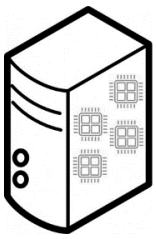
Scheduling in Distributed Systems

This project aims to develop a job scheduler for distributed systems, e.g., compute clusters and (cloud) data centres. The project shall be carried out in groups of 3 (2 in exceptional cases with the unit convenor's approval).

Commences: Week 1.
Group formation due: Week 3.
Value: 40% (Stage1: 20% and Stage 2: 20%)
Programming language to be used: Java
System to be used: Ubuntu Linux
Stage 1 due: **5pm** Monday, Week 7 (Group).
Stage 2 due: **5pm** Monday, Week 13 (Individual).
**** Project, the code in particular, must be managed through a git repository (GitHub/Bitbucket).***

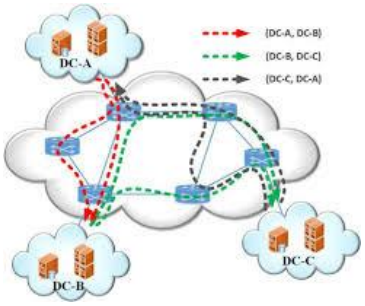
Overview

Distributed (computing) systems come in various sizes and scale (see the figure below). They range from a single workstation computer with several processors, a cluster of compute nodes (aka servers) to a federation of geographically distributed data centres with millions of servers (e.g., Google data centres, AWS clouds and etc.). The main component of distributed systems is servers. These servers are networked together and often form a single system, e.g., a compute cluster or data centre. Roughly, when these servers are virtualised, the system is called a 'cloud' data centre or simply a cloud. Note that servers of a particular distributed system cannot be assumed to be identical/homogeneous; they are often heterogeneous in terms of processor architecture and resource capacity.


(a) Single machine with multiple CPUs


(b) a rack of servers


(c) a data centre



(d) a federated cloud

A distributed system, in general, deals with a diverse set of jobs of multiple users. In other words, servers in a distributed system are shared among many jobs, such as `ls` and `gcc`, user applications like our simulator, MapReduce data-processing jobs and long-running web services. These jobs have different resource requirements in terms of the number of CPU cores, memory and disk space.

Scheduling is the key technique for ensuring the efficient use of computer systems including distributed systems. In this project, you will be *eventually* developing a new job scheduler in a simulated distributed system. The simulation is enabled by a discrete event simulator (ds-sim, https://github.com/distsys-MQ/ds-sim). In COMP3100/COMP6105, the simulation adopts the client-server model. The server-side simulator has already been developed and provided to you. It oversees the simulation. In particular, it simulates a distributed system with user specified

configurations (e.g., `ds-config01.xml`) and pairs up with the client-side simulator containing one or more scheduling policies/algorithms. The IP address and default port number to be used to connect the server-side simulator are 127.0.0.1 and 50000, respectively.

Stage 1 (due **5pm** Monday, Week 7)
Your ***group*** is required to design and implement a 'vanilla' version of client-side simulator (your ds-client) that includes basic scheduling functionalities and a simple job *dispatcher*.
** Individual contributions will be assessed based primarily on commit history in your group's git repository.*

Stage 2 (due **5pm** Monday, Week 13)
Your task (**individual**) in this stage is to design and Implement one or more new scheduling algorithms that shall be at least comparable to three baseline algorithms, First-Fit (FF), Best-Fit (BF) and Worst-Fit (WF).

** The working of your ds-client in each stage will be assessed in a demo session during a designated workshop, in addition to the formal and more detailed assessment on your submission.*
** Detailed instructions of each of these stages including marking rubric will be available in a separate document.*