CS 671 Final Project: Effects of Varying Job Distribution Rate and Worker Node Selection in a Distributed Cluster

Pawel Szczurko

Rutgers University, New Brunswick

December 11, 2016

Abstract

Distributed clusters are at times used to process a series of tasks or jobs. This sort of job processing can be structured by having one node as the master (or producer), while the remaining nodes serve as workers that process jobs given to them. This report studies the effects node selection methods and job distribution rate have in this environment. These features are examined by creating a simulated environment and running a series of tests that exhibit varying configurations. The obtained results were used to provide a generalized argument as to which configurations should be utilized during certain workload scenarios.

1 Introduction

Distributed clusters have been studied for quite some time, but have become much more widespread within the past one to two decades. High amounts of storage and computational power can be currently acquired, but likely at the cost of high power consumption. Thus it is beneficial to find optimizations in order to minimize costs of solving large and or difficult computational challenges.

Others have studied probabalistic approaches to scheduling strategies [1], while the idea behind this report is to look into scheduling at the very basic level. In this problem, two main areas of this problem will be explored by this report:

- how should the number of jobs a worker receives vary over time
- how should a worker be selected to receive jobs

In this report, the chunk of jobs a worker receives will be studied by keeping it constant as well as by varying it in a linear and exponential fashion. The worker selection will be examined by selecting a worker node randomly, sequentially, and via the shortest queue method. In order to focus on those two aspects, the performed simulation will keep certain configurations constant throughout.

All the data and any benchmarks presented in this report were ran on the Rutgers University Elf cluster[1]. This report will present information in the following way. Section 2 presents the design of the simulated environment by going into specifics of design as well as the available configurations. Section 3 explains how the simulated environment was configured to collect results. Sections 4, 5, 6 present analysis of results. Section 7 talks about future work and Section 8 concludes the report.

2 Environment Simulation Design

In the ideal case, this experiment should be done with server instances, either physical or virtual, in order to have the true environment where these situations occur. Due to resource and time constraints, the ideal environment was not created and instead one was simulated using the Pthread library. The source

[1]Rutgers Discovery Informatics Institute Elf Cluster: `http://rdi2.rutgers.edu/elf`

code of this simulated environment can be found in the 'simulation_code' directory of the repository[2]. This section will provide insight into the architectural design of the simulation by explaining the various components and how they are used. Note that throughout this report the terms thread and node will be used interchangeably since in this simulated environment server nodes will be simulated via threads.

2.1 High Level Overview

In order to launch the simulation, a configuration file with the following properties must be provided.

- Number of jobs to process
- Execution time of a single job (job type): small, mid, large, varied
- Number of workers
- Worker queue size
- Maximum number of jobs master can distribute
- Worker selection strategy: sequential, random, shortest queue
- Job distribution strategy (job chunk distribution): constant (along with size), linear, exponential

The process execution context acts as the master thread, in that it will be responsible for distributing jobs among worker threads.

The master thread first launches n worker threads where n is specified in the provided configuration file. Each worker thread is launched with its own queue, logger, and corresponding lock mechanisms to prevent race conditions. As long as there are jobs left to distribute, the master will execute as demonstrated by Figure 1. The size of job chunk to be distributed is first determined by taking into consideration whether previous job distribution has succeeded and the job distribution strategy (i.e. constant, linear, exponential) specified in the configuration file. Then as long as the job chunk has not been assigned or the number of attempts is not equal to the number of workers n, the master will first select a worker thread to distribute a job chunk to, based on strategy (i.e. sequential, random, shortest queue) specified in configuration file, attempt to distribute job chunk to selected worker thread, and then proceed accordingly depending on the result of job chunk distribution as demonstrated by Figure 1. Since this is a threaded simulation, signals are used by the master thread in order to notify a corresponding worker thread that jobs have been added to its queue. Additionally, in a real environment network delay would have a direct effect on this workflow thus in this simulated environment network delay has been simulated by having the master thread sleep between job chunk distribution attempts for a random period of time in range 0.00 - 0.05 seconds. Once there are no more jobs left to distribute, the master will wait for the workers to finish their execution. Subsequently, master then writes all the information from loggers into a file, cleans up any allocated resources, and simulation is complete.

The logic of a worker thread is very basic in that if jobs are present in its queue, it will take one and run it and otherwise it will sleep, waiting for master thread to signal that jobs have been added. Each worker's queue has a maximum capacity specified by the configuration file in order to make this simulation comparable to the ideal environment where certain constraints might exist.

2.2 Job Types

The simulation features four type of job execution time configurations: small, mid, large, varied. Before explaining these configurations, it will be helpful to first understand how the job function works. The job function consists of allocating a chunk of memory and performing dummy calculations, where these calculations take $O(nm)$ time, n being the number of iterations and m being the size of the array. The parameter passed to the job function is subtracted from the m factor and so the bigger the value that is passed, the quicker the job will execute. Thus in order to simulate a small job, a bigger parameter is passed while a small parameter is passed to simulate a longer job. The rough execution times for the three job types on a node of the Elf cluster[3] are show in Table 1. Going back to the configuration settings, specifying small, mid, or large will perform the simulation with all jobs having the same execution time whereas specifying varied will trigger the simulation to have each job's execution time be randomly picked to be either small, mid, or large.

[2]Project source code and data repository: `https://github.com/ps/dist_job_sched_simulation`

[3]In this instance, a node on the Elf cluster is an actual server node consisting of 24 cores and 256G of memory.

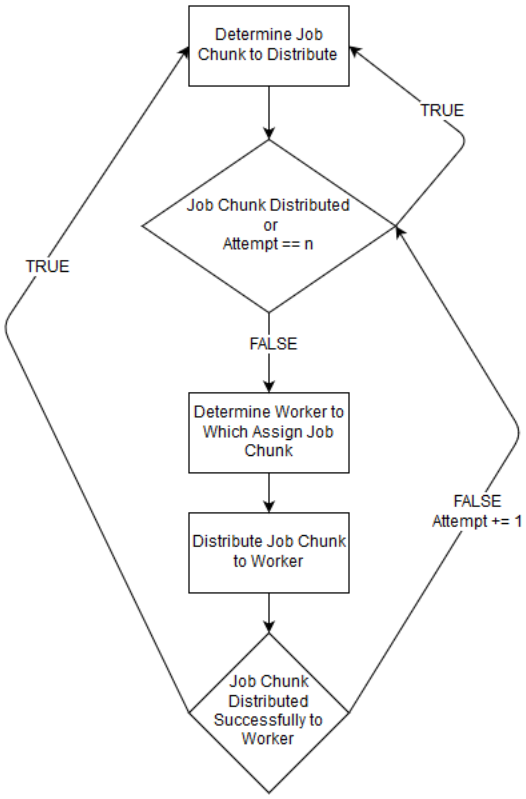


Figure 1: Main logic of master thread where n is the number of worker threads.

Job Size	SMALL	MID	LARGE
Rough Execution Time	0.7 sec	0.9 sec	1.7 sec

Table 1: Rough execution times of the three job types on Elf cluster node.

2.3 Worker Selection

As a part of the simulation, the master thread will vary the strategy it uses to determine the worker which will be given jobs. Workers will be selected sequentially, randomly, or by selecting the one with the smallest queue size. Sequential selection will work as the name implies, and when the last worker id is reached, the selection will cycle back to the beginning. Random selection will randomly pick a number in the range $[0, n)$ where n is the number of workers. Lastly, in order to determine the worker with the shortest queue, master will first sequentially look at all worker queues and pick the one with the smallest queue size. It is possible that during this iteration, queue sizes of already inspected queues may decrease, but this approach was selected so that the lock of only one queue at a time needs to be acquired. Due to the efficiency benefits, working on potentially slightly stale data did not seem detrimental.

2.4 Job Distribution

Depending on the provided configuration, job distribution strategies can be varied between constant, linear, and exponential. A constant distribution strategy means that the master will attempt to assign a constant job chunk size k every time, where k is specified in the configuration file. A linear distribution strategy will increase/decrease job chunk size in a linear matter. Specifically, job chunk size will be decreased by 1 if the previous job chunk distribution attempt failed and will be increased by 1 otherwise. In a similar fashion, exponential distribution strategy will double the job chunk size upon previous success and halve upon previous failure.

The maximum number of jobs a master can assign will be specified in the configuration file. If the linear or exponential increase exceeds this maximum limit, the job chunk size will be adjusted to match this maximum limit. On the lower spectrum, a job chunk size cannot be lower than 1. Additionally, it is logical that the maximum number of jobs that can be assigned by master should not exceed the maximum worker queue size for obvious reasons.

Upon simulation start, for linear and exponential methods job chunk size will begin at the maximum limit a master can distribute as specified in the configuration file. The reason for this is that the simulation should seek to minimize communication overhead if possible since the smaller the communication overhead the less time processing of all jobs will take.

2.5 Worker Queue

Each worker has a queue into which the master can add jobs. This queue has a maximum capacity in order to simulate a real environment. Queues are protected by mutex locks in order to avoid race conditions. The main operations on this queue implementation are add and remove. The master only utilizes the add method while the workers only utilize the remove method. Note that when adding multiple jobs, master will call the add method that accepts a list of jobs while worker remove method will only remove one job at a time.

2.6 Logging

The master and all workers all have their own separate loggers into which log messages can be inserted. A log message consists of a timestamp, message ID, and integer sized data. The log message IDs used are listed below.

- 444: Message stores job chunk size master thread is using at given timestamp.
- 555: Message stores timestamp when a thread began its processing cycle.
- 666: Message stores timestamp when a thread ended its processing cycle.
- 777: Message stores worker queue size at given timestamp.
- 999: Message stores the number jobs remaining to be distributed at given timestamp.

The master thread logs messages with IDs 555 and 666 as implied by the message descriptions and it also logs messages with IDs 777 and 999 right after the first step show in Figure 1 cycle.

3 Data Collection

Since this simulation has been written with threads, which for the most part can be considered equal in terms of resources and power, the results and conclusions presented in this report are meant to be

Configuration	Settings Studied
Worker node selection strategy	Sequential, Random, Shortest Queue
Job chunk distribution strategy	Constant ($K = 10$), Linear, Exponential
Number of workers	5, 10, 20, 30
Execution time of single job (job type)	small, mid, large, varied

Table 2: Configurations that varied during data collection.

Configuration	Setting Used
Maximum worker queue size	50
Maximum job chunk size distributed by master	30
Constant distribution strategy job chunk size	10
Number of jobs	1000

Table 3: Configurations that were kept constant during data collection.

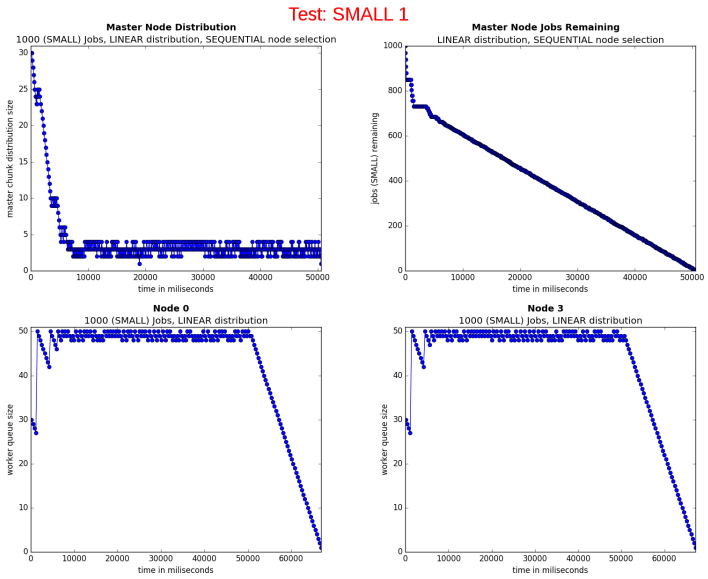
extended to a distributed cluster of server nodes where all nodes are also equal in terms of resources and power. Any of the data not shown in the report can be found in the project repository mentioned earlier, it contains all of the data that was collected along with plots that were generated from the data.

