Fractals

Investigating task farms and load imbalance

Reusing this material

http://www.archer.ac.uk

support@archer.ac.uk

The Mandelbrot Set

- The Mandelbrot Set is the set of numbers resulting from repeated iterations of the complex ($i = \sqrt{-1}$) function:

$$Z_n = Z_{n-1}^2 + C$$ with the initial condition $$Z_0 = 0$$

- $C = x_0 + iy_0$ belongs to the Mandelbrot set if $|Z_n|$ remains bounded i.e. does not diverge

$$Z_n = x_n + iy_n \,,\; Z_n^2 = (x_n^2 - y_n^2 + 2ix_ny_n) \,,\; |Z_n|^2 = (x_n^2 + y_n^2)$$

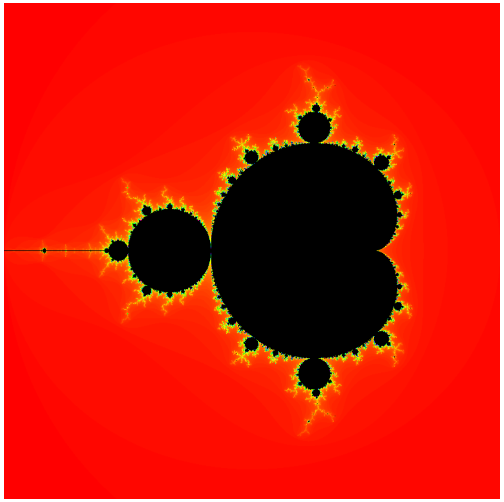
The Mandelbrot Set cont.

- Separating out the real and imaginary parts gives:

$$Z_n = Z_n^r + iZ_n^i$$

$$Z_n^r = x_{n-1}^2 - y_{n-1}^2 + x_0$$

$$Z_n^i = 2x_{n-1}y_{n-1} + y_0$$



- Take the threshold value as:

$$|Z|^2 \geq 4.0$$

- Set the maximum number of iterations to N_{max}
  - Assume that Z does not diverge at higher values of N_{max}

The Julia Set

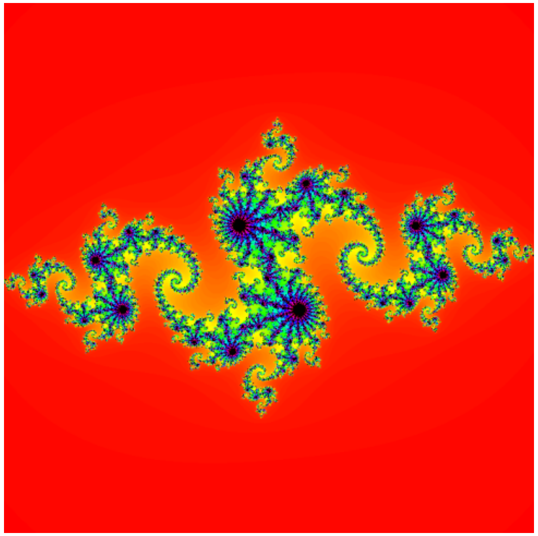
- Similar algorithm to Mandelbrot Set – recall:

$$Z_n = Z_{n-1}^2 + C, \quad C = x_0 + iy_0, \quad Z_0 = 0$$

- There are an infinite number of Julia sets, parameterised by a complex number *C*

