The College of Wiliam & Mary

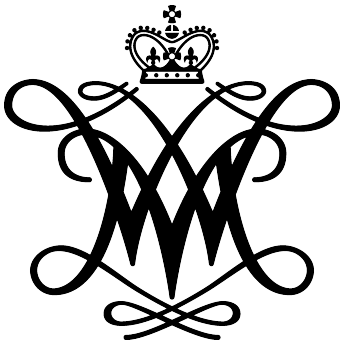
A simulation to determine the optimal queuing configurations and resource availability in an HPC cluster

High Performance Computing Cluster Resource Optimization

Author:
Nadia Aly,

Course, Professor:
CSCI 698, Leemis

December 5, 2017



1 Problem Statement

The College of William and Mary owns and operates a High-Performance Computing (HPC) cluster. The HPC cluster consists of ten sub-clusters. HPC computing provides a user with the resources necessary to run a computationally or memory intensive job that a local machine may not be able to process. We will be looking at four of the subclusters listed in Table 1 (Typhoon, Hurricane, Whirlwind, Vortex), that are open to the general student body and faculty. The current dataset contains observations from January 01, 2016 to November 28, 2016. The user will enter a virtual request with information on what resources the job requires. This includes: node type, number of nodes, number of processors per node, and an upper bound on execution time. This is formatted as:

```
qsub -l nodes={# of nodes}:{node type}:ppn={# of processors per node} -l walltime={HH:MM:SS}
scriptname
```

The goal of this simulation is to evaluate the current user and root usage, node configuration, and observe the impact of using various queuing policies on throughput and average wait times across all node types. Currently, the HPC system administration use TORQUE, a resource manager and MAUI, a job scheduling system. The default for TORQUE is a first-in-first-out (FIFO) queue configuration. Maui has a more advanced queuing system that is able to improve throughput and utilization by filling holes with shorter jobs or jobs that need less nodes using a bin packing algorithm. Maui also looks at a user's history to attempt to provide a fair share of the HPC resources to students and faculty. Maui determines priority by evaluating: job duration, current time on queue, resource usage in the past sixty days, and priority of user (i.e. research groups may be given a higher priority). Finally, Maui will reserve several nodes for jobs that take less than an hour.

Table 1: Summary of SciClone Total Resources

SubCluster	Type	Names	Available Nodes	Reserved Nodes	Cores Per Node	Total Cores	Memory
Typhoon	c9	ty01-ty60	60	0	4	240	8 GB
Typhoon	c9a	ty61-ty72	12	0	4	48	24 GB
Hurricane	c10	hu01-hu08	6	2	8	64	48 GB
Hurricane	c10a	hu09-hu12	4	0	8	32	48 GB
Whirlwind	c11	wh01-wh44	42	2	8	352	64 GB
Whirlwind	c11a	wh45-wh52	8	0	8	64	192 GB
Vortex	c18a	vx01-vx28	26	2	12	336	32 GB
Vortex	c18b	vx29-vx36	8	0	12	96	128 GB
Vortex	c18c	va01-va10	10	0	16	160	32 GB

2 Model Description

Every time a user or root submits or completes a job, a record is created detailing the request including: time entering system, initial wall time request (job time estimate), time job starts, time of job execution completion, resources requested, and resources used. An example record:

```
02/25/2016 01:19:40;E;1565231.ty00.sciclone.wm.edu;user=root group=root jobname=nodekpr-2 queue=sys
ctime=1456381030 qtime=1456381030 etime=1456381075 start=1456381076 owner=root@ty00.sciclone.wm.edu
exec_host=ty42/3+ty42/2+ty42/1+ty42/0 Resource_List.neednodes=ty42:ppn=4 Resource_List.nodect=1
Resource_List.nodes=ty42:ppn=4 Resource_List.walltime=01:00:00 session=4769 end=1456381180
Exit_status=0 resources_used.cput=00:05:58 resources_used.mem=2347972kb
resources_used.vmem=26370628kb resources_used.walltime=00:01:44
```

2.1 Model Assumptions

1. Analyzing system from time $t = 0 - 31,536,000$ seconds with a $t = 604800$ second warm-up period. The total system time is then $30,931,200$ seconds. This is equivalent to a year, with a week long warm-up period.

2. The initial conditions for the simulation are no scheduled node requests with all resources free.

3. To model the user job length time we use a trace-driven approach. We take 100 evenly spaced quantiles from 0.01...1.0, then use a discrete distribution with the associated quantile values. In SIMAN this is modeled as: DISC(0.01, 0, ...0.50,32, ...1.0, 359639.00).

4. To model user interarrival time, use a trace-driven approach. The users enter requests into the system with an interarrival time from a discrete distribution, in SIMAN modeled as: DISC(0.56, 0 , 0.69, 1, 0.70, 3, 0.71, 8, 0.72, 17, 0.73, 29,0.74, 44, 0.75, 58, 0.84, 59, 0.90, 60, 0.91, 61, 0.92, 85, 0.93, 120, 0.94, 134, 0.95, 206, 0.96, 317, 0.97, 512, 0.98, 946, 0.99, 2268, 1.0, 1731605)

5. If a user enters a request into the system and the total job time is less than one hour, then they will proceed to a reserved node queue based off the type of node request. Two nodes from Hurricane, Whirlwind, and Vortex subclusters are set aside for these users.

6. A root job is submitted for each node, every 24 hours. This root job is given the highest priority for a node, however, it is not able to seize a node that is currently being used. If the root job has been in the system for more than 24 hours, it is destroyed regardless of position on queue.

7. Each job is scheduled first-in-first out regardless of job time or nodes requested. However, the root jobs have a higher priority than user jobs.

8. There are thirteen possible requests a user can submit for a job which have different associated compatible node type, details on compatibility are in Table 2.

9. If a user submits a job that has multiple compatibility options, then it will seize the first available compatible nodes. For example c18x can run on vortex nodes va01:va10 or vortex vx01:vx36.

10. For a job to start all requested nodes and cores must be available.

11. To model the root job length time we use a trace-driven approach. We take 100 evenly spaced quantiles from 0.01...1.0,then use a discrete distribution with the associated quantile values. In SIMAN this is modeled as: DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29, 10, 0.3, 14, 0.31, 15, 0.32, 16,0.34, 17, 0.35, 19, 0.36, 21, 0.37, 22, 0.38, 23, 0.39, 26, 0.4, 28, 0.41, 32, 0.42, 37, 0.43, 42, 0.45, 43, 0.5,44, 0.52, 45, 0.53, 46, 0.54, 48, 0.55, 50, 0.56, 51, 0.57, 52, 0.58, 52, 0.6, 53, 0.62, 54, 0.63, 55, 0.64, 56,0.65, 57, 0.67, 58, 0.68, 59, 0.69, 60, 0.7, 61, 0.71, 62,0.72, 65, 0.73, 68, 0.74, 72, 0.75, 78, 0.76, 88,0.77,96,0.78, 97, 0.79, 100, 0.8, 102, 0.82, 103, 0.87, 104, 0.89, 105, 0.9, 106, 0.91, 108, 0.92, 109, 0.95, 110, 0.96, 111, 0.97, 117, 0.98, 126, 0.99, 149, 1, 7688):

2.2 Modelings Decisions

- Entity: User job request, Root job request
- Time Scale: seconds
- Attributes: Job type (User, 1 or Root, 2), Time in the System, length of job (seconds), number of cores job will require, unique job ID number, and type of node request.
- Measures of Performance: average wait time by node type, and node resource utilization.

Table 2: User Node Type Requests and Corresponding Compatibility

Node Request	Compatible Nodes	f(x)	F(x)	NodeType
c10, hurricane	h_01:h_12	0.023	0.023	1
c10a	h_09:h_12	0.006	0.030	2
c11	wh01_wh44	0.002	0.031	3
c11a	wh45_wh52	0.001	0.032	4
c11x	wh01:wh52	0.009	0.041	5
c18, c18x	va_01:va_10, vx_01:vx_36	0.042	0.083	6
c18a	vx_01:vx_28	0.018	0.101	7
c18b	vx_29:vx_36	0.017	0.118	8
vortex	vx_01:vx_36	0.018	0.136	9
c18c	va_01:va_10	0.412	0.548	10
c9, typhoon	ty_01:ty_72	0.227	0.775	11
c9a	ty_61:ty72	0.025	0.801	12
xeon, x5672	wh01:wh52, hu1:hu12	0.199	1.000	13

2.3 Trace-Driven Distribution Approximation

There following processes in the dataset needed to be fit to a distribution for the simulation:

1. User and Root Job Length (seconds)
2. User Interarrival Times
3. Request Node Type (13 possible requests total). The distribution is detailed in Table 2 and Table 3.
4. The number of cores needed for a request for each node type. The types of requests are: c10, c10a, c11, c11a, c11x, c18x, c18a, c18b, vortex, c18c, c9/typhoon, c9a, xeon/x5672. The final distributions are detailed in Table 3.

In Figure 1 we see a histogram of the user job lengths. There are spikes at several values. This can be from an individual user submitting multiple similar jobs. We looked at distributions including: exponential, beta, gamma, normal, uniform, and triangle, but the error was very large and these distributions did not capture the spikes or generate values with similar properties of the dataset. Therefore we took a trace-driven approach to model the distribution. The user job length data was taken and 100 quantiles in .01 intervals were generated from $0.00, 0.01, ...0.99, 1.0$. These values were then modeled with a discrete distibution in SIMAN. Figure 2 shows the probability distribution of the user job length (seconds) with the job length on the x-axis, and the corresponding density on the y-axis. The corresponding PDF to the approximated distribution is shown on the left. As a result of this technique and interval decision, the small spikes at various values of user job length are captured.

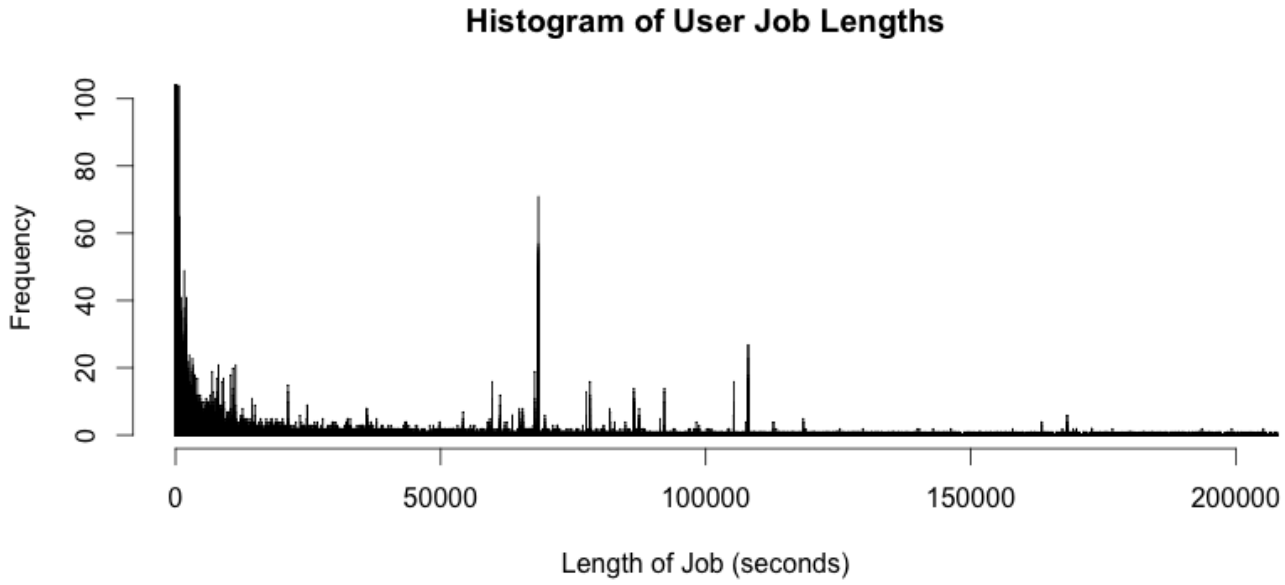
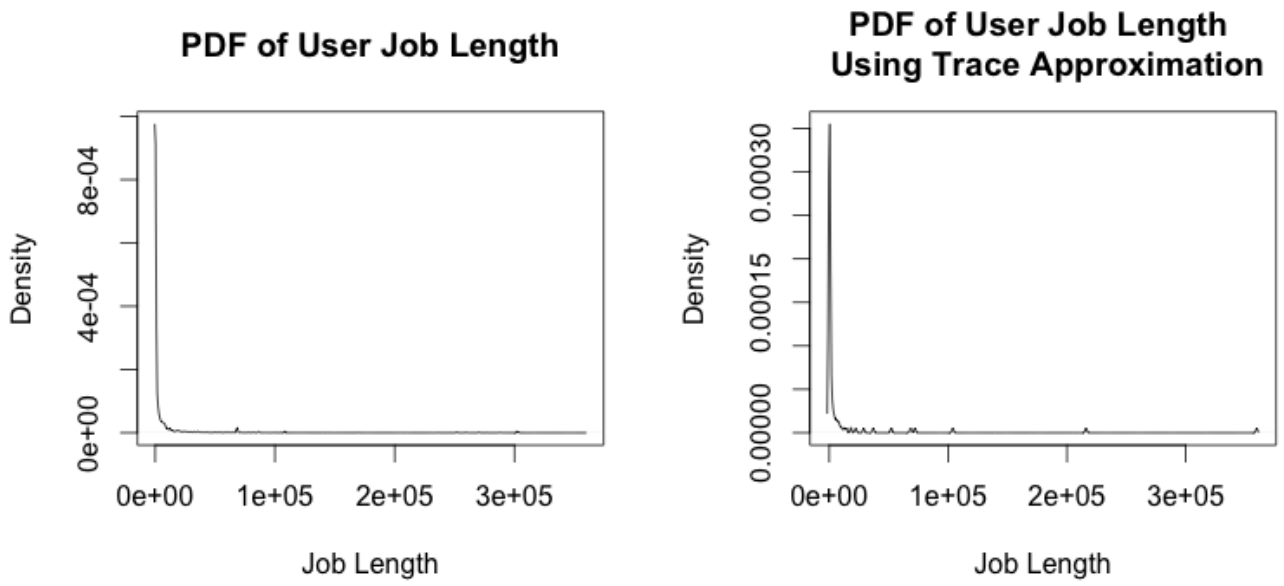


Figure 1: Histogram of non-root, user job Lengths (seconds) scale displayed with x-axis maximum of $200,000$ seconds

Figure 2: PDF of user job length (left), PDF of user job length using trace approximation (right)



A histogram of root user job lengths is displayed in Figure 3. There are a few cluster increases in frequency around: 0, 30, 45, 60, and 120 (seconds). This is most likely a consequence of changes in the cluster management. The root jobs clean up or erase disk space and ensure nodes have updated and working configurations. If the configurations change, then the process can take a different amount of time for each node. The same trace-driven, quantile technique used from the user job lengths is used again. However, in Figure 4 it is clear that all peaks were not able to be captured as well and consequently, the technique has a smoothing effect. We experimented with a smaller quantile interval of 0.005, however, the countour remained smooth. The details of the distribution of the root user job length is documented in the model source file in the Appendix.

