Beosim Visualization Agent

Specification

William M. Jones, Dan Stanzione, Louis Pang

Abstract

This document serves to outline the functionality that the visualizer needs in order to convey meaningful information in an easy to use GUI.

1 Introduction

Dan, maybe you can give an intro to Beosim here

2 GUI

The GUI, to be written in JAVA, will contain a paneled layout that will provide the user with a visualization of the data that is generated as output from Beosim. Figure 1 illustrates the basic layout of the visualizer.

Each panel will need to be scrollable, and possibly each sub-panel as well, since the number of nodes per cluster as well as the number of clusters per grid may be more than will fit within each panel. Additionally, the graphics included in the panels will need to be scalable, so that zooming is possible. Each panel will include a legend that will serve to explain what the colors represent, as well as any other pertinent information. Additionally, each panel will include a control sub-panel associated with its parent panel that will allow the user to control such features as zooming and other to be determined.

Panel 1 will depict each cluster in the grid as a grouped collection of nodes (the colored squares). The color of each node will represent the job to which the node is currently allocated. As you can see from the figure, the green job is currently allocated to nodes that reside on both cluster 1 as well as cluster 3.

Panel 2 will show a plot of the node-time graph of job allocations in the grid. Again here the colors correspond to the jobs. Here, the current time index of the simulation should be near the right-hand side of the time axis, and should data should scroll to the left as the time index is incremented.

Panel 3 will provide bar graphs that show queue depths in each cluster as well as waiting times, etc. and others to be determined.

Panel 4 will provide numerical information related to statistics provided by the simulator output file, as well as any statistics that can be derived from the data provided by the simulation data file.

