Performance of Novel Scheduling Strategies

Xiao Chen Nigel Thomas

School of Computing Science, Newcastle University, UK
xiao.chen1@ncl.ac.uk & nigel.thomas@ncl.ac.uk

Abstract

Today distributed server systems have been widely used in many areas because they enhance the computing power while being cost-effective and more efficient. Meanwhile, some novel scheduling strategies are employed to optimize the task assignment process. This project closely explored the performance of the novel scheduling strategies through computer simulation. The research was carried out regarding the simulation of a novel scheduling policy (Task Assignment by Guessing Size) and other two previous task assignment policies (Random and JSQ). The performance of novel scheduling strategy (TAGS) is achieved by comparing TAGS policy with other two preceding policies. To facilitate the performance, computer simulation is applied to perform the statistical measurements. The findings were, indeed, very interesting, showing that the novel scheduling strategy (TAGS) merely obtains an optimal performance under heavy-tail distributed computing environment. The paper concludes by summarizing the findings from the simulation and suggesting a wider study be undertaking, in order to explore the performance of the novel strategy in more depth.

1 Introduction

In today's world, distributed system has a wide application because of its powerful computing ability. In distributed system, a large task could be decomposed into small subtasks that run simultaneously on multiple server hosts communicating over a network. Thus a series of arriving tasks must be assigned to different servers to obtain the service. So as to guarantee the dispatching process, the task assignment policy is adopted in the dispatcher. Generally speaking, what we want is to complete all tasks in the system with the highest efficiency. Hence, the choice of the task assignment policy will produce an important effect on the system performance. It is no longer a new topic to find the best task assignment policy for different distributed systems, but it remains a significant research direction.

In this paper, the performance of several scheduling strategies will be explored by simulating the entire scheduling process with Java program. The aim is to simulate different task assignment policies under various computing environments and then compare the numerical results so as to analyze the performance of those policies. The most widely used task assignment policies include Random task assignment policy (Random), Join the Shortest Queue policy (JSQ) and a novel policy TAGS (Task Assignment by Guessing Size). In Random task assignment policy, each arrival will be dispatched to a host in the system at random. If the total number of hosts in the system is h, each task has a probability of 1/h to go to a single host. This policy ensures almost equivalent number of jobs being allocated to each host. In Join the Shortest Queue policy, arrivals are dispatched to the host with fewest waiting jobs at that moment. This policy attempts to balance the number of waiting jobs at each host rather than the total number of jobs being dispatched.

The last one is a novel task assignment policy named by TAGS (Task Assignment based on Guessing Size). As said by the definition of TAGS, each host in such kind of distributed system has a finite value representing the timeout. The outside arrivals are dispatched to the first host (says HOST_1) one by one, and then HOST_1 will complete the tasks which can be done within the timeout period and pass oversized tasks to the next host (HOST_2). At HOST_2, delayed tasks from the first host will restart and run until timeout or accomplishment. The timeout of each host is increased to a larger value compared with preceding host. This increasing timeout guarantees that large jobs could gain the service eventually. TAGS policy is a new policy performing a high efficiency with heavy-tailed workloads. Its excellent performance is presented through the following two factors: mean queue length and mean response time. Queue length is a mean value concerning the number of jobs waiting at each host in the

system; response time is the average value of the time duration from the time point at which job starts to the point completing its service. With the aim of gaining the related numerical results, SimJava 2.0 will be applied in simulation to aid the measurements concerning those factors noted before.

Section 1 will briefly introduce the general background information and main purpose of the project. Section 2 will demonstrate three widely used task assignment policies. Section 3 will achieve the performance of policies with exponential distribution. Section 4 will measure and analyze the performance with Pareto distribution. The last section will provide a final conclusion of the research.

2 Scheduling Strategies

2.1 Join the Shortest Queue policy

JSQ policy defines that newly arrived tasks will be immediately dispatched to the server with the fewest number of waiting tasks. As said by Winston's research, JSQ is optimal when the task size distribution is exponential and the arrival process is Poisson [3]. The optimality means that JSQ maximize the discounted number of the completed jobs within a fixed time interval. Furthermore, Ephremides, Varaiya and Walrand showed that JSQ also minimized the total time for completing all jobs by some fixed time and under an exponential task size distribution and arbitrary arrival rate [1]. In this paper, we will explore the performance of JSQ in a 2-host distributed system in order to compare with TAGS.

2.2 Random assignment policy

Random policy is a simple scheduling strategy stating that the newly incoming task is dispatched to host server i with the possibility 1/h (h is the number of hosts in the system). In this paper, to facilitate the comparison with other task assignment policies, we assume that only two server hosts exist in the distributed system (h = 2).

2.3 Task Assignment by Guessing Size

Traditionally, in distributed server system, a series of jobs are dispatched to hosts by a specified scheduling strategy. At each host, jobs are not preemptive which means once the host starts a job service, it will not terminate until the job is completed at the same host. However, in real-life distributed system, job sizes might be heavy-tailed. So we have to consider the case that workload is heavy-tailed while choosing a task assignment policy. To fit the heavy-tail distributed workload, a novel task assignment policy TAGS (Task Assignment based on Guessing Size) is proposed. This chapter will introduce TAGS algorithm and its relevant parameters based on performance.

In this section, the job flow with TAGS algorithm will be illustrated. Firstly, we assume that h is the number of hosts (h1, h2, h3…) in the server system. Secondly, for each host we set a number t to represent the timeout of this server host, where $t1 < t2 < t3 < t4$.

Outside arrivals are dispatched to the first host (h1) and put into the waiting queue before being served. At h1, some jobs (usually short jobs) will be accomplished within the timeout t1, and then terminate and leave the system. Meanwhile, those large jobs, which can not be done at h1, are killed and then forwarded to the end of queue of h2. These large jobs will restart at h2 and gain service until its completion or use up the timeout t2 and be passed to the next host. Each host serves jobs in FCFS order. Large jobs hop from one host to the next until its eventual completion. Hence, "guessing size" is achieved in the hop of jobs.

The most important benefit of TAGS is its high performance when job size is in heavy-tailed distribution. As mentioned before, we suppose that the job queue is as follow, {8, 3, 4, 7, 9, 5, 2, 191 …}. When the server systems run with TAGS scheme, few large jobs will be redelivered to the host with larger timeout, while small jobs are served without being redelivered. Otherwise, many small jobs might be delayed because some large job is processing at this host. In this situation, length queue and response time will be increased because of the delayed waiting jobs; meanwhile the system throughput will grow down. TAGS policy solves this problem by passing large jobs to the next host so as to guarantee small jobs could be served without long delay. Thereby, TAGS policy usually shows

excellent performance when job sizes are in heavy-tailed distribution.