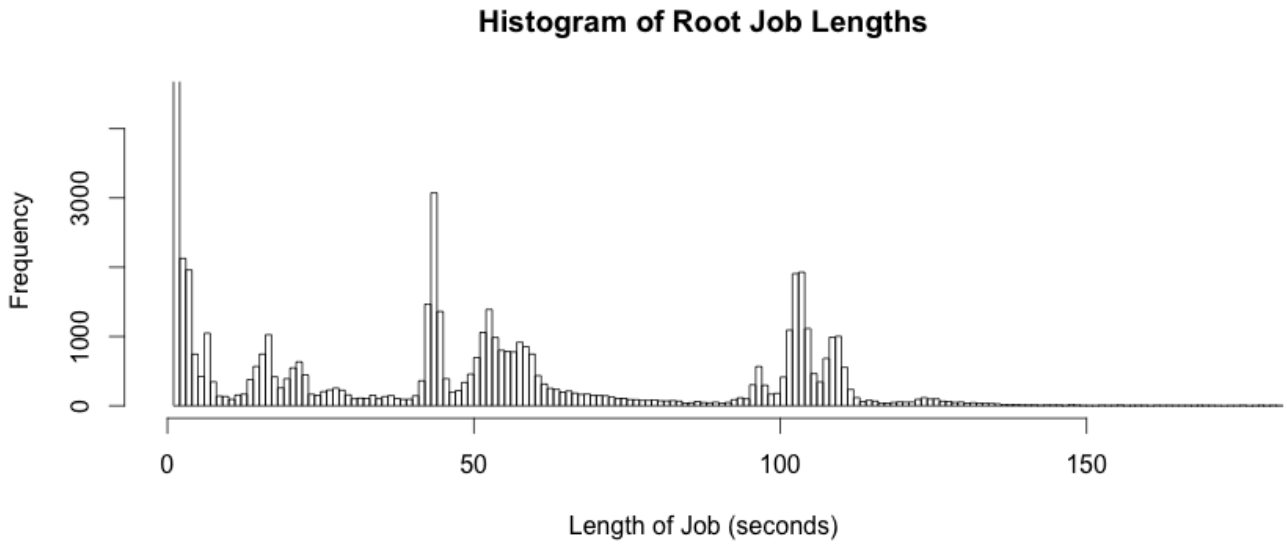
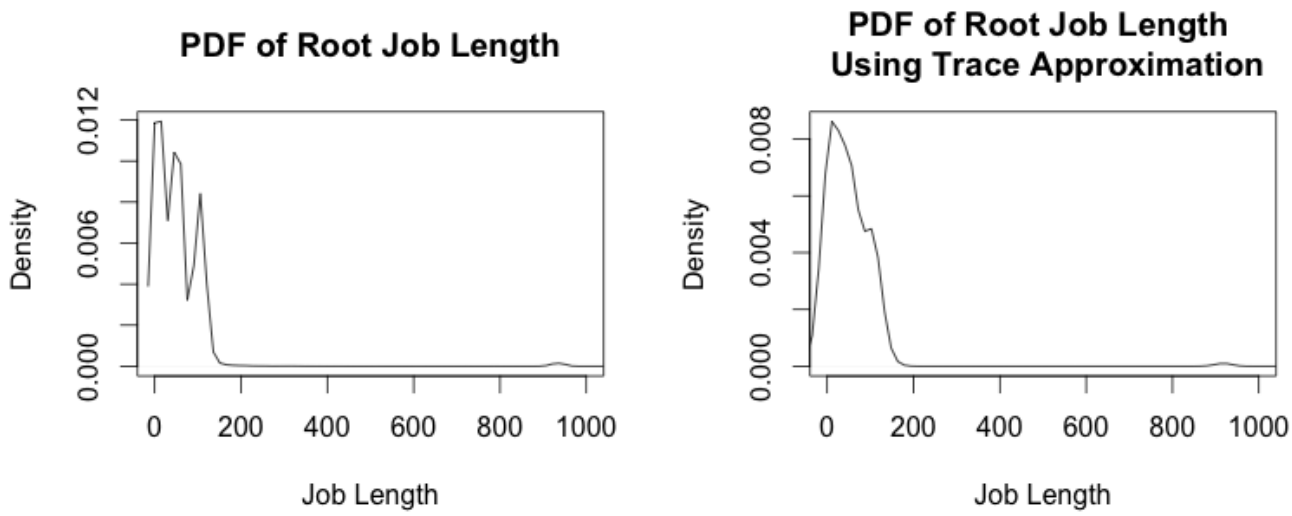


Figure 3: Histogram of Root Job Lengths (seconds) scale displayed with x-axis maximum of 150 seconds

Figure 4: PDF of root job length (left), PDF of root job length using trace approximation (right)



The user interarrival times needed to be determined to model the rate of users entering a request into the cluster. A histogram of the user arrival times from the dataset is included in Figure 5. Again, there are spikes and the distribution did not fit well to any known distribution. The same trace-driven, quantile technique used from the user job lengths is used again. The resulting PDF from this approximation technique appears to fit well and can be seen in Figure 6.

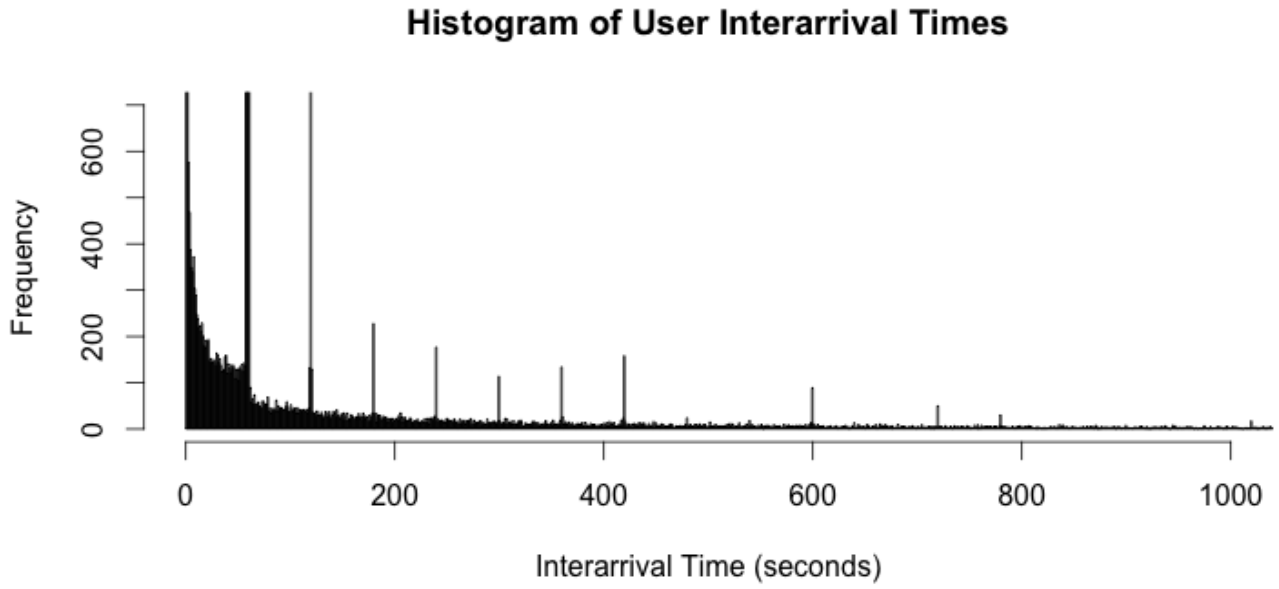


Figure 5: Histogram of user job interarrival times, scale displayed with x-axis maximum of 1000 seconds

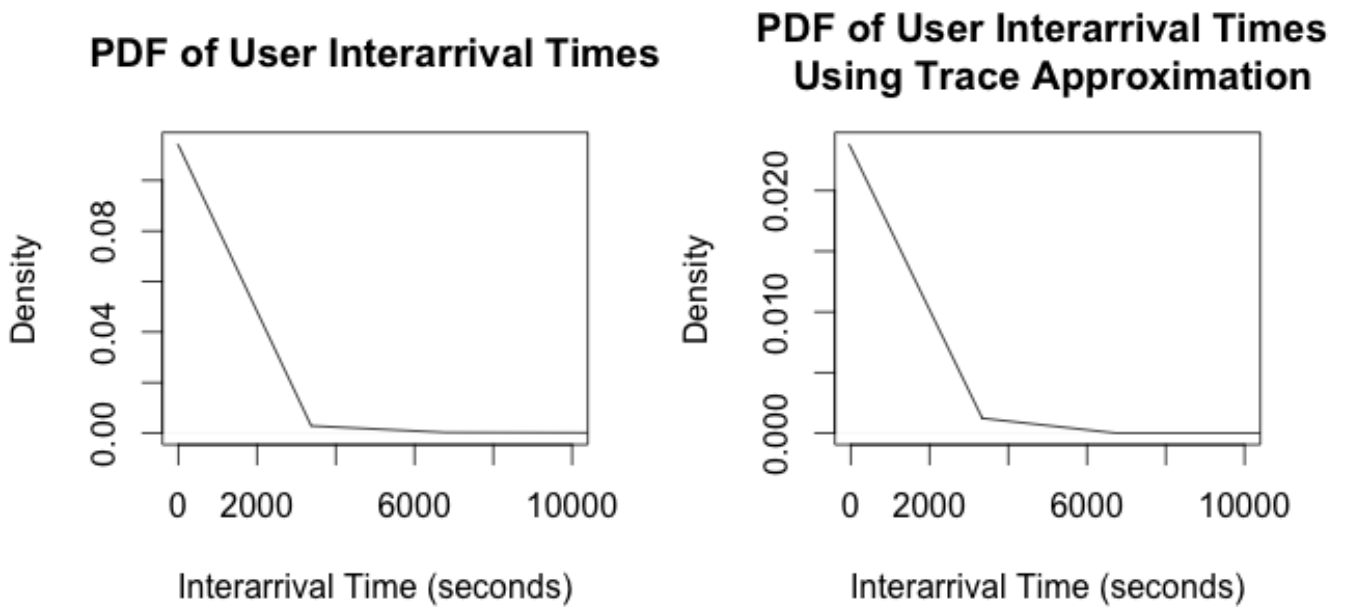


Figure 6: PDF of user job interarrival time (left), PDF of user job interarrival time using trace approximation (right)

Table 3: Distributions of Node Type Requests Determined Through Trace-Driven Approximation Technique

