Deciding how to Parallelise a Problem

Introduction

We are going to look at two separate, but closely related aspects of how to parallelise a problem:

- Parallel Decomposition
  - How to split the problem into portions that can be solved in parallel
- Parallel Communication Architecture
  - How to communicate between nodes to ensure that they have the data required to solve the problem

Parallel Decomposition

Introduction

- There are a vast number of different ways in which you can potentially split problems in order solve them in parallel
- Even for a given problem there is usually not a single way in which it might be split
  - Not even necessarily a single best way to split a problem
  - May depend on computer resources available:
    - Number of cores available
    - Amount of memory available of each node
    - Relative speed of processors vs communications
- We will therefore only be looking at a limited set of common ways to split a problem

Data Decomposition

Split the data between the processes:

- Split the output data
- Split the input data
- Split both the input and output data

Data Decomposition: Split the output data

- In many problems, given complete knowledge of the input data, different portions of the output data can be calculated independently

- Example of where it is appropriate
  - Matrix multiplication – Each element in the answer matrix can be calculated independently of the others
  - Communication is required to distribute the input data, but not between the nodes

Do Worksheet 6 Exercise 1

Data Decomposition
Split the input data

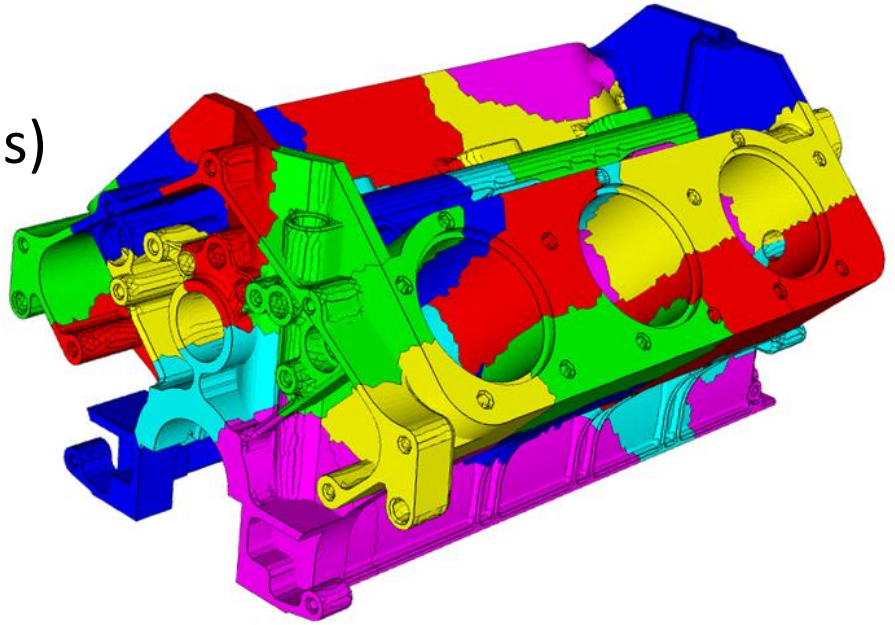
- Different portions of the input data assigned to different processes
- Contributing to a commonly held solution
  - Either globally known (e.g. a shared memory system) or known to a single primary node

Data Decomposition
Split both the input and output data

- Different portions of the input and output data in a problem may be closely related
  - E.g. When simulating a physical system there may be input data about a region or time, with some other properties of the system being calculated for those positions or times
- Very common approach in both distributed and shared memory systems
  - In distributed memory systems this will often require communication of information at the boundary of the data
  - In distributed memory system it restricts the need for blocks to memory associated with the edge of the data regions

Data Decomposition
Domain Decomposition

- Domain decomposition is a very commonly used example where both input and output data is split
- Used in the simulation of physical systems
  - The system is divided into a set of regions (domains)
  - Each process responsible for a different domain
  - Communication of data at the edge of domains



