computation, surface area communication

Example matching halos to elements

- Depth of stencil is number of boundary elements required in each direction.



- If halos swapped after every time step synchronisation needed.

The update (computation) operation

- For each time step complete the exchange before starting calculating the update.

- Better performance can be obtained by overlapping communication and computation.
  - multithreading within a task
  - non-blocking MPI communications

- Need to ensure that the correct neighbour data has been received before performing the update operation.

Overlapping compute and communication

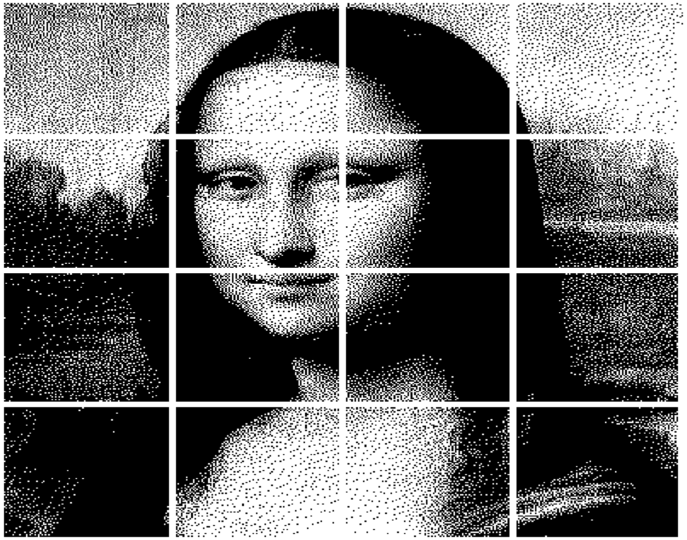
```
for (k=0; k<MAX_ITERATIONS; k++) {
  // initiate non-blocking halo swaps
  [...]
  // block for all communications to complete
  [...]
  for (i=1; i<=NX; i++) {
    rnorm = rnorm + pow(u_k[i]*2-u_k[i-1]-u_k[i+1], 2);
  }
  [...]
}
```

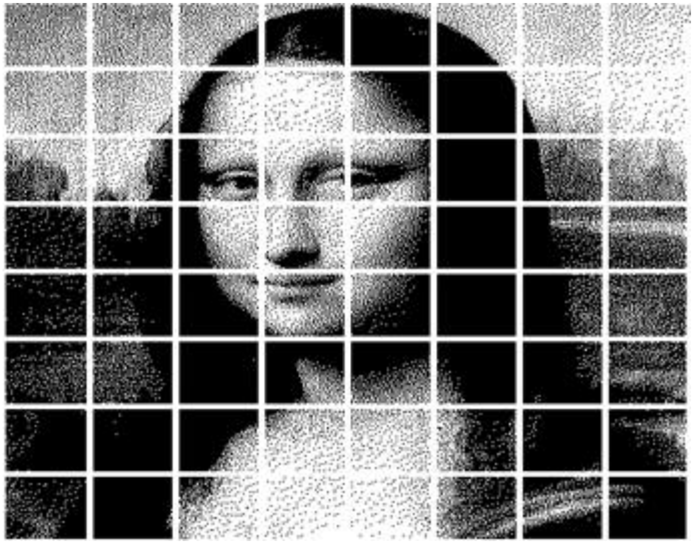
Overlapping compute and communication

```
for (k=0; k<MAX_ITERATIONS; k++) {
  // initiate non-blocking halo swaps
  [...]
  for (i=2; i<=NX-1; i++) {
    rnorm = rnorm + pow(u_k[i]*2-u_k[i-1]-u_k[i+1], 2);
  }
  // block for all communications to complete
  [...]
  rnorm = rnorm + pow(u_k[1]*2-u_k[0]-u_k[2], 2);
  rnorm = rnorm + pow(u_k[NX]*2-u_k[NX-1]-u_k[NX+1], 2);
  [...]
}
```

Task Scheduling

- One task is the update of one sub-domain.
- Tasks need to be mapped to UEs.
  - one per UE is the simplest case
  - several sub-domains per UE
    - may improve load balance
    - harder to synchronise
    - need to choose method of assignment, e.g., linear, cyclical or random





Program Structure

- Geometric Decomposition can be used with one of the following.
  - Loop Parallelism
    - an iteration of the loop corresponds to an update of one sub-domain in the system
    - maps well onto OpenMP
  - SPMD
    - one process per sub-domain
    - exchange operation corresponds to communication between processes
    - maps well onto MPI

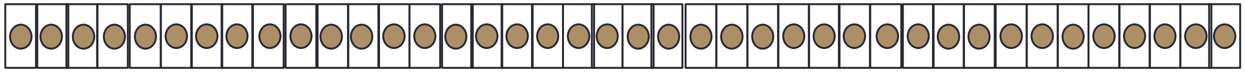
Met Office NAME example

- Numerical Atmospheric Modelling Environment
  - dispersion of particles such as volcanic ash, chemicals and pollutants
  - code is serial and simulations take days to run
  - parallelise code such that simulations can complete within hours

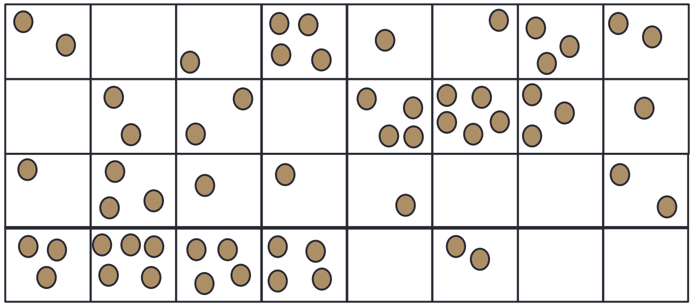
Eyjafjallajökull Volcano Plume
Courtesy of Boaworm
Wikimedia Commons, 2010