To use TAGS, the crucial problem is to compute the appropriate cutoff t, as the efficiency of this policy highly depends on the value of t. With TAGS policy, large jobs might restart many times before eventual completion. As a negative result, large jobs might be served with low efficiency. Server resource is wasted in restarting jobs time and time again. On the other hand, this policy guarantees that small jobs, which can be finished within timeout, gain service rather than being delayed for a long period when the server is occupied by large jobs. In addition, large jobs will experience more repeated services than small jobs. Thus, the larger the job is the more redundant service it experiences. Hence, in TAGS scheme, small jobs will be served and completed quickly, but large jobs have to experience longer delays.

3 Statistical Measurements and Analysis with Exponential Distribution

In this chapter, simulation statistics will be displayed with line graphs. Analysis of statistics will be carried out so as to compare different task assignment policies. As noted before, for all task assignment policies, queues are unbounded; therefore jobs will never be lost in the system. The simulated systems are all structured with two server hosts. The measurements are processed with two types of distributions, exponential distribution and Pareto distribution. Concretely, these two distributions are used to represent service time under two different situations. In next section, simulation with Pareto Distribution will be performed. However, the event generation based on an arrival rate is represented with exponential distribution no matter what type of policy is applied. To analyze the performance of each policy, measurements focus on two aspects, average queue length and average response time.

Normally, the generated random numbers with exponential distribution should fluctuate around a mean value. According to the definition of exponential distribution, a mean value must be specified before producing random numbers. To represent event generation process, exponential distribution is adopted to build a number generator and the mean of exponential distribution is specified as the reciprocal of arrival rate, namely $1/\lambda$. For the serving process at each host, the service time is presented with exponential-distributed numbers which have the mean $1/\mu$ (the reciprocal of service rate). In this section, the job sizes (namely the service demand) are based on exponential distribution. Comparison between three policies will be carried out by analyzing the graphs showing average queue length and average response time.

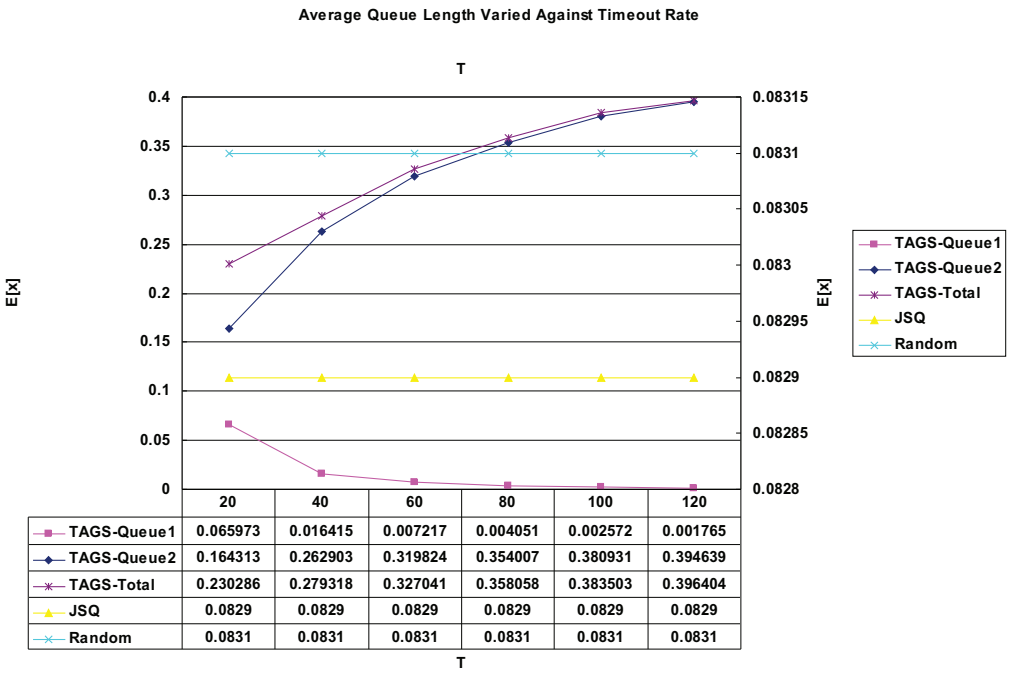


Figure 3.1 Average Queue Length Varied against Timeout Rate, λ=5, μ=10
X-axis (left): TAGS-Q1, TAGS-Q2, TAGS
X-axis (right): Random, JSQ

Figure 3.1 shows the average queue length against the timeout rate from 20 to 120. The values of arrival rate and service rate are specified as 5 and 10 respectively, which keep the same value through all measurements with exponential distribution.

From the graph, it is clear that JSQ has the shortest queue length around 0.082, while TAGS has a growing queue length from about 0.23 to 0.39 with the increase of timeout rate. For Random, its queue length is slightly less than JSQ, which is roughly 0.083. Thus, TAGS has the largest average queue length and also has large difference with other policies. As Random and JSQ policies can not be affected by timeout rate, their lines in the graph are straight and parallel to the x-axis. For TAGS, with the growth of timeout rate, the average queue length of host 1 goes down whereas the average queue length of host 2 increases rapidly. This means an increasing number of events are transmitted from host 1 to host 2 because of the decrease of timeout (equals the growth of timeout rate). As a whole, the queue length of TAGS tends to rise. According to the tendency of lines for TAGS-Q1 and TAGS-Q2, these two lines should be cross at a point at which timeout rate is less than 20. Overall, in exponential distributed computing environment, TAGS has the worst performance, while JSQ is the optimal.

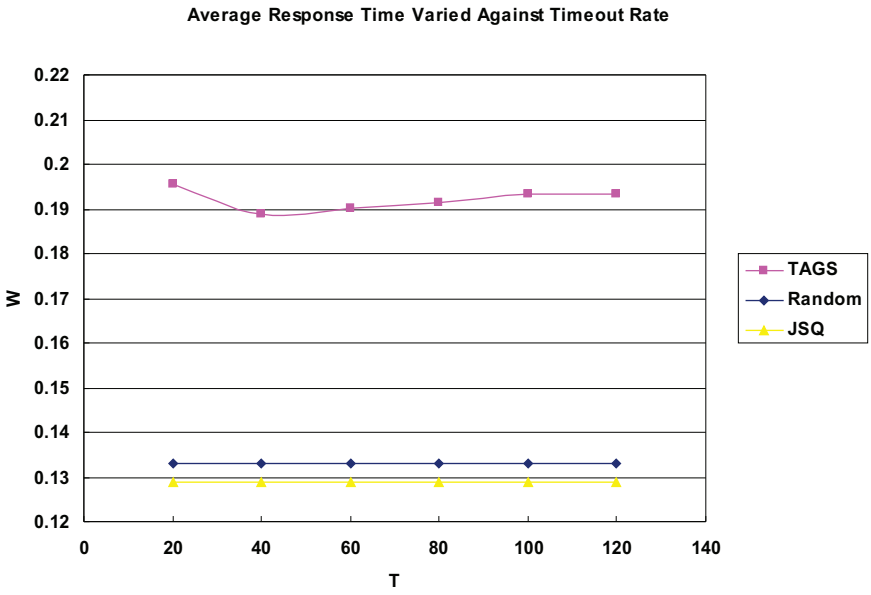


Figure 3.2 Average Response Time Varied against Timeout Rate, λ=5, μ=10

Figure 3.2 depicts the average response time of the three policies against timeout rate from 20 to 120. As shown in the graph, TASG presents the longest average response time around 0.19. However, JSQ still performs best with shortest response time which is less than 0.13. Regarding Random policy, its average response time is slightly longer than JSQ, which quite close to 0.134. The lines representing Random and JSQ are straight and parallel to x-axis with the same reason in Figure 3.1. The graph proves that TAGS has the worst performance on average response time. However, JSQ is the policy performing best.