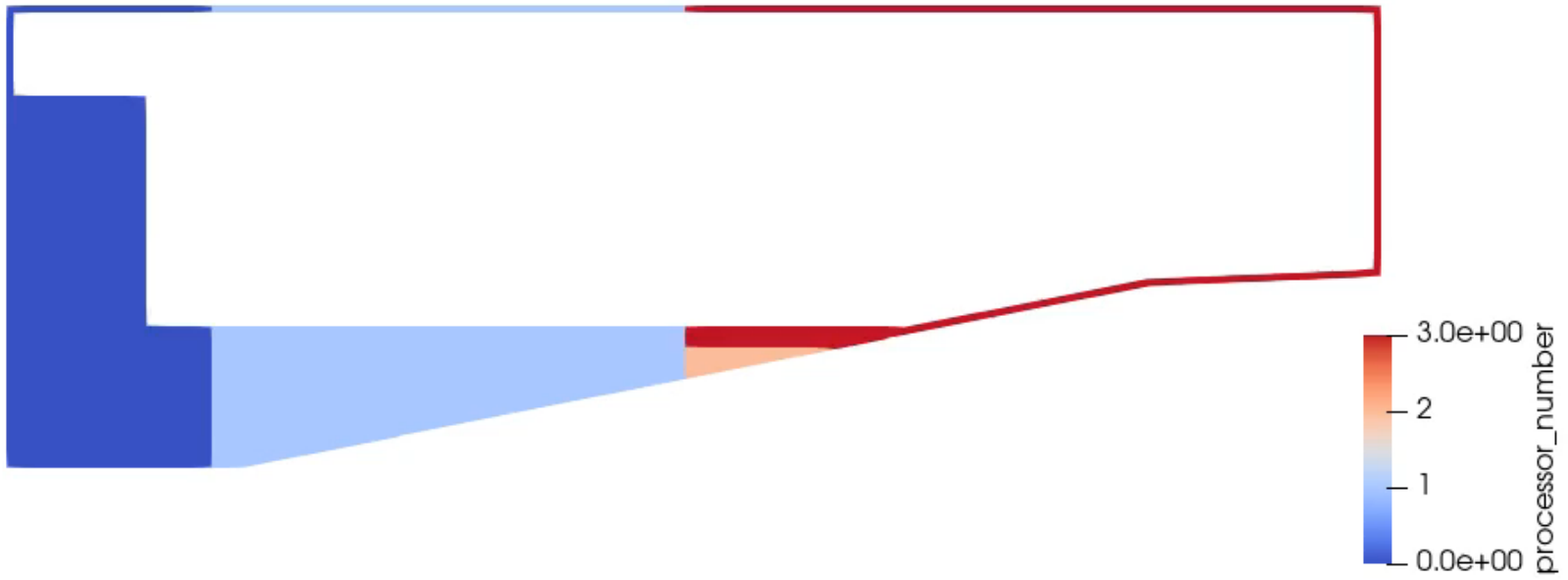
Domain decomposition will form the basis for the coursework

Data Decomposition
Load Balancing

- To ensure maximum parallel efficiency you don't want some of the processes waiting idle while other processes work
- Need to spread the computational load – try to give each process the same amount of work to do
  - When splitting should err on giving one process slightly less to do than the others rather than slightly more
  - Typically want all processes to be responsible for the same number of degrees of freedom (e.g. the same number of nodes or elements)
  - If the simulation resolution is spatially constant this may be equivalent to evenly splitting the sizes of the regions
- Need to simultaneously try to keep communication to a minimum
  - Reduce the surface area of the regions
- Sometimes hard to access the computational cost of different decompositions analytically
  - A tactic I sometimes use is to time how long a process is idle waiting for the communications on other processes to complete
    - Balance based on idle time – Give those waiting longest more to do

Load Balancing in Smoothed Particle Hydrodynamics (SPH)

- I do a lot of simulation work using SPH – Load balancing is on of the keys to efficient simulation
  - This is a simple simulation run on 4 cores – Colour on bottom video corresponds to processor number