The environment simulation described in the previous section executes one configuration. In order to analyze the effects worker node selection and job chunk distribution strategies have, various configurations need to be studied. Since there are 3 possible worker node selection strategies and 3 possible job chunk distribution strategies, that already creates at the very minimum 9 configurations that have to be ran. It is also of value to see how configurations perform when the number of workers is increased hence configurations were ran with 5, 10, 20, and 30 workers. Thus in total there were 36 configurations that were ran for one job type. These configurations were ran first for 1000 small jobs, 1000 mid jobs, 1000 large jobs, and 1000 varied jobs thus in total 144 configurations were ran. The configurations that varied along with their variations are show in Table 2.

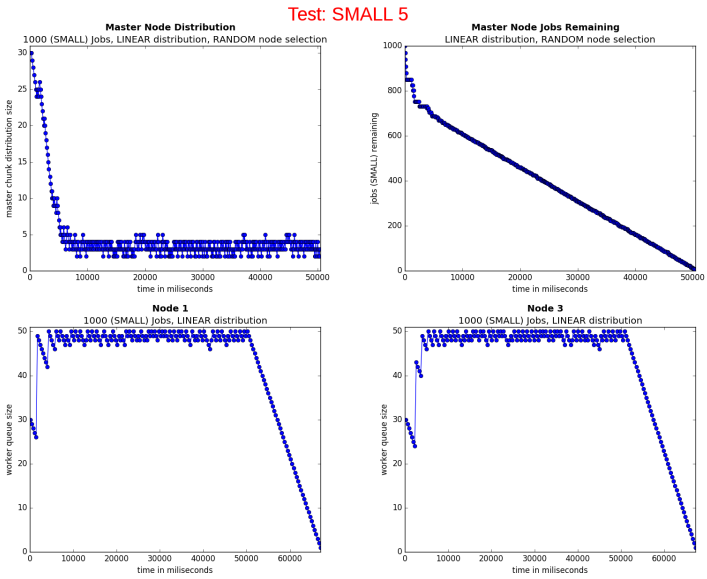
Since the focus of this simulation was on two properties, worker node selection and job chunk distribution, other configurations were kept constant throughout. The maximum worker queue size was arbitrarily set to be 50, while the maximum job chunk size was set to be 30 as it is a bit more than half of the maximum worker queue size and it is very close to the number of cores available on an Elf cluster node thus it was expected to present interesting results. The constant job chunk size was set to be 10 arbitrarily. Additionally, the number of jobs to process was kept at a constant number of 1000 as it was large enough to get some meaningful data and also took a roughly reasonable time to execute[4] . Table 3 shows a summary of these configurations.

Additionally, since this environment processed jobs via threads, it was useful to have a comparison of how long it would have taken to process the same workload sequentially. In order to make the data collection more efficient, the sequential processing was not actually performed and instead was simulated in the following way. If the task called for processing 1000 small jobs, the time it takes to execute 1 small job was obtained and multiplied by 1000.

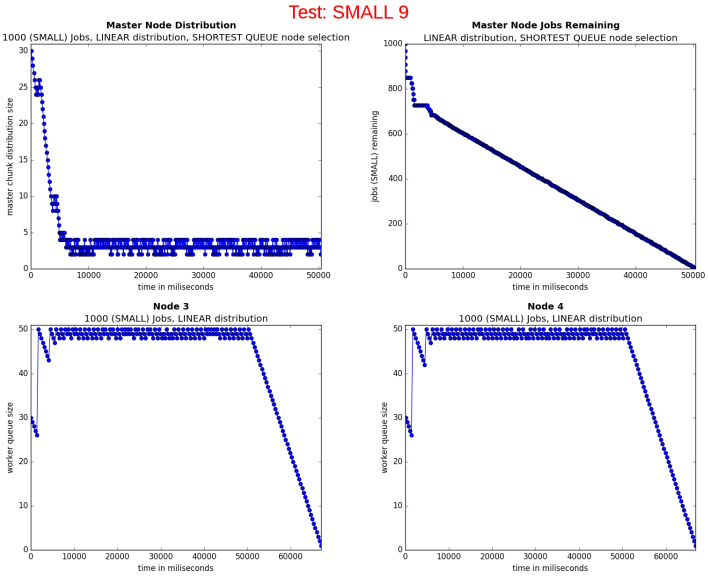
[4]Depending on the job type, processing 1000 jobs in the simulated environment took between 25 - 270 seconds.



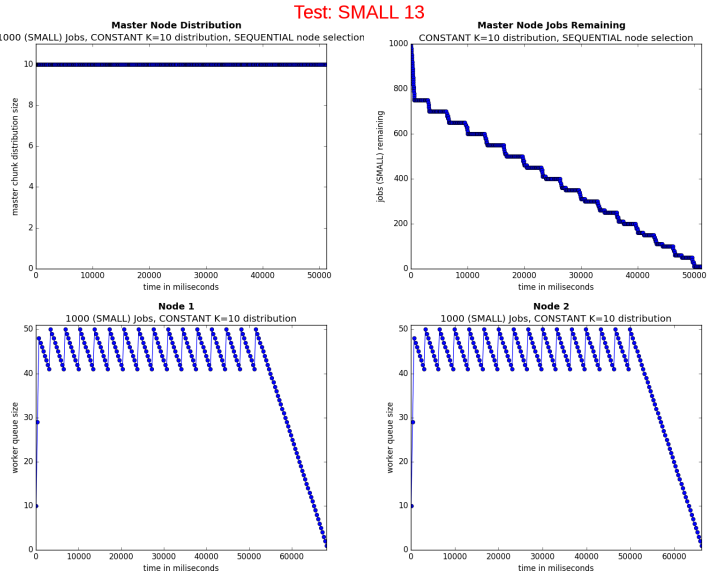
(a) Sequential node selection, Linear distribution



(b) Random node selection, Linear distribution

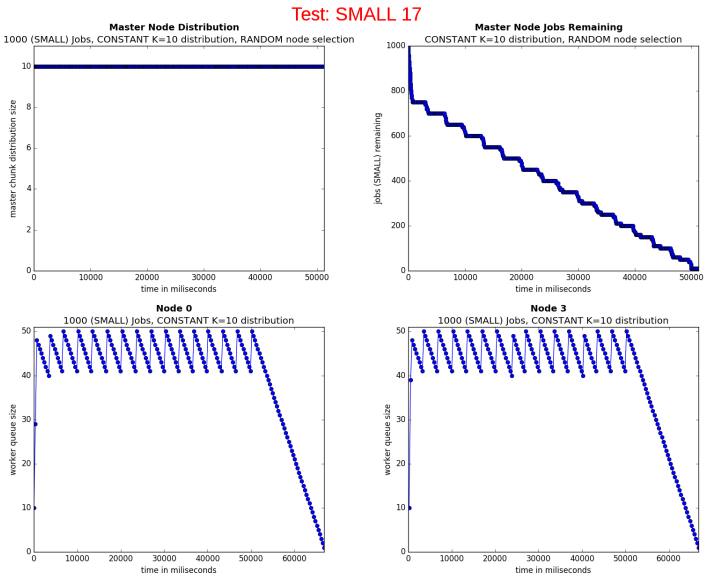


(c) Shortest Queue node selection, Linear distribution

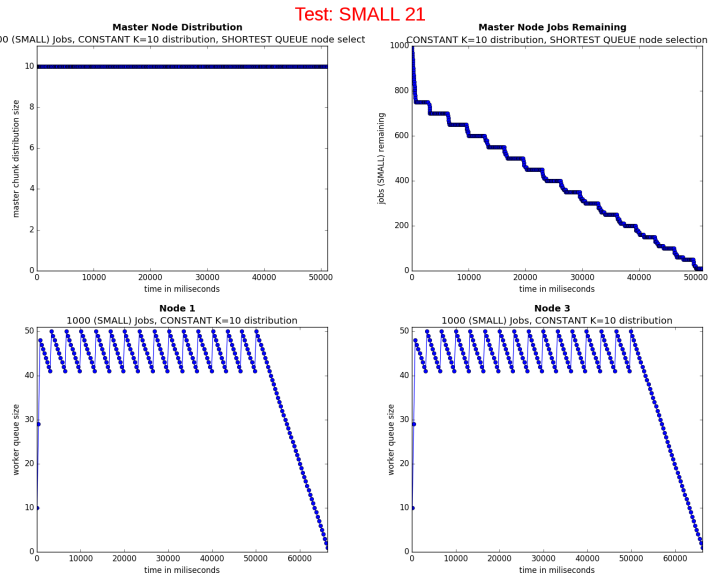


(d) Sequential node selection, Constant distribution

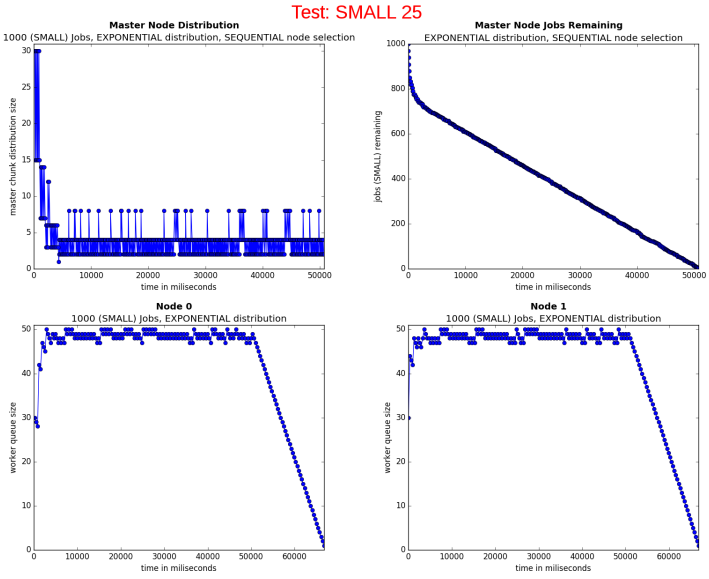
Figure 2: Overview of simulation with 1000 small jobs and 5 worker nodes part 1. These figures plot results for the test numbers 1, 5, 9, 13 as labeled at the top of each figure. The top left plot in each figure shows the master job distribution rate as time progresses. The top right plot in each figure shows the number of jobs remaining to distribute as time progresses. The bottom plots in each figure show queue sizes of two randomly selected workers.