Request	Distribution
User Arrival Time	DISC(0.56,0, 0.69, 1, 0.70,3,0.71,8, 0.72,17, 0.73, 29,0.74, 44,0.75, 58,0.84,59,0.90, 60,0.91, 61,0.92,85, 0.93,120, 0.94, 134,0.95,206, 0.96,317, 0.97, 512,0.98, 946, 0.99, 2268, 1.0, 1731605)
Node Type	DISC(0.0234,1,0.0296,2,0.0312,3,0.0317,4,0.041,5,0.083,6,0.101,7,0.118,8,0.136,9,0.548,10 ,0.775,11,0.801, 12,1.0,13)
c10, hurricane	DISC(0.0612,1, 0.6826, 2, 0.6873, 4, 0.9246, 8, 0.9286, 12, 0.9797, 16, 0.9876, 24, 0.9979,32,0.9989, 48, 1.0, 64)
c10a	DISC(0.005, 1, 0.675, 2, 0.942, 4, 1.0, 8)
c11	DISC(0.388,1,0.460,4,0.848,8,0.894,16,0.962,32,1.0,64)
c11a	DISC(0.378, 1, 0.390,4,0.683,8,0.805,16,1,24)
c11x	DISC(0.251,1,0.302,2,0.401,6,0.933,8,0.961,12,0.973,16,0.975,18,0.999,32,1,64)
c18, c18x	DISC(0.314,1, 0.343,2,0.355,4,0.394,6,0.406,8,0.408,10,0.598,12,0.600,14,0.615,16,0.617, 18, 0.663,24, 0.674, 32, 0.685, 36, 0.75, 48, 0.765, 60, 0.813, 72, 0.818, 84, 0.919, 96, 0.923, 108,0.956, 120, 0.968, 144, 0.969, 160, 0.971, 168, 0.971, 180, 0.980, 192, 0.984, 216,0.999, 304, 1.0, 360)
c18a	DISC(0.041, 1, 0.767,2, 0.773, 4, 0.778, 5, 0.782, 6, 0.786, 8, 0.796, 12, 0.887, 16,0.956, 24, 0.959, 32, 0.977, 36, 0.993, 48,0.997, 60, 0.999, 96, 1.0, 144)
c18b	DISC(0.116, 1, 0.887, 2, 0.892, 4, 0.897, 6, 0.903, 8, 0.976, 12, 0.982, 24, 0.991,48, 0.994, 60, 0.999, 84, 1.0, 96):
vortex	DISC(0.0684,1,0.0740,4,0.4092,12,0.466, 24, 0.486, 36, 0.892, 48, 0.901, 60, 0.926, 72, 0.930, 84, 0.953, 96, 0.958, 108, 0.969, 120, 0.973, 144, 0.976, 168, 0.983, 180, 0.990, 192,0.994, 204, 0.995, 264, 0.999, 300, 1.0, 360)
c18c	DISC(0.543, 1, 0.544, 4, 0.55, 12, 0.621, 16, 0.622, 24, 0.897, 32, 0.898, 36, 0.899,40, 0.909, 60, 0.912, 64, 0.913, 72, 0.958, 80, 0.960, 96, 0.999, 144, 1.0, 160)
c9, typhoon	DISC(0.757, 1, 0.830, 2, 0.979, 3, 0.980, 12, 0.985, 16, 0.986, 20, 0.987, 32, 0.99, 40, 0.99, 60, 0.992, 80, 0.993, 100, 0.995, 140,0.997, 160, 0.998, 200, 0.999, 216, 1.0, 240)
c9a	DISC(0.134, 1, 0.983, 2, 0.997, 4, 0.998, 8, 0.999, 16, 1.0, 32)
xeon, x5672	DISC(0.607, 1, 0.609, 4, 0.752, 8, 0.78, 16, 0.798, 24, 0.828, 32, 0.85, 40, 0.901, 48,0.905, 56, 0.939, 64, 0.941, 72, 0.959, 80, 0.985, 96, 0.99, 120, 0.992, 128, 0.994, 144, 0.996, 160,0.998, 184, 0.999, 240, 1.0, 280)

2.4 Systems Diagram

Figure 7 shows the systems diagram for this problem. The user will enter the system (submit request for HPC resource) and proceed to a queue based off of the node type. If the job length time is less than a hour, then the job will proceed to a reserved node queue based off of node type. Once the correct node type and needed cores are available, the entity will seize the nodes for the duration of the job. Once the job is completed, the nodes are released or are available for other jobs, and the user exits the system. On the bottom, root jobs enter the system. A root job request is submitted for each node, every 24 hours. The goal of the root jobs are to clean up disk space and ensure nodes have updated and working configurations. The root job will similarly seize the resource if the node becomes available for the length of the job, then release the nodes, and exit the system. The root job request, regardless of position on queue, is destroyed after 24 hours and a new request is issued.

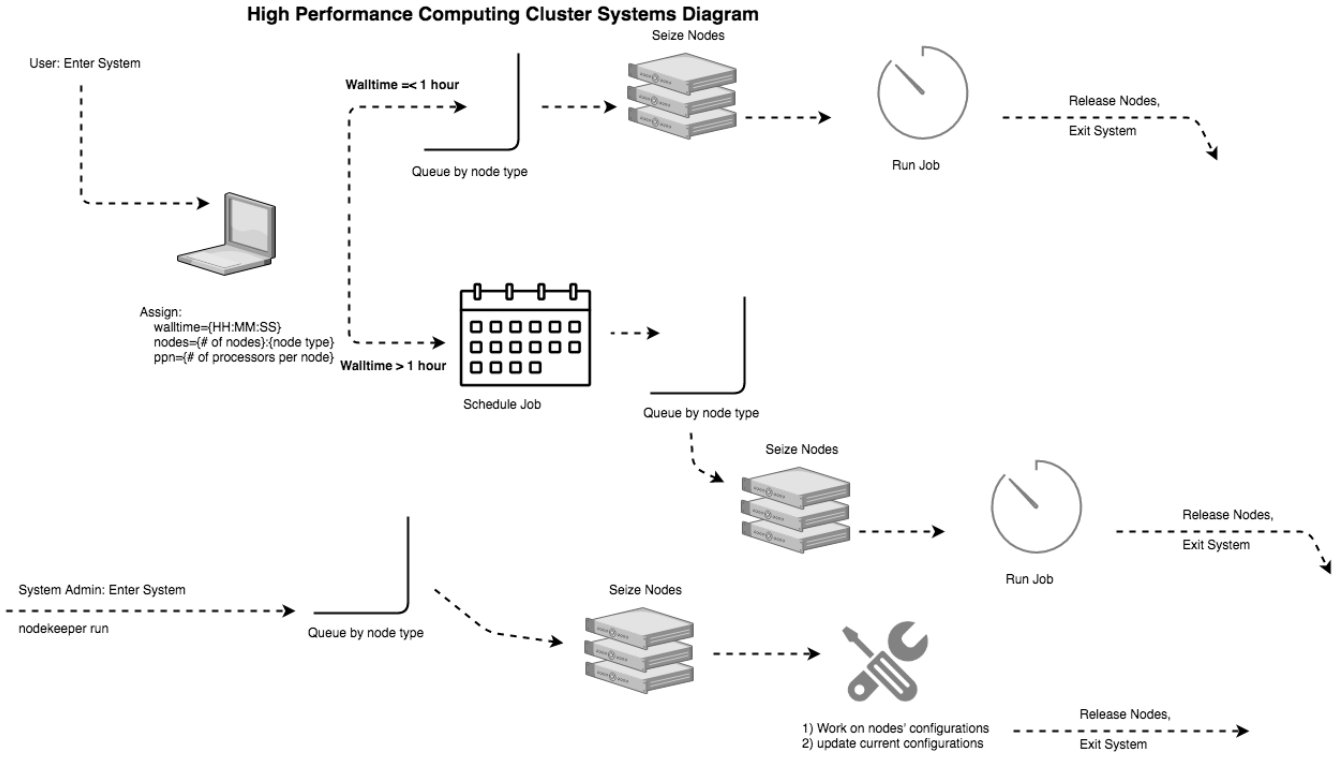


Figure 7: Systems Diagram

2.5 Block Diagram

The block diagram, in Figure 8, shows the modeling decisions, how the entities travel through the system, and how the processes are modeled. The entity will enter the system at the create blocks, two separate create blocks represent user and root users entering the system at any time.

Below the top create block is the interarrival time (time between creations), users arrive according to a discrete distribution, omitted for brevity. After the create block, the entity is assigned four attributes. The first attribute is time in the system = current time, the next attribute is nodeType, the type of request the job will run on, then a unique job ID, later used to remove duplicates from queues, and the length of the job. The type of node is modeled using the frequency of each request type in the dataset, the distribution is described in Table 3. Job Time is also described by a discrete distribution, as listed in Table 3. The entity then arrives at a branch, where it will branch by nodeType, and assign number of cores for the job length based off the node type.

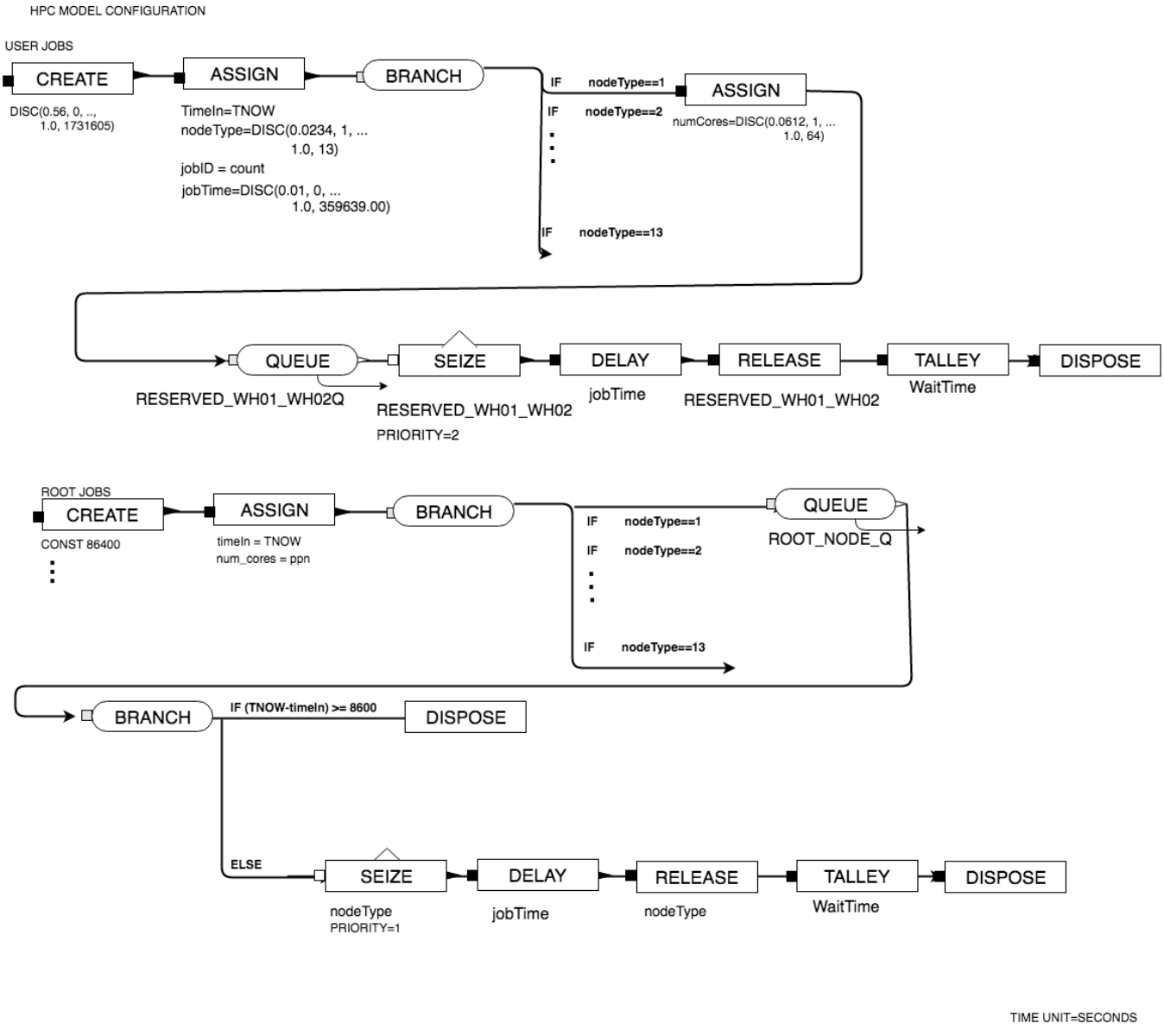


Figure 8: Block Diagram

These individual distributions are described in Table 3. This branch block is also used to clone entities that can run on multiple nodes. Here node types: 1, 5 6,9, 11, and 13 are cloned to multiple queues for different nodes. The entity then arrives at the queue block, where it will wait as a second priority, first-in-first-out or lowest value first priority by attribute job length or number of cores. When the node, and sufficient cores are available, the entity will seize that resource. The entity will then go through a branch block, if it is an entity that has been cloned, then it will go through the queues that it has been cloned in to find the entity in a search block by it's unique job ID. The entity then proceeds to a remove block, where the clone is removed from the queue given the index returned from the search block. The search and remove block are omitted for brevity. The entity then returns to the delay block, and the job time is then modeled with a delay of the jobLength attribute, early defined. The entity then releases the number of cores seized, and moves to the talley block, where the flow time is recorded by node type. The entity then moves to the dispose block where it is disposed and exits the system.
In the bottom create block, the root jobs are created. They are created once every twenty four hours or constant 86400 seconds for each node type. Each node type has a distinct number of separate nodes where the number of cores is defined in the resource block. For each node type, enough entities are created to request every node by the number of cores per node. The entity then proceeds to an assign block where the time in the system, length of job, and the number of cores equal to the processors per node are assigned. The entity then proceeds to a branch block where it will proceed by node type, here they are not cloned as above, since the entity has been created specifically for that node type. The entity then proceeds to the correct root queue block, where it has first priority, once the resource becomes available, it will branch according to time in the system. If time exceeds 24 hours, the entity is disposed, otherwise the entity will proceed to the seize block. In the seize block the entity seizes the correct number of nodes, and the job length is modeled by the a delay block for the length of the job. The length of root jobs are defined by a discrete distribution in Table 3. The entity then releases the number of cores currently seized and proceeds to a tally block where the flow time is recorded by node type. The entity then moves to the dispose block where it is disposed and exits the system.

