International Journal of u- and e- Service, Science and Technology
Vol.8, No. 9 (2015), pp.51-60
http://dx.doi.org/10.14257/ijunesst.2015.8.9.06

A Comprehensive Study of Multi-Scheduling Schemes Performance

Xiao Chen[1], Lei Jiang[2] and Jin Wang[3]

[1] *School of Computer Science and Communication Engineering, Jiangsu University, Jiangsu, 212013, China*
[2] *Petrochina Southwest Oil & Gasfield Company, Luzhou, Sichuan, 646000, China*
[3] *College of Information Engineering, Yangzhou University, Yangzhou 225009, China*

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, efficient multi-scheduling schemes are employed to optimize the task assignment process. This paper closely explored the performance of multi-scheduling schemes through computer simulation. The research was started regarding the simulation of a novel scheduling policy (Task Assignment by Guessing Size) associated with other two previous task assignment policies (Random and JSQ). The multi-scheduling schemes involve two types: Random-TAGS scheme and JSQ-TAGS scheme. To facilitate the performance, computer simulation is applied to perform the statistical measurements. The findings were, indeed, very interesting, showing that the multi-scheduling scheme obtains better performance than single scheduling strategy scheme under heavy-tail distributed computing environment. Furthermore, JSQ-TAGS scheme is more efficient and stable in contrast to Random-TAGS scheme. The paper finally concludes by summarizing the findings from the simulation and suggesting a wider study be undertaking, in order to explore the performance of multi-scheduling schemes in more depth.

Keywords: *Performance evaluation, scheduling, JSQ, Random, TAGS*

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 scheduling scheme will produce an important effect on the system performance. It is no longer a new topic to find the best scheduling scheme for different distributed systems, but it remains a significant research direction.

In this paper, the performance of multi-scheduling schemes will be explored by simulating the entire scheduling process with Java program. The aim is to simulate different multi-scheduling schemes under heavy-tail distributed environment and then compare the numerical results so as to analyze the performance of multi-schemes. The most widely used task assignment policies in multi-scheduling schemes are Random task assignment policy (Random), Join the Shortest Queue policy (JSQ) and a novel policy TAGS (Task Assignment by Guessing Size). TAGS policy is a novel scheduling policy which has optimal performance in heavy-tail distributed environment [12]. So, in this

ISSN: 2005-4246 IJUNESST

paper, the multi-scheduling schemes are base on TAGS policy, which means in the distributed system, TAGS is applied as low level scheduling policy; however, in high level of system, the scheduling scheme is designed with Random policy or JSQ policy.

Here, two types of multi-scheduling schemes, including Random-TAGS and JSQ-TAGS scheme, will be simulated and analyzed. The performance of each scheme is presented in 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. In order to facilitate the measurement process in performance, SimJava 2.0, a Java package for simulation, will be applied in simulation to aid the statistical measurements.

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. Figure 2.1 shows the structure of the distributed system.

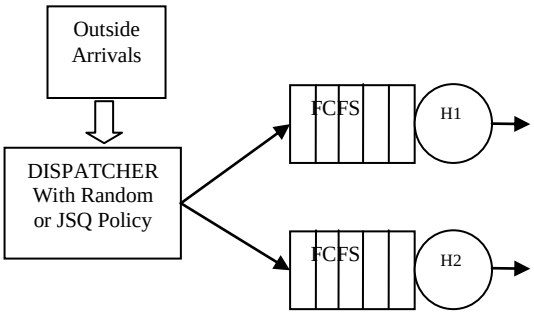

Figure 2.1. System Model with Random or JSQ

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), see Figure 2.1.

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 pre-emptive 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, see Figure 2.2. Firstly, we assume that h is the number of hosts (h1, h2, h3…) in the server system. Here, we assume there only two hosts in the 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 cannot 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.

