On Choosing a Task Assignment Policy for a Distributed Server System[1]

Ferhat Elmas

Performance Evaluation Mini Project

June 4, 2012

Introduction

- A task assignment policy for a distributed server system that composed of multiple hosts which process tasks in FCFS.

1. **Random**: One of the hosts is randomly chosen. Each of host has an equal probability.

Introduction

- A task assignment policy for a distributed server system that composed of multiple hosts which process tasks in FCFS.

1. **Random**: One of the hosts is randomly chosen. Each of host has an equal probability.
2. **Round Robin**: Tasks are assigned to hosts in cyclic fashion.

Introduction

- A task assignment policy for a distributed server system that composed of multiple hosts which process tasks in FCFS.

1. **Random**: One of the hosts is randomly chosen. Each of host has an equal probability.
2. **Round Robin**: Tasks are assigned to hosts in cyclic fashion.
3. **Dynamic**: The task is assigned to the host that has minimum expected waiting time.

Introduction

- A task assignment policy for a distributed server system that composed of multiple hosts which process tasks in FCFS.

1. **Random**: One of the hosts is randomly chosen. Each of host has an equal probability.
2. **Round Robin**: Tasks are assigned to hosts in cyclic fashion.
3. **Dynamic**: The task is assigned to the host that has minimum expected waiting time.
4. **Size Based**: Task size is partitioned and tasks that are in a certain range are assigned to a particular host.

Motivation

- Task Assignment Policies are studied extensively
  - Task Size is modelled in exponential
  - Exponential is a poor model

Motivation

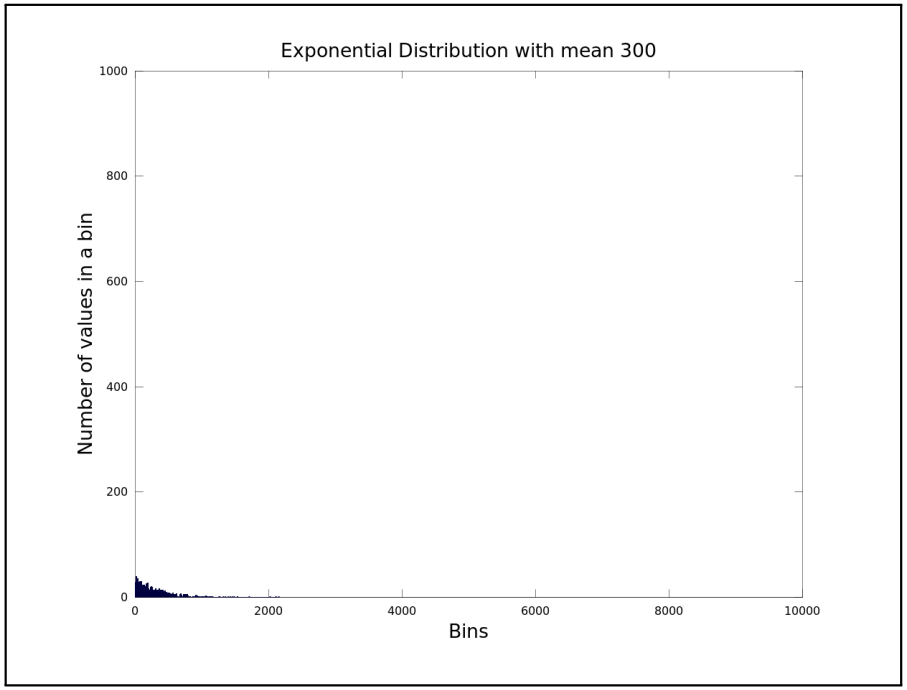
- Task Assignment Policies are studied extensively
  - Task Size is modelled in exponential
  - Exponential is a poor model
- Real task sizes show heavy tail property
  - Very small fraction of task makes nearly half of the load
  - $Pr(X > x) \sim X^{-a}$ where $0 \leq a \leq 2$
  - Lower a implies more variable task size

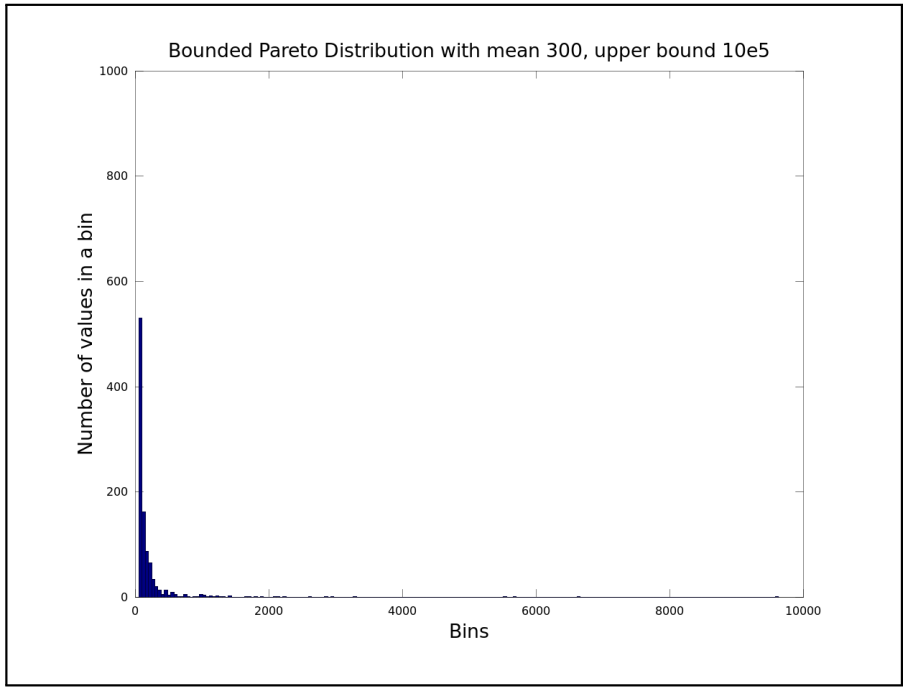
Motivation - Real Values

Task Description	a
Unix CPU requirements at BellCore	$[1, 1.25]$
Unix CPU requirements at Berkeley	~ 1
Size of files transferred through Web	$[1.1, 1.3]$
Size of FTP transfers in the internet	$[0.9, 1.1]$
I/O Times	~ 0

- That's why *Microsoft Windows* can't predict
  - How much time is left while moving, copying files

Motivation - Comparison of Exponential and Pareto





Assumptions

- Task Size is known in advance

Assumptions

- Task Size is known in advance
- Service time is equal to size of task, no scheduling overhead

Assumptions

- Task Size is known in advance
- Service time is equal to size of task, no scheduling overhead
- CPU is the only resource

Assumptions

- Task Size is known in advance
- Service time is equal to size of task, no scheduling overhead
- CPU is the only resource
- Hosts in the server are identical

Assumptions

- Task Size is known in advance
- Service time is equal to size of task, no scheduling overhead
- CPU is the only resource
- Hosts in the server are identical
- Task is assigned to a host as soon as it arrives.