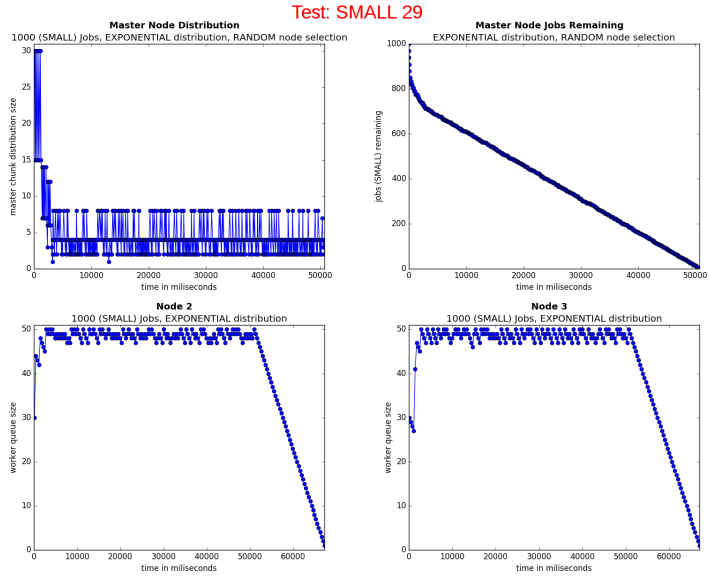
(a) Random node selection, Constant distribution



(b) Shortest queue node selection, Constant distribution

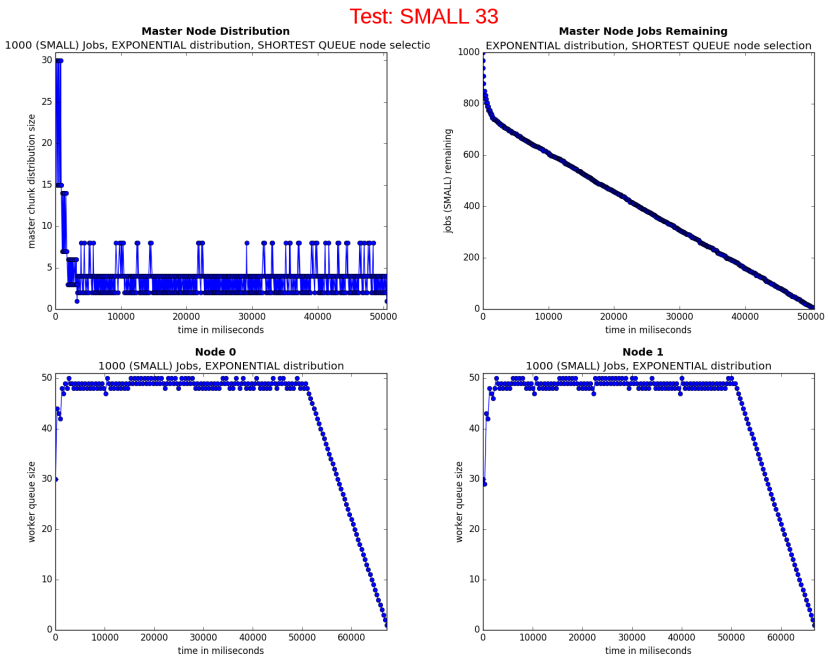


(c) Sequential node selection, Exponential distribution



(d) Random node selection, Exponential distribution

Figure 3: Overview of simulation with 1000 small jobs and 5 worker nodes part 2. These figures plot results for the test numbers 17, 21, 25, 29 as labeled at the top of each figure. The top left plot in each figure shows the master job distribution rate as time progresses. The top right plot in each figure shows the number of jobs remaining to distribute as time progresses. The bottom plots in each figure show queue sizes of two randomly selected workers.



(a) Shortest queue node selection, Exponential distribution

Figure 4: Overview of simulation with 1000 small jobs and 5 worker nodes part 3. This figure plot results for the small test numbers 33 as labeled at the top of each figure. The top left plot in each figure shows the master job distribution rate as time progresses. The top right plot in each figure shows the number of jobs remaining to distribute as time progresses. The bottom plots in each figure show queue sizes of two randomly selected workers.

4 Results: 1000 Small Jobs

Table 4 shows all the test configurations for 1000 small jobs. The last two columns specify the execution time as well as the estimated sequential execution time. Figures 2, 3, 4 show an overview of results where 5 worker nodes were used.

When there are more workers for each of the job distribution strategies, as shown in Figures 5, 6, 7, master clearly takes less time to execute, since it is quicker to distribute jobs when there are more workers present to receive jobs.

4.1 Varying Node Selection

In order to determine the effects of node selection, the tests where job distribution remained unchanged were looked at. In particular, Figures 2a, 2b, 2c are tests where job distribution remains linear. The two bottom plots of each figure, which are the node plots, do not seem to have much variation. The same goes for Figures 2d, 3a, 3b, where job distribution is constant and Figures 3c, 3d, 4a where job distribution is exponential. In order to further dive deep into the results, the plots with higher worker counts (not shown in the report) were analyzed and also did not show any variation. The reason for the lack of impact on the plots could be due to the fact that this is a simulation of the ideal environment.

The other piece of data to inspect besides the plots is the timing of these configurations. Test cases 1 through 12 in Table 4 represent linear job distribution but varying node selection. Looking at the tests where node selection differs yet the number of workers is the same, such as tests 2, 5, 10, it is evident that random node selection performs around 0.5 second worse than sequential and shortest queue distributions. Sequential and shortest queue strategies tend to perform similarly. The same can be noticed when looking at tests 3, 6, 11 and 4, 7, 12 and when repeating this inspection on tests 13-24 and 25-36. The reason sequential and shortest queue node selections perform better is due to the fact that random node selection has a higher chance of selecting the same node as previously selected, which becomes detrimental when worker queues are at maximum capacity.

4.2 Varying Job Distribution Strategy

Figure 6 shows how the linear job chunk distribution strategy performs as the number of workers grow on 1000 small jobs. As pointed out in Table 4, the tests 1, 2, 3, 4 which are shown in Figure 6 start at taking around 67 seconds and shrink down to around 22 seconds.

Figure 5 shows how the exponential job chunk distribution strategy performs as the number of workers grow on 1000 small jobs. As pointed out in Table 4, the tests 25, 26, 27, 28 which are shown in Figure 5 start at taking around 67 seconds and shrink down to around 23 seconds.

Figure 7 shows how the constant job chunk distribution strategy performs as the number of workers grow on 1000 small jobs. As pointed out in Table 4, the tests 13, 14, 15, 16 which are shown in Figure 7 start at taking around 68 seconds and shrink down to around 18 seconds.

Looking at Table 4 overall, it seems that as far as execution time goes, constant strategy performs the best out of all distribution strategies. This may be due to the case that with the constant method when there are a larger number of workers available the worker queue sizes will be increased by the constant job chunk size of 10, while in linear and exponential cases the job chunk size is very likely to be at the maximum setting of 30 thus making some workers execute longer.

4.2.1 Linear Job Distribution

As the number of workers grow from Figure 6a to Figure 6d, the variation in job chunk size, as shown by the top left plot in each figure, is clearly becoming smaller. In Figure 6a the job chunk is initially around 30 jobs, but quickly drops down to 1 or 2 jobs. While in Figure 6d there is much less variation as the number of jobs being distributed stays roughly around 25 jobs. This makes sense as there are more workers to distribute jobs to, thus in the time that it takes master to distribute chunks to some workers, other workers had time to process some of their jobs.

What is also interesting to point out is that as the number of workers grow, the plots showing the number of jobs remaining become less and less smooth. In Figure 6a the top right plot shows a somewhat smooth descent while in Figure 6d the top right plot shows a smooth descent initially and then is disrupted for a little while. This implies that during those times job chunk size must have been dropping, and that is the case as shown by Figure 6d where around the time where the top right plot has a horizontal line, the top left plot is showing a drop in job chunk size.

4.2.2 Exponential Job Distribution

The two points regarding variation in job chunk size and smoothness of the plot showing number of jobs remaining also apply to the exponential distribution. What varies though is that for tests with higher number of workers, such as Figures 5c, 5d, the variation is a bit bigger than in the linear case due to the fact that exponential strategy works by doubling and halving the job chunk size.

4.2.3 Constant Job Distribution

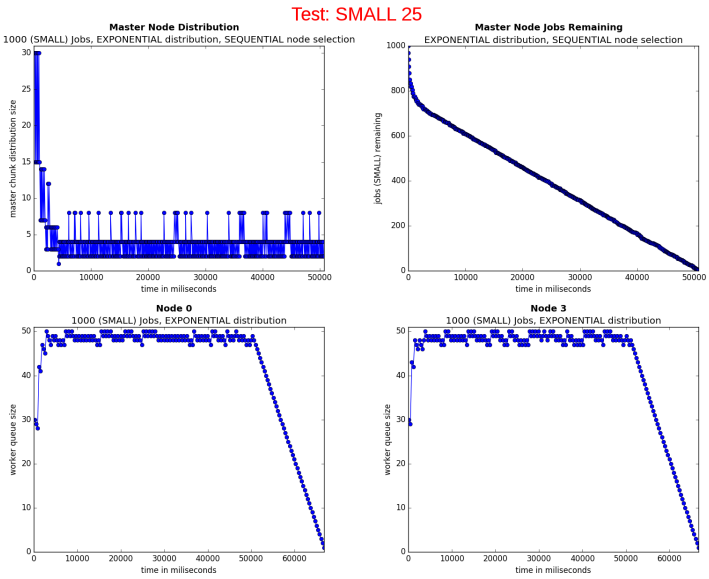
Unlike in the case of linear or exponential, the job descent occurs in a much more stepwise manner as shown in Figures 7a, 7b. In Figures 7c, 7d the bottom plots of each figure demonstrate that the worker queue sizes will only grow by the constant rate thus avoiding sudden large workload sizes as demonstrated by linear and exponential distributions.

5 Results: 1000 Mid/Large Jobs

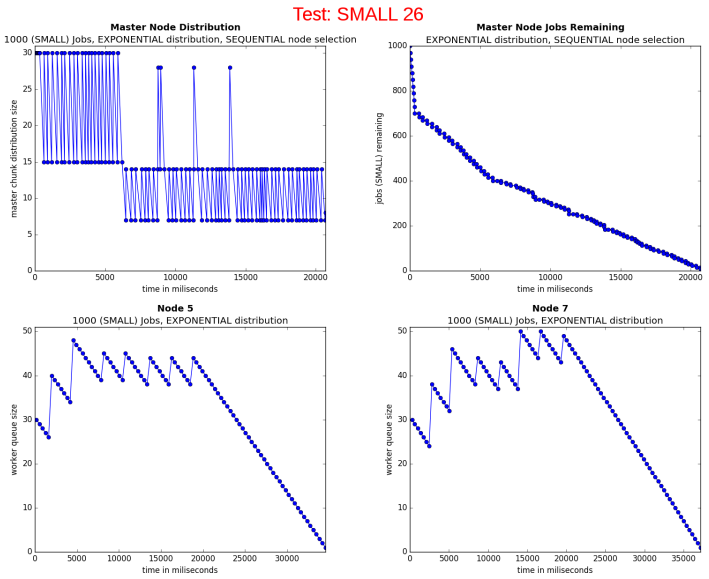
The same exact testing as shown in Table 4 was done with mid and large sized jobs. After looking through the data, the results were not much different from what was already discussed in the analysis of 1000 small jobs. Hence the data and plots have not been included in this report, but are available in the project repository for inspection.

