CS 423
Group 6
Danny Gaeta - gaeta2
Thomas Nelson - tanelso2
Shane Carey - scarey5
Christian Mahnke - mahnke2

## **Implementation and Design Decisions**

- Job Queue
  - We use a python 'list' so that we can efficiently slice the queue and send jobs to the other machine. Additionally, python lists can be treated as queues. Moreover, we wrap this 'list' in a mutex to ensure accesses are thread safe.
- Worker Thread
  - In order to sleep the Worker thread by the 'throttling value', we calculate the time that the Thread should sleep based on the throttling value and the time that it takes the job to process.
- Hardware Monitor
  - We check the CPU-use using the python package 'psutil' each second.
- Transfer Manager
  - The Transfer-manager is implemented using python sockets and the python package 'pickle'. The 'pickle' package is used to serialize python objects to a byte stream.
- State Manager
  - We defined a class called 'State', which stores all the pertinent state information. The 'State' object is updated every second and sent to the other machine.
- Load balancing Function
  - We defined a heuristic that uses the state information from each machine and calculates how many jobs should be sent (if any) to balance the loads of each machine.
  - The formula being solved is:

$$\frac{(Jobs_{local} - x)cpu_{local}}{throttling_{local}} = \frac{(Jobs_{remote} + x)cpu_{remote}}{throttling_{remote}}$$

- Adaptor
  - Our implementation of the adaptor finds how many jobs need to be sent, and we decided upon a job threshold value of 10 jobs (only send jobs if there are more than 10).
  - In addition, if there are jobs which have finished being processed on the remote server, then the adaptor on the remote server will send them to the local server.
  - Moreover, the Adaptor tells the State-manager to send state information to the other machine.