2.6 SIMAN VARIABLES

The SIMAN VARIABLES used in this model are:

1. check 03 08 ID, check wh03 wh44 ID, check wh45 wh52 ID, check ty01 ty60 ID, check ty61 ty72 ID, check va01 va10 ID, check vx03 vx28 ID, and check vx29 vx36 ID. These are all defaulted to zero and assigned before a cloned entity enters a seize block. They are used to copy the cloned entity's unique job ID, and then used in the search block to search for the cloned entity in any other queue it may appear in.
2. job count, used to make sure cloned entities were correctly being identified and disposed

2.7 SIMAN ATTRIBUTES

The SIMAN ATTRIBUTES used in this model are:

1. The time in the system is modeled with an attribute. We later use this to calculate the average flow time.
2. The job type root or user is modeled with an attribute jobType. User is set equal to type one and root is set to type two.
3. The job time or length of time the job will run for is assigned.
4. The number of cores the job needs is assigned according a unique distribution by request type.
5. The job ID is equal to the previous job ID incremented by one, i.e. the first job is set to one, second two. This is used only after the user create block, to give each entity a unique ID later used to locate cloned entities.

2.8 HOLD LOCATIONS

1. **BufferReserved wh01 wh02**: The entity will wait here on the first available reserved wh01: who02 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞
2. **BufferReserved hu01 hu02**: The entity will wait here on the first available reserved hu01: hu02 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞
3. **BuffeReserved vx01 vx02**: The entity will wait here on the first available reserved vx01: vx02 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞

4. **Buffer ty01 ty60**: The entity will wait here on the first available reserved wh01: who02 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞
5. **Buffer ty61 ty72**: The entity will wait here on the first available ty61 ty72 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞
6. **Buffer hu03 hu08**: The entity will wait here on the first available reserved hu03: hu08 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞
7. **Buffer hu09 hu12**:The entity will wait here on the first available reserved hu09: hu12 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞
8. **Buffer wh03 wh44**: The entity will wait here on the first available reserved wh03: wh44 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞
9. **Buffer wh45 wh52**: The entity will wait here on the first available reserved wh45: who52 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞
10. **Buffer vx03 vx28**: The entity will wait here on the first available reserved vx03: vx28 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞
11. **Buffer vx29 vx36**: The entity will wait here on the first available reserved vx29: vx36 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞
12. **Buffer va01 va10**: The entity will wait here on the first available reserved va01: va10 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞
13. **rootBufferReserved wh01 wh02**: The root entity will wait here on the first available reserved wh01: who02 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞, the root entity has highest priority.
14. **rootBufferReserved hu01 hu02**: The root entity will wait here on the first available reserved wh01: who02 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞, the root entity has highest priority.
15. **rootBufferReserved vx01 vx0**: The root entity will wait here on the first available reserved wh01: who02 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞, the root entity has highest priority.
16. **rootBuffer ty01 ty60**: The root entity will wait here on the first available reserved wh01: who02 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞, the root entity has highest priority.
17. **rootBuffer ty61 ty72**: The root entity will wait here on the first available reserved wh01: who02 type node with sufficient cores, queue type: FIFO, LVF(jobTime), LVF(num cores), capacity: ∞, the root entity has highest priority.

Results

Table 4: Average Wait Time Results from FIFO Queue Configuration

Identifier	Avg Wait Time	95% CI	Avg Min	Avg Max	Avg Job Count
reserved_wh01_wh02	63.187	(15.38, 111.00)	0	160	0.5
reserved_hu01_hu0	365.45	(321.21, 409.70)	7.40	161.20	28.91
reserved_vx01_vx02	104.54	(55.92, 153.16)	0.64	2549.34	2.32
ty01_ty60	12384.31	(35123.03, 61020.40)	3.12	388.46	407.04
ty61_ty72	9102.97	(44608.37, 57195.43)	0	364916.6	98.81
hu03_hu08	48071.71	(46807.77, 50677.69)	10123.84	164228.03	8.12
hu09_hu12	50901.9	(37922.02, 56675.21)	5136.01	228981.29	10.91
wh03_wh44	48742.73	(46807.77, 50677.69)	3973.95	356815.5	76.68
wh45_wh52	47298.611	(37922.02, 56675.21)	5209.7	205058.99 12.29	12.29
vx03_vx28	15737.46	(14117.03, 17357.89)	57.8	294163.5	83.54
vx29_vx36	34538.29	(19691.19, 49385.39)	6970.64	164481.83	17.56
va01_va10	79205.36	(56451.36, 101959.37)	12106.64	434820.75	236.42

Table 5: Average Wait Time Results from LVF Job Length Queue Configuration

Identifier	Avg Wait Time	95% CI	Avg Min	Avg Max	Avg Job Count
reserved_wh01_wh02	31.021	(4.68, 57.37)	0.17	193.55	7.62
reserved_hu01_hu0	322.41	(283.4927, 361.33)	0.33	2566.33	35.31
reserved_vx01_vx02	60.52	(33.80, 87.24)	0.56	413.14	9.32
ty01_ty60	11317.81	(10157.77, 12477.85)	0	364310	407.37
ty61_ty72	5310.5	(4295.00, 6325.99)	0	208615.94	206.33
hu03_hu08	29639.84	(20663.79, 38615.88)	7630.49	167725.06	21.59
hu09_hu12	50578.87	(44373.12, 56784.61)	5123.04	228981.71	10.93
wh03_wh44	48658.84	(46718.43, 50599.25)	3973.95	356815.5	76.68
wh45_wh52	31010.16	(24104.02, 37916.29)	4079.84	204291.37	26.37
vx03_vx28	15182.33	(13624.60, 16740.04)	0.13	295802.59	85.31
vx29_vx36	12184.27	(8608.03, 15760.50)	0.47	137411.37	29.64
va01_va10	19820.61	(15020.17, 24621.12)	0	357603.27	374.41

Table 6: Average Wait Time Results from LVF of number of cores requested Priority Queue Configuration

Identifier	Avg Wait Time	95% CI	Avg Min	Avg Max	Avg Job Count
reserved_wh01_wh02	63.19	(15.37, 110.99)	7.4	161.20	0.49
reserved_hu01_hu0	397.23	(368.40, 426.06)	0.27	3118.23	43.55
reserved_vx01_vx02	402.20	(380.17, 424.24)	0.12	3345.63	61.11
ty01_ty60	10080.24	(9622.32, 10538.16)	0	353341.2	407.8
ty61_ty72	9975.98	(9544.51, 10407.43)	0	356830.1	451.16
hu03_hu08	64890.36	(57171.69, 72609.03)	7889.416	402184.1	58.43
hu09_hu12	50273.05	(44318.27, 56227.83)	5123.047	229956.83	11
wh03_wh44	48680.03	(46745.4, 504614.62)	3973.95	356789	76.73
wh45_wh52	59396.15	(55043.60, 63748.69)	4015.603	390056.8	60.39
vx03_vx28	14849.89	(13685.56, 16014.22)	0.13	298599.2	87.62
vx29_vx36	19485.49	(17598.43, 21372.56)	0.28	294624.24	74.29
va01_va10	21011.54	(19727.15, 22295.93)	0	469954.6	691.39

The average user wait times for each node type, a 95% confidence interval of the average wait time, the average minimum wait time, and the average maximum wait time are displayed in Table 4, 5, and 6 for the results from 100 trails. Table 4 contains the results from the model run using a FIFO queue, Table 5 contains the results from the model run using a lowest-value-first by assigned job length priority queue, and Table 6 contains the results from the

model using a lowest-value-first by number of cores needed priority queue. The left-hand column displays the name of the resource or node type.

Table 7: Average Utilization Results from FIFO Queue Configuration

Identifier	Average Utilization	95% CI	Average Min	Average Max
reserved_wh01_wh02	0.02	(0.022, 0.024)	0	16
reserved_hu01_hu0	0.02	(0.023, 0.025)	0	16
reserved_vx01_vx02	0.03	(0.031, 0.035)	0	24
ty01_ty60	1.14	(0.29, 1.97)	0	240
ty61_ty72	0.13	(0.11, 0.14)	0	48
hu03_hu08	3.51	(1.30, 5.70)	0.96	48
hu09_hu12	0.18	(0.06, 0.29)	0	32
wh03_wh44	2.34	(2.16, 2.52)	0	352
wh45_wh52	4.61	(1.70, 7.51)	0.64	64
vx03_vx28	1.60	(1.40, 1.80)	0	336
vx29_vx36	2.85	(0.08, 5.79)	0.96	96
va01_va10	13.27	(6.09, 20.45)	0	160

Table 8: Average Utilization Results from LVF Job Length Queue Configuration

Identifier	Average Utilization	95% CI	Average Min	Average Max
reserved_wh01_wh02	0.02	(0.022, 0.024)	0	16
reserved_hu01_hu0	0.02	(0.023, 0.025)	0	16
reserved_vx01_vx02	0.03	(0.032, 0.036)	0	24
ty01_ty60	1.13	(0.29, 1.97)	0	240
ty61_ty72	0.13	(0.11, 0.14)	0	48
hu03_hu08	2.67	(0.78, 4.55)	0.48	48
hu09_hu12	0.18	(0.065, 0.29)	0	32
wh03_wh44	2.34	(2.17, 2.52)	0	352
wh45_wh52	4.17	(1.39, 6.95)	0.64	64
vx03_vx28	1.61	(1.40, 1.81)	0	336
vx29_vx36	2.44	(0.06, 4.95)	0	96
va01_va10	11.62	(4.73, 18.52)	1.6	160

Table 9: Average Utilization Results from LVF by number of cores Queue Configuration

Identifier	Average Utilization	95% CI	Average Min	Average Max
reserved_wh01_wh02	0.0229	(0.021, 0.024)	0	16
reserved_hu01_hu0	0.02	(0.023, 0.026)	0	16
reserved_vx01_vx02	0.0394	(0.037, 0.041)	0	24
ty01_ty60	1.1337	(0.29, 1.97)	0	240
ty61_ty72	0.3064	(0.28, 0.33)	0	48
hu03_hu08	10.2919	(7.48, 13.10)	0.96	48
hu09_hu12	0.1813	(0.067, 0.29)	0	32
wh03_wh44	2.3449	(2.16, 2.52)	0	352
wh45_wh52	9.1917	(5.58, 12.79)	0.64	64
vx03_vx28	1.6459	(1.44, 1.84)	0	336
vx29_vx36	5.4572	(1.75, 9.15)	0.96	96
va01_va10	27.3543	(18.97, 35.73)	0	160

The average utilization for each node type, a 95% confidence interval of the average wait time from 100 trials, the average minimum wait time, and the average maximum wait time are displayed in Table 7, 8, and 9. Table 7 contains the results from the model run using a FIFO queue, Table 8 contains the results from the model run using

a lowest-value-first by assigned job length priority queue, and Table 9 contains the results from the model using a lowest-value-first by number of cores needed priority queue.

Analysis

The average wait time results for the FIFO queues in Table 4, show very high average wait times across all queues, except for the reserved queues. However, the average number of user jobs for these resources is only 0.5, 28.91, 2.32 respectively, from 100 trails. This is most likely because the probability of a user both having a job of less than one hour and requesting those nodes is very low. The corresponding utilization for these resource are $0.02, 0.02$, and 0.03. The low utilization and wait time suggest that the reserved wh01:wh02 node could be better utilized if opened to all job times. A similar conclusion can be made for the reserved vx01:vx02 nodes where the average wait time is less than two minutes, and the average total job count from 51 weeks is only 2.32. The highest average wait time across all nodes is for the whirlwind 45:52 resource, this average wait time is 47298.61 seconds or 13.13 hours. This is a reasonable wait time considering some jobs may have a long runtime and require a lot of resources or a large number of cores. The problem with the FIFO priority configuration is that a job that needs a large number of cores can hold up the rest of the queue waiting on a sufficient number of cores, and jobs which require a smaller number of cores could be running during this time. The second configuration, LVF by job length has lower average wait times for all nodes compared to the FIFO priority configuration. We typically see the average wait times most significantly drop in the nodes with a larger number of total cores, including va01:va10, vx29:vx36, ty60:ty72, and hu03:hu08. The average wait is not calculated using root jobs.

Overall, the throughput across all nodes is relatively low for the FIFO configuration. The average job counts for the FIFO configuration are for a total of 51 weeks, with respective average counts of: 0.5, 28.91, 2.32, 407.04, 98.81, 8.12, 10.91, 76.68, 12.29, 83.54, 17.56, and 236.42. The throughput is improved using the LVF by job length queue across all jobs compared to the FIFO configuration. Similarly, the throughput for all nodes is greater or unchanged comparing the FIFO and the LVF by number of cores configuration. The LVF by job length had a greater throughput than the LVF by number of cores in the reserved wh01:wh02, ty61:ty72, hu09:hu12, and the LVF by number of cores had a greater throughput in all other nodes, in some cases, by significant margins. However it is important to note, in both the LVF priority queues, at times, users with large job times will sit on the queue and never run, which would need to be addressed in a real-world application. These job statistics are not reflected in the average results in the table, because these entities will not arrive at the Tally block, which marks the wait time after a entity seizes a resource.

