Course Project

Vineet Patel, Parth Mehul Desai, Heng Li

Instructions

Requirements:

1. Scala
2. Sbt
3. Docker

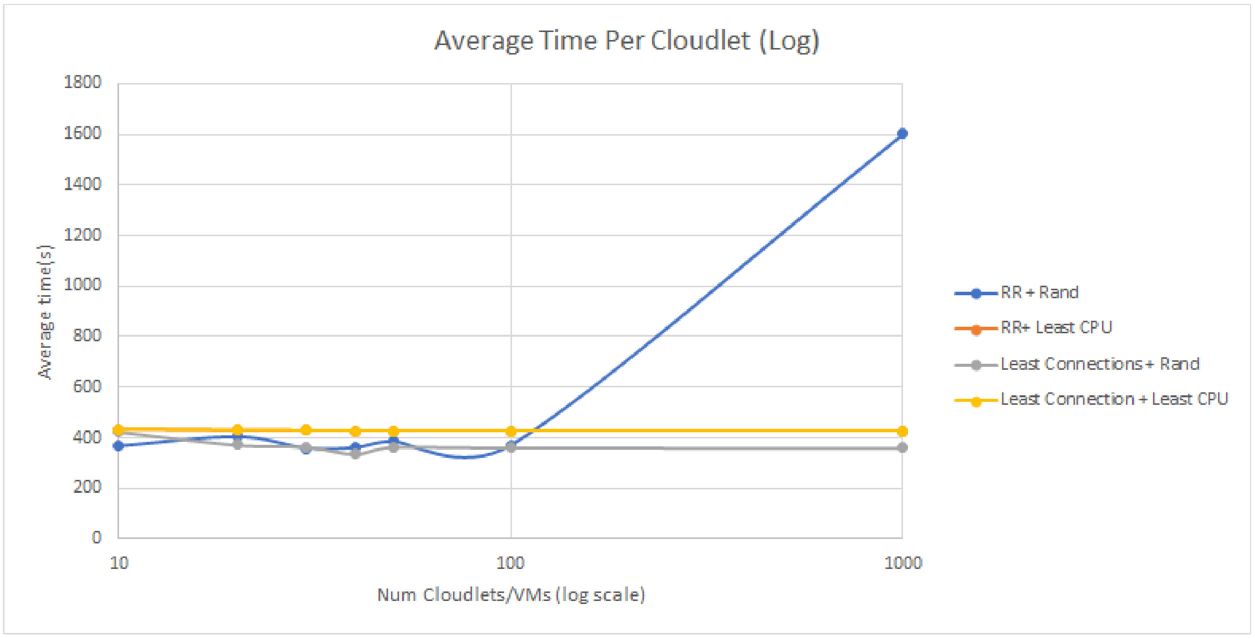
Run the files using `` `*sbt clean compile run*` ``.
Tests can be run using `` `*sbt test*` ``.

###Dockerfile
You can pull the image from
https://cloud.docker.com/u/vineet1999/repository/docker/vineet1999/cs441_group_project .
Do build the image run `` `*docker build -t <image name you would like>*` ``.
Make sure when you build, your current directory is the root directory.
of this project, as the files will be copied over when building the image.

To run run the container run
`` `docker run -it <image name>` ``
Note it is important that you include the -it flag, as user input is need
to select which simulation you wish to run.

###Simulation overview
Our project is centered around the effect of different load balancing algorithms. We have utilized 2 algorithms for this purpose: Random and Least Cpu Usage. The Random algorithm uses the java.util.Random class to decide the index of the vm to which the cloudlet is allocated. The LeastCpuUsage load balancing algorithm will select the vm with the least amount of cpu utilization and assign the incoming cloudlet to the vm.
Furthermore, we have provided 2 vm allocation policies: Least Connection and Round Robin.
Least Connection allocates the vm to the host which has the least number of vm's already assigned. Round Robin assigns vms to the hosts in a round robin fashion.

