Project Proposal

Farm Generator

Ali Tabatabaee

1 Brief Description

A load-balanced farm consists of several processes that work as servers which solve their given sub-problems (tasks) and also a single central process which generates new tasks, schedule the tasks over multiple servers in a way to balance the workload among the processes and keep them busy as much as possible, and handles the results outputted by the servers. The target of the farm is to minimize the total processing time for the whole problem.

The objective of this project is to write a program (in Python) which gets as input three functions *generate_task*(), *process_task*(), and *process_result*() written in C and creates a C source code for a farm consisting of N processes using MPI libraries. One of the processes is the central process which calls *generate_task*() and chooses a server to process the generated task while managing the workload of the processes. $N-2$ processes act as servers and call *process_task*() to process the tasks given in their buffers. Finally, another process works as a receiver which gets the results from servers and calls *process_result*() to further process them. The possibility of merging the central and receiving processes into a single process will be investigated.

2 Resources

In addition to the research papers that will potentially be used for implementing some real-world distributed algorithms in order to test the performance of the farm, three source codes below will be used to achieve more insight on important details and configurations in farms. Parts of these source codes might be further used in the implementations.

- farm (dynamically fed farm code written by the course instructor, Dr. Alan Wagner)
- adlb
- mrlib

3 Milestones

1. The first step would be developing a simple farm using MPI in C to get more insight on how farms work
2. The next step is to write a python code to generate the code of such simple farm based on some simple configuration parameters
3. The third step is to add more complexity to the farm to make the scheduling algorithms and communications more efficient
4. The project could be further extended by recognizing more configuration parameters and therefore giving the user more control on how the farm works
5. Developing more complicated real-world examples would also be helpful in order to test the performance of the farm and improve the scheduling algorithms used by the central process and the communication between the processes