This model overall included many simplifications from the current cluster setup. It would be informational to tailor the distributions to the hour, day of week, and year to observe the trends across these different time frames. Times of high academic activity such as mid-semester may have a higher job submission rate and times of lower academic activity such as the summer and winter break may have a lower job submission rate. When viewing a histogram of the job count by day in Appendix, Figure 9, at times, there are a sudden burst in the number of jobs submitted for a day. Modeling these burst and their impact on the cluster would create more realistic results. Also, it would be informational to collect additional statistics on average queue length to observe why the utilization is so low on various nodes. In conclusion, this model produces a proof of concept of the cluster's processes, resources, configuration, and job flow, however, further modifications are needed to produce any helpful application or insight.

APPENDIX

Experiment Source File:

```
BEGIN, YES;
PROJECT,    HPC Cluster, Nadia Aly, 12/01/16;

ATTRIBUTES: 1, jobType:
            2, TimeIn:
            3, jobTime:
            4, num_cores:
            5, jobID:
            6, nodeType:
            Picture;
PICTURES:  Picture.Woman:
            Picture.Man;

ENTITIES: Entity 1, Picture.Woman:
            Entity 2, Picture.Man;
VARIABLES:
        check_09_12_ID, 0:
check_03_08_ID, 0:
check_wh03_wh44_ID, 0:
check_wh45_wh52_ID, 0:
check_ty01_ty60_ID,0:
check_ty61_ty72_ID,0:
check_va01_va10_ID,0:
check_vx03_vx28_ID,0:
check_vx29_vx36_ID,0:
job_count, 0;
STATIONS:
        vx01_station:
vx01_pass_station:
wh01_station:
wh01_pass_station:
ty01_station:
ty01_pass_station:
hu03_station:
hu03_pass_station:
hu09_station:
hu09_pass_station:
wh03_station:
wh03_pass_station:
wh45_station:
wh45_pass_station:
vx03_station:
vx03_pass_station:
vx29_station:
vx29_pass_station:
va01_station:
va01_pass_station;
QUEUES: Buffer_reserved_wh01_wh02:
    Buffer_reserved_hu01_hu02:
    Buffer_reserved_vx01_vx02:
    Buffer ty01_ty60:
    Buffer_ty61_ty72:
        Buffer_hu03_hu08:
        Buffer_hu09_hu12:
```

```
        Buffer_wh03_wh44:
        Buffer_wh45_wh52:
        Buffer_vx03_vx28:
        Buffer_vx29_vx36:
        Buffer_va01_va10:
      root_Buffer_reserved_wh01_wh02:
    root_Buffer_reserved_hu01_hu02:
    root_Buffer_reserved_vx01_vx02:
    root_Buffer_ty01_ty60:
        Buffer_ty61_ty72:
        root_Buffer_hu03_hu08:
     root_Buffer_hu09_hu12:
     root_Buffer_wh03_wh44:
     root_Buffer_wh45_wh52:
     root_Buffer_vx03_vx28:
        root_Buffer_vx29_vx36:
        root_Buffer_va01_va10;
RESOURCES:
        reserved_wh01_wh02, 16:
    reserved_hu01_hu02, 16:
    reserved_vx01_vx02, 24:
    ty01_ty60, 240:
        ty61_ty72, 48:
        hu03_hu08, 48:
        hu09_hu12, 32:
        wh03_wh44, 352:
        wh45_wh52, 64:
vx03_vx28, 336:
        vx29_vx36, 96:
        va01_va10, 160;
 DSTATS: NR(reserved_wh01_wh02), Reserved wh01_wh02 Utilization:
    NR(reserved_hu01_hu02), Reserved_hu01_hu02 Utilization:
    NR(reserved_vx01_vx02), Reserved vx01_vx02 Utilization:
    NR(ty01_ty60), ty01_ty60 Utilization:
        NR(ty61_ty72), ty61_ty72 Utilization:
        NR(hu03_hu08), hu03_hu08 Utilization:
        NR(hu09_hu12), hu09_hu12 Utilization:
        NR(wh03_wh44), wh03_wh44 Utilization:
        NR(wh45_wh52), wh45_wh52 Utilization:
        NR(vx03_vx28), vx03_vx28 Utilization:
        NR(vx29_vx36), vx29_vx36 Utilization:
        NR(va01_va10), va01_va10 Utilization;
COUNTERS: 1, check_queue;
TALLIES: 1, reserved_wh01_wh02_WaitTime:
        2, reserved_hu01_hu02_WaitTime:
    3, reserved_vx01_vx02_WaitTime:
        4, ty01_ty60_WaitTime:
        5, ty61_ty72_WaitTime:
        6, hu03_hu08_WaitTime:
        7, hu09_hu12_WaitTime:
        8, wh03_wh44_WaitTime:
        9, wh45_wh52_WaitTime:
        10, vx03_vx28_WaitTime:
        11, vx29_vx36_WaitTime:
        12, va01_va10_WaitTime;
REPLICATE,  100,0, 31536000,,,604800;
END;
```