$$Z_n = Z_{n-1}^2 + C, \quad Z_0 = x_0 + iy_0$$

  - for example, *C* = 0.8 + *i* 0.156

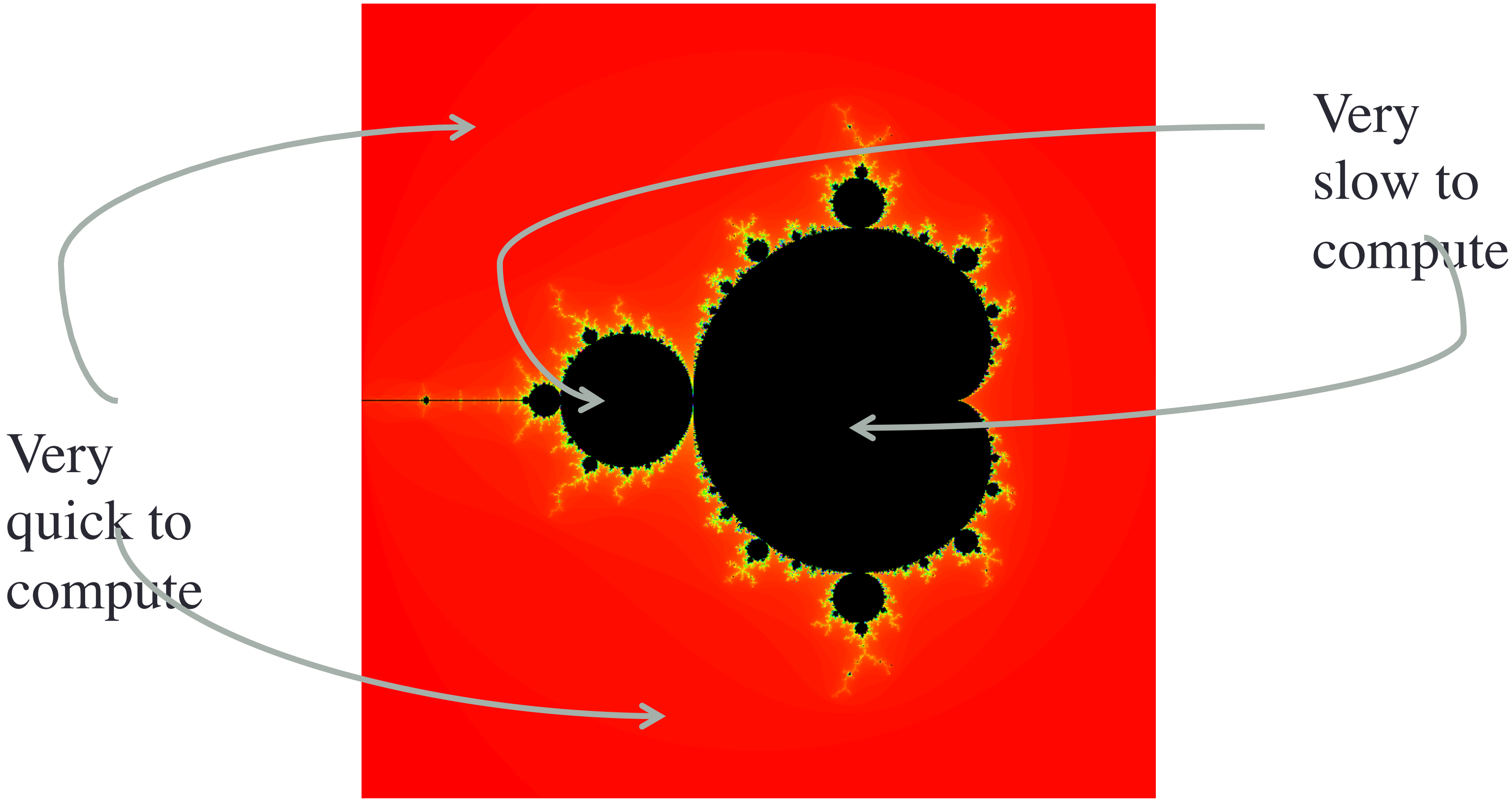


Visualisation

To visualise a Mandelbrot/Julia set:

- Represent the complex plane as a 2D grid where complex numbers correspond to points on the grid (x, y)
- Calculate number of iterations N for the series to diverge (exceed the threshold) for each point on the grid
  - If it does not diverge, $N = N_{max}$
- Convert the value of N to a colour and plot this on the grid