According to the previous statistical analysis, JSQ policy is the optimal task assignment policy when the job sizes are based on exponential distribution. Compared with JSQ and Random, TAGS is always going to be inferior. From the analysis, TAGS is unsuitable for the exponential distributed computing environment. In fact, TAGS usually performs well when the variability of the service demand distribution augments.

4 Statistical Measurements and Analysis with Pareto Distribution

Heavy-tailed distribution (Pareto distribution) will be considered in this section, as it is necessary to explore how the distribution of job sizes affects the decision of which task assignment policy to use.

The previous measurements have used exponential distribution to capture the distribution of job sizes. According to the preceding conclusion, exponential distributed computing environment is poor to

perform TAGS policy. In fact, under TAGS policy, the perfect variability of job sizes should be like this: there are typically many short jobs and fewer large jobs. To represents such a job sizes distribution, heavy-tailed distribution is considered as an accurate way from the recent research. Pareto distribution is the simplest heavy-tailed distribution. In this section, all measurements will be carried out with Pareto distribution. Before starting the measurements, two parameters used for Pareto distribution must be specified, which are the shape symbolized by k and the scale symbolized by Xm.

In accordance with the previous research, a series of jobs with heavy-tailed distributed size should obtain the following properties [1]:

- Decreasing failure rate: Especially, the longer a job has run, the longer it is expected to continue running.
- Infinite variance: If the k is less than 1, the mean is towards infinite.
- A small proportion (< 1%) of very large jobs comprise over half of the total load.[1]

The smaller the value k, the more variable the distribution, which means the sizes of jobs are more heavy-tailed. According to others' research, heavy-tailed distribution performs excellently in many recent measurements, as follows:

Unix process CPU requirements measured at Bellcore: $1 \leq k \leq 1.25$ [7];
Unix process CPU requirements measured at UC Berkeley: $k \approx 1$ [8];
Sizes of files transferred through the Web: $1.1 \leq k \leq 1.3$ [9, 10];
Sizes of FTP transfers in the Internet: $0.9 \leq k \leq 1.1$ [11];

In most situations, the value of k is somewhat larger than 1. In this section, we will analyze and explore the measurements based on $1.0 < k < 2.0$. In simulation process, varied value of k is applied to observe the effect caused by the changing of the distribution.

4.1 Pareto Distributed Measurements, k = 1.1

As mentioned in preceding section, when the shape parameter k is set between 1.1 and 1.3, it represents the sizes of files transferred on the Web. Subsequently, we will analyze the performance of three task assignment policies by comparing their average queue length and average response time.

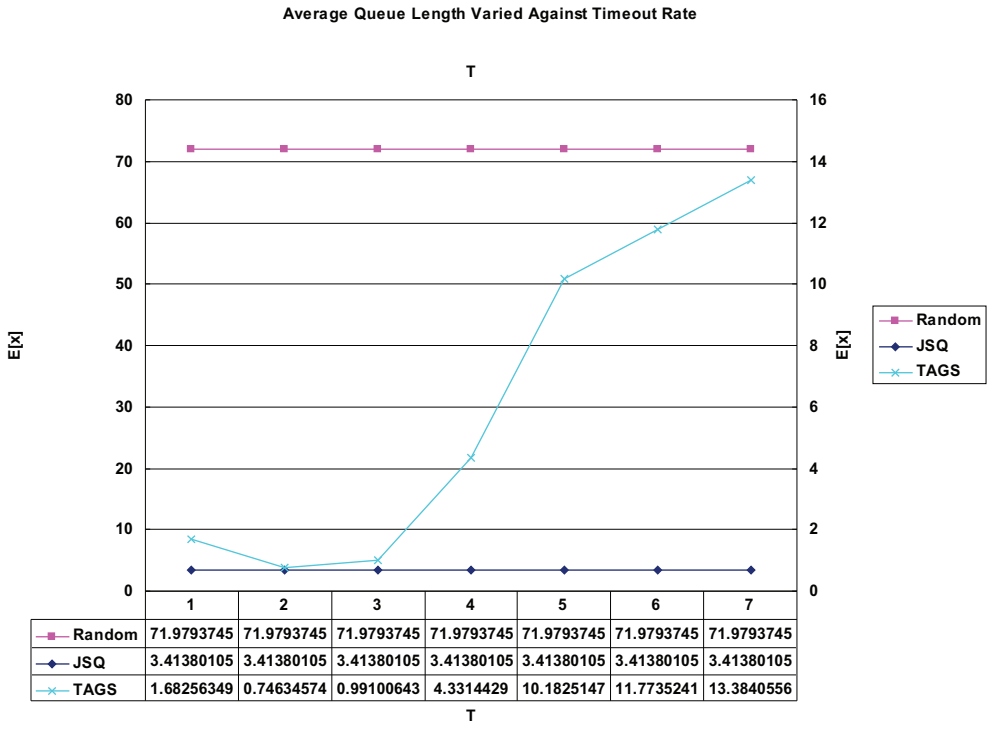
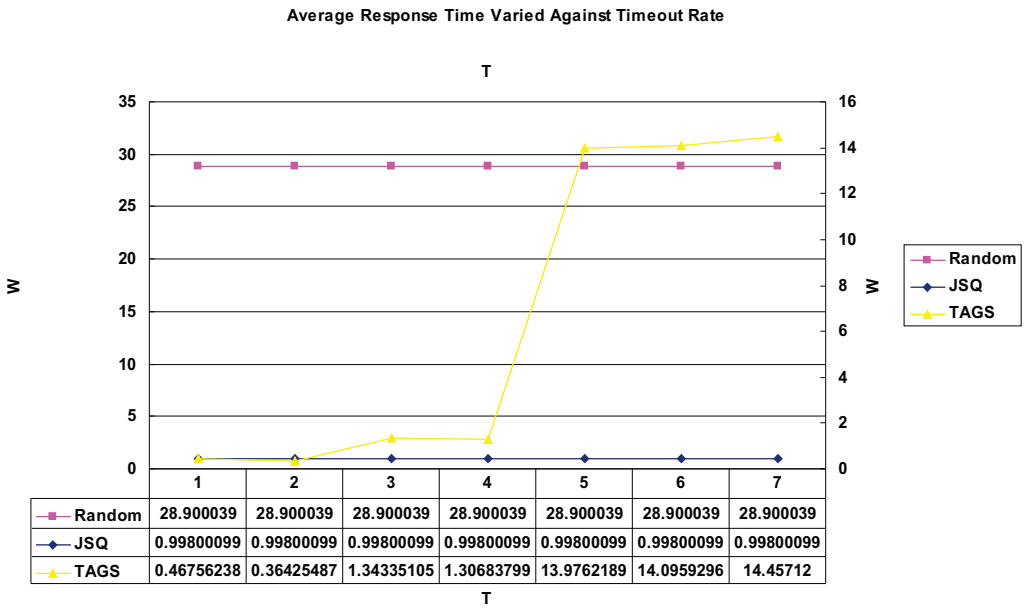