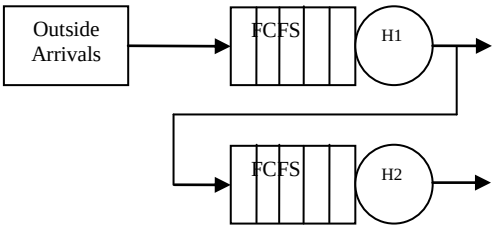

Figure 2.2. System Model with TAGS

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 cut-off 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. Performance of Single Scheduling Schemes

In this section, performances of three task assignment policies will be explored so as to provide statistical evidence for the design of multi-scheduling schemes in the following section. Here, the performance is based on computer simulation with Java programming. In order to simulate different scheduling policies in distributed system models and compare their performances conveniently, we assume that the computing environment is

based on exponential distribution and heavy-tail distribution respectively. In addition, the distributed system has only two hosts and single task assignment policy is applied for the scheduling process. Figure 2.1 and Figure 2.2 in previous section show system structures with those three single task assignment policies.

3.1. Performance Based on Exponential Distribution

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 in the simulation program and the mean of exponential distribution is specified as the reciprocal of arrival rate, namely $1/\lambda$. For the serving process at each server host, the service time is presented with a series of exponential-distributed numbers which have the mean $1/\mu$ (the reciprocal of service rate).

In this part, 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 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 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.

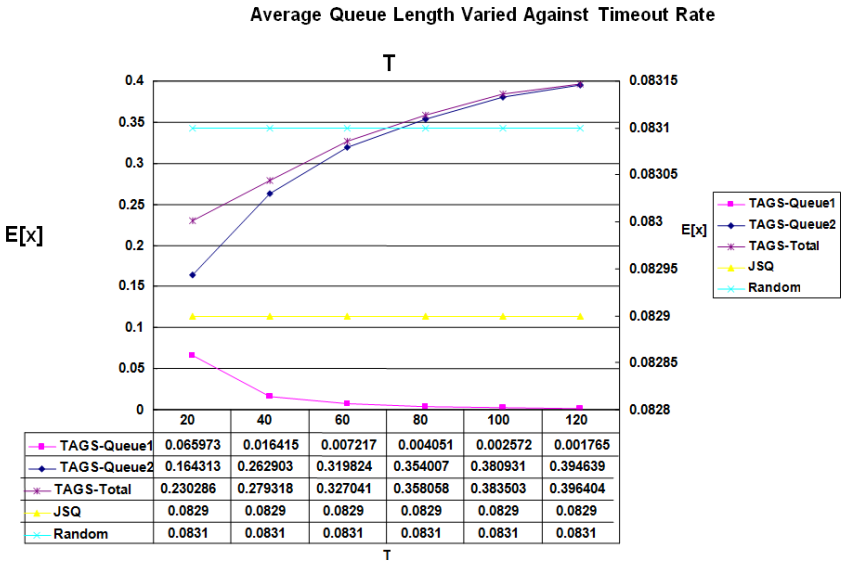

Figure 3.1. Average Queue Length Varied against Timeout Rate, λ=5, μ=10 X-Axis (Left): TAGS-Q1, TAGS-Q2, TAGS X-axis (right): Random, JSQ

Thus, in exponential distributed computing environment, TAGS has the worst performance in average queue length, while JSQ is the optimal.