Test #	Job Number & Size	# of Workers	Node Selection	Distribution	Execution Time (sec)	Sequential Time (sec)
1	1000 SMALL JOBS	5	SEQUENTIAL	LINEAR	67.52325	320.64
2	1000 SMALL JOBS	10	SEQUENTIAL	LINEAR	37.449792	320.663
3	1000 SMALL JOBS	20	SEQUENTIAL	LINEAR	22.213429	320.626
4	1000 SMALL JOBS	30	SEQUENTIAL	LINEAR	22.88186	320.749
5	1000 SMALL JOBS	5	RANDOM	LINEAR	67.548033	320.557
6	1000 SMALL JOBS	10	RANDOM	LINEAR	38.254254	321.306
7	1000 SMALL JOBS	20	RANDOM	LINEAR	22.682357	320.759
8	1000 SMALL JOBS	30	RANDOM	LINEAR	23.76065	321.227
9	1000 SMALL JOBS	5	SQS	LINEAR	67.556903	320.532
10	1000 SMALL JOBS	10	SQS	LINEAR	37.937056	320.745
11	1000 SMALL JOBS	20	SQS	LINEAR	22.250906	320.485
12	1000 SMALL JOBS	30	SQS	LINEAR	23.883818	321.599
13	1000 SMALL JOBS	5	SEQUENTIAL	CONSTANT (K = 10)	68.455644	320.276
14	1000 SMALL JOBS	10	SEQUENTIAL	CONSTANT (K = 10)	36.764797	320.393
15	1000 SMALL JOBS	20	SEQUENTIAL	CONSTANT (K = 10)	18.851795	321.838
16	1000 SMALL JOBS	30	SEQUENTIAL	CONSTANT (K = 10)	18.298678	320.765
17	1000 SMALL JOBS	5	RANDOM	CONSTANT (K = 10)	68.715863	320.684
18	1000 SMALL JOBS	10	RANDOM	CONSTANT (K = 10)	37.237259	321.706
19	1000 SMALL JOBS	20	RANDOM	CONSTANT (K = 10)	21.887619	320.886
20	1000 SMALL JOBS	30	RANDOM	CONSTANT (K = 10)	20.737932	320.615
21	1000 SMALL JOBS	5	SQS	CONSTANT (K = 10)	68.596678	320.484
22	1000 SMALL JOBS	10	SQS	CONSTANT (K = 10)	36.807229	320.732
23	1000 SMALL JOBS	20	SQS	CONSTANT (K = 10)	18.706516	321.487
24	1000 SMALL JOBS	30	SQS	CONSTANT (K = 10)	17.415236	320.802
25	1000 SMALL JOBS	5	SEQUENTIAL	EXP	67.439778	321.095
26	1000 SMALL JOBS	10	SEQUENTIAL	EXP	37.614362	320.896
27	1000 SMALL JOBS	20	SEQUENTIAL	EXP	22.393486	321.518
28	1000 SMALL JOBS	30	SEQUENTIAL	EXP	23.397359	321.743
29	1000 SMALL JOBS	5	RANDOM	EXP	67.554958	320.581
30	1000 SMALL JOBS	10	RANDOM	EXP	38.649991	320.255
31	1000 SMALL JOBS	20	RANDOM	EXP	22.816328	320.638
32	1000 SMALL JOBS	30	RANDOM	EXP	22.246627	320.45
33	1000 SMALL JOBS	5	SQS	EXP	67.476154	320.872
34	1000 SMALL JOBS	10	SQS	EXP	37.554753	320.53
35	1000 SMALL JOBS	20	SQS	EXP	22.29204	320.773
36	1000 SMALL JOBS	30	SQS	EXP	23.512812	320.611

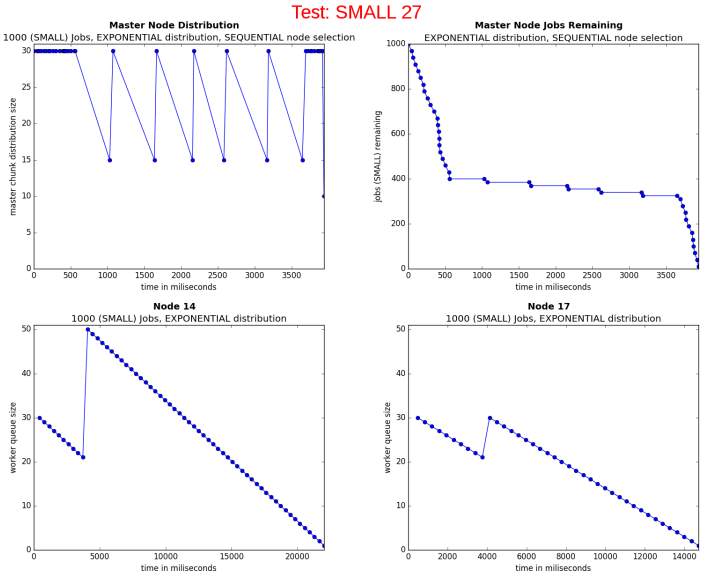
Table 4: Listing of all test cases ran on 1000 small jobs.



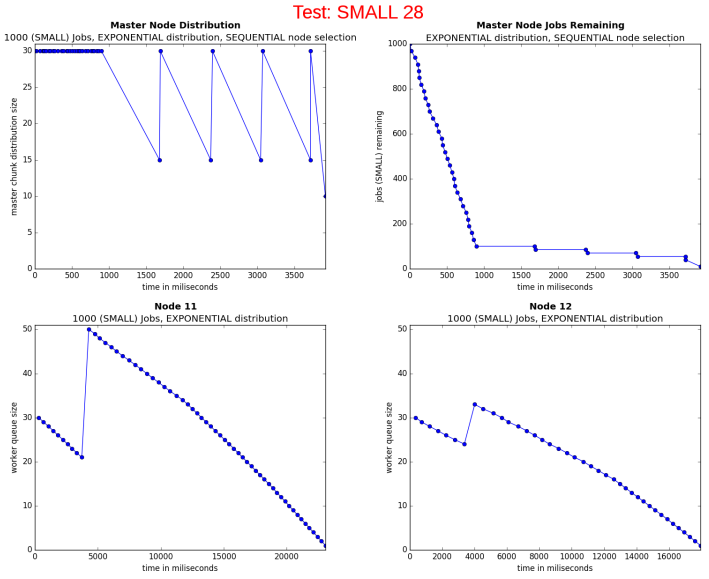
(a) 5 workers



(b) 10 workers

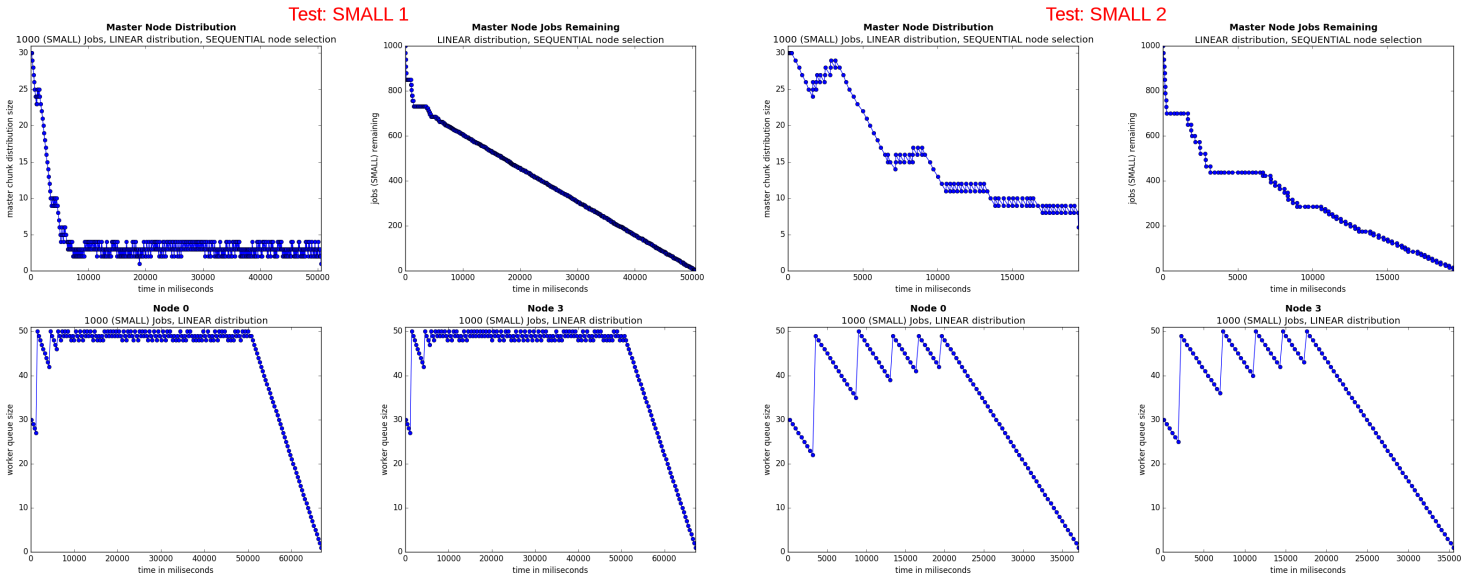


(c) 20 workers



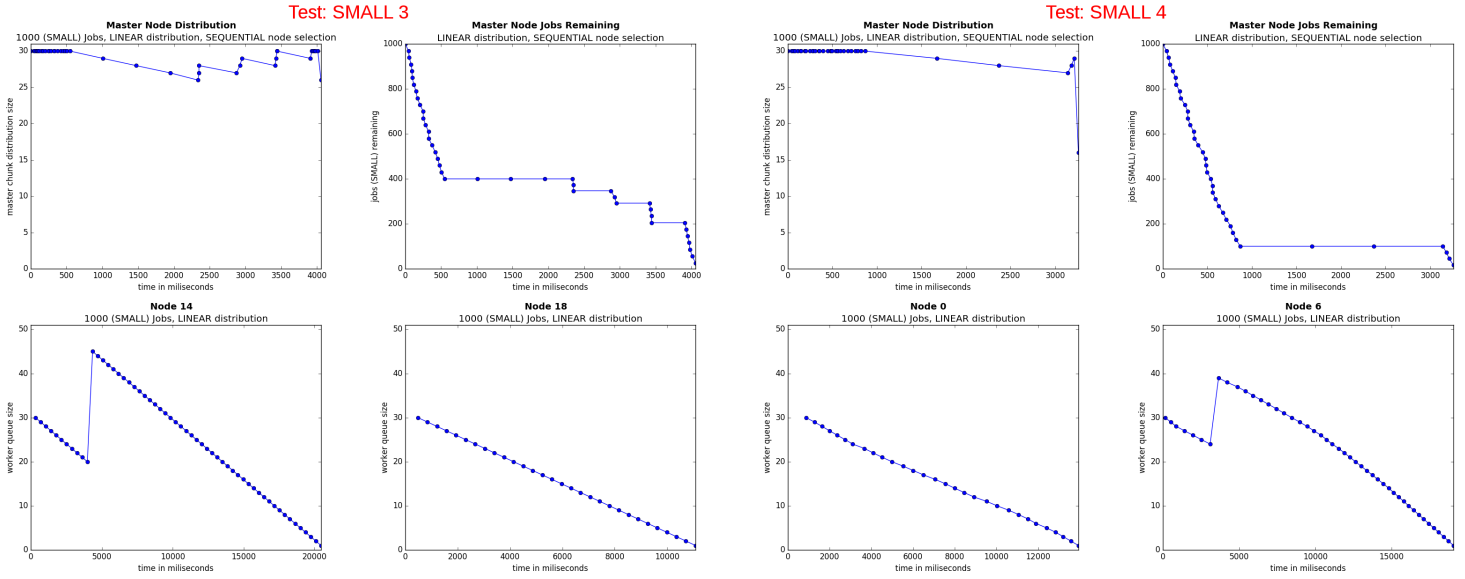
(d) 30 workers

Figure 5: These tests focus on exponential job chunk distribution with 1000 small jobs to process and varying worker size. The figures plot results for the small test numbers 25, 26, 27, 28 as labeled at the top of each figure. The top left plot in each figure shows the master job distribution rate as time progresses. The top right plot in each figure shows the number of jobs remaining to distribute as time progresses. The bottom plots in each figure show queue sizes of two randomly selected workers.