Figure 4.1 Average Queue Length Varied against Timeout Rate, λ=5, μ=10, k=1.1
X-axis (left): Random, JSQ
X-axis (right): TAGS

Figure 4.1 shows the average queue length against varied timeout rate from 1 to 7. The main feature of this graph is that TAGS has the best performance when timeout rate lies between 1 and 3 inclusively. However, Random policy is the worst having a quite high value in its average queue length which is nearly 72. JSQ, having the value around 3.4, is slightly higher than TAGS when the timeout rate is less that 4. However, when the timeout rate is larger than 4, the average queue length of TAGS increases over the JSQ. As the timeout rate increasing, for TAGS policy a growing number of events are delayed and retransmitted to the second host; as a result, the number of waiting events at the second host increases greatly. Such delayed events will be restarted at host 2 which leads to a poor performance in average queue length. For Random policy, it is shown as the worst choice in such a distribution.



	1	2	3	4	5	6	7
Random	28.900039	28.900039	28.900039	28.900039	28.900039	28.900039	28.900039
JSQ	0.99800099	0.99800099	0.99800099	0.99800099	0.99800099	0.99800099	0.99800099
TAGS	0.46756238	0.36425487	1.34335105	1.30683799	13.9762189	14.0959296	14.45712

Figure 4.2 Average Response Time Varied against Timeout Rate, λ=5, μ=10, k=1.1
X-axis (left): Random, JSQ
X-axis (right): TAGS

Figure 4.2 depicts the average response time of three task assignment policies. From the graph, TAGS policy has the least average response time around 0.4 when the timeout rate is less than 3. Meanwhile, JSQ policy uses a bit more response time than TAGS. With the rise of the timeout rate, the average response time of TAGS has a dramatic growth especially the rate varying from 3 to 4. This is caused by the increased number of events that are redelivered to host 2. For each event, once it is retransmitted from the first host to the second, the time used at host 1 is wasted, but this time is still added to the total response time. So this means that the more the retransmitted events the larger the average response time. Moreover, JSQ policy also has a short average response time (keeping the value less than 1) even though somewhat higher than TAGS when the timeout rate is at a lower level. However, Random policy has a value of average response time around 30 that means Random is inefficient as a scheduling scheme in this situation.

To sum up, with the Pareto distribution (k = 1.1), TAGS performs best especially when timeout rate is specified with small values. JSQ is also acceptable, as sometimes its performance is quite close to TAGS. The main advantage of JSQ is that its average queue length and average response time always keeps the same without being interfered by the timeout rate. For Random policy, it always gives the worst performance in each measurement. So Random policy is quite unsuitable to be applied in Pareto distributed (k=1.1) computing environment.

4.2 Pareto Distributed Measurements, k = 1.3

In this section, the scale k of Pareto distribution is increased to 1.3. As I said before, with the increase of k, the variability of distribution will goes down. The measurements of each policy are still processed in the same conditions, λ=5, μ=10.

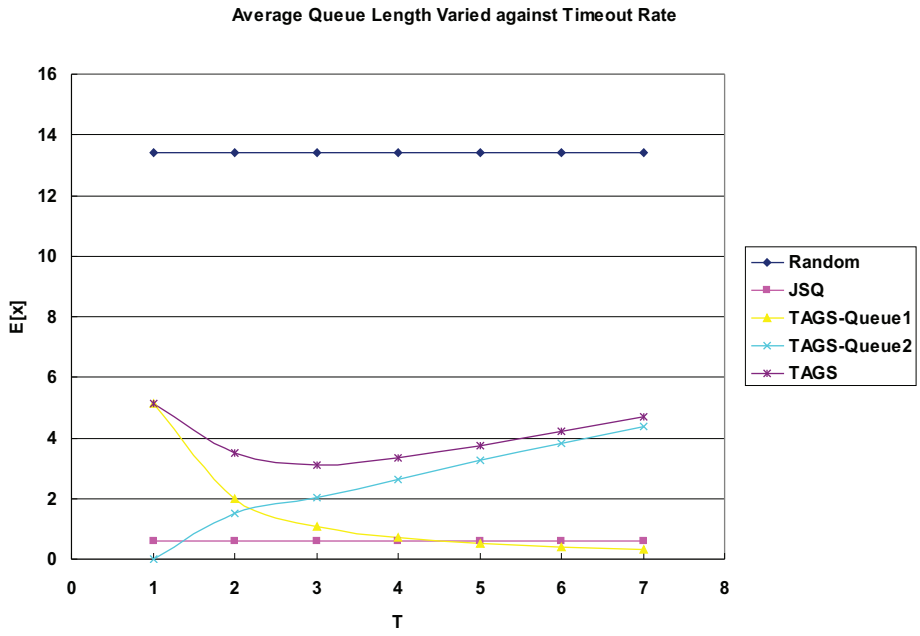


Figure 4.3 Average Queue Length Varied against Timeout Rate, λ=5, μ=10, k=1.3

Figure 4.3 displays the average queue length varied against timeout rate from 1 to 7. From the line graph, JSQ has the least average length queue around 0.6, while the average queue length of Random policy is the largest roughly 13.4. For TAGS, the average queue length of the first host starts from a higher value but it tends to a lower level as the timeout rate is increasing. Oppositely, at the second host, the average queue length starts from a small value to a higher level as the timeout rate goes up. Along with the growth timeout rate, the timeout at host 1 is growing. As a result, a growing number of events will be passed to the second host. So the average queue length at host 2 tends to increase. In total, the average queue length of TAGS is around 4.

In contrast to the Pareto distribution k = 1.1 (Figure 4.1), the average queue length of TAGS has a tendency to a higher level along with the rise of k. The performance of TAGS in the lower level of timeout rate is not as good as it is in Figure 4.1 (k = 1.1). In contrast to TAGS, for both Random and JSQ, the average queue length has a decrease comparing with the value in Figure 4.1. Hence, as the value of k is increasing, the variability of distribution is reduced; the performance of TAGS becomes worse. However, Random and JSQ have improvements in performance.