Mandelbrot Set



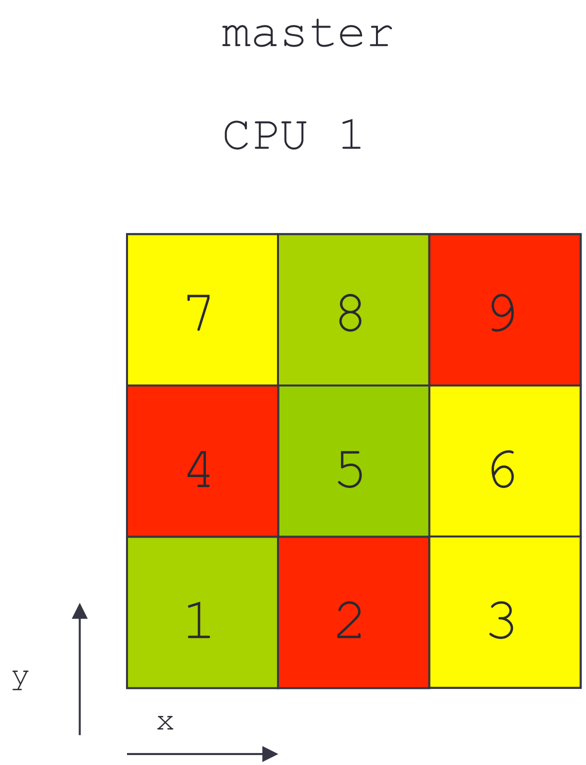
Parallelisation

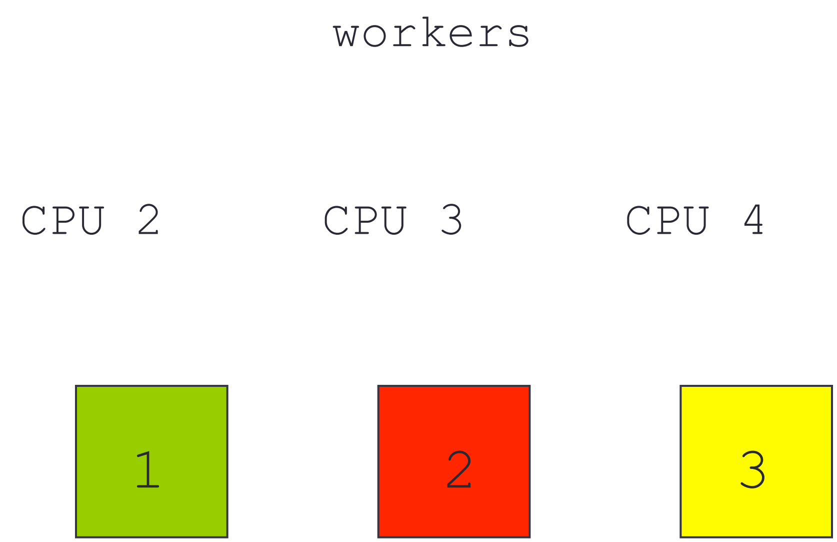
- Values for each coordinate depend only on the previous values at that coordinate.
  - decompose 2D grid into equally sized blocks
  - no communications between blocks needed.
- Don't know in advance how much work is needed.
  - number of iterations across the blocks varies.
  - work dynamically assigned to workers as they become available.

Implementation

- Split the grid into blocks:
  - each block corresponds to a task.
  - **master** process hands out tasks to **worker** processes.
  - workers return completed task to master.

