INTRODUCTION

- Goal: Create a framework for completing tasks written in C# .NET Core. Used to teach concepts related to parallel and distributed systems with new and upcoming technology.
- .NET Core is entirely cross platform with capabilities including: threading, networking and image processing.
- .NET Core includes languages like the multi-paradigm F#, encompassing functional as well as object oriented programming.
- .NET Core supports fully featured C# Task Parallel Library.

FEATURES

- Application to setup system with server, backend nodes, and scheduler along with user program submission and feedback.
- Job, JobTask, and JobResult classes for representing an abstract distributed task.
- Cross platform implementation, ability to run nodes and execute code on multiple environments simultaneously.
- Example programs including, image processing, distributed file sorting, and cryptographic computations.
- API Reference: https://acf5118.github.io/MS-Project/

SYSTEM ARCHITECTURE

EXAMPLE: BUDDhabrot

```csharp
public class BuddhabrotDis : Job
{
    private const int DesiredNodes = 4;
    ...
    public override int RequestedNodes() {
        return DesiredNodes;
    }
    ...
    public override void Main(string[] args) {
        for (var i = 0; i < DesiredNodes; i++) {
            AddTask(new BuddhabrotTask(Width, Height, 
                                        Xmin, Xmax, Iter, Samples));
        }
    }
    ...
}
```

EXAMPLE: RESULTS

Figure 2: A different way of rendering the Mandelbrot set.
Instead of selecting initial points on the real-complex plane, one for each pixel, points are selected randomly. Each initial point is iterated using the standard Mandelbrot function in order to test whether it "escapes" from the region near the origin. Escaped pixels get reiterated, counting the selected pixels each iteration generates the image pictured above. 

Figure 3: Example of running Time vs Cores

REFERENCES