Assumptions

- Task Size is known in advance
- Service time is equal to size of task, no scheduling overhead
- CPU is the only resource
- Hosts in the server are identical
- Task is assigned to a host as soon as it arrives.
- Execution is in FCFS and non-preemptive

System Model

- Task arrival rate is poisson.

System Model

- Task arrival rate is poisson.
- Task size is bounded pareto distribution.

System Model

- Task arrival rate is poisson.
- Task size is bounded pareto distribution.
- Defined as $B(k, p, a)$
  - $P(x) = \frac{a \cdot k^a \cdot x^{-a-1}}{1 - \frac{k}{p}^a}$ where $k <= x <= p$

System Model

- Task arrival rate is poisson.
- Task size is bounded pareto distribution.
- Defined as $B(k, p, a)$
  - $P(x) = \frac{a \cdot k^a \cdot x^{-a-1}}{1-\frac{k}{p}^a}$ where $k <= x <= p$
- a shape parameter and controls variance
  - Smaller $a \rightarrow$ Higher variance
  - Varies in range of $[1, 2]$

Performance Measures

- Average Waiting Time

Performance Measures

- Average Waiting Time
- Average Slow Down
  - $\frac{WaitingTime}{TaskSize}$

System Parameters

- Number of Servers

System Parameters

- Number of Servers
- Load

System Parameters

- Number of Servers
- Load
- *a* shape parameter of pareto distribution
  - while *a* is changing, *k* is adjusted to keep the mean same

Schedule Algorithms

- **Random**:
  - Host $h \leftarrow$ Task t where $\frac{h}{|H|} \leq u < \frac{h+1}{|H|}$ where $u \sim U[0,1]$

Schedule Algorithms

- **Random**:
  - Host $h \leftarrow$ Task t where $\frac{h}{|H|} \leq u < \frac{h+1}{|H|}$ where $u \sim U[0,1]$
- **Round Robin**:
  - Host $h \leftarrow$ Task t where $t \equiv h \pmod{|H|}$

Schedule Algorithms

- **Random**:
  - Host $h \leftarrow$ Task t where $\frac{h}{|H|} \leq u < \frac{h+1}{|H|}$ where $u \sim U[0,1]$
- **Round Robin**:
  - Host $h \leftarrow$ Task t where $t \equiv h \pmod{|H|}$
- **Dynamic**:
  - Host $h \leftarrow$ Task t where $h = getHostWithMinTaskQueue()$

Schedule Algorithms

- **Random**:
  - Host $h \leftarrow$ Task t where $\frac{h}{|H|} \leq u < \frac{h+1}{|H|}$ where $u \sim U[0,1]$
- **Round Robin**:
  - Host $h \leftarrow$ Task t where $t \equiv h \pmod{|H|}$
- **Dynamic**:
  - Host $h \leftarrow$ Task t where $h = getHostWithMinTaskQueue()$
- **Size Based**:
  - Host $h \leftarrow$ Task t where $bounds[h] \leq t.size < bounds[h+1]$

Schedule Algorithms - Size Based Bound Calculation

Bounds x_i

$$k = x_0 < x_1 < x_2 < \cdots < x_{h-1} < x_h = p$$

$$\int_{k=x_0}^{x_1} x \cdot dF(x) = \int_{x_1}^{x_2} x \cdot dF(x) = \cdots = \int_{x_{h-1}}^{x_h=p} x \cdot dF(x) = \frac{\mu}{h} = \frac{\int_{k}^{p} x \cdot dF(x)}{h}$$

Simulation Values

- To study variance
  - $a \leftarrow (1.1 + 0.1 \cdot step)$ where $0 \leq step < 9$
- $p = 100000, \mu = 300$
  - k is adjusted

$$k = 51 \leftarrow a = 1.1$$
$$k = 65 \leftarrow a = 1.2$$
$$k = 78 \leftarrow a = 1.3$$
$$k = 91 \leftarrow a = 1.4$$
$$k = 103 \leftarrow a = 1.5$$
$$k = 114 \leftarrow a = 1.6$$
$$k = 125 \leftarrow a = 1.7$$
$$k = 134 \leftarrow a = 1.8$$
$$k = 142 \leftarrow a = 1.9$$