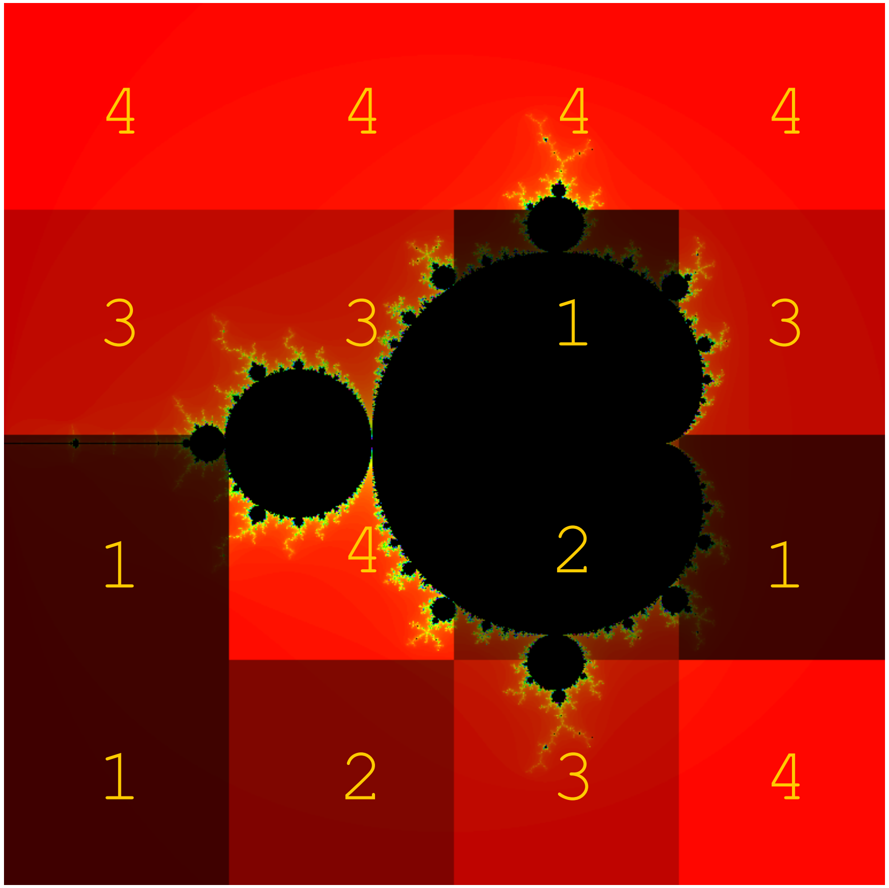
Example: Parallelisation on 4 CPUs





- In diagram, colour represents which worker did the task
  - number gives the task id
  - tasks scan from left to right, moving upwards

Parallelisation cont.



- in supplied code
  - shading represents worker
  - here we have added worker id as a number by hand

- e.g. taskfarm run on 5 CPUs
  1 master
  4 workers
- total number of tasks = 16

Exercise

- You are supplied with source code etc.
- Compile and run on ARCHER
  - visualise results
- Quantify performance results
- For a fixed number of workers
  - improve load balance by increasing number of tasks (decrease size)
  - compute LIF to estimate minimum achievable runtime
  - is this minimum ever reached?

Fractals

Outcomes

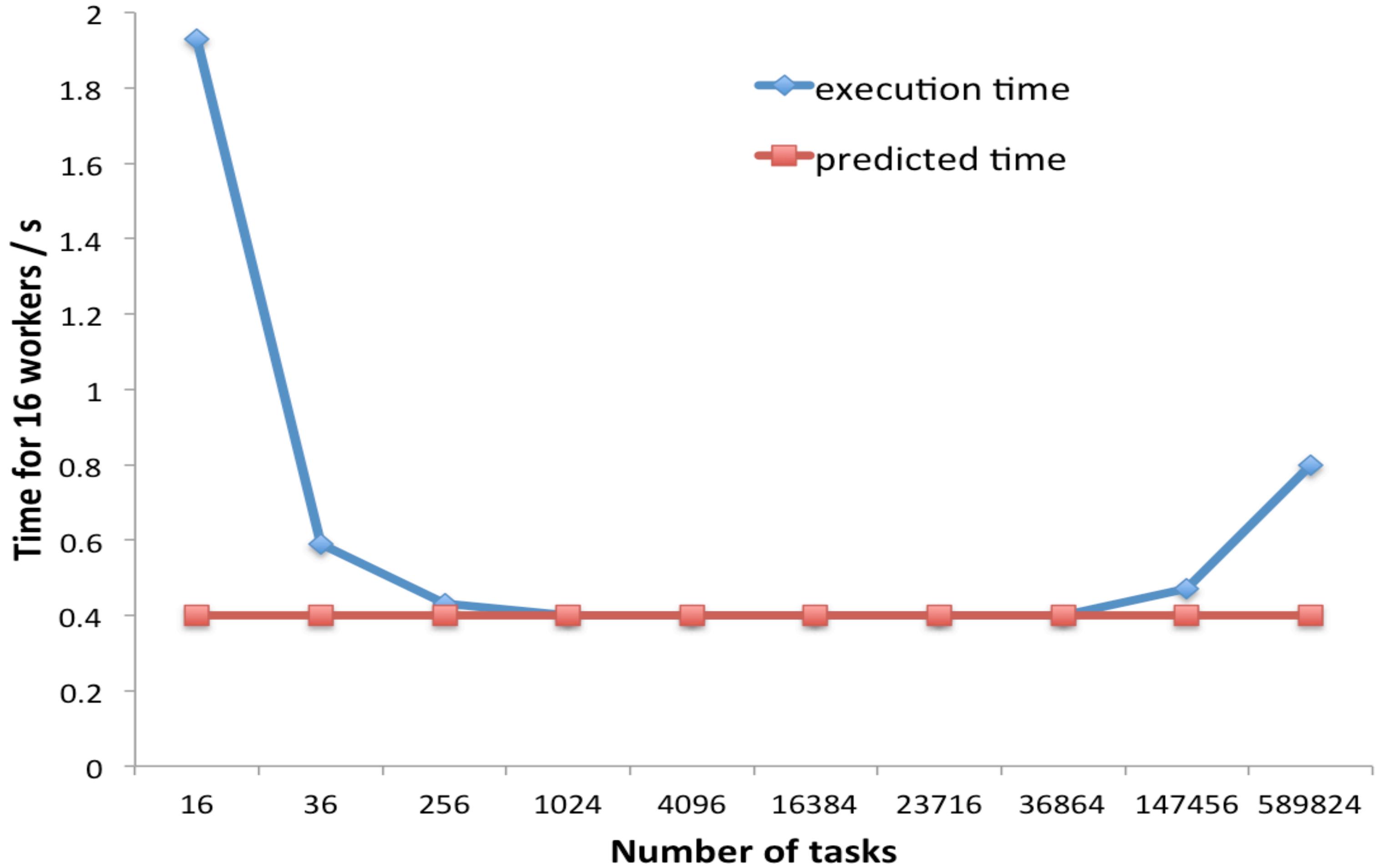
Example results – fixed number of workers