Model Source File:

```
BEGIN;
       CREATE, 1: DISC(0.56,0, 0.69, 1, 0.70,3,0.71,8, 0.72,17, 0.73, 29,0.74, 44,0.75, 58,
           0.84,59, 0.90, 60,0.91, 61,0.92,85, 0.93,120, 0.94, 134,0.95,206, 0.96,317, 0.97, 512,
           0.98, 946, 0.99, 2268, 1.0, 1731605):


MARK(TimeIn);
ASSIGN:
nodeType=DISC(0.0234,1,0.0296,2,0.0312,3,0.0317,4,0.041,5,0.083,6,0.101,7,0.118,8,0.136,9,
0.548, 10,0.775,11,0.801, 12,1.0,13):

job_count = job_count+1:
jobType=1:
jobID = job_count:
jobTime=DISC(0.01,0.00,0.02,0.00, 0.03,0.00,0.05,1.00,0.2,2,0.25,3,0.27,4,0.3,5,0.32,6,
0.34,7,0.36, 8,0.37,9,0.38,10, 0.39,11,0.4,12,0.41,13,0.42,14,0.43,15,0.44,17,0.45,18
,0.46,20, 0.47,23,0.48,26,0.49,29,0.5,32,0.51,37,0.52,44,0.53, 52.00,0.54, 66.00, 0.55,
81.00, 0.56, 91.00,  0.57,116,0.58,148,0.59,187,0.6,237,0.61,271,0.62,328,0.63,396, 0.64,
482, 0.65,589.00, 0.66, 687.00,  0.67,805.00, 0.68, 925.00, 0.69, 1052.00, 0.7, 1161.00,
0.71, 1301.00, 0.72, 1451.00, 0.73,  1610.00,0.74, 1770.00,0.75, 1959.96, 0.76, 2184.00, 0.77,
2481.00, 0.78, 2749.00, 0.79, 3161.00,0.80, 3501.00, 0.81, 3948.00, 0.82, 4467.00, 0.83,
5068.00, 0.84, 5763.00, 0.85,6352.36,0.86,  7042.00,0.87,  7891.44,0.88,   8626.48, 0.89,
10153.00, 0.9, 12224.00,0.91, 14408.00,0.92,18223.48, 0.93,22263.68,  0.94, 28915.00,0.95,
36923.52, 0.96,52196.80, 0.97, 68402.00,0.98, 103909.64, 0.99,216070.40, 1.0,  359639.00):

nodeType=DISC(0.0234,1,0.0296,2,0.0312,3,0.0317,4,0.041,5,0.083,6,0.101,7,0.118,8,0.136,9,
0.548, 10,0.775,11,0.801, 12,1.0,13):
NEXT(assign_cores);


assign_cores      BRANCH, 1:
IF, (nodeType==1), assign_nodetyp1:
IF, (nodeType==2), assign_nodetyp2:
IF, (nodeType==3), assign_nodetyp3:
IF, (nodeType==4), assign_nodetyp4:
IF, (nodeType==5), assign_nodetyp5:
IF, (nodeType==6), assign_nodetyp6:
IF, (nodeType==7), assign_nodetyp7:
IF, (nodeType==8), assign_nodetyp8:
IF, (nodeType==9), assign_nodetyp9:
IF, (nodeType==10), assign_nodetyp10:
IF, (nodeType==11), assign_nodetyp11:
IF, (nodeType==12), assign_nodetyp12:
IF, (nodeType==13), assign_nodetyp13;

assign_nodetyp1  ASSIGN:
num_cores = DISC(0.0612,1, 0.6826, 2, 0.6873, 4, 0.9246, 8, 0.9286,
    12, 0.9797, 16, 0.9876, 24, 0.9979, 32,0.9989, 48, 1.0, 64):
NEXT(reg_branch);

assign_nodetyp2  ASSIGN:
num_cores = DISC(0.005, 1, 0.675, 2, 0.942, 4, 1.0, 8):
NEXT(reg_branch);

assign_nodetyp3  ASSIGN:
num_cores = DISC(0.388,1,0.460,4,0.848,8,0.894,16,0.962,32,1.0,64):
```

```
NEXT(reg_branch);


assign_nodetyp4  ASSIGN:
num_cores = DISC(0.378, 1, 0.390,4,0.683,8,0.805,16,1,24):
NEXT(reg_branch);

assign_nodetyp5  ASSIGN:
num_cores = DISC(0.251,1,0.302,2,0.401,6,0.933,8,0.961,12,0.973,16,0.975,18,0.999,32,1,64):
NEXT(reg_branch);
assign_nodetyp6  ASSIGN:
num_cores = DISC(0.314,1, 0.343,2,0.355,4,0.394,6,0.406,8,0.408,10,0.598,12,0.600,14,0.615,
16,0.617,18,
 0.663,24, 0.674, 32, 0.685, 36, 0.75, 48, 0.765, 60, 0.813, 72, 0.818, 84,
 0.919, 96, 0.923, 108,
   0.956, 120, 0.968, 144, 0.969, 160, 0.971, 168, 0.971, 180, 0.980, 192, 0.984, 216,
 0.999, 304, 1.0, 360):
NEXT(reg_branch);
assign_nodetyp7  ASSIGN:
num_cores = DISC(0.041, 1, 0.767,2, 0.773, 4, 0.778, 5, 0.782, 6, 0.786, 8, 0.796, 12, 0.887, 16,0.956,
24, 0.959, 32, 0.977, 36, 0.993, 48,0.997, 60, 0.999, 96,  1.0, 144 ):
NEXT(reg_branch);

assign_nodetyp8  ASSIGN:
num_cores = DISC(0.116, 1, 0.887, 2, 0.892, 4, 0.897, 6, 0.903, 8, 0.976, 12, 0.982, 24, 0.991,
 48, 0.994, 60, 0.999, 84, 1.0, 96 ):
NEXT(reg_branch);

assign_nodetyp9  ASSIGN:
num_cores = DISC(0.0684,1,0.0740,4,0.4092,12,0.466, 24, 0.486, 36, 0.892, 48, 0.901, 60, 0.926, 72,
0.930, 84, 0.953, 96, 0.958, 108, 0.969, 120, 0.973, 144, 0.976, 168, 0.983, 180, 0.990, 192,
0.994, 204, 0.995, 264, 0.999, 300, 1.0, 360 ):
  NEXT(reg_branch);

assign_nodetyp10  ASSIGN:
num_cores = DISC(0.543, 1, 0.544, 4, 0.55, 12, 0.621, 16, 0.622, 24, 0.897, 32, 0.898, 36, 0.899, 40,
0.909, 60, 0.912, 64, 0.913, 72, 0.958, 80, 0.960, 96, 0.999, 144, 1.0, 160):
NEXT(reg_branch);

assign_nodetyp11  ASSIGN:
num_cores = DISC(0.757, 1, 0.830, 2, 0.979, 3, 0.980, 12, 0.985, 16, 0.986, 20, 0.987, 32, 0.99,
40, 0.99, 60, 0.992, 80, 0.993, 100, 0.995, 140, 0.997, 160, 0.998, 200, 0.999, 216, 1.0, 240):
NEXT(reg_branch);

assign_nodetyp12  ASSIGN:
num_cores = DISC(0.134, 1, 0.983, 2, 0.997, 4, 0.998, 8, 0.999, 16, 1.0, 32):
NEXT(reg_branch);

assign_nodetyp13  ASSIGN:
num_cores = DISC(0.607, 1, 0.609, 4, 0.752, 8, 0.78, 16, 0.798, 24, 0.828, 32, 0.85, 40, 0.901, 48,
0.905, 56, 0.939, 64, 0.941, 72, 0.959, 80, 0.985, 96, 0.99, 120, 0.992, 128, 0.994, 144, 0.996, 160,
0.998, 184, 0.999, 240, 1.0, 280):
NEXT(reg_branch);

CREATE, 2: 86400:
       MARK(TimeIn);
       ASSIGN:
```

```
jobType=2:
               jobTime=DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29,
                10, 0.3, 14, 0.31, 15, 0.32, 16,0.34, 17, 0.35, 19, 0.36, 21, 0.37, 22,
                0.38, 23, 0.39, 26, 0.4, 28, 0.41, 32, 0.42, 37, 0.43, 42, 0.45, 43,
                0.5,44, 0.52, 45, 0.53, 46, 0.54,  48, 0.55, 50, 0.56, 51, 0.57, 52, 0.58, 52, 0.6,
                53, 0.62, 54, 0.63, 55, 0.64, 56,0.65, 57, 0.67, 58, 0.68, 59, 0.69, 60,  0.7, 61,
                0.71, 62,0.72, 65, 0.73, 68, 0.74, 72, 0.75, 78, 0.76, 88,0.77,96,0.78, 97, 0.79,
                100, 0.8, 102, 0.82, 103, 0.87,104, 0.89, 105, 0.9, 106, 0.91, 108, 0.92, 109, 0.95,
                110, 0.96, 111, 0.97, 117, 0.98, 126, 0.99, 149, 1, 7688):

NEXT(root_reserved_wh01_wh02);


CREATE, 2: 86400:
       MARK(TimeIn);
       ASSIGN:  jobType=2:
               jobTime=DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29,
                10, 0.3, 14, 0.31, 15, 0.32, 16,0.34, 17, 0.35, 19, 0.36, 21, 0.37, 22,
                0.38, 23, 0.39, 26, 0.4, 28, 0.41, 32, 0.42, 37, 0.43, 42, 0.45, 43,
                0.5,44, 0.52, 45, 0.53, 46, 0.54,  48, 0.55, 50, 0.56, 51, 0.57, 52, 0.58, 52, 0.6,
                53, 0.62, 54, 0.63, 55, 0.64, 56,0.65, 57, 0.67, 58, 0.68, 59, 0.69, 60,  0.7, 61,
                0.71, 62,0.72, 65, 0.73, 68, 0.74, 72, 0.75, 78, 0.76, 88,0.77,96,0.78, 97, 0.79,
                100, 0.8, 102, 0.82, 103, 0.87,104, 0.89, 105, 0.9, 106, 0.91, 108, 0.92, 109, 0.95,
                110, 0.96, 111, 0.97, 117, 0.98, 126, 0.99, 149, 1, 7688):
NEXT(root_reserved_hu01_hu02);

CREATE, 2: 86400:
       MARK(TimeIn);
jobType=2:
               jobType=2:
               jobTime=DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29,
                10, 0.3, 14, 0.31, 15, 0.32, 16,0.34, 17, 0.35, 19, 0.36, 21, 0.37, 22,
                0.38, 23, 0.39, 26, 0.4, 28, 0.41, 32, 0.42, 37, 0.43, 42, 0.45, 43,
                0.5,44, 0.52, 45, 0.53, 46, 0.54,  48, 0.55, 50, 0.56, 51, 0.57, 52, 0.58, 52, 0.6,
                53, 0.62, 54, 0.63, 55, 0.64, 56,0.65, 57, 0.67, 58, 0.68, 59, 0.69, 60,  0.7, 61,
                0.71, 62,0.72, 65, 0.73, 68, 0.74, 72, 0.75, 78, 0.76, 88,0.77,96,0.78, 97, 0.79,
                100, 0.8, 102, 0.82, 103, 0.87,104, 0.89, 105, 0.9, 106, 0.91, 108, 0.92, 109, 0.95,
                110, 0.96, 111, 0.97, 117, 0.98, 126, 0.99, 149, 1, 7688):
NEXT(root_reserved_vx01_vx02);


CREATE, 60: 86400:
       MARK(TimeIn);
       ASSIGN:  jobType=2:
               jobTime=DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29,
                10, 0.3, 14, 0.31, 15, 0.32, 16,0.34, 17, 0.35, 19, 0.36, 21, 0.37, 22,
                0.38, 23, 0.39, 26, 0.4, 28, 0.41, 32, 0.42, 37, 0.43, 42, 0.45, 43,
                0.5,44, 0.52, 45, 0.53, 46, 0.54,  48, 0.55, 50, 0.56, 51, 0.57, 52, 0.58, 52, 0.6,
                53, 0.62, 54, 0.63, 55, 0.64, 56,0.65, 57, 0.67, 58, 0.68, 59, 0.69, 60,  0.7, 61,
                0.71, 62,0.72, 65, 0.73, 68, 0.74, 72, 0.75, 78, 0.76, 88,0.77,96,0.78, 97, 0.79,
                100, 0.8, 102, 0.82, 103, 0.87,104, 0.89, 105, 0.9, 106, 0.91, 108, 0.92, 109, 0.95,
                110, 0.96, 111, 0.97, 117, 0.98, 126, 0.99, 149, 1, 7688):
NEXT(root_ty01_ty60);

CREATE, 12: 86400:
       MARK(TimeIn);
       jobType=2:
```

```
               jobTime=DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29,
                10, 0.3, 14, 0.31, 15, 0.32, 16,0.34, 17, 0.35, 19, 0.36, 21, 0.37, 22,
                0.38, 23, 0.39, 26, 0.4, 28, 0.41, 32, 0.42, 37, 0.43, 42, 0.45, 43,
                0.5,44, 0.52, 45, 0.53, 46, 0.54,  48, 0.55, 50, 0.56, 51, 0.57, 52, 0.58, 52, 0.6,
                53, 0.62, 54, 0.63, 55, 0.64, 56,0.65, 57, 0.67, 58, 0.68, 59, 0.69, 60,  0.7, 61,
                0.71, 62,0.72, 65, 0.73, 68, 0.74, 72, 0.75, 78, 0.76, 88,0.77,96,0.78, 97, 0.79,
                100, 0.8, 102, 0.82, 103, 0.87,104, 0.89, 105, 0.9, 106, 0.91, 108, 0.92, 109, 0.95,
                110, 0.96, 111, 0.97, 117, 0.98, 126, 0.99, 149, 1, 7688):
NEXT(root_ty61_ty72);


CREATE, 6: 86400:
       MARK(TimeIn);
       jobType=2:
               jobTime=DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29,
                10, 0.3, 14, 0.31, 15, 0.32, 16,0.34, 17, 0.35, 19, 0.36, 21, 0.37, 22,
                0.38, 23, 0.39, 26, 0.4, 28, 0.41, 32, 0.42, 37, 0.43, 42, 0.45, 43,
                0.5,44, 0.52, 45, 0.53, 46, 0.54,  48, 0.55, 50, 0.56, 51, 0.57, 52, 0.58, 52, 0.6,
                53, 0.62, 54, 0.63, 55, 0.64, 56,0.65, 57, 0.67, 58, 0.68, 59, 0.69, 60,  0.7, 61,
                0.71, 62,0.72, 65, 0.73, 68, 0.74, 72, 0.75, 78, 0.76, 88,0.77,96,0.78, 97, 0.79,
                100, 0.8, 102, 0.82, 103, 0.87,104, 0.89, 105, 0.9, 106, 0.91, 108, 0.92, 109, 0.95,
                110, 0.96, 111, 0.97, 117, 0.98, 126, 0.99, 149, 1, 7688):
NEXT(root_hu03_hu08);

CREATE,4: 86400:
       MARK(TimeIn);
       ASSIGN:
jobType=2:
               jobTime=DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29,
                10, 0.3, 14, 0.31, 15, 0.32, 16,0.34, 17, 0.35, 19, 0.36, 21, 0.37, 22,
                0.38, 23, 0.39, 26, 0.4, 28, 0.41, 32, 0.42, 37, 0.43, 42, 0.45, 43,
                0.5,44, 0.52, 45, 0.53, 46, 0.54,  48, 0.55, 50, 0.56, 51, 0.57, 52, 0.58, 52, 0.6,
                53, 0.62, 54, 0.63, 55, 0.64, 56,0.65, 57, 0.67, 58, 0.68, 59, 0.69, 60,  0.7, 61,
                0.71, 62,0.72, 65, 0.73, 68, 0.74, 72, 0.75, 78, 0.76, 88,0.77,96,0.78, 97, 0.79,
                100, 0.8, 102, 0.82, 103, 0.87,104, 0.89, 105, 0.9, 106, 0.91, 108, 0.92, 109, 0.95,
                110, 0.96, 111, 0.97, 117, 0.98, 126, 0.99, 149, 1, 7688):
NEXT(root_hu09_hu12);

CREATE,44: 86400:
       MARK(TimeIn);
       ASSIGN:  jobType=2:
               jobTime=DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29,
                10, 0.3, 14, 0.31, 15, 0.32, 16,0.34, 17, 0.35, 19, 0.36, 21, 0.37, 22,
                0.38, 23, 0.39, 26, 0.4, 28, 0.41, 32, 0.42, 37, 0.43, 42, 0.45, 43,
                0.5,44, 0.52, 45, 0.53, 46, 0.54,  48, 0.55, 50, 0.56, 51, 0.57, 52, 0.58, 52, 0.6,
                53, 0.62, 54, 0.63, 55, 0.64, 56,0.65, 57, 0.67, 58, 0.68, 59, 0.69, 60,  0.7, 61,
                0.71, 62,0.72, 65, 0.73, 68, 0.74, 72, 0.75, 78, 0.76, 88,0.77,96,0.78, 97, 0.79,
                100, 0.8, 102, 0.82, 103, 0.87,104, 0.89, 105, 0.9, 106, 0.91, 108, 0.92, 109, 0.95,
                110, 0.96, 111, 0.97, 117, 0.98, 126, 0.99, 149, 1, 7688):
NEXT(root_wh03_wh44);


CREATE,8: 86400:
       MARK(TimeIn);
       ASSIGN: Picture=Picture.Man:
jobType=2:
               jobTime=DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29, 10, 0.3, 14, 0
```

```
NEXT(root_wh45_wh52);



CREATE,28: 86400:
       MARK(TimeIn);
       ASSIGN: Picture=Picture.Man:
jobType=2:
               jobType=2:
               jobTime=DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29,
                10, 0.3, 14, 0.31, 15, 0.32, 16,0.34, 17, 0.35, 19, 0.36, 21, 0.37, 22,
                0.38, 23, 0.39, 26, 0.4, 28, 0.41, 32, 0.42, 37, 0.43, 42, 0.45, 43,
                0.5,44, 0.52, 45, 0.53, 46, 0.54,  48, 0.55, 50, 0.56, 51, 0.57, 52, 0.58, 52, 0.6,
                53, 0.62, 54, 0.63, 55, 0.64, 56,0.65, 57, 0.67, 58, 0.68, 59, 0.69, 60,  0.7, 61,
                0.71, 62,0.72, 65, 0.73, 68, 0.74, 72, 0.75, 78, 0.76, 88,0.77,96,0.78, 97, 0.79,
                100, 0.8, 102, 0.82, 103, 0.87,104, 0.89, 105, 0.9, 106, 0.91, 108, 0.92, 109, 0.95,
                110, 0.96, 111, 0.97, 117, 0.98, 126, 0.99, 149, 1, 7688):
NEXT(root_vx03_vx28);


CREATE,8: 86400:
       MARK(TimeIn);
       ASSIGN:  jobType=2:
               jobTime=DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29,
                10, 0.3, 14, 0.31, 15, 0.32, 16,0.34, 17, 0.35, 19, 0.36, 21, 0.37, 22,
                0.38, 23, 0.39, 26, 0.4, 28, 0.41, 32, 0.42, 37, 0.43, 42, 0.45, 43,
                0.5,44, 0.52, 45, 0.53, 46, 0.54,  48, 0.55, 50, 0.56, 51, 0.57, 52, 0.58, 52, 0.6,
                53, 0.62, 54, 0.63, 55, 0.64, 56,0.65, 57, 0.67, 58, 0.68, 59, 0.69, 60,  0.7, 61,
                0.71, 62,0.72, 65, 0.73, 68, 0.74, 72, 0.75, 78, 0.76, 88,0.77,96,0.78, 97, 0.79,
                100, 0.8, 102, 0.82, 103, 0.87,104, 0.89, 105, 0.9, 106, 0.91, 108, 0.92, 109, 0.95,
                110, 0.96, 111, 0.97, 117, 0.98, 126, 0.99, 149, 1, 7688):
NEXT(root_vx29_vx36);



CREATE,10: 86400:
       MARK(TimeIn);
       ASSIGN:  jobType=2:
               jobTime=DISC(0.11,1, 0.17,2, 0.21, 3, 0.24, 4, 0.25, 5, 0.26, 6,0.28, 7, 0.29,
                10, 0.3, 14, 0.31, 15, 0.32, 16,0.34, 17, 0.35, 19, 0.36, 21, 0.37, 22,
