Distributed Computing - Assignment No.1

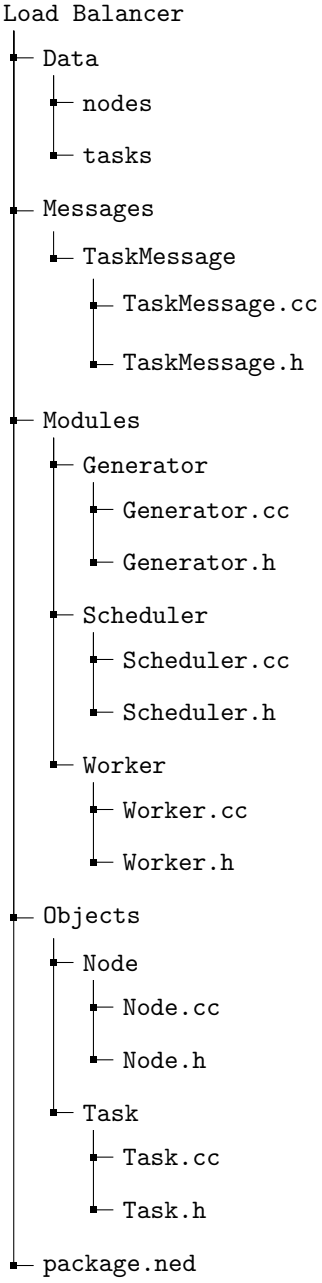
Ali Azhar

Wednesday, October 06, 2018

Contents

1 Code Structure

To understand how the simulation works, please first have a look at the code structure:



1.1 Data

The Data folder contains information about the nodes and tasks. These files are read by the Scheduler and Generator modules respectively to dynamically generate cModule and TaskMessage objects.

1.2 Messages

The Messages folder contains classes inherited from cMessage class. The TaskMessage class is used to communicate requirements and current status of the tasks. Additionally, it contains startTime and endTime fields to track time taken from it's initialization to it's completion.

1.3 Modules

The Modules folder contains classes inherited from cSimpleModule. The Generator module simply generates TaskMessages every simulation time unit and forwards them to the Scheduler.
The Scheduler on initialization, boots up the Worker modules. It also stores all the incoming TaskMessages in a queue. Whenever Scheduler gets a new TaskMessage, it checks if it can send any task stored in the queue to any of the Worker nodes.
The Worker Module simply receives a task, calculates how long it would take it to complete it, and sends the Scheduler notification after this time has passed.

1.4 Objects

The Objects folder contains Node and Task classes. They simply act as a wrapper around MIPS, Storage and RAM properties. The Node class also contains pointer to it's cModule.

2 Analysis of Scheduling Algorithms

Efficiency of algorithms is measured by averaging the time taken by all processes since their initialization to their completion.

2.1 FCFS

The basic algorithm that was implemented to schedule tasks was a naive first come first serve technique. Tasks are stored in the queue and sent to free and eligible workers. The workers are sorted in the same order as the file in which they were listed. I am calling this algorithm Naive FCFS.

2.2 RSW

The first optimization to this technique was to sort the worker nodes in descending order of their MIPS. Now the tasks are assigned to the free worker with the highest MIPS. This resulted in considerable decrease in the average completion time of processes. This algorithm is referred to as Reverse Sorted Workers(RSW).

2.3 SJB

The second optimization was to sort the pending tasks in the increasing order of their MIPS as well. Now, the task with the lowest MIPS is sent to the worker with the highest MIPS. This is effectively Shortest Job First(SJF). This didn't seem to have any effect on the average. I assume this is because the number of tasks are very less to exhaustively analyze the algorithm.

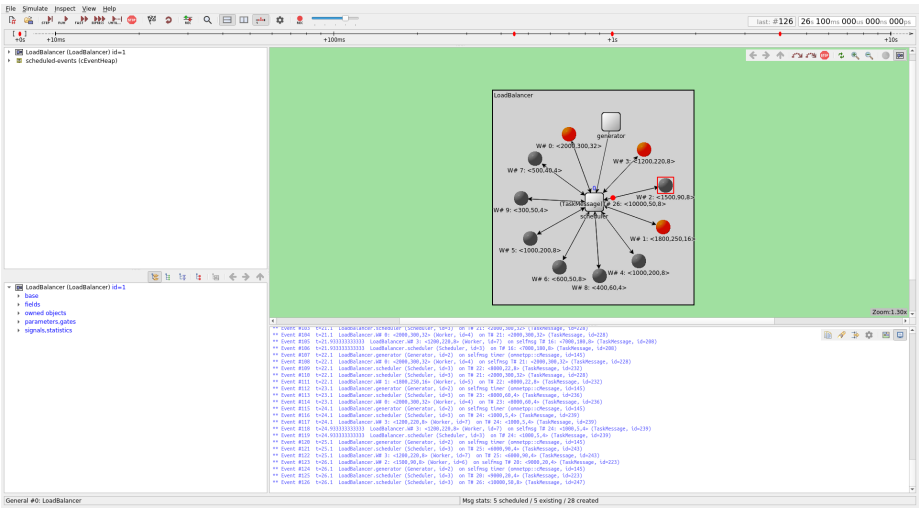
2.4 LJB

To find how prioritizing the longest job first would effect the average time, the pending tasks were reverse sorted. Surprisingly, this didn't have any effect on the average either. Again, I think this is because of the small number of tasks.

2.5 Comparison

Algorithm	Average Completion Time
Naive FCFS	3.98086
RSW	2.91235
SJF	2.91235
LJF	2.91235

3 Screenshot



Red Nodes represent busy workers.

3.1 GitHub

https://github.com/imAliAzhar/Load-Balancing-Simulation