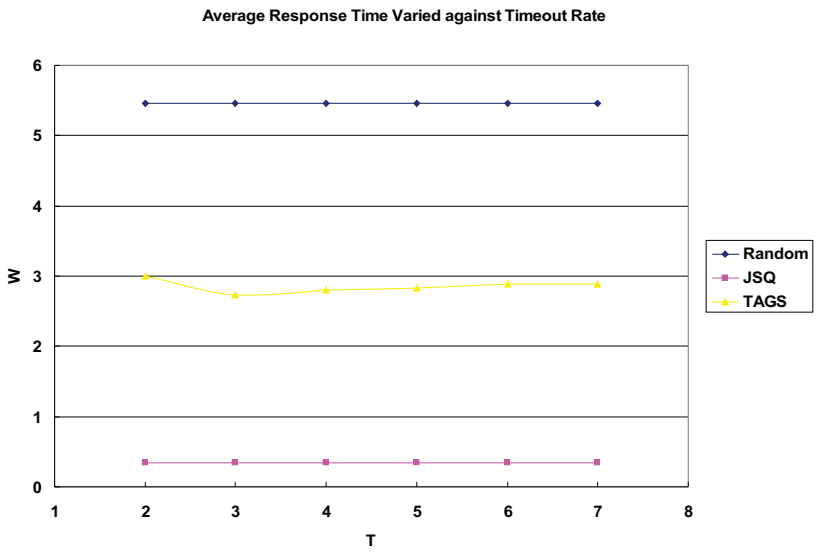


Figure 4.4 Average Response Time Varied against Timeout Rate, λ=5, μ=10, k=1.3

Figure 4.4 is about the average response time of each policy against timeout rate from 2 to 7. According to the graph, JSQ has the least average response time with the value about 0.3, but Random policy is the highest, namely around 5.5. For TAGS, the average response time is close to 3, which is between Random and JSQ.

Compared with the situation in Figure 4.2 (k = 1.1), the average queue length in both Random and JSQ is decreased which means the performance is better when the value of k is added. However, TAGS policy has the different change. When the timeout rate is less than 5, the average response time is increased with the rise of k. Consequently, when the shape parameter k is added to a larger value, the performance of TAGS (in average response time) will become worse. Nevertheless, Random and TAGS have completely opposite change in average response time, which means their performance goes to be better.

To conclude, after increasing the shape parameter k to 1.3, the performance of TAGS is not as good as it is when k = 1.1. For TAGS, the performance in both average queue length and average response time turn to a worse quality as the value of k becomes larger. As k goes up, the distribution of job sizes is becoming less variable which means the job sizes are less heavy-tailed. From the information given by graphs, it clearly demonstrates that TAGS policy usually gives an excellent performance in the distribution with high variability. When the variability of distribution goes down, TAGS will lose its high performance; however, in this situation, JSQ will appear to be superior. When the real-life system environment is Pareto-distributed with the shape k=1.3, there is on doubt that JSQ is the optimal policy.

4.3 Pareto Distributed Measurements, k = 1.5

In this section, the shape k of Pareto distribution is added to 1.5. Measurements for Random, JSQ and TAGS will be carried out based on average queue length and average response time.

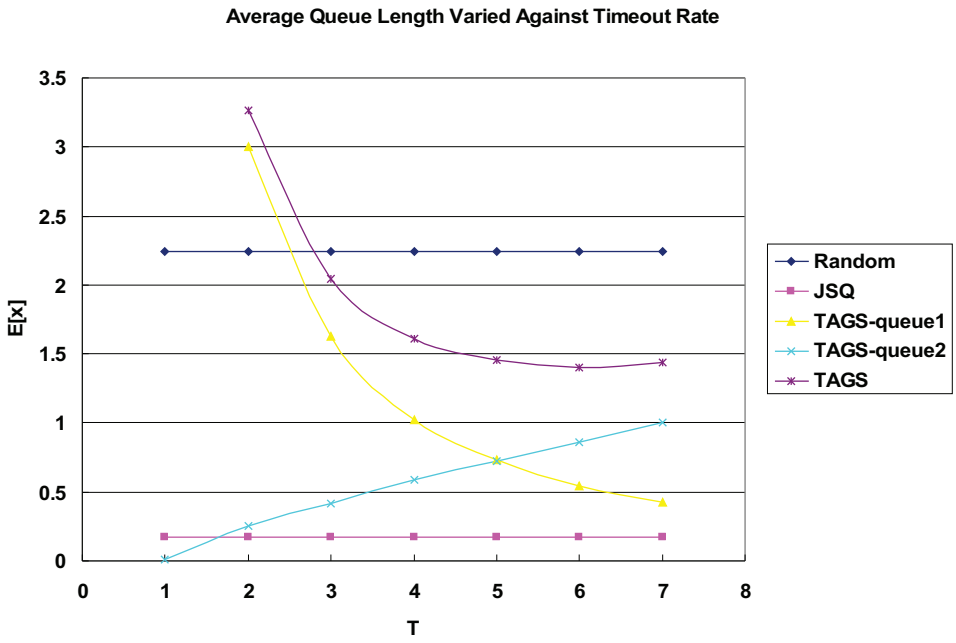


Figure 4.5 Average Queue Length Varied against Timeout Rate, λ=5, μ=10, k=1.5

Figure 4.5 shows the average queue length based on the timeout rate from 1 to 7. The graph clearly displays that JSQ has the least average queue length with the value about 0.18. Furthermore, when the timeout rate is less than 3, the average queue length of TAGS is larger than that of Random policy. However, as the timeout rate going up, the average queue length of TAGS grows down to a stable level around 1.5. When the arrival rate is greater than 3, the average queue length of TAGS reduces to a lower level than Random policy. In short, the performance of TAGS varies greatly at the beginning and then goes to a stable level.

In contrast to the situation in Figure 4.3 (k = 1.3), the performance of Random policy has an obvious

improvement. It denotes that the average queue length of Random has a dramatic decrease in contrast to other two policies. With the increase of k, the average queue length of Random keeps growing down and comes closely to the level of TAGS. In other words, compared with Random and JSQ, TAGS policy has lost its advantage in performance as k growing up.

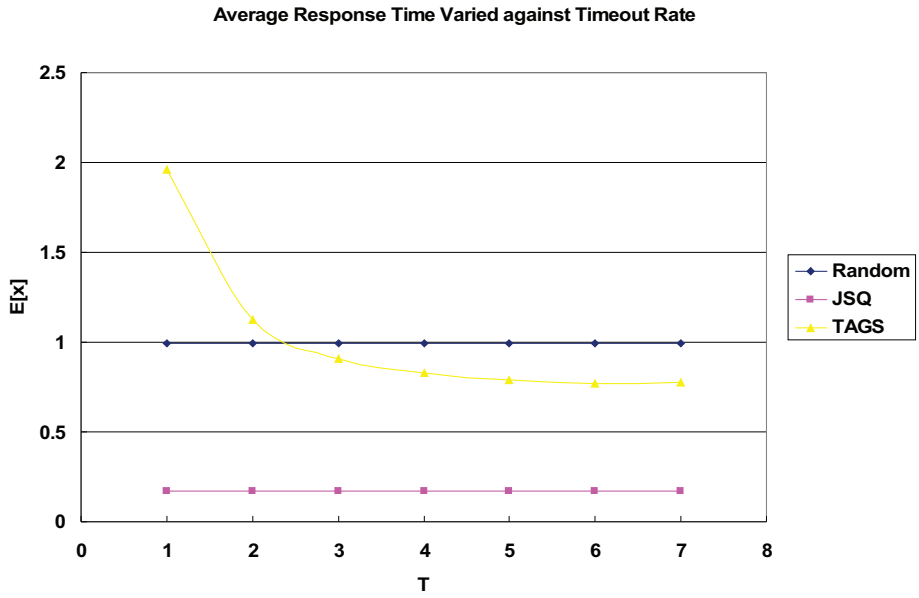


Figure 4.6 Average Response Time Varied against Timeout Rate, λ=5, μ=10, k=1.5

Figure 4.6 displays the average response time against timeout rate from 1 to 7. As shown in the line graph, JSQ is the policy with the least average response time nearly 0.17. TAGS policy has a larger value Random when the timeout rate is at a lower level, but its trend is towards a lower position and then it reaches a stable level around 0.8 under the line standing for Random policy.