                0.38, 23, 0.39, 26, 0.4, 28, 0.41, 32, 0.42, 37, 0.43, 42, 0.45, 43,
                0.5,44, 0.52, 45, 0.53, 46, 0.54,  48, 0.55, 50, 0.56, 51, 0.57, 52, 0.58, 52, 0.6,
                53, 0.62, 54, 0.63, 55, 0.64, 56,0.65, 57, 0.67, 58, 0.68, 59, 0.69, 60,  0.7, 61,
                0.71, 62,0.72, 65, 0.73, 68, 0.74, 72, 0.75, 78, 0.76, 88,0.77,96,0.78, 97, 0.79,
                100, 0.8, 102, 0.82, 103, 0.87,104, 0.89, 105, 0.9, 106, 0.91, 108, 0.92, 109, 0.95,
                110, 0.96, 111, 0.97, 117, 0.98, 126, 0.99, 149, 1, 7688):
NEXT(root_va01_va10);


reg_branch       BRANCH,1 :
IF, jobTime<3600, reserved_branch:
                  ELSE, next_choice;

reserved_branch  BRANCH,1:
IF, (nodeType==3).OR.(nodeType==4).OR.(nodeType==5).OR.(nodeType==13), goto_reserved_wh01_wh02:
IF, (nodeType==1).OR.(nodeType==2),goto_reserved_hu01_hu02:
IF, (nodeType==7).OR.(nodeType==8).OR.(nodeType==9), goto_reserved_vx01_vx02:
```

```
       ELSE, next_choice;

next_choice       BRANCH, 3:
IF, (nodeType==1), goto_hu03_hu08:
IF, (nodeType==1), goto_hu09_hu12:
IF, (nodeType==2), goto_hu09_hu12:
IF, (nodeType==3), goto_wh03_wh44:
IF, (nodeType==4), goto_wh45_wh52:
IF, (nodeType==5), goto_wh03_wh44:
IF, (nodeType==5), goto_wh45_wh52:
IF, (nodeType==6), goto_va01_va10:
IF, (nodeType==6), goto_vx03_vx28:
IF, (nodeType==6), goto_vx29_vx36:
IF, (nodeType==7), goto_vx03_vx28:
IF, (nodeType==8), goto_vx29_vx36:
IF, (nodeType==9), goto_vx03_vx28:
IF, (nodeType==9), goto_vx29_vx36:
IF, (nodeType==10), goto_va01_va10:
IF, (nodeType==11), goto_ty01_ty60:
IF, (nodeType==11), goto_ty61_ty72:
IF, (nodeType==12), goto_ty61_ty72:
IF, (nodeType==13), goto_wh03_wh44:
IF, (nodeType==13), goto_wh45_wh52:
IF, (nodeType==13), goto_hu03_hu08:
IF, (nodeType==13), goto_hu09_hu12;

goto_reserved_wh01_wh02         QUEUE,  Buffer_reserved_wh01_wh02;
SEIZE, 2: reserved_wh01_wh02, num_cores;
TALLY: 1, INT(TimeIn);
                          DELAY: jobTime;
STATION, wh01_station;
ROUTE:  0, wh01_pass_station;
STATION, wh01_pass_station;
                          RELEASE: reserved_wh01_wh02, num_cores;

              DISPOSE;

goto_reserved_hu01_hu02
QUEUE, Buffer_reserved_hu01_hu02;
SEIZE, 2: reserved_hu01_hu02, num_cores;
TALLY: 2, INT(TimeIn);

                         DELAY: jobTime;
                           RELEASE: reserved_hu01_hu02, num_cores;
              DISPOSE;

goto_reserved_vx01_vx02  QUEUE, Buffer_reserved_vx01_vx02;
   SEIZE, 2: reserved_vx01_vx02, num_cores;
   TALLY: 3, INT(TimeIn);
                            DELAY: jobTime;
STATION, vx01_station;
ROUTE:  0, vx01_pass_station;
STATION, vx01_pass_station;
                            RELEASE: reserved_vx01_vx02, num_cores;
              DISPOSE;

goto_ty01_ty60         QUEUE, Buffer_ty01_ty60;
```

```
          SEIZE, 2:ty01_ty60, num_cores;
ASSIGN: check_ty01_ty60_ID = jobID;
BRANCH, 1:
IF, nodeType==11, check_ty01_ty60_queue_11:
ELSE, continue_ty01_ty60;
continue_ty01_ty60
TALLY: 4, INT(TimeIn);
DELAY: jobTime;
STATION, ty01_station;
ROUTE:  0, ty01_pass_station;
STATION, ty01_pass_station;
                          RELEASE: ty01_ty60, num_cores;
             DISPOSE;

goto_ty61_ty72 QUEUE, Buffer_ty61_ty72;
          SEIZE, 2: ty61_ty72, num_cores;
ASSIGN: check_ty61_ty72_ID = jobID;
BRANCH, 1:
IF, nodeType==11, check_ty61_ty72_queue_11:
ELSE, continue_ty01_ty60;
continue_ty61_ty72
TALLY: 5, INT(TimeIn);
DELAY: jobTime;

                          RELEASE: ty61_ty72, num_cores;
              DISPOSE;

goto_hu03_hu08 QUEUE, Buffer_hu03_hu08;
          SEIZE, 2: hu03_hu08, num_cores;
ASSIGN: check_03_08_ID = jobID;
BRANCH, 1:
IF, nodeType==1, check_hu03_hu08_queue_1:
IF, nodeType==13, check_hu03_hu08_queue_13:
ELSE, continue_hu03_hu08;

continue_hu03_hu08
             TALLY: 6, INT(TimeIn);
DELAY: jobTime;
STATION, hu03_station;
ROUTE:  0, hu03_pass_station;
STATION, hu03_pass_station;
                          RELEASE: hu03_hu08, num_cores;
              DISPOSE;

goto_hu09_hu12 QUEUE, Buffer_hu09_hu12;
          SEIZE, 2: hu09_hu12, num_cores;
ASSIGN: check_09_12_ID = jobID;
BRANCH, 1:
IF, nodeType==1, check_hu09_hu12_queue_1:
IF, nodeType==13, check_hu09_hu12_queue_13:

ELSE, continue_hu09_hu12;

continue_hu09_hu12
TALLY: 7, INT(TimeIn);
                DELAY: jobTime;
ROUTE:  0, hu09_pass_station;
```

```
STATION, hu09_pass_station;
                         RELEASE: hu09_hu08, num_cores;
                        RELEASE: hu09_hu12, num_cores;
              DISPOSE;

goto_wh03_wh44 QUEUE, Buffer_wh03_wh44;
           SEIZE, 2: wh03_wh44, num_cores;
ASSIGN: check_wh03_wh44_ID = jobID;
BRANCH, 1:
IF, nodeType==5, check_wh03_wh44_queue_5:
IF, nodeType==13, check_wh03_wh44_queue_13:

ELSE, continue_wh03_wh44;

continue_wh03_wh44
TALLY: 8, INT(TimeIn);
                DELAY: jobTime;
                RELEASE: wh03_wh44, num_cores;
              DISPOSE;

goto_wh45_wh52 QUEUE, Buffer_wh45_wh52;
           SEIZE, 2: wh45_wh52, num_cores;
          TALLY: 9, INT(TimeIn);
    DELAY: jobTime;
                         RELEASE: wh45_wh52, num_cores;
              DISPOSE;

goto_vx03_vx28 QUEUE, Buffer_vx03_vx28;
           SEIZE, 2: vx03_vx28, num_cores;

ASSIGN: check_vx29_vx36_ID = jobID;
BRANCH, 1:
IF, (nodeType==6).OR.(nodeType==9), check_vx03_vx28_queue_6:
ELSE, continue_vx03_vx28;

continue_vx03_vx28
                    TALLY: 10, INT(TimeIn);
                    DELAY: jobTime;
        RELEASE: vx03_vx28, num_cores;
              DISPOSE;

goto_vx29_vx36 QUEUE, Buffer_vx29_vx36;
           SEIZE, 2: vx29_vx36, num_cores;

ASSIGN: check_vx29_vx36_ID = jobID;

BRANCH,1:
IF,(nodeType==6).OR.(nodeType==9), check_vx29_vx36_queue_6:
ELSE, continue_vx29_vx36;

continue_vx29_vx36
                TALLY: 11, INT(TimeIn);
                DELAY: jobTime;
                RELEASE: vx29_vx36, num_cores;
              DISPOSE;

goto_va01_va10 QUEUE,Buffer_va01_va10;
```

```
          SEIZE, 2: va01_va10, num_cores;

ASSIGN: check_va01_va10_ID = jobID;
BRANCH, 1:
IF, nodeType==6, check_va01_v10_queue_6:
ELSE, continue_va01_v10;

continue_va01_v10 TALLY: 12, INT(TimeIn);
                    DELAY: jobTime;
                    RELEASE: va01_va10, num_cores;
               DISPOSE;

root_reserved_wh01_wh02         QUEUE, root_Buffer_reserved_wh01_wh02;
SEIZE, 1: reserved_wh01_wh02, 8;
BRANCH,1:

IF, (TNOW-TimeIn)>86400, exit_jobs:
                 ELSE, continue_job1;

continue_job1                 DELAY: jobTime;
                           RELEASE: reserved_wh01_wh02, 8;
               DISPOSE;


root_reserved_hu01_hu02        QUEUE, root_Buffer_reserved_hu01_hu02;
SEIZE, 1: reserved_hu01_hu02, 8;
BRANCH,1:
                                IF, (TNOW-TimeIn)>86400, exit_jobs:
                   ELSE, continue_job2;
continue_job2                 DELAY: jobTime;
                           RELEASE: reserved_hu01_hu02, 8;
               DISPOSE;

root_reserved_vx01_vx02         QUEUE, root_Buffer_reserved_vx01_vx02;
SEIZE, 1: reserved_vx01_vx02, 12;
BRANCH,1:
                                IF, (TNOW-TimeIn)>86400, exit_jobs:
                   ELSE, continue_job3;
continue_job3                 DELAY: jobTime;
                           RELEASE: reserved_vx01_vx02, 12;
               DISPOSE;

root_ty01_ty60          QUEUE, root_Buffer_ty01_ty60;
SEIZE, 1: ty01_ty60, 4;
BRANCH,1:
                                IF, (TNOW-TimeIn)>86400, exit_jobs:
                   ELSE, continue_job4;
continue_job4                 DELAY: jobTime;
                           RELEASE: ty01_ty60, 4;
               DISPOSE;

root_ty61_ty72         QUEUE, root_Buffer_ty61_ty72;
SEIZE, 1: ty61_ty72, 4;
BRANCH,1:
                                IF, (TNOW-TimeIn)>86400, exit_jobs:
                   ELSE, continue_job5;
continue_job5                 DELAY: jobTime;
```

```
                        RELEASE: ty61_ty72, 4;
             DISPOSE;

exit_jobs                       COUNT: check_queue;
DISPOSE;


root_hu03_hu08         QUEUE, root_Buffer_hu03_hu08;
SEIZE, 1: hu03_hu08, 8;
BRANCH,1:
                                IF, (TNOW-TimeIn)>86400, exit_jobs:
                  ELSE, continue_job6;
continue_job6                 DELAY: jobTime;
                        RELEASE: hu03_hu08, 8;
             DISPOSE;

exit_job6                       COUNT: check_queue;
DISPOSE;

root_hu09_hu12        QUEUE, root_Buffer_hu09_hu12;
SEIZE, 1: hu09_hu12, 8;
BRANCH,1:
                                IF, (TNOW-TimeIn)>86400, exit_jobs:
                  ELSE, continue_job7;
continue_job7                 DELAY: jobTime;
                        RELEASE: hu09_hu12, 8;
             DISPOSE;

exit_jobs                       COUNT: check_queue;
DISPOSE;


root_wh03_wh44       QUEUE, root_Buffer_wh03_wh44;
SEIZE, 1: wh03_wh44, 8;
BRANCH,1:
                                IF, (TNOW-TimeIn)>86400, exit_jobs:
                  ELSE, continue_job8;
continue_job8                 DELAY: jobTime;
                        RELEASE: wh03_wh44, 8;
             DISPOSE;

root_wh45_wh52       QUEUE, root_Buffer_wh45_wh52;
SEIZE, 1: wh45_wh52, 8;
BRANCH,1:
                                IF, (TNOW-TimeIn)>86400, exit_jobs:
                  ELSE, continue_job9;
continue_job9                 DELAY: jobTime;
                        RELEASE: wh45_wh52, 8;
             DISPOSE;

root_vx03_vx28       QUEUE, root_Buffer_vx03_vx28;
SEIZE, 1: vx03_vx28, 12;
BRANCH,1:
                                IF, (TNOW-TimeIn)>86400, exit_jobs:
                  ELSE, continue_job10;
continue_job10                 DELAY: jobTime;
                        RELEASE: vx03_vx28, 12;
```

```
             DISPOSE;

root_vx29_vx36       QUEUE, root_Buffer_vx29_vx36;
SEIZE, 1: vx29_vx36, 12;
BRANCH,1:
                                IF, (TNOW-TimeIn)>86400, exit_jobs:
                  ELSE, continue_job11;
continue_job11                 DELAY: jobTime;
                         RELEASE: vx29_vx36, 12;
             DISPOSE;

root_va01_va10       QUEUE, root_Buffer_va01_va10;
SEIZE, 1: va01_va10, 16;
BRANCH,1:
                                IF, (TNOW-TimeIn)>86400, exit_jobs:
                  ELSE, continue_job12;
continue_job12                 DELAY: jobTime;
                         RELEASE: va01_va10, 16;
             DISPOSE;

exit_jobs                       COUNT: check_queue;
DISPOSE;


check_hu03_hu08_queue_1         SEARCH, Buffer_hu09_hu12: jobID==check_03_08_ID;
REMOVE: J, Buffer_hu09_hu12, exit_jobs:
NEXT(continue_hu03_hu08);

check_hu03_hu08_queue_13        SEARCH, Buffer_wh03_wh44: jobID==check_03_08_ID;
REMOVE: J, Buffer_wh03_wh44, exit_jobs;
SEARCH, Buffer_wh45_wh52: jobID==check_03_08_ID;
REMOVE: J, Buffer_wh45_wh52, exit_jobs;
SEARCH, Buffer_hu03_hu08: jobID==check_03_08_ID;
REMOVE: J, Buffer_hu03_hu08, exit_jobs:
NEXT(continue_hu03_hu08);

check_hu09_hu12_queue_1         SEARCH, Buffer_hu03_hu08: jobID==check_09_12_ID;
REMOVE: J, Buffer_hu03_hu08, exit_jobs:
NEXT(continue_hu09_hu12);

check_hu09_hu12_queue_13        SEARCH, Buffer_wh03_wh44: jobID==check_09_12_ID;
REMOVE: J, Buffer_wh03_wh44, exit_jobs;
SEARCH, Buffer_wh45_wh52: jobID==check_09_12_ID;
REMOVE: J, Buffer_wh45_wh52, exit_jobs;
SEARCH, Buffer_hu03_hu08: jobID==check_09_12_ID;
REMOVE: J, Buffer_hu03_hu08, exit_jobs:
NEXT(continue_hu09_hu12);


check_wh03_wh44_queue_5        SEARCH, Buffer_wh45_wh52: jobID==check_wh03_wh44_ID;
REMOVE: J, Buffer_wh45_wh52, exit_jobs:
NEXT(continue_wh03_wh44);

check_wh03_wh44_queue_13     SEARCH, Buffer_hu09_hu12: jobID==check_wh03_wh44_ID;
REMOVE: J, Buffer_wh03_wh44, exit_jobs;
SEARCH, Buffer_wh45_wh52: jobID==check_wh03_wh44_ID;
REMOVE: J, Buffer_wh45_wh52, exit_jobs;
```

```
SEARCH, Buffer_hu03_hu08: jobID==check_wh03_wh44_ID;
REMOVE: J, Buffer_hu03_hu08, exit_jobs:
NEXT(continue_wh03_wh44);

check_wh45_wh52_queue_5       SEARCH, Buffer_wh03_wh44: jobID==check_09_12_ID;
REMOVE: J, Buffer_wh03_wh44, exit_jobs:
NEXT(continue_wh45_wh52);


check_w45_wh52_queue_13  SEARCH, Buffer_wh03_wh44: jobID==check_09_12_ID;
REMOVE: J, Buffer_wh03_wh44, exit_jobs;
SEARCH, Buffer_wh45_wh52: jobID==check_09_12_ID;
REMOVE: J, Buffer_wh45_wh52, exit_jobs;
SEARCH, Buffer_hu03_hu08: jobID==check_09_12_ID;
REMOVE: J, Buffer_hu03_hu08, exit_jobs:
NEXT(continue_wh45_wh52);


check_ty01_ty60_queue_11 SEARCH, Buffer_ty61_ty72: jobID==check_ty01_ty60_ID;
REMOVE: J, Buffer_ty61_ty72, exit_jobs:
NEXT(continue_ty01_ty60);

check_ty61_ty72_queue_11 SEARCH, Buffer_ty01_ty60: jobID==check_ty61_ty72_ID;
REMOVE: J, Buffer_ty01_ty60, exit_jobs:
NEXT(continue_ty61_ty72);

check_va01_va10_queue_6 SEARCH, Buffer_vx03_vx28: jobID==check_va01_va10_ID;
REMOVE: J, Buffer_vx03_vx28, exit_jobs;

SEARCH, Buffer_vx29_vx36: jobID==check_va01_va10_ID;
REMOVE: J, Buffer_vx29_vx36, exit_jobs:
NEXT(continue_va01_va10);

check_vx03_vx28_queue_6 SEARCH, Buffer_va01_va10: jobID==check_vx03_vx28_ID;
REMOVE: J, Buffer_va01_va10, exit_jobs;

SEARCH, Buffer_vx29_vx36: jobID==check_vx03_vx28_ID;
REMOVE: J, Buffer_vx29_vx36, exit_jobs:

NEXT(continue_vx03_vx28);

check_vx29_vx36_queue_6 SEARCH, Buffer_va01_va10: jobID==check_vx29_vx36_ID;
REMOVE: J, Buffer_va01_va10, exit_jobs;

SEARCH, Buffer_vx03_vx28: jobID==check_vx29_vx36_ID;
REMOVE: J, Buffer_vx03_vx28, exit_jobs:

NEXT(continue_vx29_vx36);


END;
```