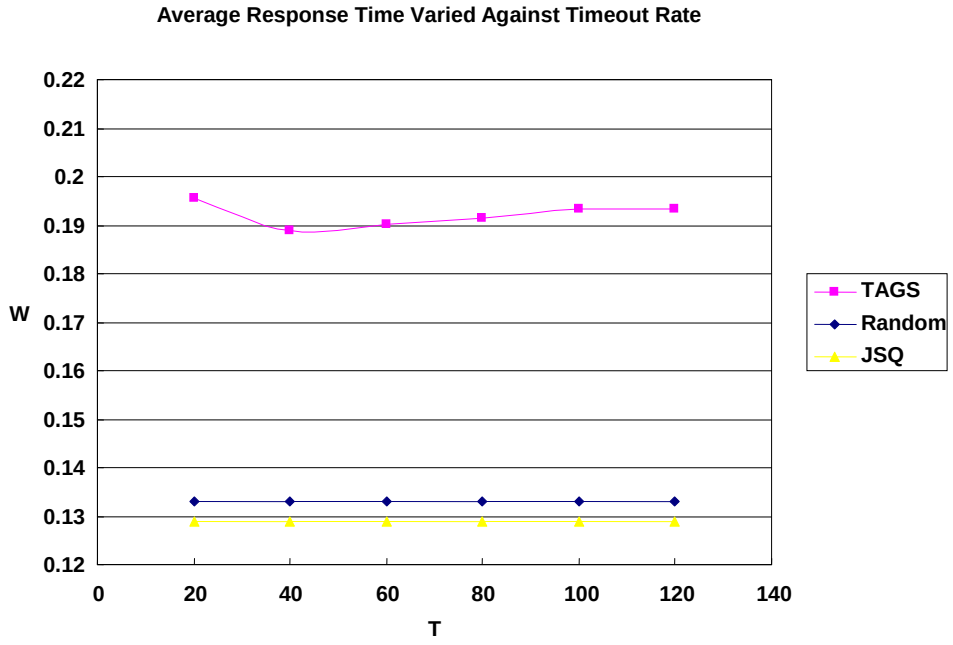

Figure 3.2. Average Response Time Varied against Timeout Rate, λ=5, μ=10

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.

To sum up, 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.

3.2. Performance Based on 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];

Generally speaking, in most situations, the value of k is somewhat larger than 1. Thus, in the Pareto distributed environment, TAGS policy has better performance when the variability of distribution is much high, namely the parameter of k is specified with a value much close to 1. In this paper, k value is specified with 1.05 to create much heavy tailed distribution.

Figure 3.3 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.

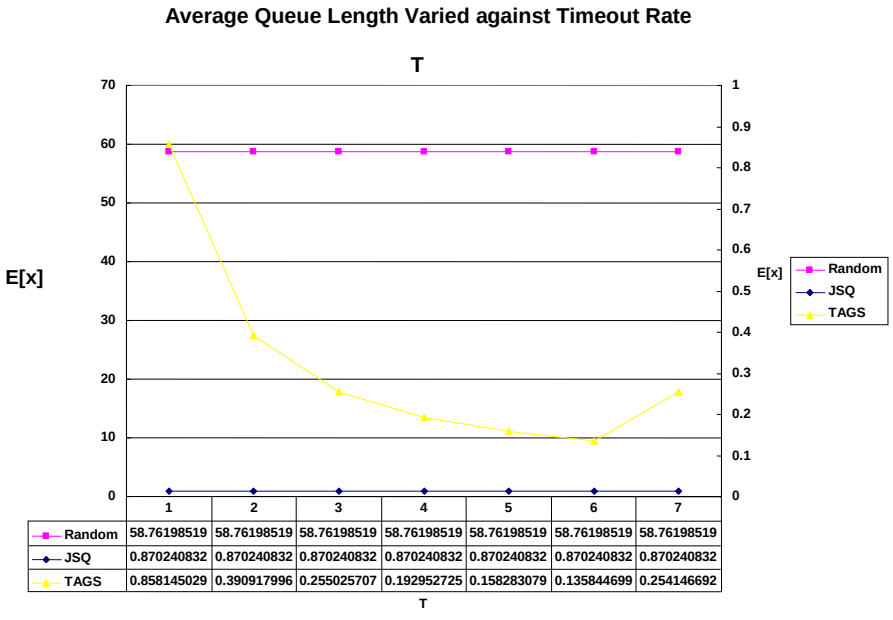

Figure 3.3. Average Queue Length Varied against Timeout Rate, λ=5, μ=10, k=1.05 X-axis (left): Random, JSQ X-axis (right): TAGS

Figure 3.4 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.

