

Fractals

Outcomes

EPSRC



The Mandelbrot Set

- The Mandelbrot Set is the set of numbers resulting from repeated iterations of the complex function:

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

- $C = x_0 + iy_0$ belongs to the Mandelbrot set if $|Z|$ converges.

$$Z = x + iy \rightarrow Z^2 = x^2 + i2xy - y^2$$



The Mandelbrot Set cont.

- Separating out the real and imaginary parts gives:

$$Z^r = x^2 - y^2 + x_0$$

$$Z^i = 2xy + y_0$$

- Take the threshold value as:

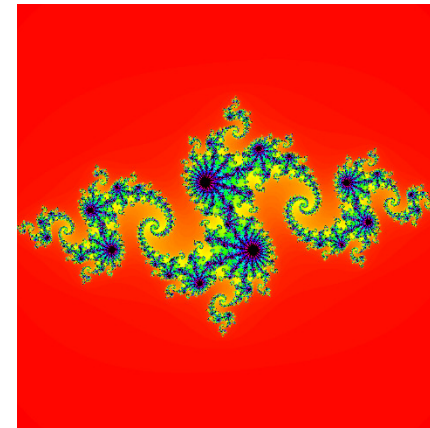
$$|Z|^2 \geq 4.0$$

- Set the maximum number of iterations N
 - Assume that Z does not diverge at higher values of N



The Julia Set

- Similar algorithm to Mandelbrot Set
- Starting coordinates x_0 and y_0 represent fixed point from inside the Mandelbrot set



Visualisation

To visualise a Mandelbrot/Julia set:

- Represent the complex plane as a 2D grid.
- Calculate number of iterations N for complex numbers C corresponding to points on the grid.
- Convert the value of N to a colour and plot this on the grid.



Parallelisation

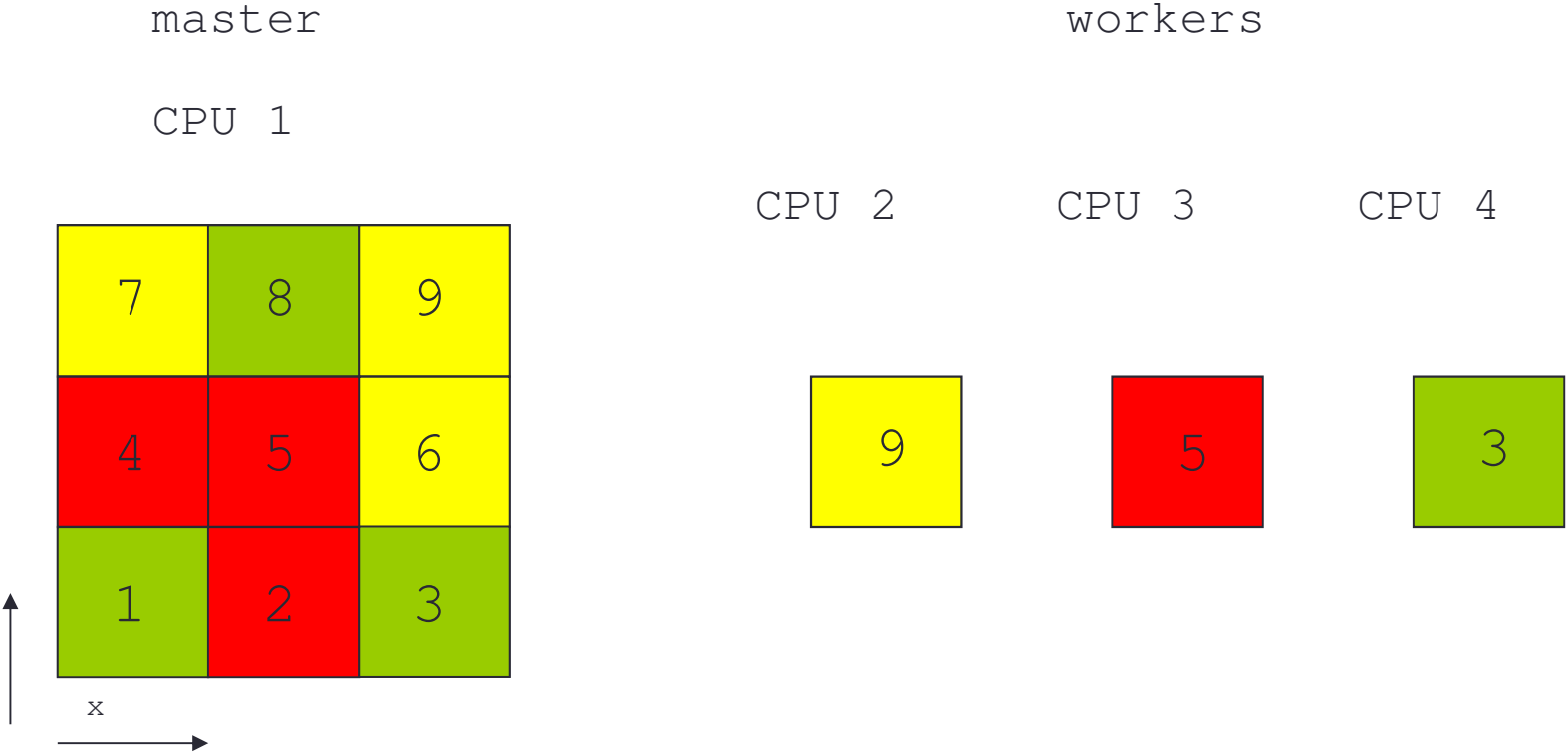
- Values for each coordinate depend only on the previous values at that coordinate.
 - decompose 2D grid into equally sized blocks
 - no communications between blocks needed.
- Don't know in advance how much work is needed.
 - number of iterations across the blocks varies.
 - work dynamically assigned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