- Load is fixed at 0.8 so arrival rate is
  - $Rate = \frac{h \cdot load}{\mu}$

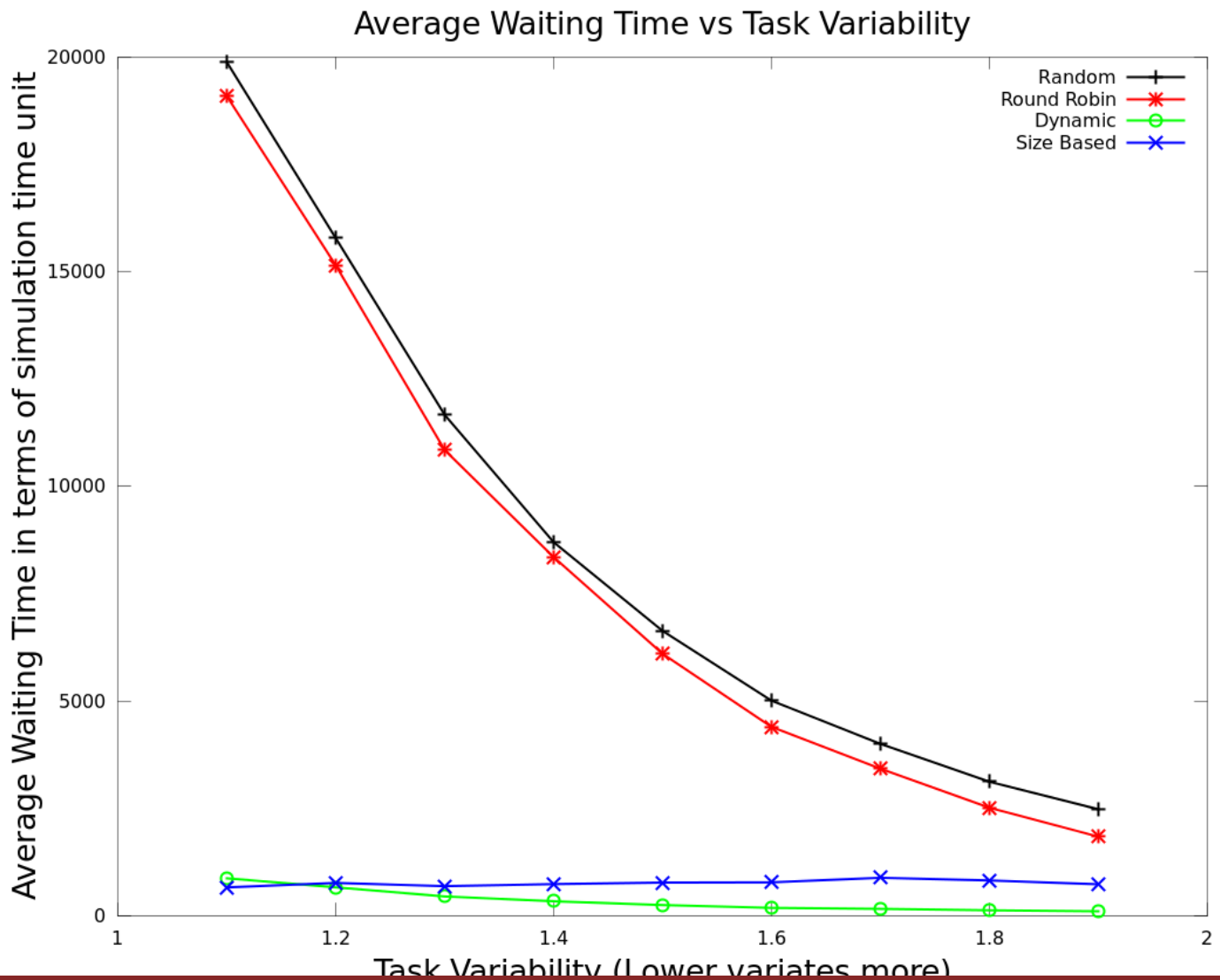
Simulation Values - 2

- Number of arrival tasks
  - Sufficient condition to finish simulation
  - $t = 10^7$
- Number of Runs
  - Required condition to get confidence
  - $r = 100$
- Steady state
  - Required condition to sense heavy tail
  - After $2 * 10^6$ task arrivals, 20% of all tasks

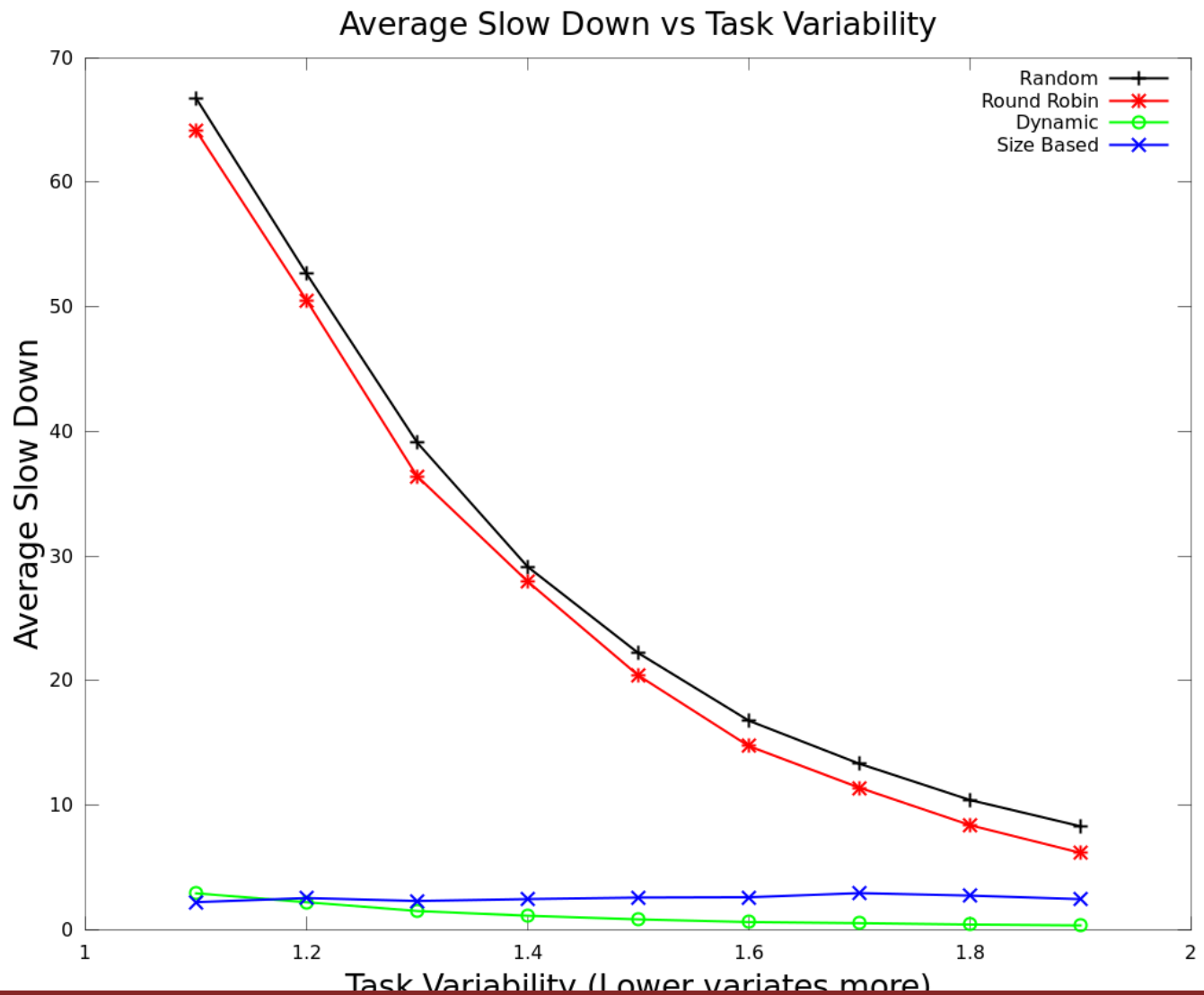
Average Time Comparison to Exponential

	Exponential		Bounded Pareto($a = 1.1$)	
Policy	Waiting Time	SlowDown	Waiting Time	SlowDown
Random	1262.5670	4.2152	19890.4074	66.6756
Round Robin	630.2895	2.1033	19103.7297	64.1288
Dynamic	122.1085	0.4078	924.5439	3.0929
Size Based	693.9924	2.3162	650.5760	2.1863

Graphs


Average Waiting Time vs Task Variability
Average Waiting Time in terms of simulation time unit
Task Variability (Lower variates more)
Random
Round Robin
Dynamic
Size Based
20000
15000
10000
5000
0
1
1.2
1.4
1.6
1.8
2