Comparing with the Figure 4.4(k = 1.3), both TAGS and Random have a decrease in average response time which indicates that their performance is better than that showing in Figure 4.4. However, the value of Random is closer to TAGS in more depth. Hence, TAGS gradually loses its superior performance in contrast to Random.

To conclude, when the shape parameter k is increased from 1.3 and 1.5, the distribution becomes less variable than before. In such distributed environment, all three policies have improvements in performance. The performance of Random and JSQ tends to be better; however, TAGS policy gradually loses its advantage in performance. TAGS policy is no longer the best policy with less heavy-tailed distribution. In this situation, JSQ shows its optimal performance just as it performs under exponential distribution. Thus we could get the following conclusion:

TAGS policy is merely the optimal in much heavy-tailed distribution environment; the more heavy-tailed the distribution the better the performance of TAGS. On the other hand, JSQ is the best policy when the computing environment is not heavy-tailed or less heavy-tailed in distribution.

In order to provide more evidence for the preceding conclusion, further measurements will be carried on in next sub-section. The measurements are processed under highly heavy-tailed distribution in which k is quite close to 1 and much less heavy-tailed distribution with a larger value for k (usually k≥2).

4.4 Further Measurements with Pareto Distribution

In this section, further measurements will be processed in order to measure the performance of three policies under the two special Pareto distributions, as follows:

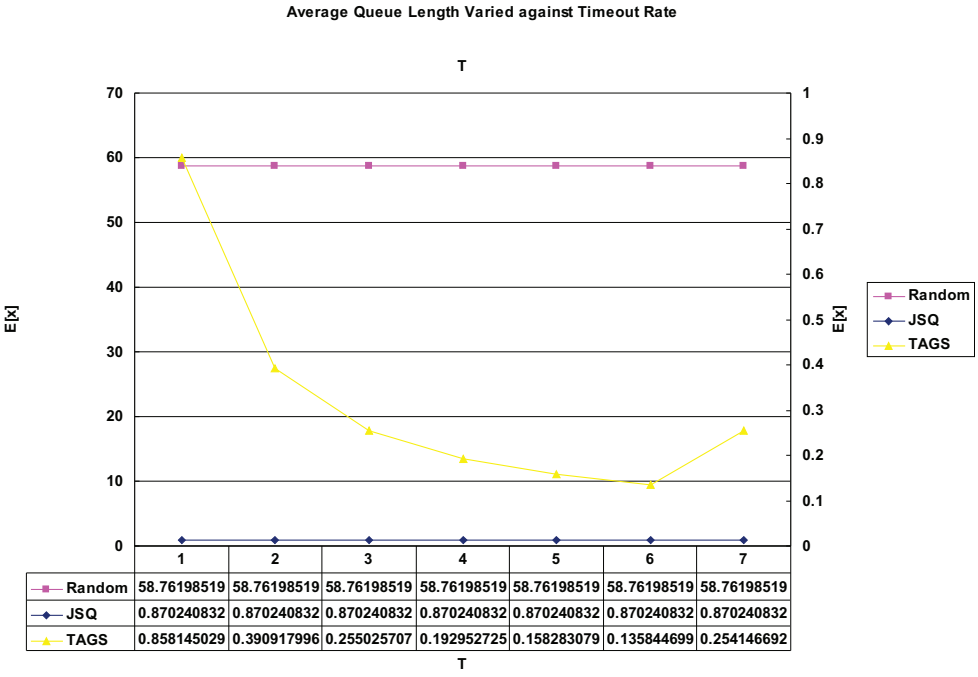
◆ the variability of distribution is at much higher level. To facilitate this environment, the shape parameter k of Pareto distribution needs to be specified with a value much close to 1. In this section,

the value 1.05 will be taken for k.
◆ the variability of distribution lies in a quite low level which means the heavy-tailed situation is unobvious. To achieve this, the shape parameter k should be set with a large value. Here the number 4.0 will be used for k.

The reason to measure the performance under such distinct conditions is to prove the previous conclusion in more depth. The conclusion can be described as follows: TAGS shows the optimal performance under highly heavy-tailed distribution, but its optimal performance will gradually fade away with the loss of the variability in distribution. Finally, its optimal performance will completely disappear in a non-heavy-tailed computing environment.

4.4.1 Measurements with Pareto Distribution, k = 1.05

This subsection illustrates the performance of three policies based on two aspects average queue length (against timeout rate) and average response time (against timeout rate). The shape of distribution is set to 1.05.

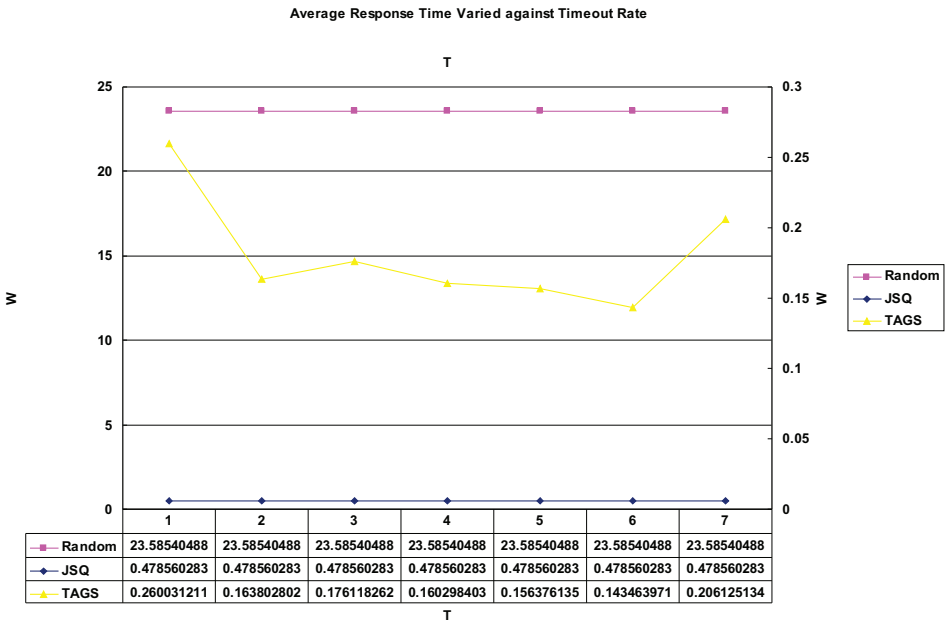


	1	2	3	4	5	6	7
Random	58.76198519	58.76198519	58.76198519	58.76198519	58.76198519	58.76198519	58.76198519
JSQ	0.870240832	0.870240832	0.870240832	0.870240832	0.870240832	0.870240832	0.870240832
TAGS	0.858145029	0.390917996	0.255025707	0.192952725	0.158283079	0.135844699	0.254146692