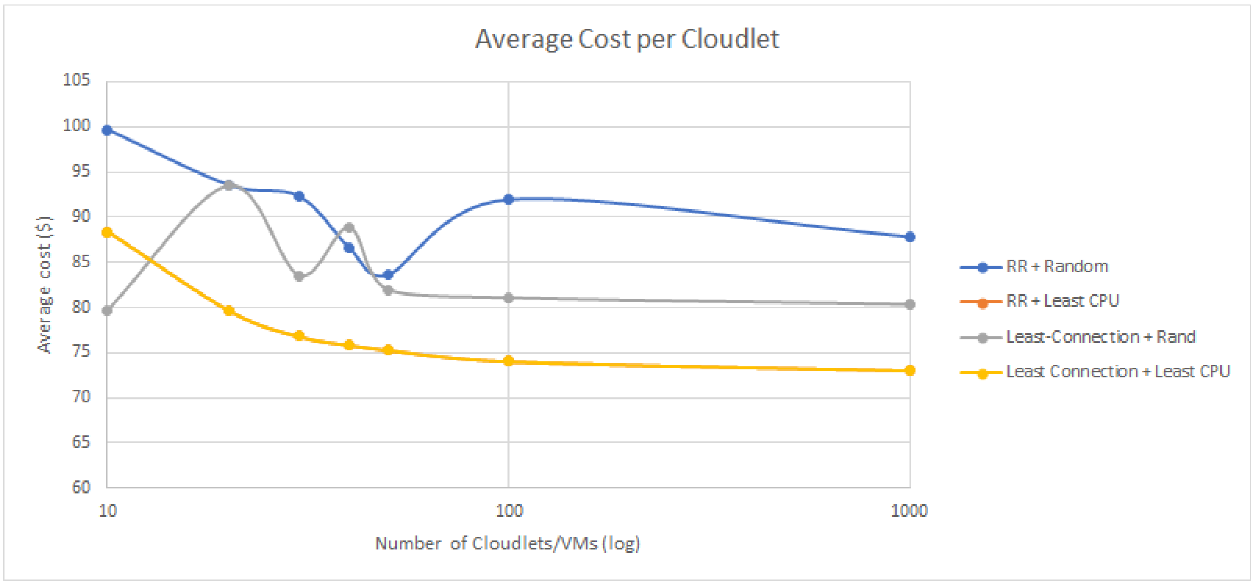
###Results
The cloudlets and vms used for the simulations are of 3 types: standard, which have low computational resource requirements, memory-intensive, which have high memory requirements and cpu-intensive, which are highly cpu intensive. These are equally represented. The simulations themselves are of 3 types- normal, power and network simulations. We have used cost, time, power and data transferred as the metrics to analyse these simulations.



Number of vms	Per cloudlet time			
	RR + random	RR + Least CPU	Least Connection + Random	Least Connection + Least CPU
10	366.743	434.31	425.07	434.31
20	403.164	430.03	372.38	430.03
30	358.250	428.61	364.86	428.61
40	360.05	428.147	334.72	428.14
50	385.06	427.86	363.95	427.86
100	370.482	427.25	361.20	427.25
1000	1600.43	426.76	359.672	426.76

Number of vms	Total time			
	RR + random	RR + Least CPU	Least Connection + Random	Least Connection + Least CPU
10	800.32	800.22	800.32	800.22
20	1600.43	800.22	800.32	800.22
30	880.43	800.22	800.32	800.22
40	800.32	800.22	880.43	800.22
50	880.43	800.22	880.43	800.22
100	880.43	800.22	880.43	800.22
1000	68580.80	800.22	880.43	800.22