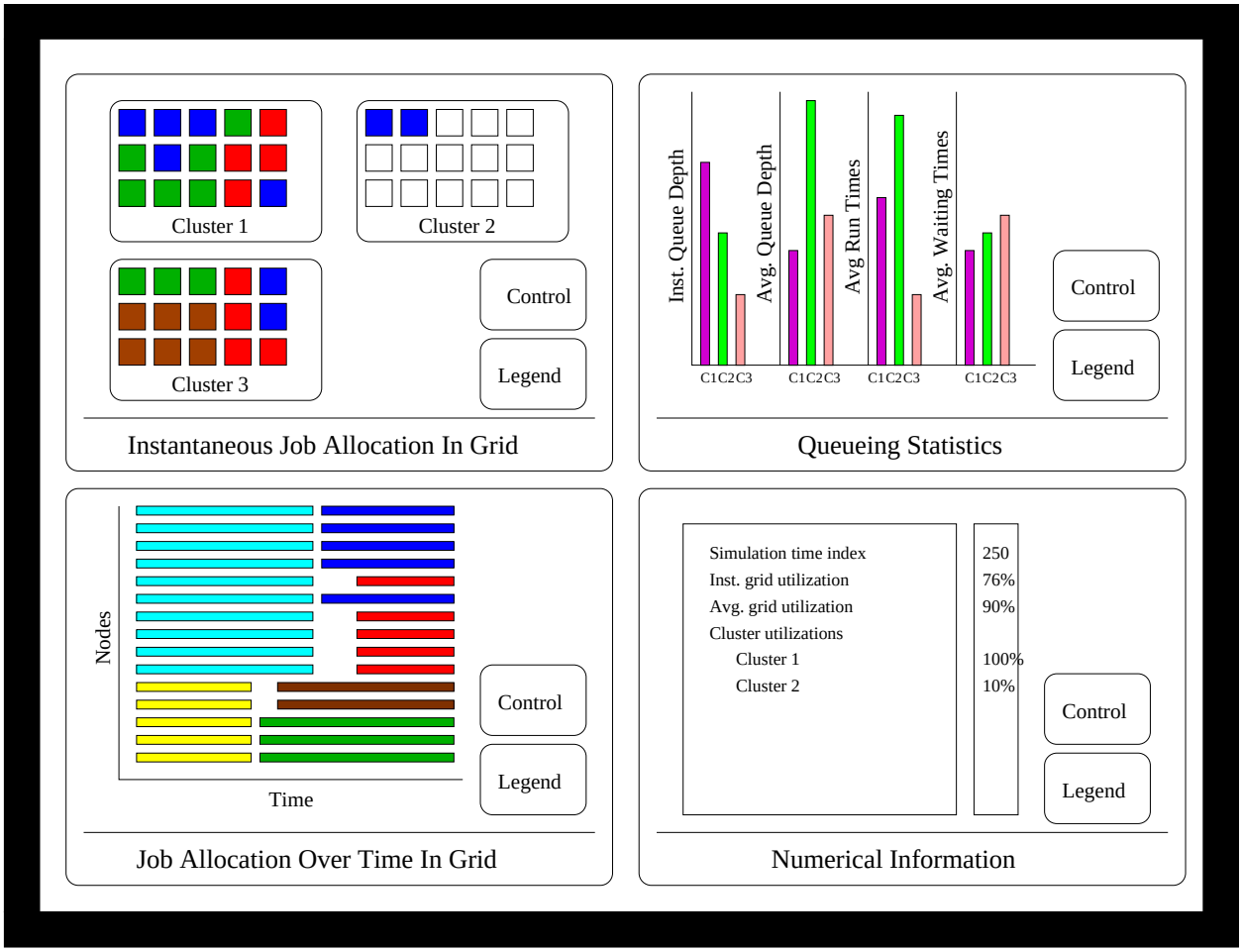

Figure 1: GUI Concept Figure

3 File Format

The information provided in the simulators output file will be as follows:

Dan, Louis, Will: we need to decide on this! One of my questions about this is whether the simulator should provide all the information that the GUI is to display in such a way that the GUI need not keep track of nor calculate any information during its runtime.

I think we initially decided that the simulator will produce a listing of data for each time step of the simulator. Since the simulator is event driven, the simulator does not step through each time increment, but rather only time increments that an event occurs. If we only write out this information, then the visualizer may have to animate between event times so that the the visualization proceeds at a linear rate through time. However, if we are not interested in this (and it may not be desirable), then we can simply print out all the information needed by the visualizer at each event time.

Probably the most important thing to discuss upfront so the student can get stared is what the header information will be, i.e. information that will setup the number of clusters, number of nodes per cluster, etc. That way he can go ahead and get the basic layout working.

The second most important question is what is printed at each time index. The following need to be included:

1. time index
2. number of jobs
3. list of jobs with allocated node ranges
4. queueing information
   (a) inst. and avg. Q depth per queue per cluster
   (b) inst. and avg. grid and cluster utilizations
   (c) avg. wait times and run-times for each job

I am not sure if the simulator currently calculates a running average of the above fields at each event index, but we need to.

3.1 Visualization file format

Here is the file format.

```
Header
#clusters    N1 N2 ... Nm
TimeStamp: numJobsRunning [clusterID,jobID]<NC1,NC2,..,NCm> ... Nq1 Nq2 .. Nqm
...
...
...
Footer
avRunTime avNumInQueue avQueueTime avRunTime avUtilization
avRunTime avNumInQueue avQueueTime avRunTime avUtilization
...
...
```

Here is a sample output:

```
Header
2 2 2
0.000000: 1 [0,1]<2,0>  0  0
0.000000: 2 [0,1]<2,0> [1,1]<0,2>  0  0
22.000000: 2 [0,1]<2,0> [1,1]<0,2>  0  1
54.000000: 2 [0,1]<2,0> [1,1]<0,2>  0  2
85.000000: 2 [1,1]<0,2> [1,2]<2,0>  0  1
152.000000: 2 [1,1]<0,2> [1,2]<2,0>  1  1
153.000000: 2 [1,1]<0,2> [1,2]<2,0>  1  2
279.000000: 2 [1,1]<0,2> [1,2]<2,0>  1  3
333.000000: 2 [1,1]<0,2> [1,3]<2,0>  1  2
412.000000: 2 [1,3]<2,0> [1,4]<0,2>  1  1
443.000000: 2 [1,4]<0,2> [1,5]<2,0>  1  0
456.000000: 2 [1,4]<0,2> [1,5]<2,0>  2  0
470.000000: 2 [1,4]<0,2> [1,5]<2,0>  3  0
521.000000: 2 [1,4]<0,2> [1,5]<2,0>  4  0
583.000000: 2 [1,5]<2,0> [0,4]<0,1>  3  0
601.000000: 2 [0,4]<0,1> [0,2]<2,1>  2  0
611.000000: 1 [0,2]<2,1>  2  0
723.000000: 1 [0,3]<2,1>  1  0
770.000000: 1 [0,5]<2,1>  0  0
789.000000: 0  0  0
Footer
60.2 1.36629 215.6 60.2 1
219.8 0.969582 153 219.8 0.886081
```