READSD and STQA Combined Assignment

Problem Scenario

Your university is in the process of acquiring a new supercomputer to support research activities. The IT department has to design and implement a software solution for batch job submission, control and accounting. To do this, it needs a simulation tool to model the behaviour of the computing platform and to explore alternative accounting strategies.

The assignment is to design, implement and test a simulator of the new computing/job control system.

User Requirements

The simulated computing system will be heterogeneous, containing:

- A set of "traditional" computing nodes, i.e. shared-memory multicore CPUs;
- A set of "accelerated" compute nodes, same as above but with 1 or 2 attached GPUs dedicated to computations;
- A set specialized nodes, with GPUs dedicated to rendering and visualization of complex data;
- A set of storage devices.

There will be at least 128 nodes with at least 16 processor cores per node. The simulated users of the system can be classified as:

- IT support;
- Researchers;
- Students;

Researchers are divided into groups, where each group has an allocation of resources, but individual researchers may have grants entitling them to additional resource usage.

Students are grouped by the curriculum they are enrolled in, and all students have a cap on the maximum usable resources, both cumulative and instantaneous, which may depend on their group.

The batch system shall match job requests to resources; there will be different classes of jobs for different uses, e.g. short, medium, long running, interactive, etc.

The job queues are maintained in an order established by an auxiliary algorithm called a scheduler; the interface to the scheduler should be designed so that it would be easy to test different algorithms. Indeed, the overall performance of the computing system in actual operation strongly depends on the scheduler, therefore the evaluation of a given scheduler is one of the most important features of the simulation. The proper design and implementation of an efficient scheduler is a task beyond the scope of the current course; hence, for the purposes of this assignment, it is sufficient to implement and interface a simple first-come first-served scheduler, provided the interfacing mechanism is designed to allow for later experimentation with other schedulers.

Note that the system should be used "optimally", but different stakeholders may have different optimality criteria; for instance, users typically want to minimize waiting time, whereas administrators strive to maximize system utilization.

The policy constraints are:

- There are at least four different job queues:
    1. Short, interactive jobs that can take up to 2 nodes for no more than 1 hour; a certain subset (say 10%) of the machine must be reserved for this queue;
    2. Medium-sized jobs that can take up to 10% of the total number of cores, and can last up to 8 hours; another subset (say 30%) of the machine must be reserved for this queue
    3. Large jobs, that can take up to 16 hours and up to 50% of the total core count;
    4. Huge, active only from 0500pm Friday to 0900am Monday, where the jobs can potentially reserve the whole machine. During these times the other job queues do not serve requests.
- Each job queue has associated a cost for number of machine-hours requested.
- Each job will request a certain number of processor cores for a certain amount of time. For the sake of simplicity you can assume that, at any given time, the queueing software will grant access under a first-come first-served basis, up to the available capacity of the machine.
- At the end of the week, there will be a cutoff time such that no new jobs will start if their estimated completion time will go beyond the end of the work week (thereby leaving the machine free for the weekend queue).

The machine can be assumed to have a constant operational cost per hour.

In the simulation program, users will be modeled as producers of requests, with each user having a certain budget; each user will make requests up to his available budget.

Each run of the simulator shall define a specific scenario comprising the set of simulated computer users, i.e. their number, distribution in classes and budgets; there can be mixes of classes of small, medium, and big users. These parameters are chosen by the simulator users, i.e. the IT department staff that are investigating the policies for the new supercomputer.

During the course of the simulation, each simulated user will produce requests; the time between any two successive job requests and the size of the request (within the user class) shall be modeled by an exponential probability distribution, with parameters dependent on the class of user.

The output from the simulation should include

- The number of jobs processed in each queue (throughput) per week;
- The actual number of machine-hours consumed by user jobs;
- The resulting price paid by the users;
- The average wait time in each queue;
- The average turnaround time ratio, i.e. the time from placing the job request to completion of the job divided by the actual runtime of the job;
- The economic balance of the centre, calculated by subtracting from the actual price the operating costs.

Assignment Details

1. Produce a set of use case diagrams detailing the operation of the simulation system.
2. Choose appropriate models to represent the requirements (functional and non-functional); it is suggested to create a structured model, a behavioural model and a data flow model to capture the requirements of the simulation system.
3. Develop a test plan for the software. Explain what needs to be tested and how you will implement the tests. Ensure you have adequately considered both lower level unit and integration testing, and explain your choice of test inputs.
4. Implement the software and run the testing plan, showing output from your application tests. You can use a combination of unit tests and any further validation testing you think appropriate.
5. Determine the test coverage for each of your components that your test plan has achieved. Specify the percentage coverage of statement, decision

and path coverage for each component, and thereby make a statement of the coverage across your whole application.

6. Identify the quality metrics you have used in your system and demonstrate how they have influenced your design. You are free to concentrate on any quality measures that you think are suitable.
7. Explore further operations for the system

Assignment Deliverables

Ensure that all source code can be compiled and executed, by including the relevant make/project files

- Software Requirements Document;
- Test Plan Document;
- Working Source Code;
- Working Test Harnesses;
- Test results document and code coverage;

Marking Scheme

20 Software implementation: structure, internal documentation;

35 Requirements specification document, project plan;

30 Test plan document;

15 Test results document, including code coverage.

Notes

You are free to make any reasonable assumption to supplement any missing or not fully specified requirements, but you have to state explicitly your additional assumptions in the documents. You are not assessed on the monetary value that is produced by the simulation program under these assumptions.

You are encouraged to make use of standard documentation formats to structure your reports; the IEEE standard templates (among the most commonly used) are available in the support material on Blackboard.

Submission deadline: 1:30 pm (UK), January 8th, 2018