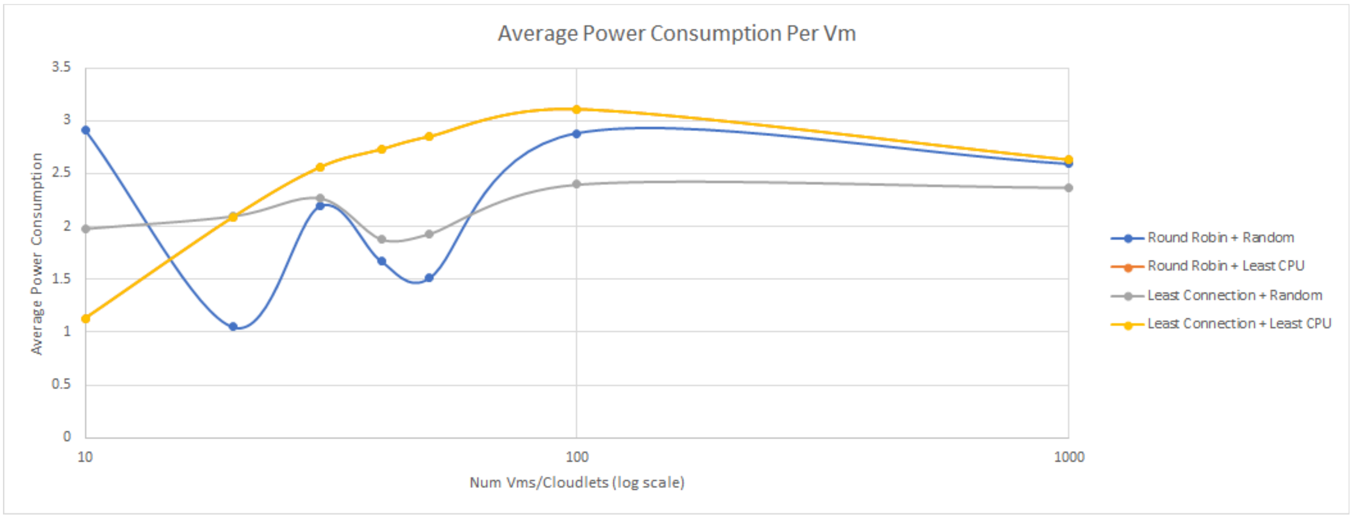
For lower number of cloudlets, we found that performance was pretty identical. However, when the number of cloudlets increased exponentially, the disadvantages of random allocation were clearly evident. This was especially so in case of the combination of Round Robin VM allocation and Random load balancing. The arbitrary assignment of cloudlets to vms led to multiple resource intensive cloudlets being assigned to certain vms and others being left lightly loaded.



Number of vms	Per cloudlet cost			
	RR + random	RR + Least CPU	Least Connection + Random	Least Connection + Least CPU
10	99.66	88.34	79.67	88.34
20	93.56	79.69	93.47	79.69
30	92.30	76.80	83.52	76.80
40	86.63	75.87	88.81	75.87
50	83.58	75.30	81.95	75.30
100	91.93	74.07	81.02	74.07
1000	87.81	73.06	80.29	73.06

Number of vms	Total cost			
	RR + random	RR + Least CPU	Least Connection + Random	Least Connection + Least CPU
10	597.96	618.40	637.40	618.40
20	1232.84	1275.04	1215.16	1275.04
30	2030.684	2150.56	2088.16	2150.56
40	2685.64	2807.20	2664.36	2807.20
50	3343.48	3463.84	3359.96	3463.845
100	6711.53	7184.81	6766.976	7184.81
1000	68580.800	72848.92	68415.67	72848.92

The average cost per cloudlet was varying significantly in case of random load balancing. However, it was decreasing gradually in all the other cases. This was mainly due to the randomization when assigning vms. Most of the time the algorithm does a good job of balancing the workload across all the vms, however sometimes it might assigns multiple cloudlets to the same vm.



Number of vms	Avg Power			
	RR + random	RR + Least CPU	Least Connection + Random	Least Connection + Least CPU
10	2.91	1.13	1.98	1.13
20	1.05	2.09	2.10	2.09
30	2.19	2.56	2.27	2.56
40	1.67	2.73	1.88	2.73
50	1.51	2.85	1.93	2.85
100	2.88	3.11	2.40	3.11
1000	2.59	2.63	2.37	2.63

As surmised by the random nature of load balancing, power consumption per vm for Round Robin + Random is all over the map. On the other hand, Least CPU load balancing allows equivalent load across all the vms, allowing average power utilization to be lower and variance to be reduced. As power utilization increases dramatically when the system is under heavy load, the Least Connection+Least CPU combination is the best for power consumption in the beginning.
However, as the scale of the datacenter increases, the average power consumptions across all simulations starts to converge until there’s no significant difference between them.

For NetworkSimulation, we were suppose to simulate network topology and data transfer across different hosts within the data center. After researching and experimenting ourselves and consulting other students with same problem on piazza, we found out that due to the limitation of the cloudsimplus we aren’t able to fully utilize the sendTask and receiveTask in the cloudlet. So we ended up using only execution tasks in the cloudlet. Therefore our network simulation doesn’t utilize the network topology and other features. Therefore it has no different than non-NetworkSimulation.

Conclusion:

The combination of the Least Connection Vm Allocation Policy and Least CPU load balancing policy provide the best overall results across all metrics. We think that due to the allocation of vm/cloudlets to their respective hosts/cloudlets with a measure of cpu utilization and previously waiting workload, this combination is superior to the other arbitrary allocation policies used.