(a) 5 workers

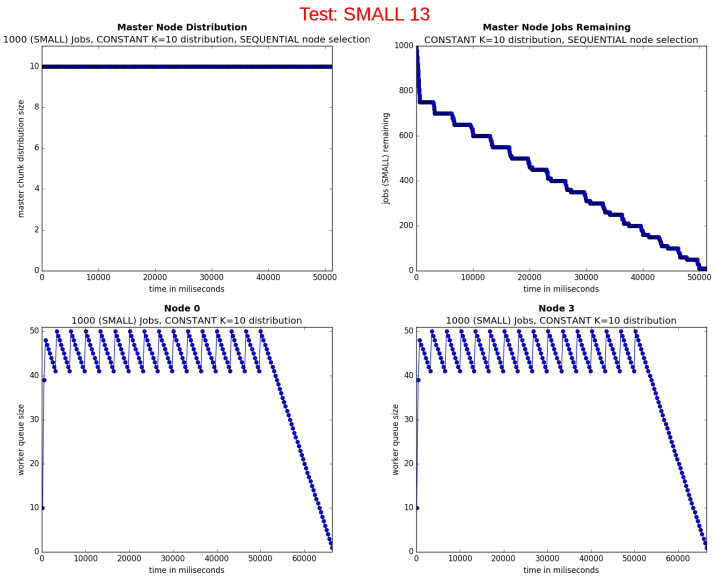
(b) 10 workers



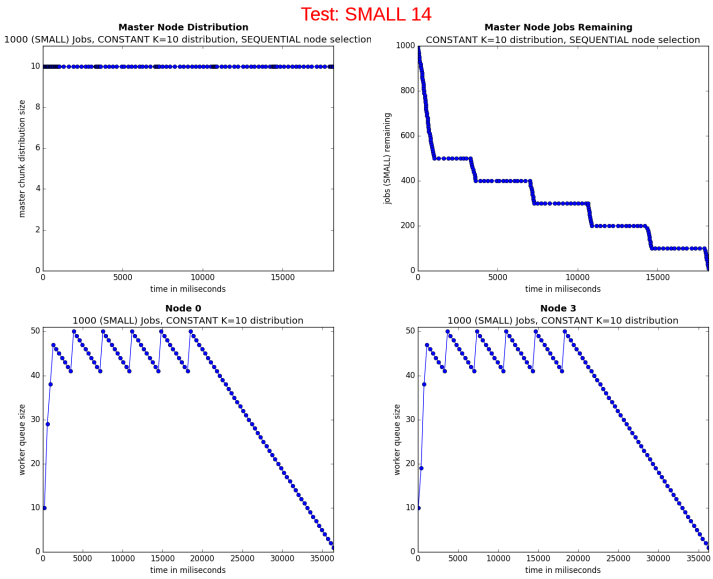
(c) 20 workers

(d) 30 workers

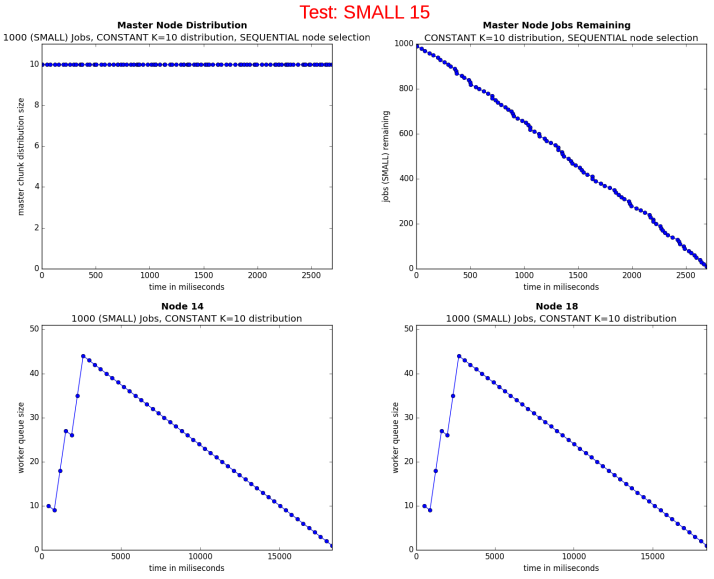
Figure 6: These tests focus on linear job chunk distribution with 1000 small jobs to process and varying worker size. The figures plot results for the small test numbers 1, 2, 3, 4 as labeled at the top of each figure. The top left plot in each figure shows the master job distribution rate as time progresses. The top right plot in each figure shows the number of jobs remaining to distribute as time progresses. The bottom plots in each figure show queue sizes of two randomly selected workers.



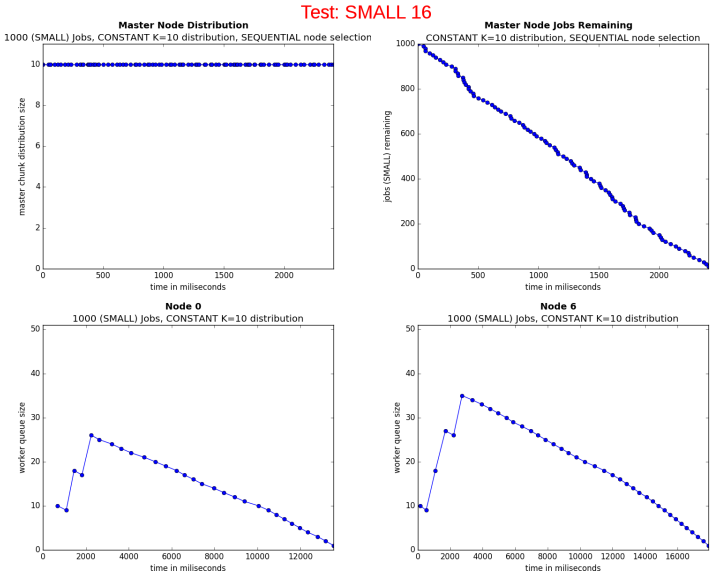
(a) 5 workers



(b) 10 workers



(c) 20 workers



(d) 30 workers

Figure 7: These tests focus on constant job chunk distribution with 1000 small jobs to process and varying worker size. The figures plot results for the small test numbers 13, 14, 15, 16 as labeled at the top of each figure. The top left plot in each figure shows the master job distribution rate as time progresses. The top right plot in each figure shows the number of jobs remaining to distribute as time progresses. The bottom plots in each figure show queue sizes of two randomly selected workers.

6 Results: 1000 Varied Jobs

Table 5 shows all the test configurations for 1000 varied jobs. The last two columns specify the execution time as well as the estimated sequential execution time. Figures 8, 9, 10 show an overview of results where 5 worker nodes were used.

6.1 Varying Node Selection

Similarly as in small job type, node selection does not seem to have much of an effect on the plots. This can be inspected in Figures 8a, 8b, 8c where job distribution is linear while node selection is varied. Although, the same conclusion can be reached regarding random node selection performing worse than others when inspecting Table 5. In this instance, random performs equal or on average about 1.5 seconds worse than the other two strategies.

6.2 Varying Job Distribution Strategy

Similarly, as in Section 6.2, inspecting the time in Table 5 shows that constant distribution strategy performs around 1 second better than the other two strategies.

Additionally, as the number of workers grow, the job chunk size variation does not decrease as much as it did with 1000 small jobs. For instance in Figure 11b job chunk size goes from around 30 jobs to around 5-4 jobs. While in the small jobs case Figure 6b goes from around 30 jobs to around 10 jobs. This means that for varied workloads, linear and exponential distribution strategies are more likely to spread the job load evenly when the number of workers is small.

6.2.1 Linear Job Distribution

In general patterns occurring in Section 4.2.1 for workloads composed of small jobs subjected to a linear job distribution also occurred for workloads composed of varied jobs subjected to a linear job distribution. This can be seen in Figure 11.

6.2.2 Exponential Job Distribution

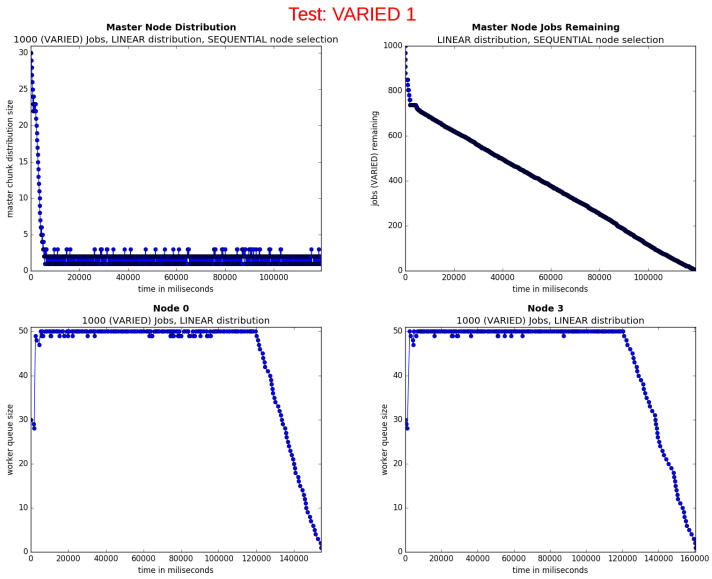
It is interesting to note that the master job chunk size is much less likely to return to the maximum value in the varied case. In Figure 5b the rate can be observed on two occasions to nearly reach the 30 maximum job chunk size, while in Figure 12b the job chunk size after a certain time does not go beyond 15.

6.2.3 Constant Job Distribution

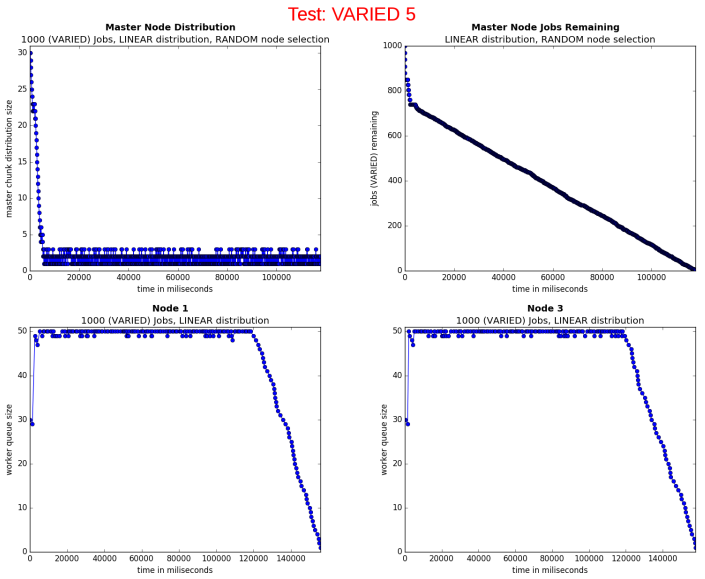
In the varied case, the plots in top right of Figures 13c, 13d showing the number of jobs left, are not displaying as much of a stepwise descent as they did in the case of small jobs. The variation in jobs must have allowed the master node to distribute a bit better.