Example results for the default image size (768 × 768 pixels), fixed number of iterations (5000), fixed number of workers (16) and varying number of tasks :

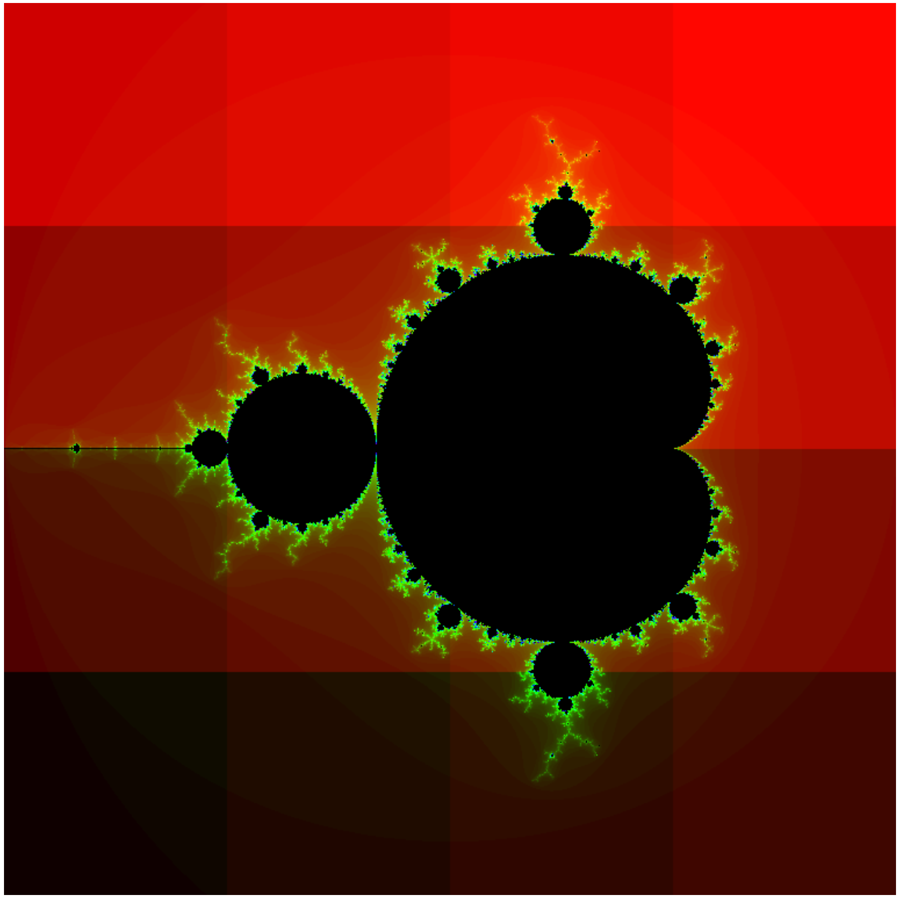
Number of Tasks (Task Size)	Time (s)	Load Imbalance Factor
16 (192 × 192)	1.93	5.034
64 (96 × 96)	0.59	1.501
256 (48 × 48)	0.43	1.108
4096 (12 × 12)	0.4	1.017
36864 (4 × 4)	0.4	1.003
147456 (2 × 2)	0.47	1.017
589824 (1 × 1)	0.80	1.006

Table 2: Example execution Times for 16 workers and varying number of Tasks.

Results cont.