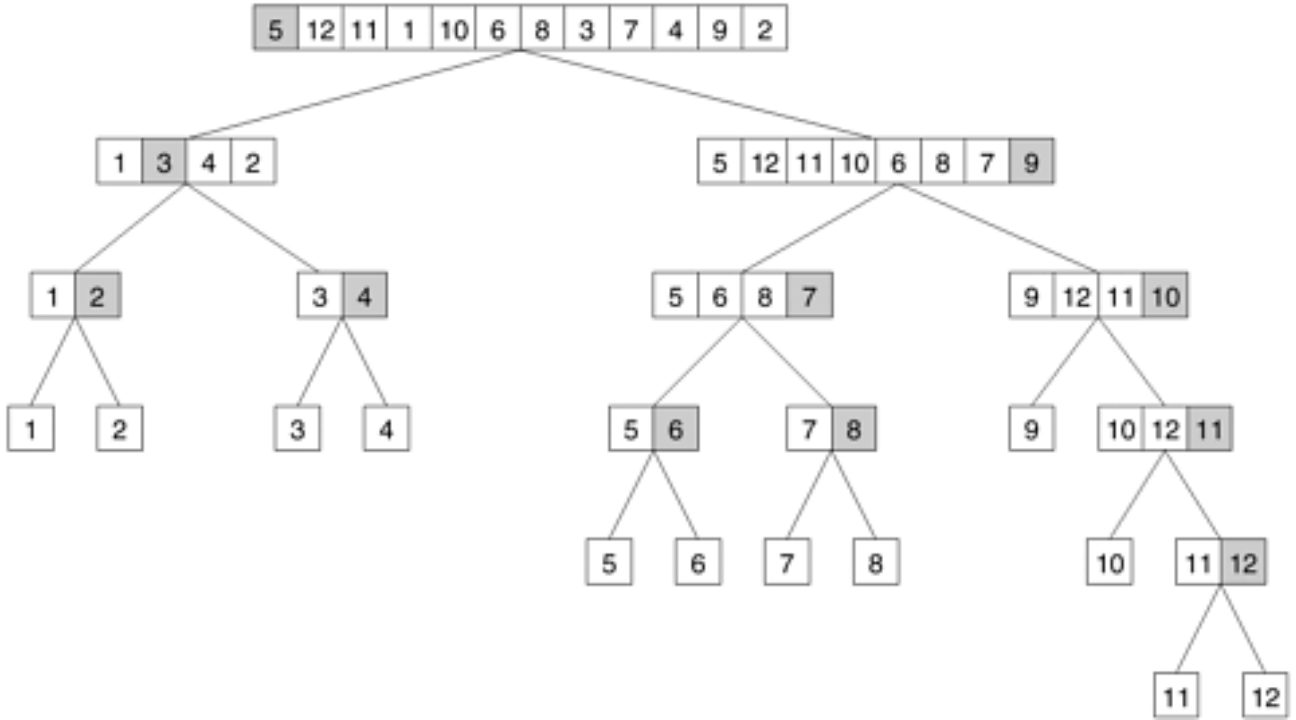


Recursive Decomposition

- Progressively split a problem into smaller portions and then combine the solutions
  - Works best for problems where the computational cost of the solution increases faster than linearly with the size of the problem and where the cost of splitting and combining solutions is comparatively cheap
  - So called “divide and conquer” algorithms
  - Often used in serial algorithms, but can also be used as the basis for parallel decomposition

Recursive Decomposition - Quicksort

- Classic example of recursive decomposition is the quicksort algorithm
  - Brute force sorting of *n* items is O(n^2)
  - …,but you can split lists in O(*n*) time and combine them in O(*1*) time with the total time taken being approximately O(*n log(n)*)