Implementation

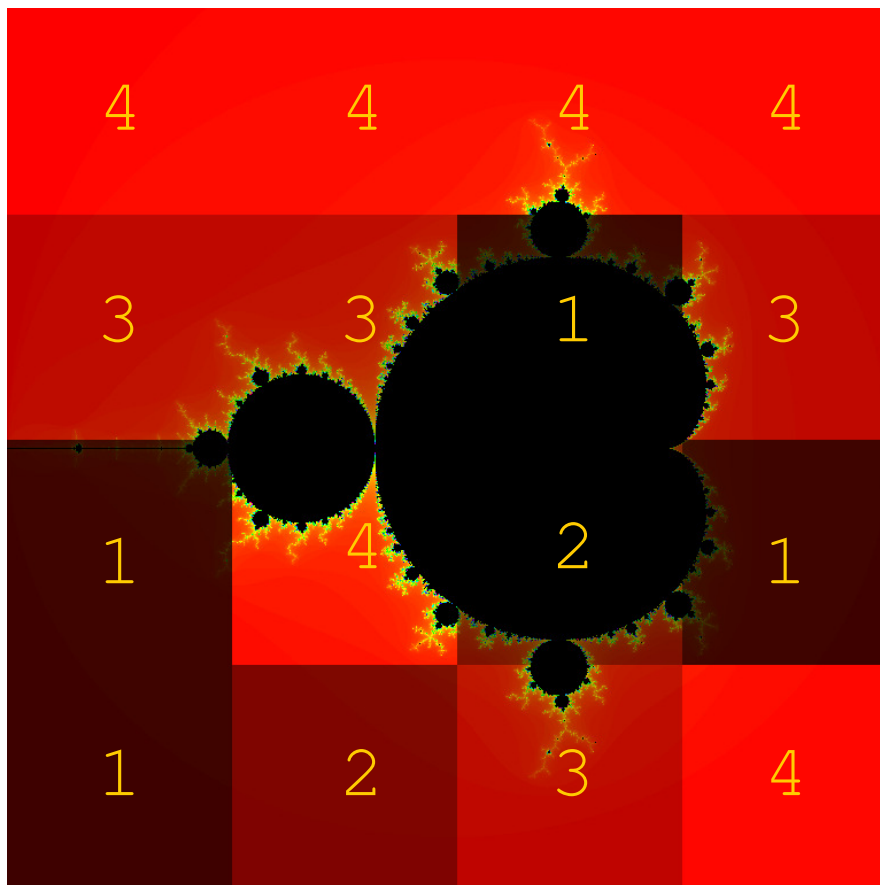
- Split the grid into blocks:
 - each block corresponds to a task.
 - **master** process hands out tasks to **worker** processes.
 - workers return completed task to master.



Example: Parallelisation on 4 CPUs



Parallelisation cont.



