CS681 Simulation Project Part-1

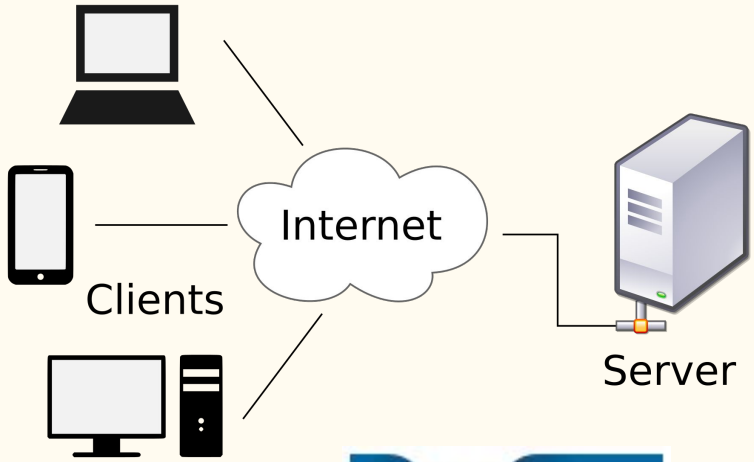
—

130070009 - Anand Dhoot
13D100032 - Anchit Gupta

The Model

We consider the model of Volunteer Computing, specifically describes BOINC - Berkeley Open Infrastructure for Network Computing.

In such a system, the server is a repository of jobs. The client requests jobs from the server and is allotted a job, decided by some algorithm at the server. The client itself performs scheduling, to choose which jobs to run.



Through this project, we plan to implement various strategies for the server and client side scheduling using a Discrete Event Simulator and evaluate them based on various metrics to present a comparative study of the same.

Implementation

We have made C++ based Discrete Event Simulator. We used OOP (inheritance, virtual functions, etc..) to make the code efficient and easily extensible.

In the main loop of our program,

- We handle the earliest event in the event list
- This in turn results in one or more subsequent events being pushed into the event list
- The event handlers in turn use the scheduling algorithms of the clients to schedule the next job

Code Design

Main Classes

- Events
- Clients
- Jobs

Data Structures Used

For the Discrete Event Simulator itself, we stored a list of events, such that the most recent event is at the head of the queue.

We have used lists and arrays to store the individual client job buffer, client request buffer, server job buffer, etc.

Key Events (& Handlers)

- Arrival of the job at the server - Serviced by adding the job to the job buffer at the server.
- Server Interrupt - Interrupt at the server to allocate some job from the buffer to a available client. This client is chosen by a suitable scheduling algorithm.
- Client Interrupt - This event happens at every Client CPU. This facilitates the job scheduling at individual CPUs, removing completed jobs from the client job buffer and if client job buffer not full, requests the server for more jobs.

Current Status

All the basic code skeleton is ready. Specifically, after extensive debugging, we have implemented

- Two types of Client Schedulers
- One server scheduler
- The main event handler loop along with all of its required machinery
- Trace printing
- Rudimentary statistics collection

TODO

- Implement the remaining client, server schedulers
- Run tests, varying parameters and noting down the metrics
- Analyse the metrics and interpret them
- Draw conclusions and explain using theoretical knowledge
- Make report, graphs etc.