Graphs


Average Slow Down vs Task Variability
Average Slow Down
Task Variability (Lower variates more)
Random
Round Robin
Dynamic
Size Based
0
10
20
30
40
50
60
70
1
1.2
1.4
1.6
1.8
2

Confidence Intervals

- 100 runs

Exponential		
Policy	Lower Bound	Upper Bound
RANDOM	1249.26593	1263.79357
ROUND ROBIN	622.54191	632.53330
DYNAMIC	121.35404	123.49327
SIZE BASED:	684.83063	696.14331

Confidence Intervals - 2

- 100 runs

Bounded Pareto - Random Policy		
a	Lower Bound	Upper Bound
1.1	19428.98410	20437.87776
1.2	14978.03231	16344.14195
1.3	11089.38970	11961.20672
1.4	8399.67174	9102.56048
1.5	6437.75149	6913.98037
1.6	4755.01100	5129.61958
1.7	3834.98348	4183.03872
1.8	2997.05842	3172.11419
1.9	2324.65254	2571.22622

Confidence Intervals - 3

- 100 runs

Bounded Pareto - Round Robin Policy		
a	Lower Bound	Upper Bound
1.1	18749.38675	20086.56348
1.2	14379.42287	15940.56209
1.3	10391.33202	11662.84883
1.4	7853.41338	8423.94786
1.5	5795.79636	6169.45627
1.6	4178.87500	4475.55269
1.7	3159.88026	3567.81041
1.8	2373.26957	2695.34599
1.9	1718.44651	2002.97348

Confidence Intervals - 4

- 100 runs

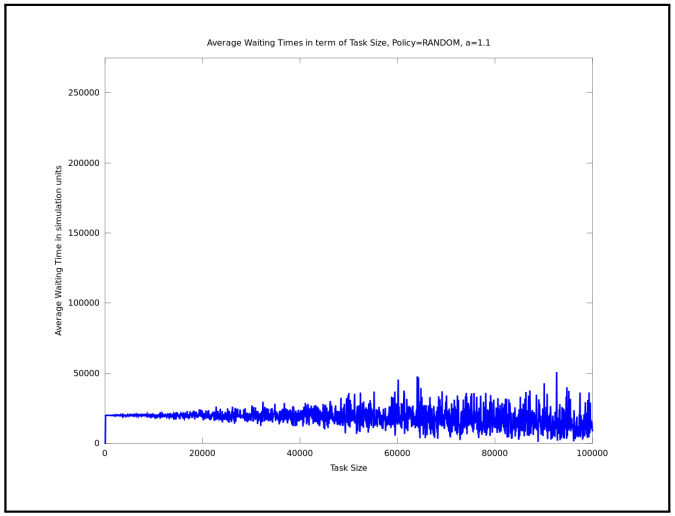
Bounded Pareto - Dynamic Policy		
a	Lower Bound	Upper Bound
1.1	804.91955	985.55327
1.2	619.41650	781.61605
1.3	441.59131	530.17471
1.4	339.75903	404.82852
1.5	264.03451	285.85370
1.6	205.73459	233.99490
1.7	186.39173	194.75100
1.8	154.97537	165.03551
1.9	135.49628	141.11565

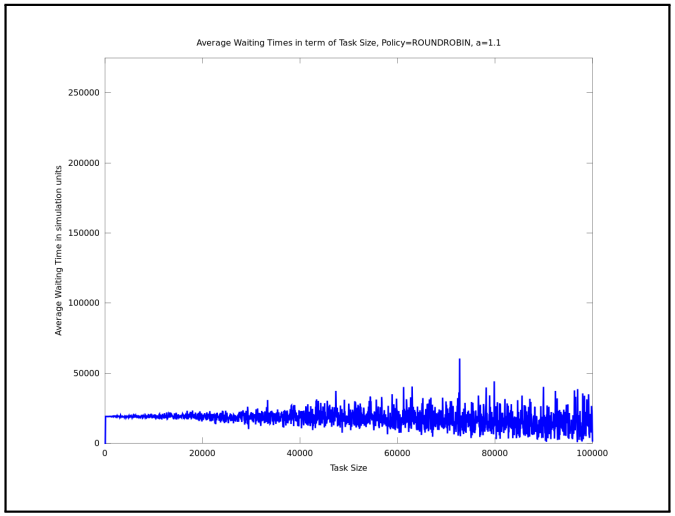
Confidence Intervals - 5

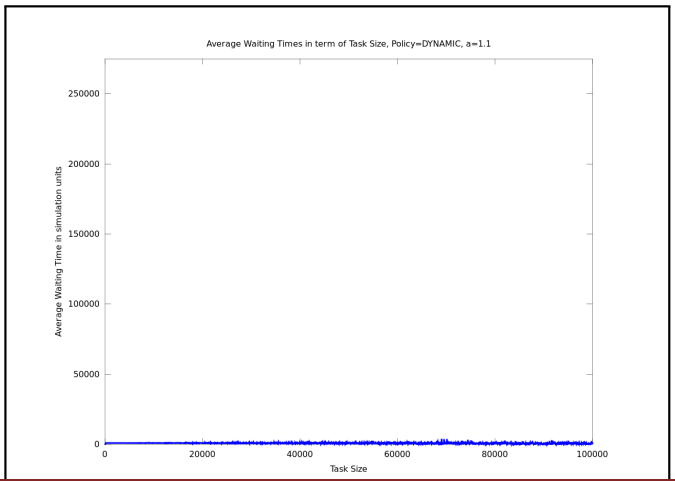
- 100 runs

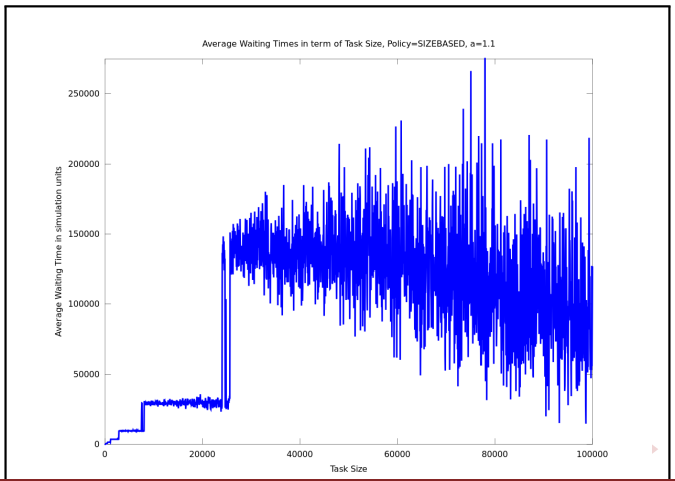
Bounded Pareto - Size Based Policy		
a	Lower Bound	Upper Bound
1.1	641.64633	688.31200
1.2	665.11589	830.02852
1.3	656.61129	730.20091
1.4	707.65831	795.26298
1.5	739.64542	827.10732
1.6	730.64636	847.75423
1.7	827.62977	948.85723
1.8	810.10281	881.40799
1.9	718.54123	756.09282