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



Example results

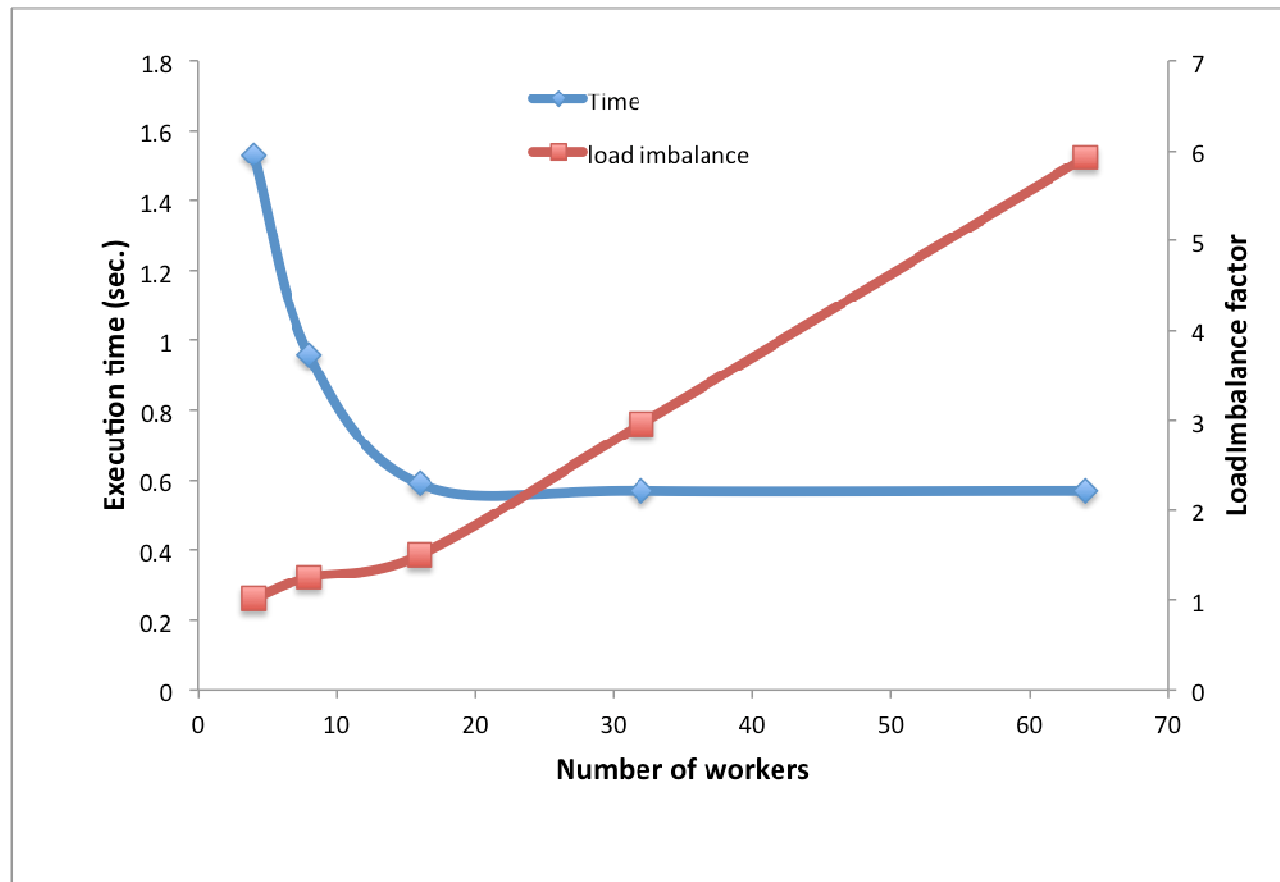
Example results for the default image size (768 × 768 pixels), fixed number of iterations (5000), 64 tasks of the size of 96 x 96 pixels and varying number of workers :

| Workers | Time (s) | Avg. Workload | Max Workload | Min Workload | Load Imbalance Factor |
|---------|----------|---------------|--------------|--------------|-----------------------|
| 4 | 1.52 | 124505763 | 126124383 | 122696852 | 1.01 |
| 8 | 0.96 | 62252881 | 77744803 | 51117022 | 1.25 |
| 16 | 0.59 | 31126440 | 46737752 | 10968369 | 1.50 |
| 32 | 0.57 | 15563220 | 46114456 | 67614 | 2.96 |
| 64 | 0.57 | 7781610 | 46089216 | 9246 | 5.92 |

Table 1: Example Run-Times for different number of workers and their Avg/Min/Max Workloads.



