Person wants to ZOOM in DEEP into the Mandelbrot Set...

...but they are limited by their weak device...

Distributed computing will save the day.

User Assembles a Request

```
{
START: [0.29340, -1.3941],
ZOOM: "8x",
NUM_FRAMES: 128,
FPS: 20,
THEME: "Galaxy",
FORMAT: "mp4"
}
```

CLUSTER LEADER (PHASE I)

Sequence generation "task" is broken down into the following steps:

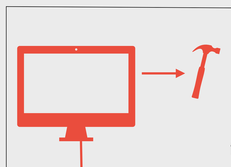
1. Compute the sets of (real, imaginary) coordinate pairs that define all of the frames needed to construct a request.

2. Check if frames have been generated in the past before starting computation.

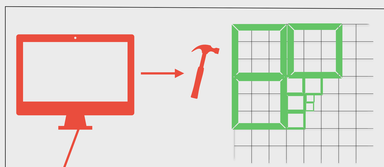
3. Assemble an array of promised 'futures' that represent atomic units of workload. (@ray.remote)

4. Execute the futures using Ray.io to gather all available GPU/CPU cores and summon them (@ray.get)

WORKER NODES

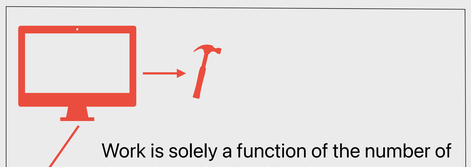


Each worker executes pyopencl instructions that accommodate CPU, GPU, and many other optimizations



INFINITE inductive bias makes this an INFINITELY scalable problem.

Type to enter text



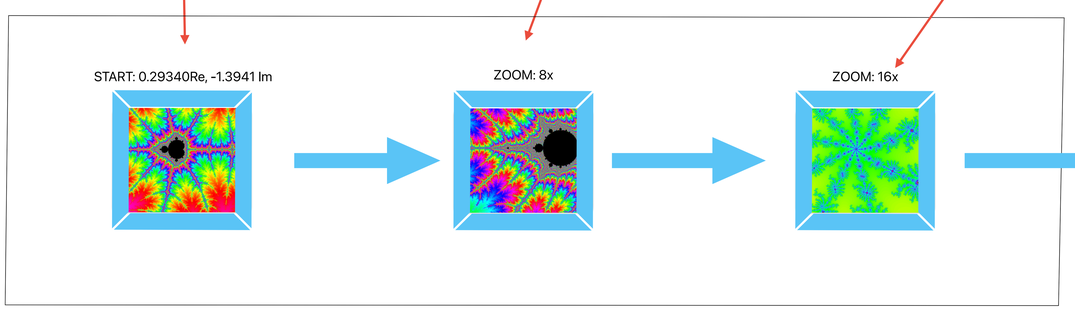
Work is solely a function of the number of pixels for which the iterative convergence/ divergence process is preformed. Frames, grids, sequences, are all just recursively built up from pixels.

• • •

Monitor Nodes using dialogue. Launch Dashboard for performance stats.

Each workers fulfills promises by processing N frames.

Frames encoded as b64 strings.



CLUSTER LEADER (PHASE II)

1. Persist any novel frames.

2. Stitch everything together and serve the users request!