Test #	Job Number & Size	# of Workers	Node Selection	Distribution	Execution Time (sec)	Sequential Time (sec)
1	1000 VARIED JOBS	5	SEQUENTIAL	LINEAR	161.844156	754.053213
2	1000 VARIED JOBS	10	SEQUENTIAL	LINEAR	89.742808	738.409125
3	1000 VARIED JOBS	20	SEQUENTIAL	LINEAR	53.982696	736.704731
4	1000 VARIED JOBS	30	SEQUENTIAL	LINEAR	54.338283	751.345197
5	1000 VARIED JOBS	5	RANDOM	LINEAR	158.770141	753.83557
6	1000 VARIED JOBS	10	RANDOM	LINEAR	91.350818	738.928114
7	1000 VARIED JOBS	20	RANDOM	LINEAR	53.600079	747.684029
8	1000 VARIED JOBS	30	RANDOM	LINEAR	51.587202	753.658543
9	1000 VARIED JOBS	5	SQS	LINEAR	158.538569	743.892494
10	1000 VARIED JOBS	10	SQS	LINEAR	89.631225	732.140331
11	1000 VARIED JOBS	20	SQS	LINEAR	55.47384	755.899796
12	1000 VARIED JOBS	30	SQS	LINEAR	48.615173	754.232638
13	1000 VARIED JOBS	5	SEQUENTIAL	CONSTANT (K = 10)	159.512476	745.68618
14	1000 VARIED JOBS	10	SEQUENTIAL	CONSTANT (K = 10)	91.318658	747.735918
15	1000 VARIED JOBS	20	SEQUENTIAL	CONSTANT (K = 10)	46.774289	749.278662
16	1000 VARIED JOBS	30	SEQUENTIAL	CONSTANT (K = 10)	40.697084	731.82836
17	1000 VARIED JOBS	5	RANDOM	CONSTANT (K = 10)	162.045801	755.382666
18	1000 VARIED JOBS	10	RANDOM	CONSTANT (K = 10)	93.745082	753.163448
19	1000 VARIED JOBS	20	RANDOM	CONSTANT (K = 10)	51.682266	780.054384
20	1000 VARIED JOBS	30	RANDOM	CONSTANT (K = 10)	49.178355	738.993456
21	1000 VARIED JOBS	5	SQS	CONSTANT (K = 10)	155.964292	733.491056
22	1000 VARIED JOBS	10	SQS	CONSTANT (K = 10)	89.77435	736.275724
23	1000 VARIED JOBS	20	SQS	CONSTANT (K = 10)	48.082729	736.751697
24	1000 VARIED JOBS	30	SQS	CONSTANT (K = 10)	41.264443	745.566809
25	1000 VARIED JOBS	5	SEQUENTIAL	EXP	155.947948	737.656006
26	1000 VARIED JOBS	10	SEQUENTIAL	EXP	87.764681	741.957524
27	1000 VARIED JOBS	20	SEQUENTIAL	EXP	56.013402	739.046668
28	1000 VARIED JOBS	30	SEQUENTIAL	EXP	48.870783	738.489488
29	1000 VARIED JOBS	5	RANDOM	EXP	162.617345	743.977726
30	1000 VARIED JOBS	10	RANDOM	EXP	91.397033	746.30314
31	1000 VARIED JOBS	20	RANDOM	EXP	55.964803	742.597571
32	1000 VARIED JOBS	30	RANDOM	EXP	50.481499	762.706475
33	1000 VARIED JOBS	5	SQS	EXP	161.504917	745.572964
34	1000 VARIED JOBS	10	SQS	EXP	91.471311	764.73493
35	1000 VARIED JOBS	20	SQS	EXP	51.42736	745.500791
36	1000 VARIED JOBS	30	SQS	EXP	43.326807	738.88087

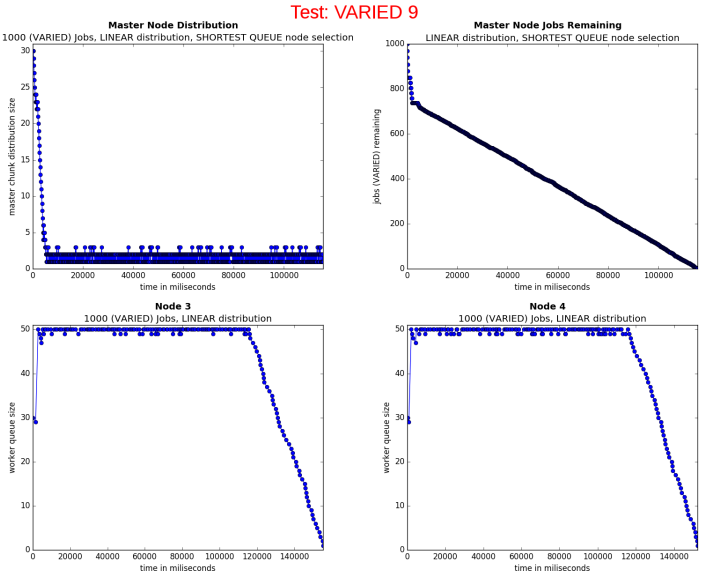
Table 5: Listing of all test cases ran on 1000 varied jobs.



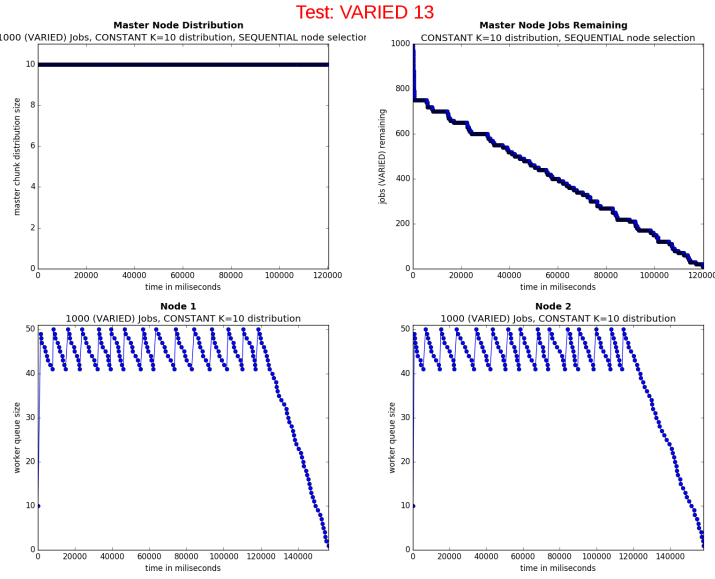
(a) Sequential node selection, Linear distribution



(b) Random node selection, Linear distribution

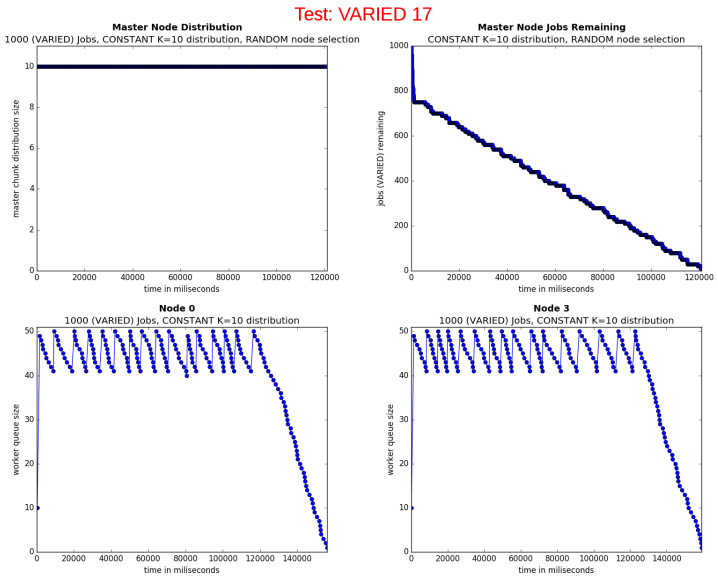


(c) Shortest Queue node selection, Linear distribution

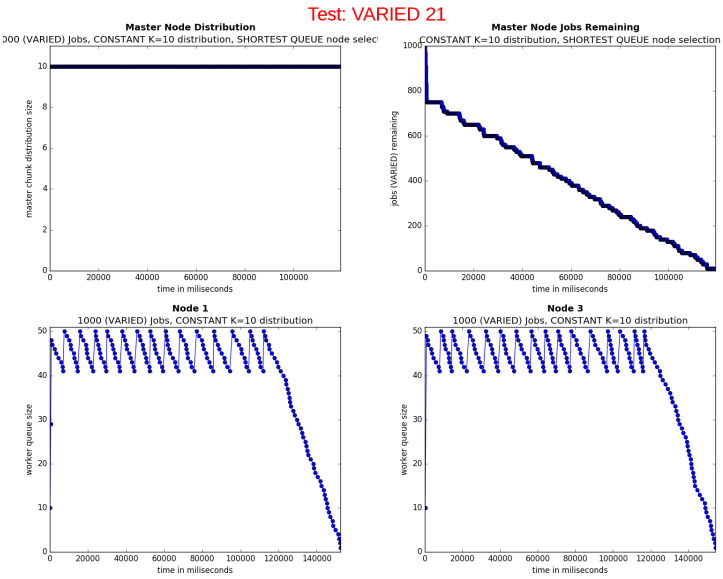


(d) Sequential node selection, Constant distribution

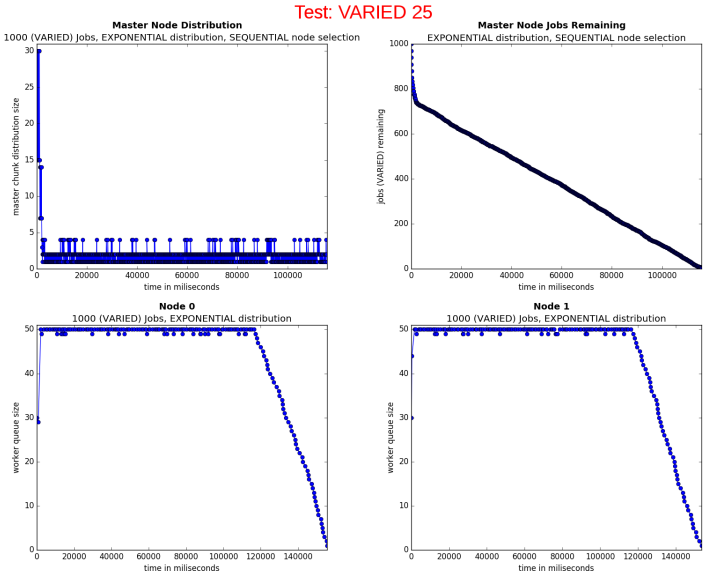
Figure 8: Overview of simulation with 1000 varied jobs and 5 worker nodes part 1. These figures plot results for the varied test numbers 1, 5, 9, 13 as labeled at the top of each figure. The top left plot in each figure shows the master job distribution rate as time progresses. The top right plot in each figure shows the number of jobs remaining to distribute as time progresses. The bottom plots in each figure show queue sizes of two randomly selected workers.