Results cont.



Example results

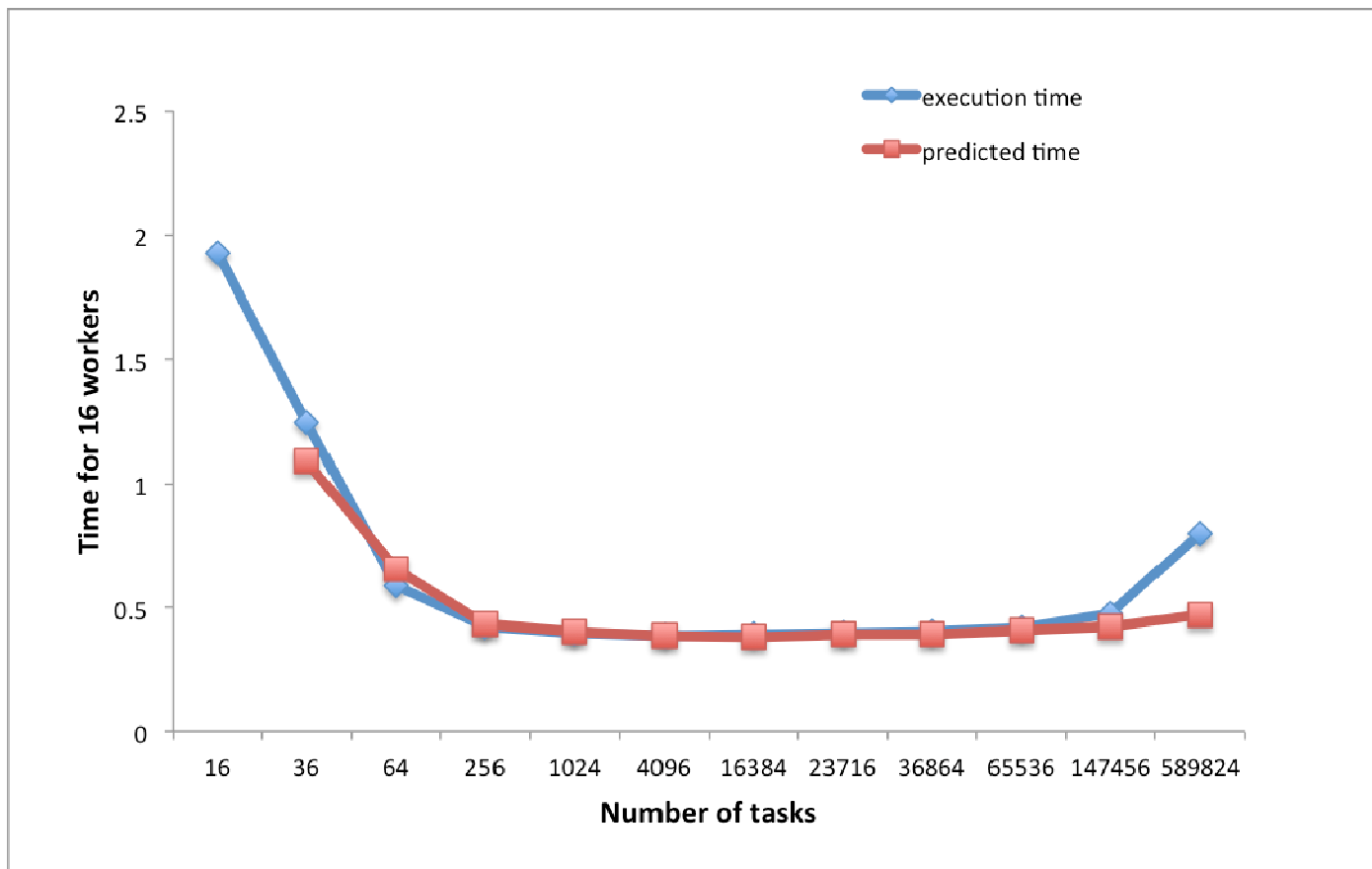
Example results for the default image size (768×768 pixels), fixed number of iterations (5000), fixed number of workers (16) and varying number of tasks :

| 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.



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 workers processes.
- 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.



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. For a task farm, the tasks should as far as possible be discrete units of work from the overall

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?