Average Waiting Times by Task Size, a=1.1

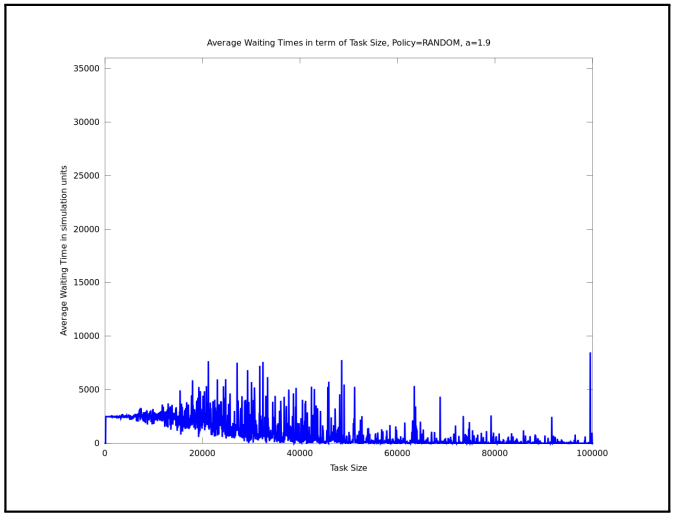


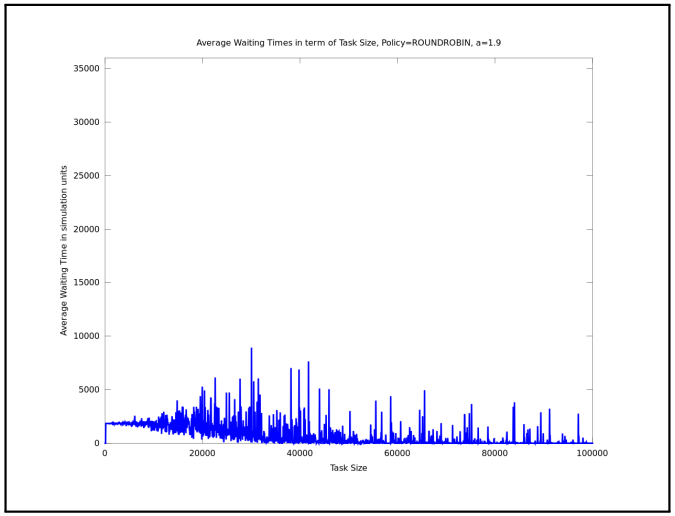


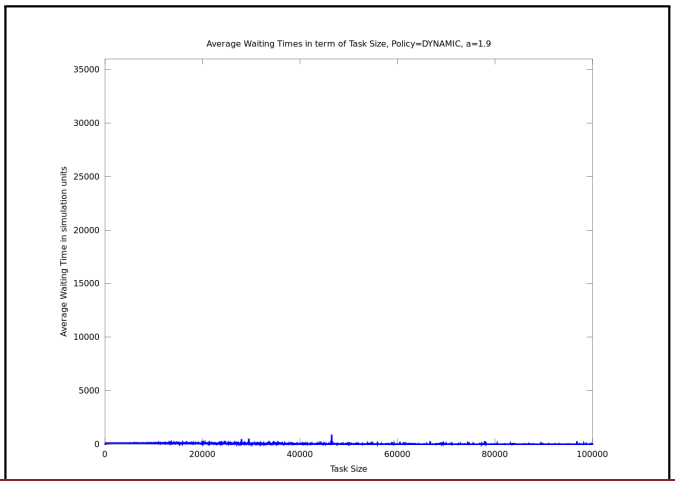


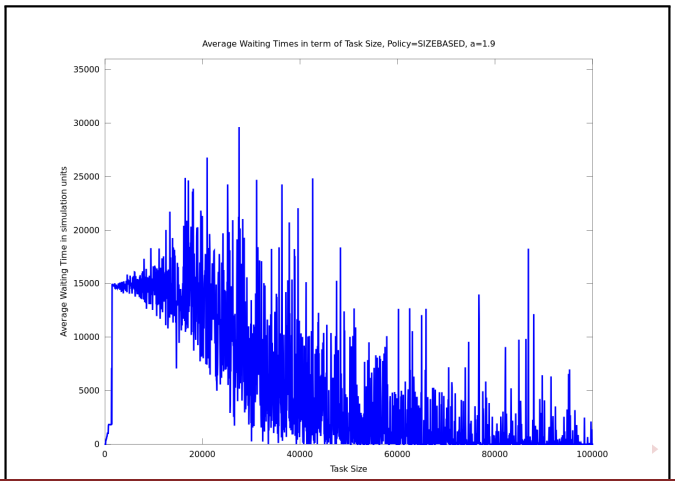


Average Waiting Times by Task Size, a=1.9

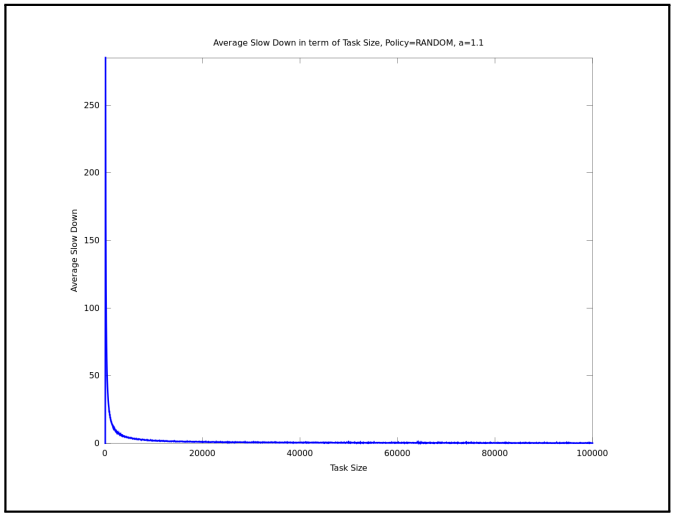


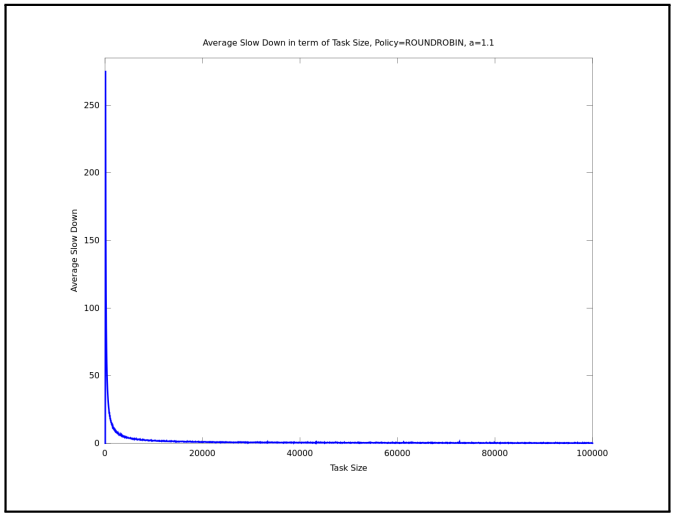


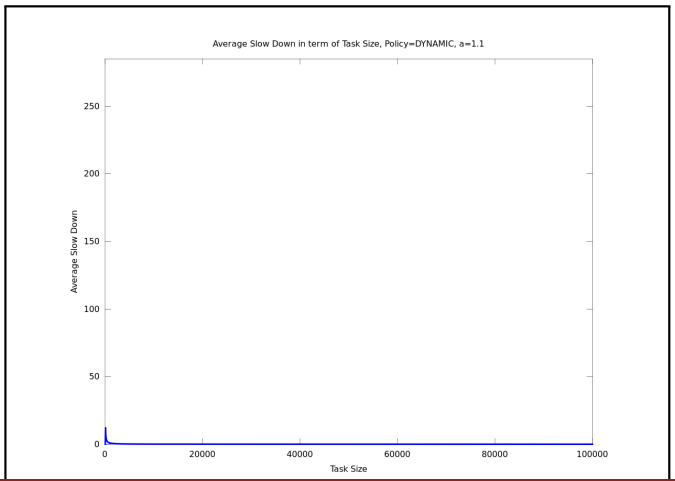


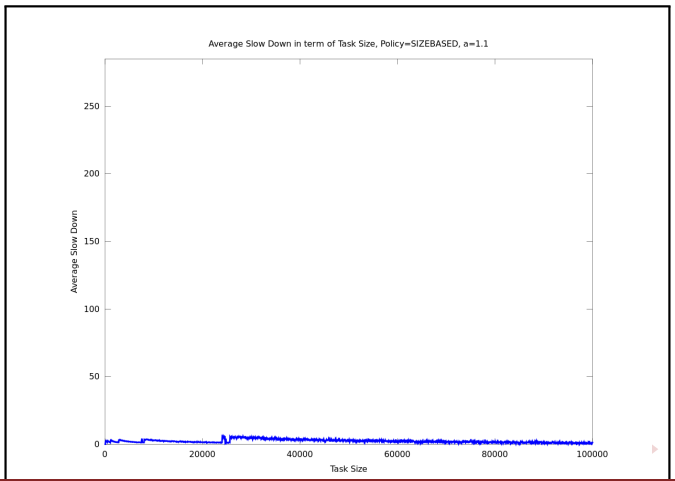


