Parallel and Distributed Computing
COS786

Quinton Weenink[1]

[1]Department of Computer Science – University of Pretoria
Pretoria, South Africa

qweenink@gmail.com, u13176545@tuks.co.za

1. Distributed System Architecture

1.1. Leadership

A master-slave approach was decided on due to the architecture and functionality of *rummy.pyc* file. Mainly due to *rummy.pyc* not being able to resend the clues that need to be solved and that multiple agents can't request there clues separately.

The master starts up as many agents as specified automatically and the remaining agents could be started up manually by the user. Automatic restarting of unresponsive agents can only be done on the main host (the computer that the master is running on). A "heath-check" over all distributed hosts which restarts unresponsive agents could improve the solution.

```
for agent in agentList:
    if agent.timeSinceSeen > 25 seconds:
        agent.restartAgent() # Start the agent locally
```

1.2. Corruption

After each map iteration the corruption percentage of the agents are used to decide weather they should be removed. For this implementation it was decided on removing agents with a failure rate greater than 7.5%

```
for agent in agentList:
    if (failureRate > 7.5):
        agentList.remove(agent) # Remove poor performing agent
        startNewAgent() # Starts agent on master host-name
```

1.3. Networking

Sockets were used for the communication between the slave and master due to their simple nature. All communication happens through the same port on the master host (eg. 12345)

```
while True:
    socket.accept() # Accept all connections
    addClueToSovedClueList() # Add solved clue to list that will be
   verified
    sendNewClue() # Send agent new clue to solve
```

Due to the master-slave architecture this solution will not perform well with a large network of agents, with the master being the bottleneck of the solution.

2. Compilation and execution instruction instructions

Note: This project was written for Python 2.7 on Linux (Ubuntu).

Firstly ensure that the *data* directory as well as *rummy.pyc* are in the root directory of the project. Also ensure that all of the Python files have execute rights:

```
chmod +x *.py *.pyc
```

Run the master (quartermaster.py) with:

```
./quartermaster.py <PORT> <NUM-PIRATES> <NUM-MASTER-START-PIRATES>

Eg: ./quartermaster.py 12345 10 10

Eg: time -p ./quartermaster.py 12345 10 10 # can be used to measure the
    execution time
```

PORT
: The port all communications will take place on

NUM-PIRATES
: The number of pirates the execution will use to solve the solution (± CPU Cores + 2)

NUM-MASTER-START-PIRATES
: The number of automatically started pirates by the master

To start a pirate on your own (Start before 25 seconds into execution):

```
./pirate.py <PORT>
```

3. Concerns

3.1. Security

The master accepts any requests from unverified agents therefor this solution is not secure. This can be improved by validating a unique agent key that is assigned to each agent on start-up.

3.2. Resilience

The master-slave architecture is heavily dependent on the master process and will fail if the master is killed or blocked.