Figure 4.7 Average Queue Length Varied against Timeout Rate, λ=5, μ=10, k=1.05
X-axis (left): Random, JSQ
X-axis (right): TAGS

Figure 4.7 displays the average queue length against timeout rate varying from 1 to 7. As can be seen from the line graph, TAGS policy shows itself with the optimal performance, as its average queue length is the least of all. In contrast, Random policy has far more average queue length than both TAGS and JSQ, so Random has the worst performance. For JSQ, its average queue length is close to TAGS, but slightly higher around 0.87.

Comparing to the previous graphs (Figure 4.1, 4.3 and 4.5), TAGS has the optimal performance in a wider range of timeout rate. It means that the performance of TAGS is improved with the decrease of k.



	1	2	3	4	5	6	7
Random	23.58540488	23.58540488	23.58540488	23.58540488	23.58540488	23.58540488	23.58540488
JSQ	0.478560283	0.478560283	0.478560283	0.478560283	0.478560283	0.478560283	0.478560283
TAGS	0.260031211	0.163802802	0.176118262	0.160298403	0.156376135	0.143463971	0.206125134

Figure 4.8 Average Response Time Varied against Timeout Rate, λ=5, μ=10, k=1.05
X-axis (left): Random, JSQ
X-axis (right): TAGS

Figure 4.8 presents the average response time against timeout rate varied from 1 to 7. In the line graph, TAGS policy uses the least average response time, while Random policy has the largest value around 23 in its average response time. Regarding JSQ, its average response time is about 0.48 which is quite close to TAGS. Compared with the average response time showing in Figure, 4.2, 4.4 and 4.6, TAGS exhibits its first-rate performance in average response time. It means the performance of TAGS is improved gradually when the variability of distribution goes up (k is closer to 1.0).

4.4.2 Measurements with Pareto Distribution, k = 4

In this subsection, the measurements are based on the average queue length and average response time (against timeout rate). The shape parameter k uses the value 4.0. It is necessary to note that measurements here are processed against timeout rate starting from 10 to 30. According to the feature of Pareto distribution, when k is set to 4.0 each random value in such distribution becomes larger. In order to optimize the measurements, the timeout rate needs to be increased.

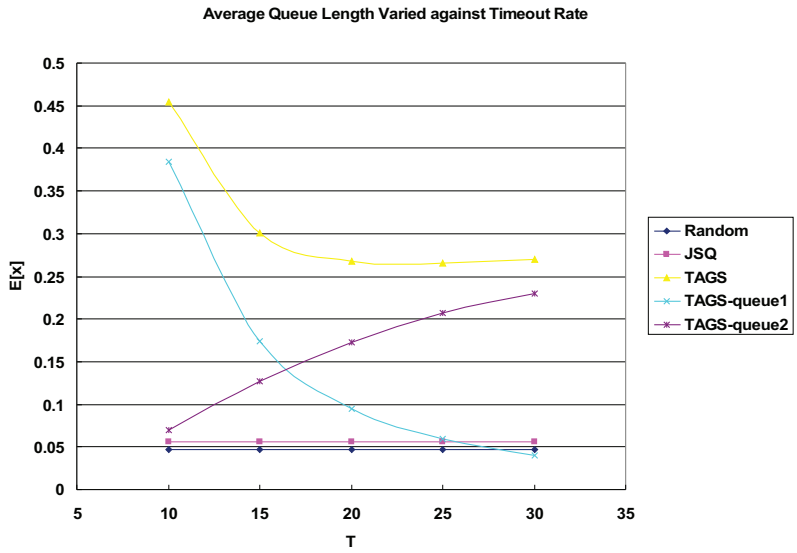


Figure 4.9 Average Queue Length Varied against Timeout Rate, λ=5, μ=10, k=4

Figure 4.9 represents the average queue length based on the timeout rate from 10 to 30. According to the graph, JSQ and Random performs better than TAGS policy. JSQ has the least average queue length which is under 0.05, but its value is quite close to Random whose average queue length is a little higher than 0.05. However, TAGS appears with a high value in average queue length that is above 0.25.

In contrast to the preceding graph (Figure 4.1, 4.3 and 4.5), the performance of TAGS goes from bad to worse. When the shape parameter k equals 1.5, TAGS even performs better than Random policy. However, after k is added to the value 4, the performance of TAGS becomes the worst of the three. The same thing also occurs in the comparison of average response time. This can be proved by the following figure, Figure 4.10. This graph shows the status of each policy by comparing the average response time against timeout rate from 10 to 30.

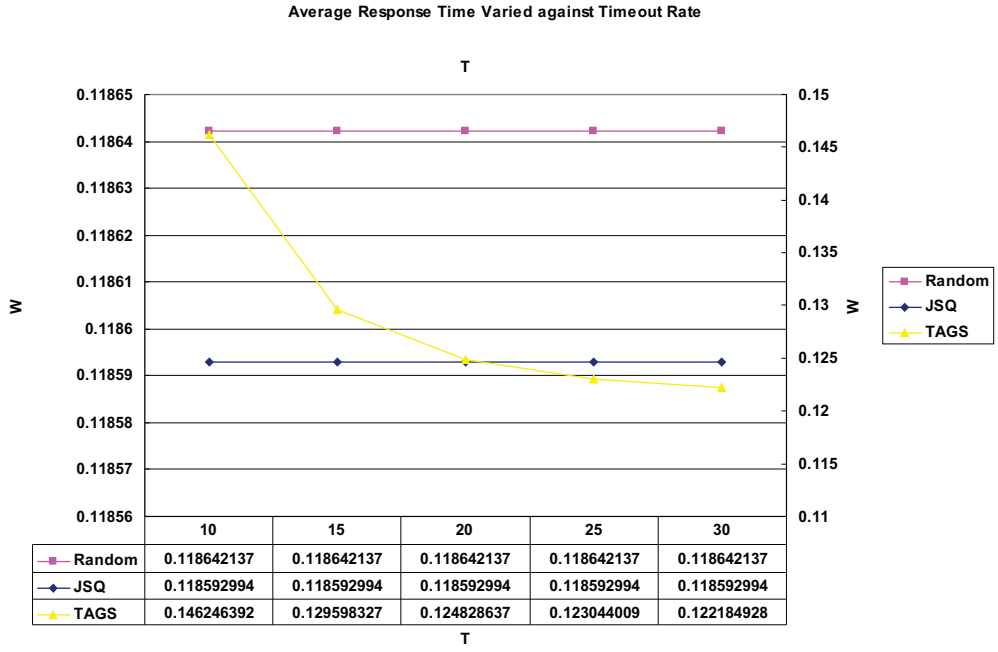


Figure 4.10Average Response Time Varied against Timeout Rate, λ=5, μ=10, k=4
X-axis (left): Random, JSQ
X-axis (right): TAGS

In Figure 4.10, TAGS uses the longest average response time over 0.12, while both JSQ and Random have very short values which are quite close to each other.