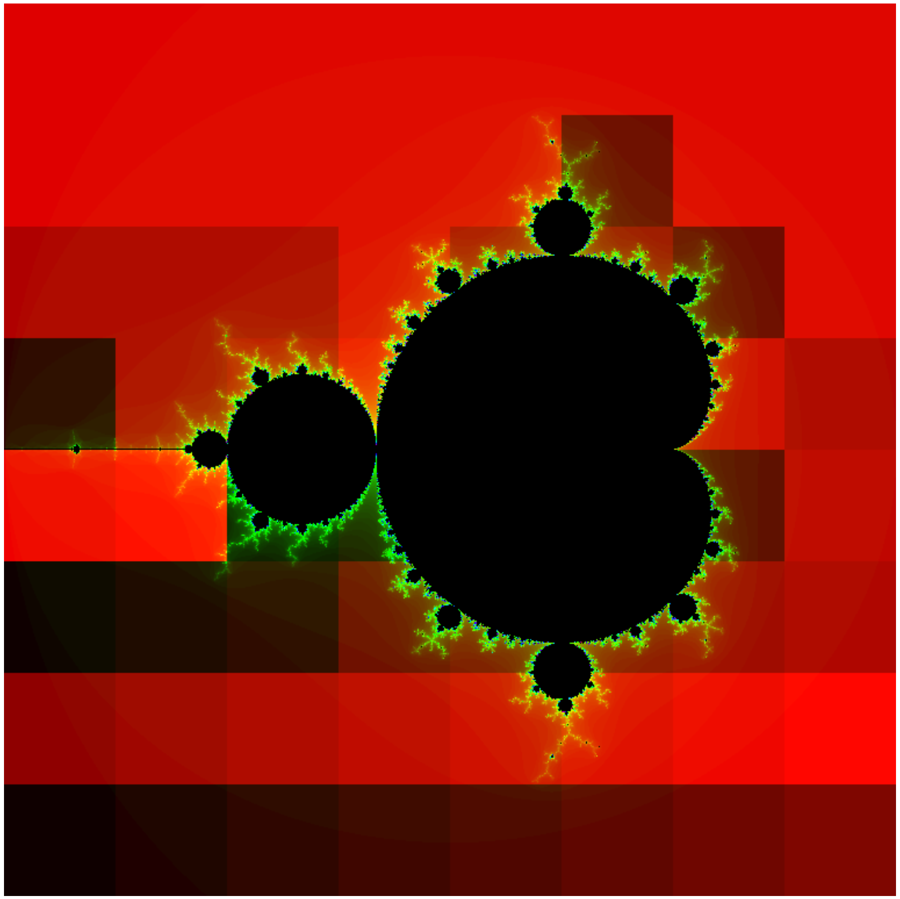
16 workers and 16 tasks



```
-----Workload Summary (number of
iterations)----

Total Number of Workers: 16
Total Number of Tasks:   16

Total   Worker Load: 498023053
Average Worker Load: 31126440
Maximum Worker Load: 156694685
Minimum Worker Load: 62822

Time taken by 16 workers was
1.929219 (secs)
Load Imbalance Factor: 5.034134
```

archer epcc THE UNIVERSITY OF EDINBURGH

16 workers and 64 tasks



```
-----Workload Summary (number of
iterations)---------

Total Number of Workers: 16
Total Number of Tasks:   64

Total   Worker Load: 498023053
Average Worker Load: 31126440
Maximum Worker Load: 46743511
Minimum Worker Load: 10968369

Time taken by 16 workers was
0.586923 (secs)
Load Imbalance Factor: 1.501730
```

Key points to take away

TASK FARMS

- Also known as the master/worker pattern
- Allows a master process to distribute work to a set of worker processors.
- Can be used for other types of tasks but it complicates the situation and other patterns may be more suitable for implementing.
- Master process is responsible for creating, distributing and gathering the individual jobs.
- Can improve load balance by using more tasks than workers
  - with some overhead
- Load imbalance adversely affects performance
  - especially as number of processors increases

Key points to take away

TASKS

- Units of work
- Vary in size, do not have to be of consistent execution time. If execution times are known it can help with load balancing.

QUEUES

- Master generates a pool of tasks and puts them in a queue
- Workers assigned task from queue when idle

Key points to take away

LOAD BALANCING

- How a system determines how work or tasks are distributed across workers (processes or threads)
- Successful load balancing avoids idle processes and overloading single cores
- Poor load balancing leads to under-utilised cores, reducing performance.

Key points to take away

COST

- Increasingly important
- Finite budgets require optimal use of resources requested.
- Load balancing is just one method of ensuring optimal usage and avoiding wasting resources.
- More power and resources do not necessarily mean improved performance.
- Always ask – is it necessary to run this on 4000 cores or could it be run on 2000 more efficiently?