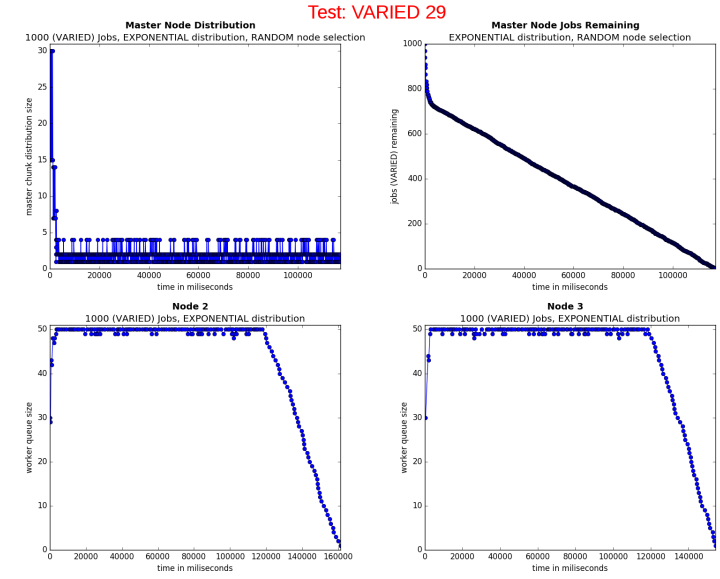
(a) Random node selection, Constant distribution



(b) Shortest queue node selection, Constant distribution

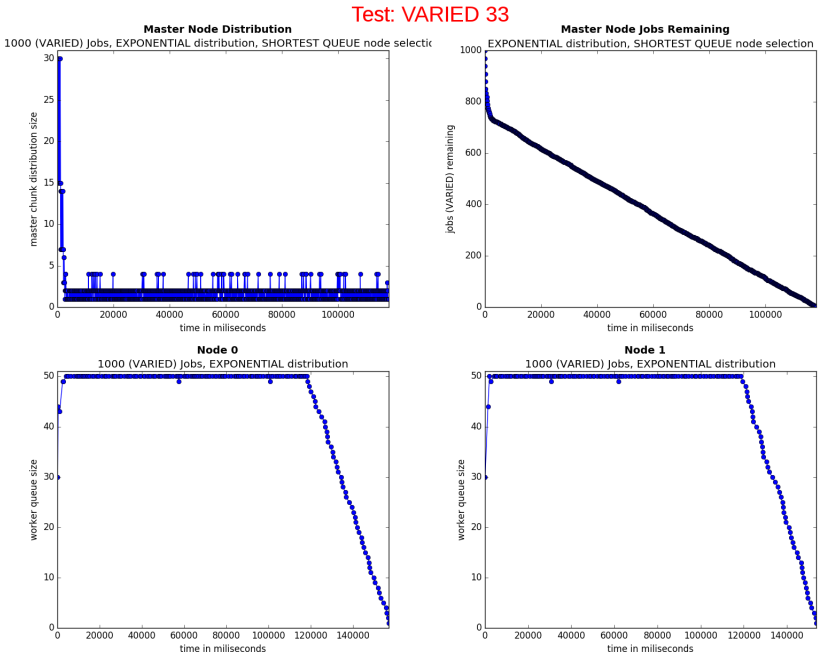


(c) Sequential node selection, Exponential distribution



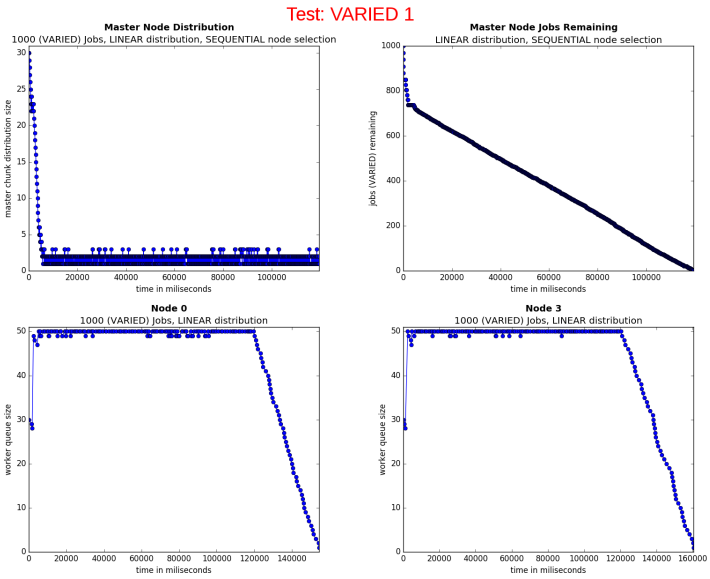
(d) Random node selection, Exponential distribution

Figure 9: Overview of simulation with 1000 varied jobs and 5 worker nodes part 2. These figures plot results for the varied test numbers 17, 21, 25, 29 as labeled at the top of each figure. The top left plot in each figure shows the master job distribution rate as time progresses. The top right plot in each figure shows the number of jobs remaining to distribute as time progresses. The bottom plots in each figure show queue sizes of two randomly selected workers.

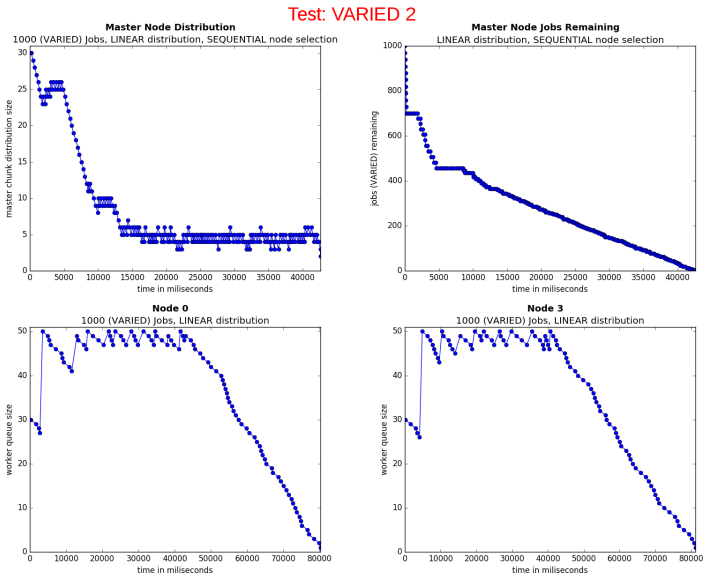


(a) Shortest queue node selection, Exponential distribution

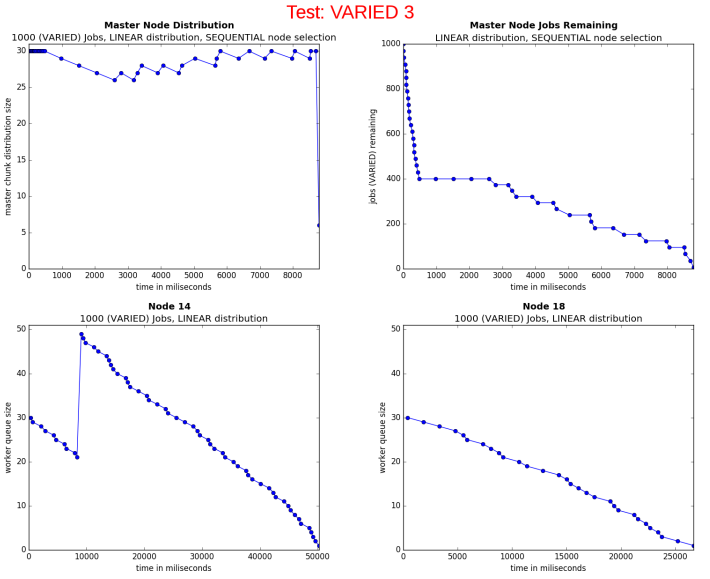
Figure 10: Overview of simulation with 1000 varied jobs and 5 worker nodes part 3. This figure plot results for the varied test numbers 33 as labeled at the top of each figure. The top left plot in each figure shows the master job distribution rate as time progresses. The top right plot in each figure shows the number of jobs remaining to distribute as time progresses. The bottom plots in each figure show queue sizes of two randomly selected workers.