SIMAN EXAMPLE OUTPUT: FIFO priority queues

```
SIMAN Run Processor Version 8.01.00
Copyright 1982-2004 by Rockwell Software, Inc.  All rights reserved.
Reading program file: GPU_ANIM.P
Beginning replication 1 of 100
```

ARENA Simulation Results
Information Technology

Summary for Replication 1 of 100

Project: HPC Cluster Run execution date :11/30/16
Analyst: Nadia Aly Model revision date:12/ 1/16

Replication ended at time : 31536000.0 Hours
Statistics were cleared at time: 604800.0 Hours (Thursday, November 29, 2085, 00:00:00)
Statistics accumulated for time: 30931200.0 Hours
Base Time Units: Hours

TALLY VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Observations
reserved_wh01_wh02_WaitTime	0	0	0	0	0
reserved_hu01_hu02_WaitTime	186.65	(Insuf)	.00000	2749.0	55
reserved_vx01_vx02_WaitTime	139.50	(Insuf)	2.0000	805.00	6
ty01_ty60_WaitTime	8071.7	6333.9	.00000	3.5964E+05	453
ty61_ty72_WaitTime	11274.	(Insuf)	.00000	3.5964E+05	101
hu03_hu08_WaitTime	2.4177E+05	(Insuf)	2.1607E+05	2.6747E+05	2
hu09_hu12_WaitTime	43780.	(Insuf)	3948.0	2.1607E+05	18
wh03_wh44_WaitTime	49806.	(Insuf)	3948.0	3.5964E+05	92
wh45_wh52_WaitTime	1.3413E+05	(Insuf)	52196.	2.1607E+05	2
vx03_vx28_WaitTime	15464.	(Insuf)	.00000	3.5964E+05	108
vx29_vx36_WaitTime	51564.	(Insuf)	2.0000	3.5964E+05	7
va01_va10_WaitTime	13392.	(Insuf)	.00000	3.5964E+05	89

DISCRETE-CHANGE VARIABLES

Identifier	Average	Half Width	Minimum	Maximum	Final Value
Reserved wh01_wh02 Utilization	.02202	.00966	.00000	16.000	16.000
Reserved_hu01_hu02 Utilization	.03397	.01138	.00000	16.000	16.000
Reserved vx01_vx02 Utilization	.05185	.02116	.00000	24.000	24.000
ty01_ty60 Utilization	.53273	.15154	.00000	240.00	240.00
ty61_ty72 Utilization	.11876	.08049	.00000	48.000	48.000
hu03_hu08 Utilization	.21584	.29102	.00000	48.000	48.000
hu09_hu12 Utilization	.10441	.05542	.00000	32.000	32.000
wh03_wh44 Utilization	2.3364	1.6166	.00000	352.00	352.00
wh45_wh52 Utilization	.21586	.28199	.00000	64.000	64.000
vx03_vx28 Utilization	1.6252	1.2081	.00000	336.00	336.00
vx29_vx36 Utilization	.14517	.04158	.00000	96.000	96.000
va01_va10 Utilization	.40760	.20676	.00000	160.00	160.00

COUNTERS

```
Identifier                                        Count   Limit
-------------------------------------------------------------

check_queue                                          41  Infinite
```

SIMAN EXAMPLE OUTPUT: LVF priority queues

```
SIMAN Run Processor Version 8.01.00
Copyright 1982-2004 by Rockwell Software, Inc.  All rights reserved.
Reading program file: LOW.P
Beginning replication 1 of 100

                                    ARENA Simulation Results
                                     Information Technology

                                  Summary for Replication 1 of 100

Project: HPC Cluster                                                Run execution date :12/ 1/16
Analyst: Nadia Aly                                                  Model revision date:12/ 1/16

Replication ended at time      : 31536000.0 Hours
Statistics were cleared at time: 604800.0 Hours (Thursday, November 29, 2085, 00:00:00)
Statistics accumulated for time: 30931200.0 Hours
Base Time Units: Hours

                                          TALLY VARIABLES

Identifier                                Average   Half Width  Minimum    Maximum   Observations
-------------------------------------------------------------------------------------------------

reserved_wh01_wh02_WaitTime                .33333     (Insuf)    .00000     1.0000         21
reserved_hu01_hu02_WaitTime                183.33     (Insuf)    .00000     2749.0         56
reserved_vx01_vx02_WaitTime                70.250     (Insuf)    .00000     805.00         12
ty01_ty60_WaitTime                         7864.3     5615.4     .00000     3.5964E+05    465
ty61_ty72_WaitTime                         6251.9     (Insuf)    .00000     3.5964E+05    173
hu03_hu08_WaitTime                         19333.     (Insuf)    3948.0     2.2112E+05     36
hu09_hu12_WaitTime                         43780.     (Insuf)    3948.0     2.1607E+05     18
wh03_wh44_WaitTime                         49806.     (Insuf)    3948.0     3.5964E+05     92
wh45_wh52_WaitTime                         16080.     (Insuf)    3948.0     2.1607E+05     31
vx03_vx28_WaitTime                         14302.     (Insuf)    .00000     3.5964E+05    108
vx29_vx36_WaitTime                         51564.     (Insuf)    2.0000     3.5964E+05      7
va01_va10_WaitTime                         2919.3     4025.3     .00000     3.5964E+05    387

                                    DISCRETE-CHANGE VARIABLES

Identifier                                Average   Half Width  Minimum    Maximum   Final Value
-------------------------------------------------------------------------------------------------

Reserved wh01_wh02 Utilization             .02267     .01125     .00000     16.000     16.000
Reserved_hu01_hu02 Utilization             .02820     .01024     .00000     16.000     16.000
Reserved vx01_vx02 Utilization             .03965     .01553     .00000     24.000     24.000
ty01_ty60 Utilization                      .55331     .15138     .00000     240.00     240.00
ty61_ty72 Utilization                      .11118     .07962     .00000     48.000     48.000
hu03_hu08 Utilization                      .19163     .14129     .00000     48.000     48.000
hu09_hu12 Utilization                      .11189     .05884     .00000     32.000     32.000
wh03_wh44 Utilization                      2.3758     1.6058     .00000     352.00     352.00
```

```
wh45_wh52 Utilization                    .29740     .30955     .00000     64.000     64.000
vx03_vx28 Utilization                    1.5925     1.2065     .00000     336.00     336.00
vx29_vx36 Utilization                    .12748     .03619     .00000     96.000     96.000
va01_va10 Utilization                    .34899     .20744     .00000     160.00     160.00

                                          COUNTERS

Identifier                                 Count   Limit
-------------------------------------------------------------

check_queue                                   20  Infinite
```

Referenced Figures

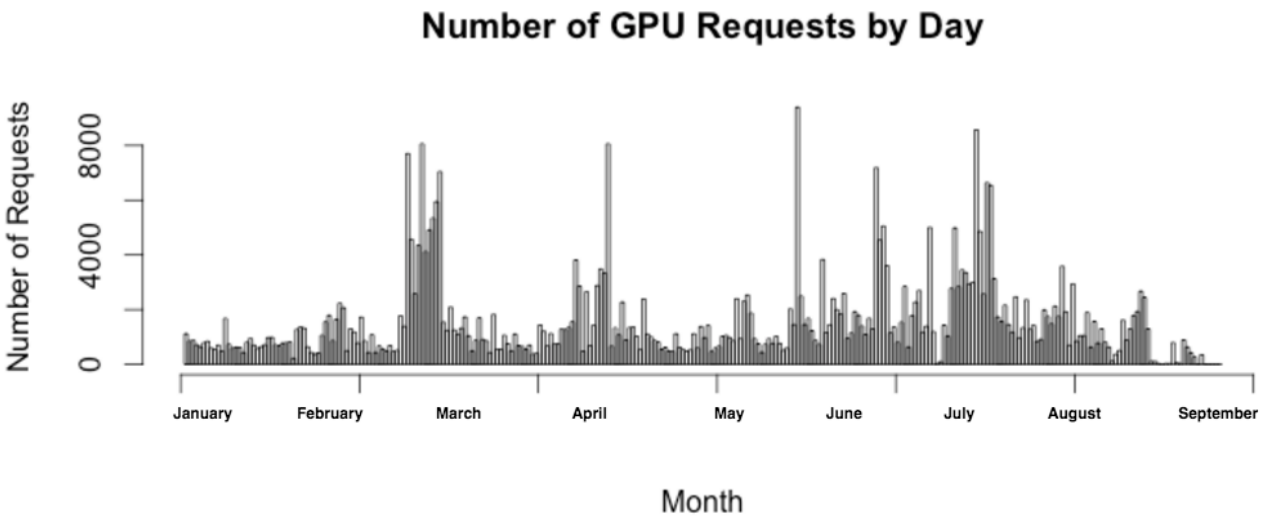


Figure 9: Histogram of total user jobs and root jobs by day and month

Command-line Code

- Search for all xeon node requests, excluding staff jobs, print the node count and the processors per node requested. This can be done similarly for each node type, replacing xeon for desired node:

```
cat * | grep -w E  | egrep -v 'root|sysops|hpcstaff' | grep xeon | awk '{print$13,$14}'
```

- Logged into Sciclone, check which nodes are compatible with node

```
pbsnodes -a | grep {node}
```