Met Office NAME example

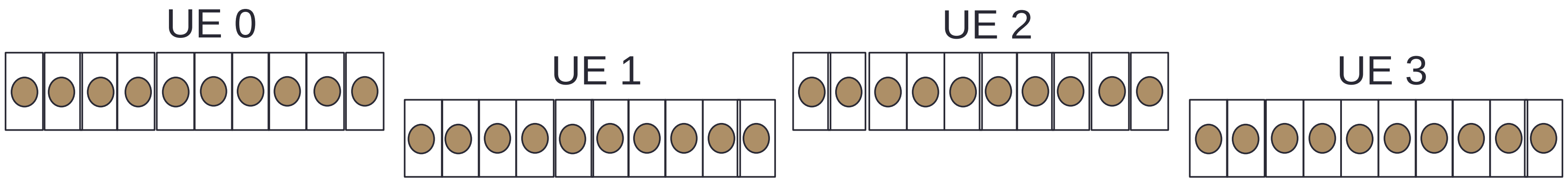
- 1D array of particles, where each particle is associated with a set of properties.
  - e.g., mass, size, position, velocity



- There are three computational steps.
  - update position
  - update velocities using averaged values for wind speed or temperature
  - particle properties can also be altered via chemical reactions
    - requires knowledge of neighbours



Two choices for parallelism (1D)



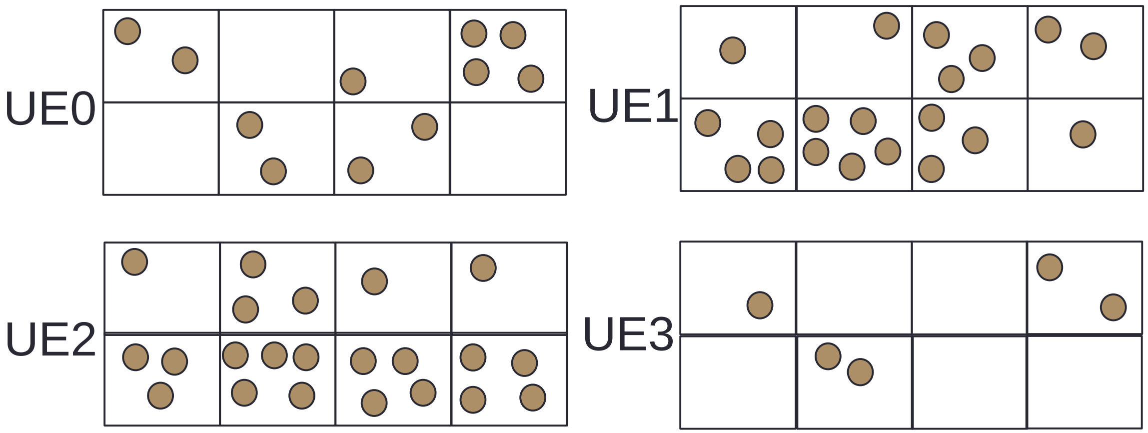
pros

- simple 1D decomposition of particle array
- work is evenly balance balanced across UEs for first two computational steps (*x*,*y*,*z* and ***v***)

cons

- a particle's neighbours may be distributed across all UEs, which maximises communication required to determine impact of chemical reactions

Two choices for parallelism (2D)



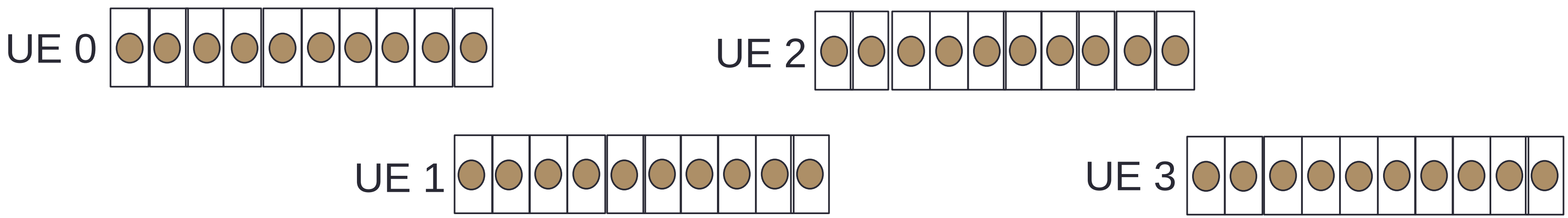
pros

- minimised communications required for chemical reactions has many such calculations will be between particles within the same domain

cons

- non-uniform particle distribution would cause load imbalance
  - could be remedied by irregular decomposition but this adds complexity
- particles have to be transferred to between UEs

Choose 1D decomposition



- Particle array is partitioned irrespective of geographical location.
  - straightforward to code
  - load balancing is included by default
  - chemical interactions could be computed using reductions
    - every particle receives an input from each UE that is calculated from all chemical interactions of neighbouring particles
    - reductions are implemented efficiently on modern machines

Conclusion

- Geometric decomposition is a very common pattern.
- A number of choices (such as UE assignment, task scheduling etc) need to be made during implementation in order to tune for performance and scalability.

- If you're simulating a physical system where most (if not all) interactions are local then a geometric decomposition is usually the best strategy.
  - also maps closely on to problems modelled as (discretised) differential equations