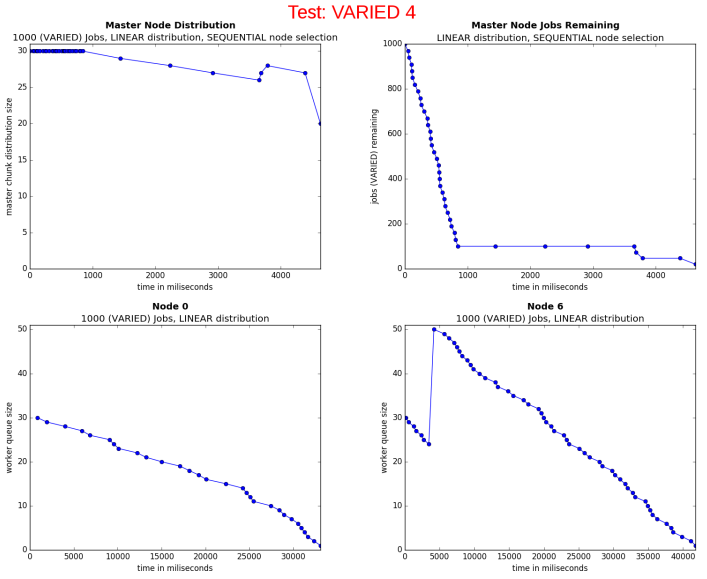
(a) 5 workers



(b) 10 workers

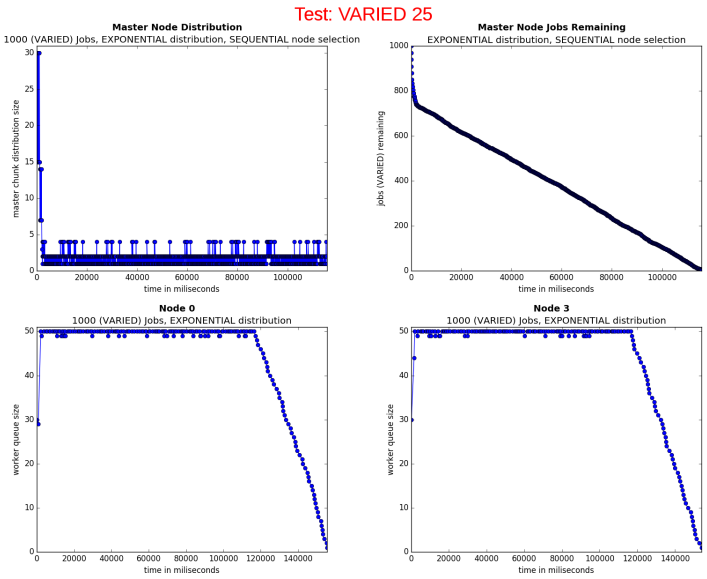


(c) 20 workers

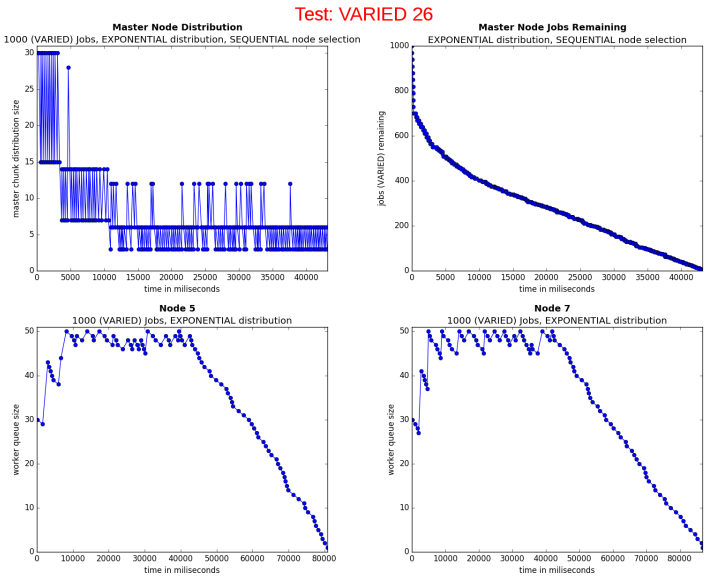


(d) 30 workers

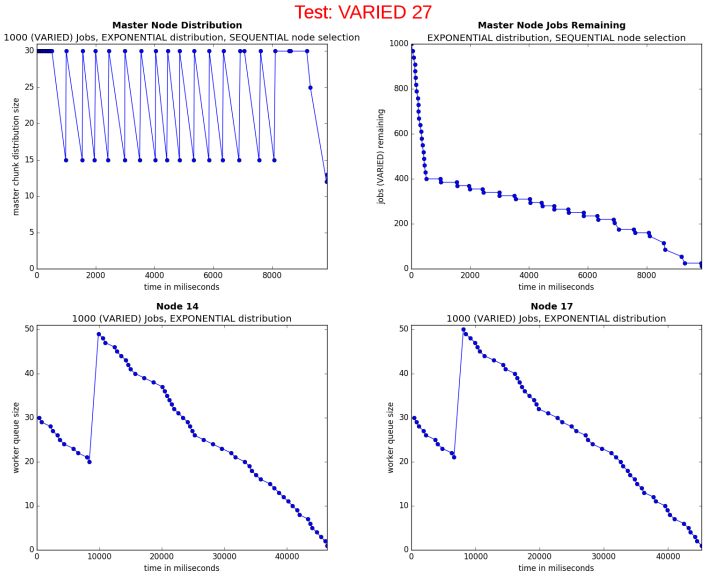
Figure 11: These tests focus on linear job chunk distribution with 1000 varied jobs to process and varying worker size. These figures plot results for the varied test numbers 1, 2, 3, 4 as labeled at the top of each figure. The top left plot in each figure shows the master job distribution rate as time progresses. The top right plot in each figure shows the number of jobs remaining to distribute as time progresses. The bottom plots in each figure show queue sizes of two randomly selected workers.



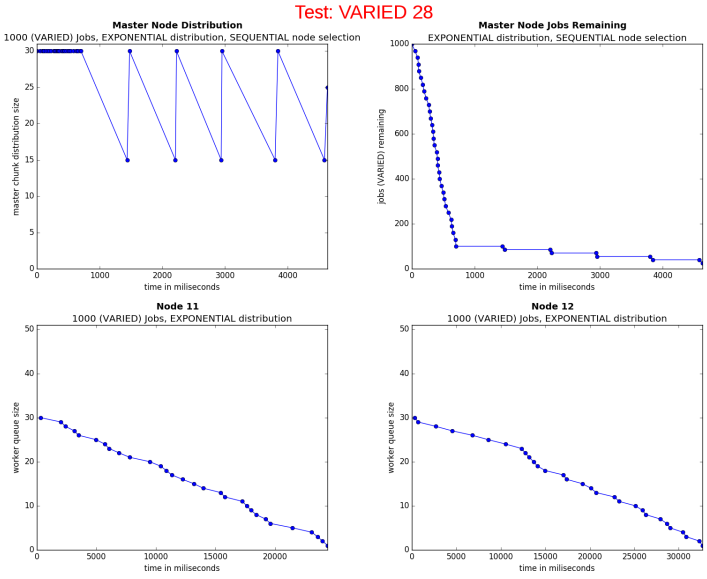
(a) 5 workers



(b) 10 workers

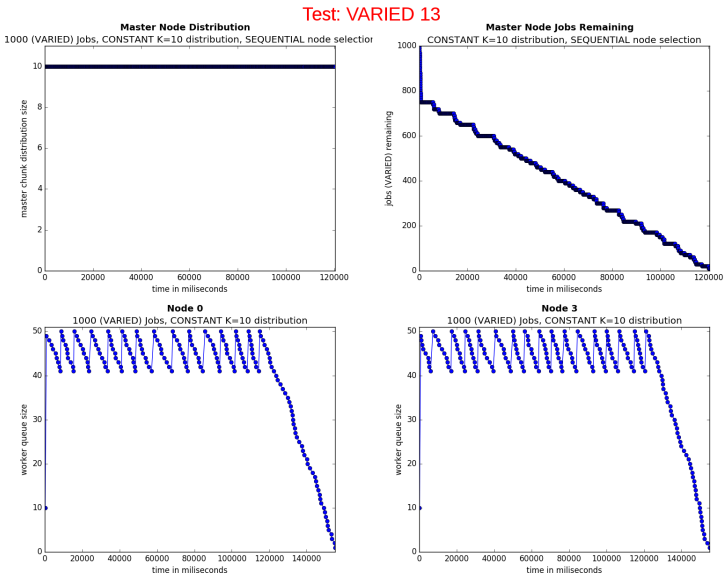


(c) 20 workers

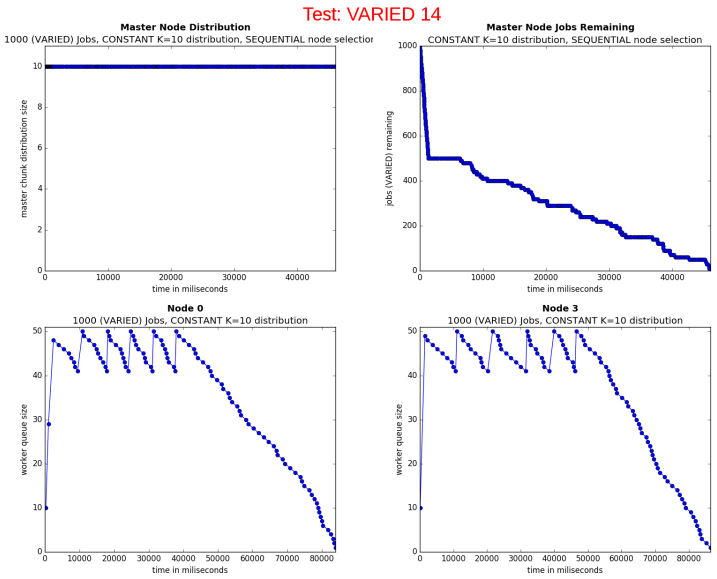


(d) 30 workers

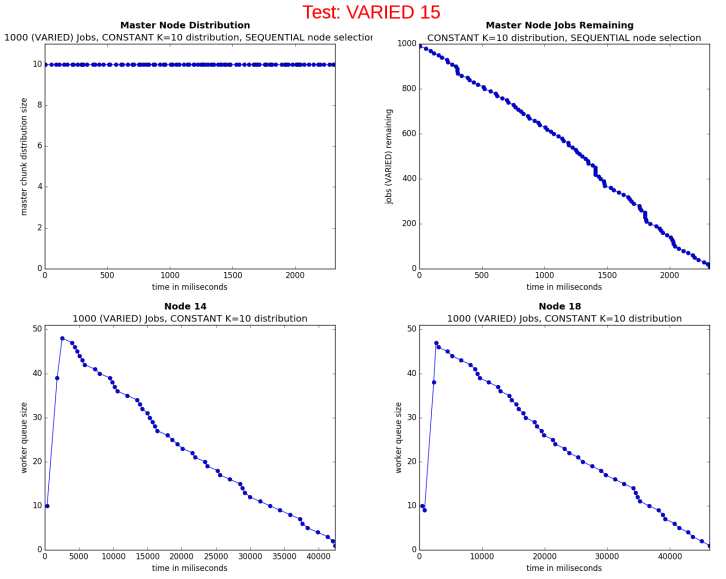
Figure 12: These tests focus on exponential job chunk distribution with 1000 varied jobs to process and varying worker size. These figures plot results for the varied test numbers 25, 26, 27, 28 as labeled at the top of each figure. The top left plot in each figure shows the master job distribution rate as time progresses. The top right plot in each figure shows the number of jobs remaining to distribute as time progresses. The bottom plots in each figure show queue sizes of two randomly selected workers.



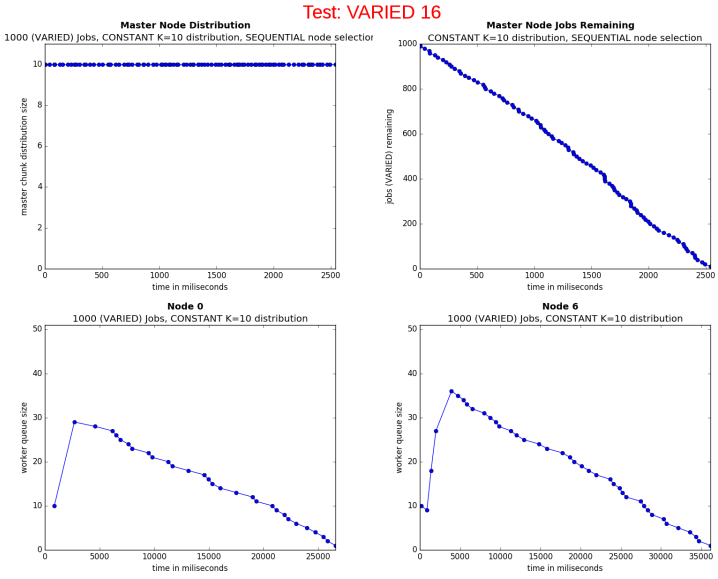
(a) 5 workers



(b) 10 workers



(c) 20 workers



(d) 30 workers

Figure 13: These tests focus on constant job chunk distribution with 1000 varied jobs to process and varying worker size. These figures plot results for the varied test numbers 13, 14, 15, 16 as labeled at the top of each figure. The top left plot in each figure shows the master job distribution rate as time progresses. The top right plot in each figure shows the number of jobs remaining to distribute as time progresses. The bottom plots in each figure show queue sizes of two randomly selected workers.

7 Future Work

This simulation focused on varying two configurations while keeping others constant. As the next steps, it would be beneficial to vary some of the configurations that were kept constant as shown in Table 3. Aside from varying configurations it would definitely be beneficial to setup a simulation in an actual distributed cluster, which should be somewhat trivial to setup through Amazon AWS.

Additionally, this simulation studied the push method where the master continuously attempted to distribute, or push out, resources to the workers. It would be interesting to study the pull method where workers come to the master to receive more jobs. In this setup the configurations would clearly be different, but this scheduling strategy may prove to be beneficial in certain scenarios.

8 Conclusion

The results show that in workloads with very similar job execution times and workloads with varying job execution times, randomly selecting a worker node will have a detrimental effect. Additionally, it also seemed to be the case that constant job distribution yielded the fastest completion time in both workloads. Exponential and linear job distribution strategies ended up being more favorable for spreading job load evenly across workers when there is a relatively small number of workers available. When there is an abundant number of workers, constant job distribution strategy tended to spread the job load evenly as it is less likely to make worker queues suddenly jump to higher sizes. Thus overall, when faced with distributing jobs in a cluster of uniform nodes, the following generalizations could be made based on the presented results:

- Random worker node selection tends to have a detrimental effect.
- If strictly looking for best performance, distribute a constant number of jobs at all times.
- If looking for ideal workload spread, distribute a constant number of jobs when there is an abundant number of workers, and vary the distribution in a linear or exponential rate when there are a few workers available.

References

[1] H Karatza and R Hilzer. Parallel Job Scheduling in Homogeneous Distributed System, 2004.