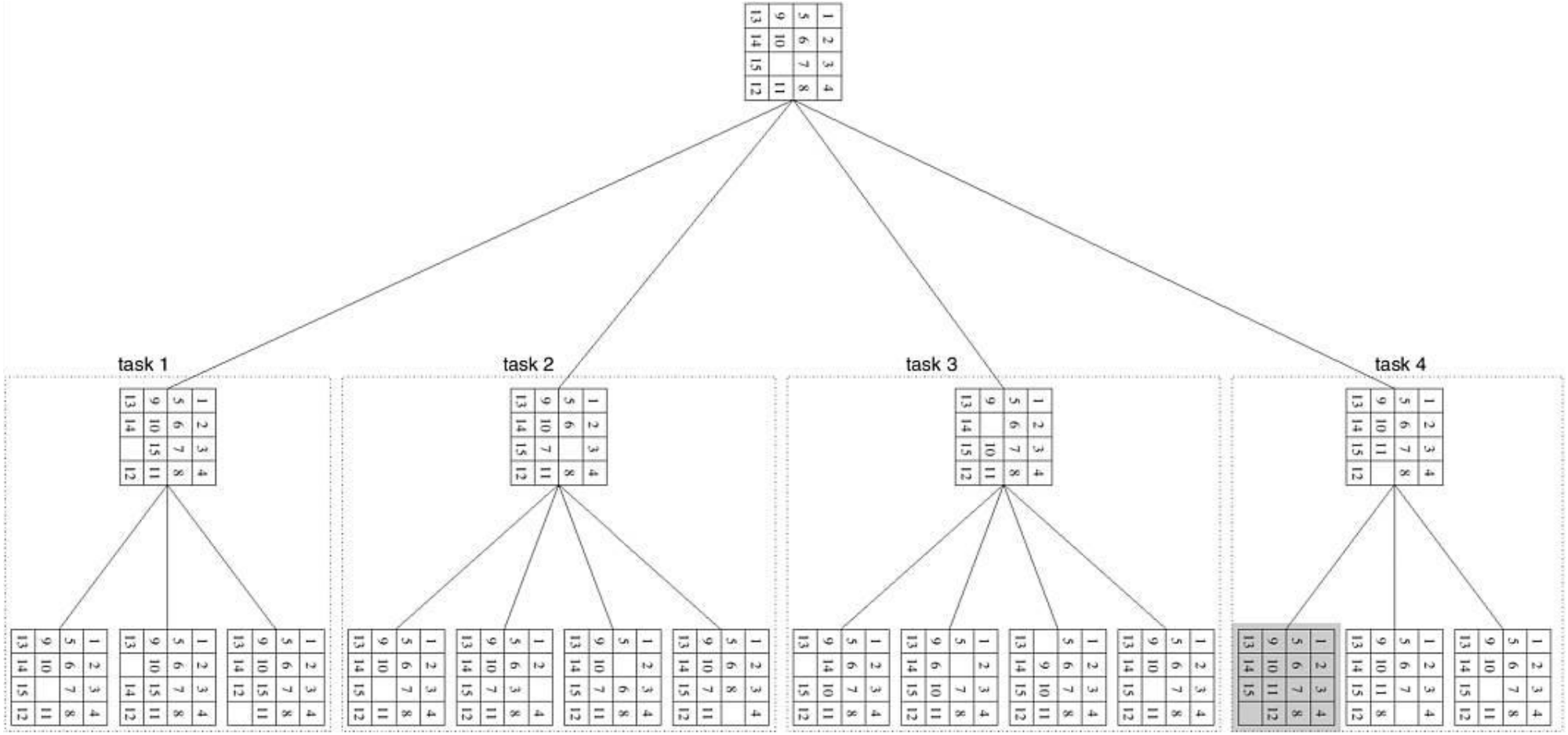
Do Worksheet 6 Exercise 2

Exploratory Decomposition

- Used for searching for solutions in multi-step problems
- Has some similarity to data decomposition in that the initial search space is split between processes
  - Note that this might not be a true split of data, but could be a split of a parameter space
- The next stage is that a new set of data/parameters are produced from the previous stage and these are then split
  - Can be split onto new processes if available, otherwise the current processes become responsible for more states
  - Could allow processes to become available again if their search hits a dead-end

Exploratory Decomposition Example

- Finding solution to 15 puzzle problem



This is the Workshop Exercise

Speculative Decomposition

- Can be used in problems where subsequent tasks depend on the outcome of earlier tasks
  - E.g. two different tasks to complete depending on whether the solution to an earlier task is true or false
- In speculative decomposition you carry out all the subsequent tasks without waiting for the result from the earlier task
- Particularly advantageous if the earlier tasks take a larger or similar amount of time compared to all of the subsequent tasks

Speculative Decomposition Example

- Simulation of system with multiple interacting components