Average Slow Downs by Task Size, a=1.1

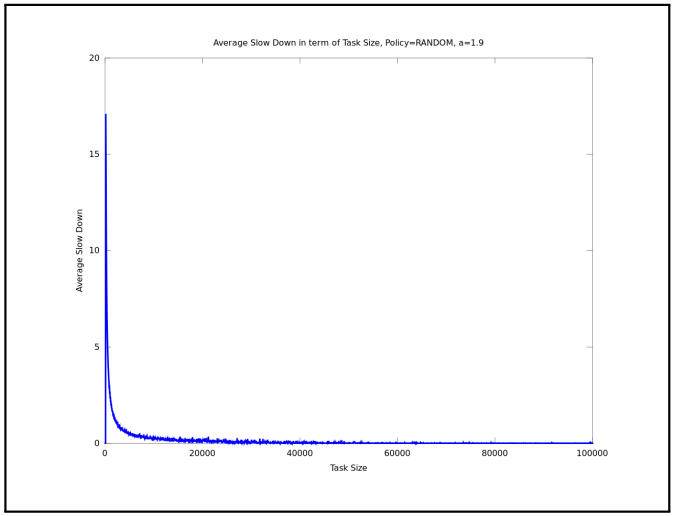


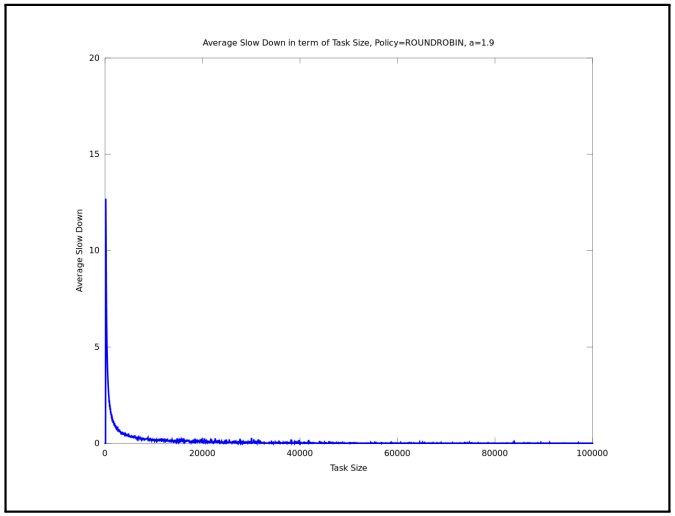


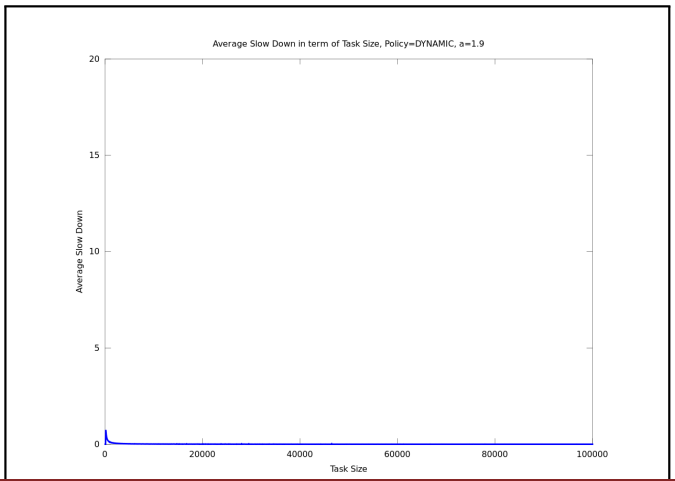


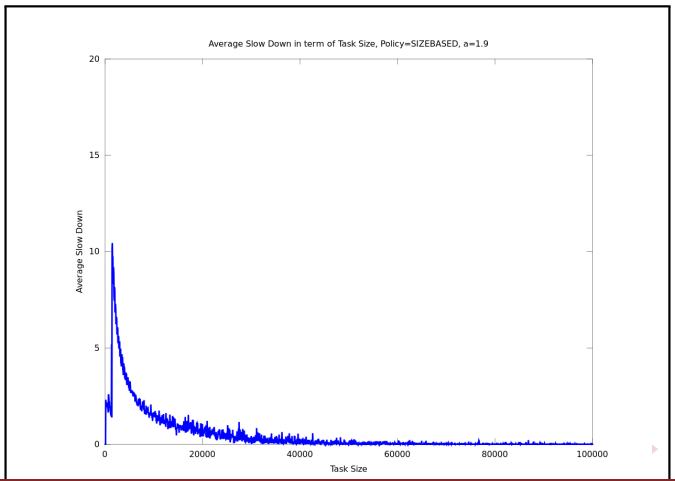


Average Slow Downs by Task Size, a=1.9

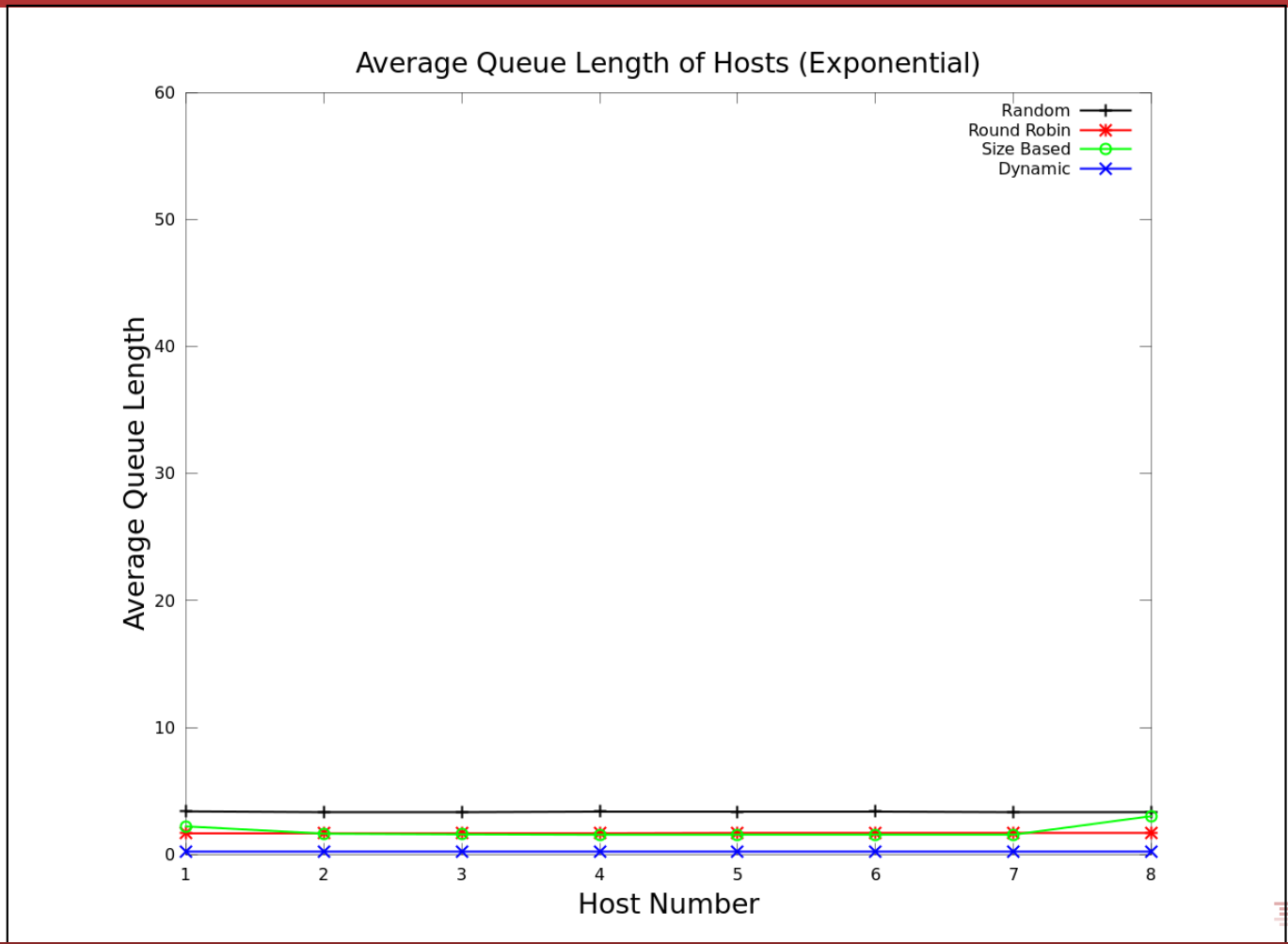




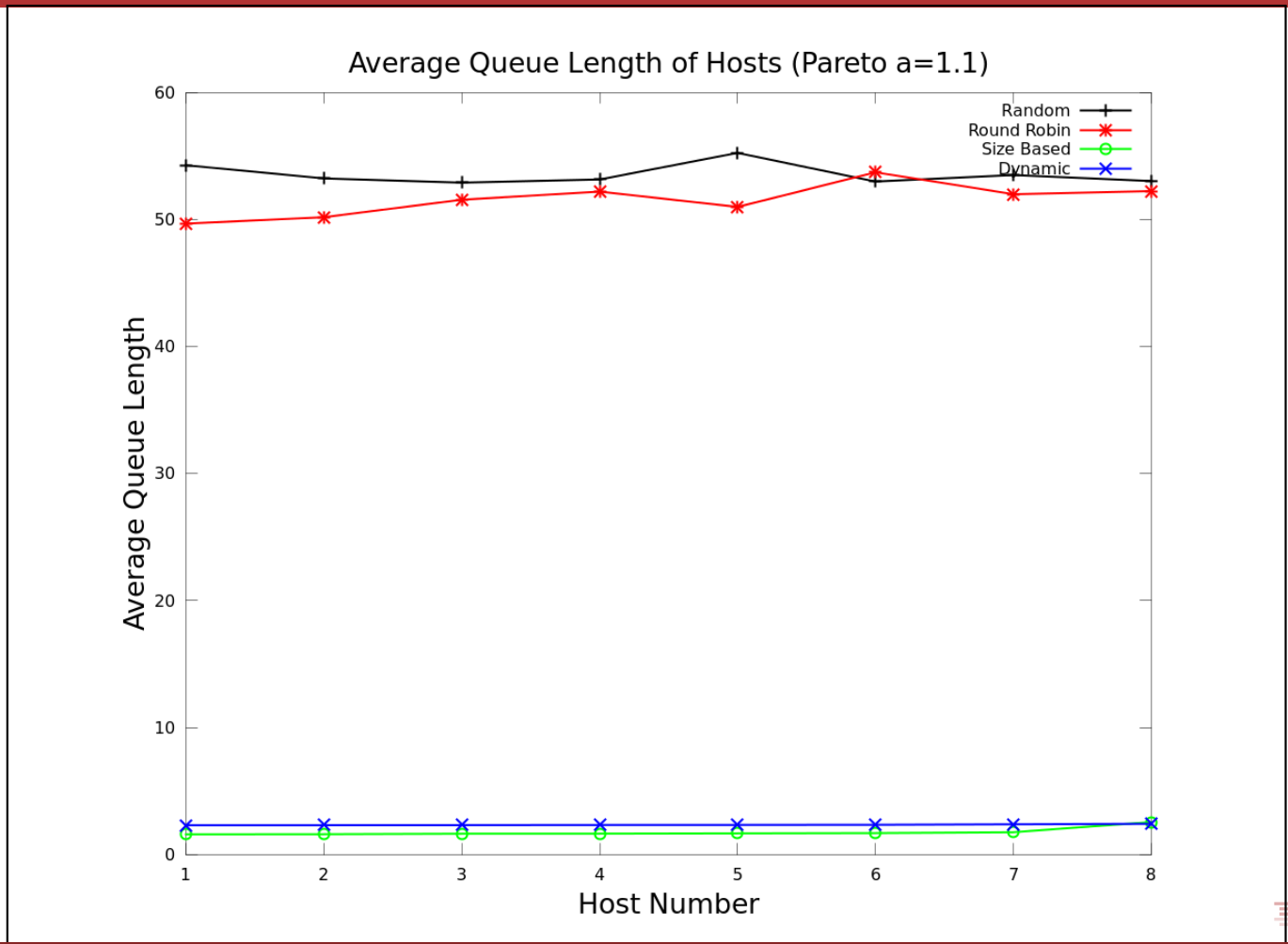




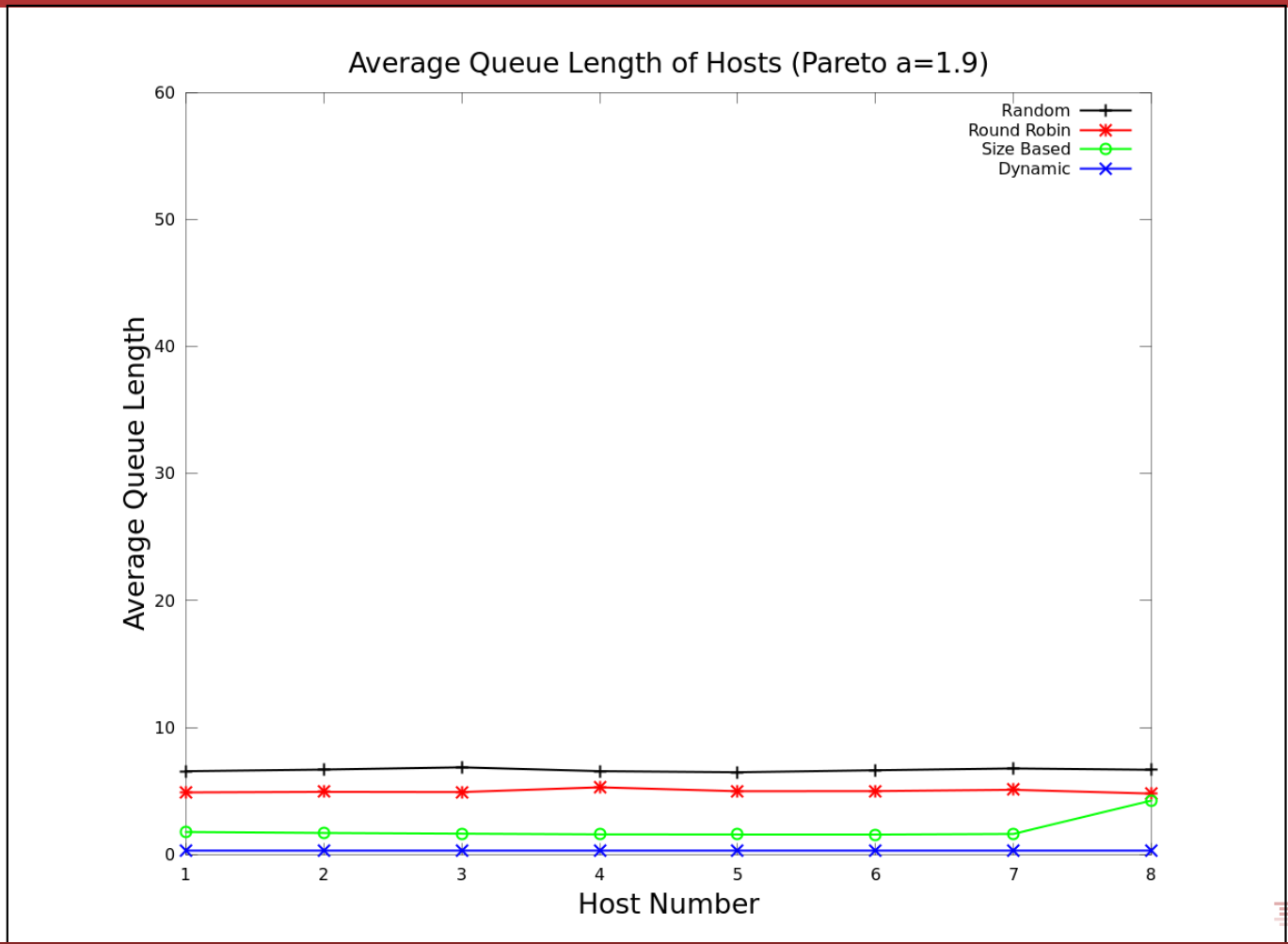
Average Queue Length - Exponential



Average Queue Length - Pareto $a = 1.1$



Average Queue Length - Pareto $a = 1.9$



Conclusion

- With high variance, size based policy does the best
- While variability decreases, dynamic policy gets better
  - *Mathematically proofed*
- Size-based and dynamic always have much smaller waiting time and slowdown than random and round robin policies
- Size based policy isn't affected a lot by variability in task size

Conclusion - Comparison of Simulation to Paper

- What is done in the simulation is correlated by findings of the paper

- **Thanks for attention!**

References

M. Harchol-Balter, M. E. Crovella, and C. D. Murta.
On choosing a task assignment policy for a distributed server system.
Journal of Parallel and Distributed Computing, Proceedings of Performance Tools, 1468:231–242, 1999.