In conclusion, TAGS policy has the optimal performance in highly heavy-tailed distribution environment in which k is close up to the value 1.0. However, it tends to be inefficient and useless when the variability of distribution grows down (increasing value for k). In addition, JSQ is the policy having the optimal performance when the distribution is non-heavy-tailed. Random policy, being the simplest, usually has the similar performance to JSQ when the environment is non-heavy-tail distributed.

4.5 Summary and Discussion

In the preceding sections, measurements of three task assignment policies are processed under two sorts of distributed computing environments that are exponential distributed and Pareto distributed. According to measurement statistics, these three policies have shown different characteristics in performance under two different distributions.

In exponential distributed environment, JSQ task assignment policy is the optimal scheduling strategy. Furthermore, Random task assignment policy takes the second place in performance. However, Task Assignment by Guessing Size policy presents a negative performance from the measurements, and has obvious difference in ability of assigning tasks comparing with other polices. Consequently, TAGS policy is unsuitable for exponential distributed computing environment.

For Pareto distribution, it is an ideally excellent distribution to achieve the heavy-tail distributed environment. Pareto distribution is controlled by modifying both shape parameter k ($k > 1$) and scale parameter Xm. Theoretically Pareto distribution has a significant characteristic which is the smaller the value of k the more heavy-tailed the distribution. Based on the measurement results, when the shape k is specified with a much smaller value being quite close to 1.0, for example the values 1.05 and 1.1 used in this project, TAGS policy supports the optimal performance of all. Moreover, TAGS policy's performance has the tendency to become better while the shape k grows down. On the other hand, when the value of shape parameter k reaches a high level (such as, 1.5 or 4.0), the variability of Pareto distribution is not that remarkable any more. In this situation, TAGS policy gradually loses its optimal performance with the growth of k. On the contrary, JSQ policy appears to be the optimal in performance as the shape k goes up. As such, Random policy tends to a high quality in its performance.

In conclusion, TAGS policy has the optimal performance in heavy-tail distributed environment. On the other hand, both TAGS and JSQ are good policies as long as they are used in their own suitable environment. To select the best scheduling strategy for a distributed system, it is really essential to choose a task assignment policy being consistent with its related computing environment.

5 Conclusions

This project has aimed to delve into the performance of the novel scheduling strategy by means of computer based simulation. The project is interesting not only because it proposes a novel task assignment policy, but also it refers to the scheduling process that is a core section in distributed systems.

According to the measurements towards three target task assignment policies, the final conclusion regarding their performance is summarized as follows: Firstly, in exponential distributed computing environment, the novel task assignment policy, namely TAGS, was put into the shade. On the contrary, the other two traditional task assignment policies, JSQ and Random, showed their pre-eminent performance in the same environment especially JSQ policy which was the optimal of all.

Secondly, in heavy-tail distributed environment, each policy showed various characteristics in its performance and such variation occurred with the change of distribution. Whenever the variability of distribution was at a high level, which means the distribution was much more heavy-tailed, TAGS was the optimal policy. Its performance was improving with the growth of variability in distribution. In contrast, JSQ and Random lost their excellent performance in heavy-tail distributed environment and tended to a worse situation as the variability of distribution went up. In short, it was exactly proved that for TAGS the more heavy-tailed the distribution the better the performance. However, for JSQ and Random, their performance would go from bad to worse as the variation grew up.

Thirdly, regarding less heavy-tailed distribution environment, in which the variability of distribution reduced to a low level, TAGS policy turned to a bad performance which was the same as the performance in exponential distributed environment. However, in this situation, JSQ and Random performed better than in highly heavy-tailed environment. When the variability of distribution reached a quite low level, TAGS completely lost its optimal performance.

Finally, in order to optimize the performance of the whole system, it was shown that multiple task assignment policies could be combined as a new multi-scheme. This multi-scheme could make a considerable improvement in the performance of the system. This will be explored in the future work.

The strength of the project is that it clearly produced quite honest and accurate statistics to accomplish the performance of the novel scheduling strategy. Such strong point relies on the exact computer based simulation as well as correct measurement scheme. Perhaps the weakness of the chosen method of research lies in the fact that no attempt was made to explore the performance through creating a mathematical model. This is not achieved because of my limited mathematic background. If this had been done, the performance of novel scheduling strategy could have been analyzed in more depth.

Consequently, this project advocates that a further study should be undertaken to investigate a set of subjects referring to the novel scheduling strategy in more depth, and to achieve a greater understanding of the aforementioned factors.

Reference

[1] M. Harchol-Balter, Task Assignment with Unknown Duration, *Journal of ACM*, 49(2). 2002.

[2] N. Thomas, Modeling Job Allocation where Service Duration is Unknown, *Journal of IEEE*. 2006.

[3] W. Winston, Optimality of the Shortest Line Discipline, *Journal of Applied Probability.* 1977.

[4] B. Schroeder and M. Harchol- Balter, Evaluation of Task Assignment Policies for Supercomputing Servers: The Case for Load Unbalancing and Fairness*, Journal of IEEE*. 2000.

[5] V. Gupta, M. Harchol Balter, K. Sigman and W. Whitt, Analysis of join-the-shortest-queue Routing for Web Server Farms, *Elsevier B.V.* 2007.

[6] S. Kang and L. R. Lipsky, Steady-State Solution for Join-the-Shortest-Queue Policy in General Server Systems. 1996.

[7] W. E. Leland and T. J. Ott, Load-balancing Heuristics and Process Behavior, *In Proceedings of Performance and ACM Sigmetrics*. 1986.

[8] Mor Harchol-Balter and A. Downey, Exploiting Process Lifetime Distributions for Dynamic Load Balancing, *ACM Transactions on Computer Systems*. 1997.

[9] M. E. Crovella and A. Bestavros, Self-similarity in World Wide Web Traffic: Evidence and Possible Causes, *IEEE/ACM Transaction on Networking*. 1997.

[10] M. E. Crovella, M. S. Taqqu and A. Bestavros, Heavy-tailed Probability Distributions in the World Wide Web. *Chapman & Hall*, New York, 1998.

[11] V. Paxson and S. Floyd, Wide-area Traffic: The Failure of Poisson Modeling, *IEEE/ACM Transaction on Networking*. June, 1995.