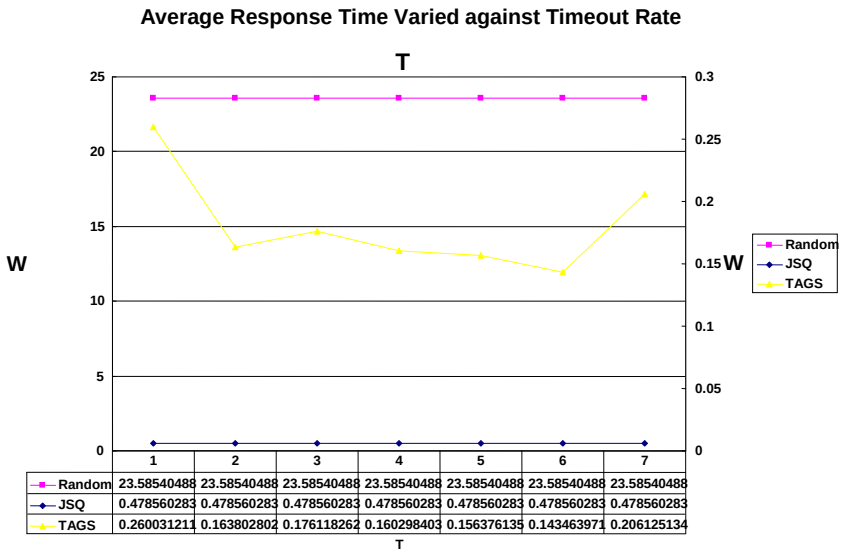

Figure 3.4 Average Response Time Varied against Timeout Rate, λ=5, μ=10, k=1.05 X-axis (left): Random, JSQ X-axis (right): TAGS

According to simulation results, TAGS get an optimal performance in both aspects, average queue length and average response time, when variability of distribution is quite high.

4. Design of Multi-Scheduling Schemes

In previous section, all measurements are based on the system which has only 2 hosts and single task assignment policy. Actually, there are more than two server hosts existing in the real-world distributed system. In such system, the scheduling process usually adopts multiple task assignment polices so that the efficiency of the scheduling process could be improved to a high level.

When more hosts and multi-scheduling scheme are applied, the system should run more efficiently. Hence, the multiple scheduling schemes should have better performance than that with single task assignment policy. This conclusion will be proved in this section.

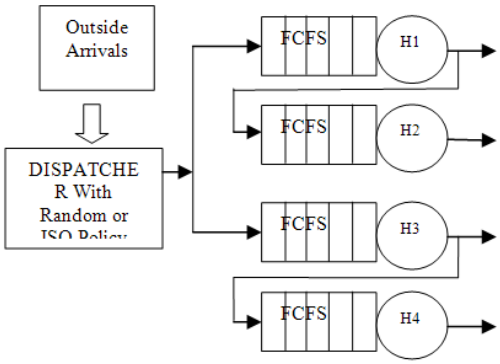

Figure 4.1. Distributed Systems with Multi-Scheduling Policies

To facilitate measurements for the system with multi-scheme, it is necessary to modify the previous system models. Since in Pareto distributed environment TAGS has the optimal performance, it should be applied in the low level subsystem. On the high level of the system, Random and JSQ can be employed. Thus, TAGS policy is used in company with Random and JSQ respectively. According to the preceding model design for Random and JSQ, each host in the system can be replaced by a 2-host subsystem which is organized with TAGS policy. The system structure is described in Figure 4.1.

In such a system, outside jobs are allocated to subsystems by means of Random or JSQ policy, and each subsystem serves incoming jobs with TAGS policy. Hence, TAGS policy is the head and front in such multi-scheme.

5. Measurements and Analysis

From the previous section, we know that TAGS policy has the worst performance in exponential distribution, while it has the optimal performance when the distribution is heavy-tailed. As this paper is aimed to explore the performance of TAGS policy in multi-schemes, it is necessary to carry out the simulation in Pareto distribution so that TAGS policy could produce the optimal performance.

As displayed in Figure 4.1, subsystems undertake the serving process in which hosts are organized with TAGS policy. In order to make hosts work in the highest efficiency and compare with single scheduling scheme, measurements will be processed in Pareto distributed computing environment. The shape parameter k is set to 1.05.

Figure 5.1 represents the average queue length of two multi-schemes against timeout rate from 2 to 7. The graph clearly shows that the average queue length of JSQ-TAGS multi-scheme is slightly shorter than that of Random-TAGS. When the arrival rate is at 5, both two multi-policies have almost the same average queue length. What is more, it is necessary to denote that the average queue length of each multi-scheme is less than 0.1.

In addition, with the rise of timeout rate, the average queue length of Random-TAGS scheme appears quite unstable, while JSQ-TAGS policy has a steadily decreased tendency.

In contrast to Figure 3.3 (the scheme with single policy and two server hosts), for both JSQ-TAGS scheme and Random-TAGS scheme, the average queue length has a dramatic decrease. This means the performance of multi-scheme is growing to be superior. Hence, when more server hosts are used in the system as well as multiple task assignment policies, the performance of the system will be improved to a much higher level.

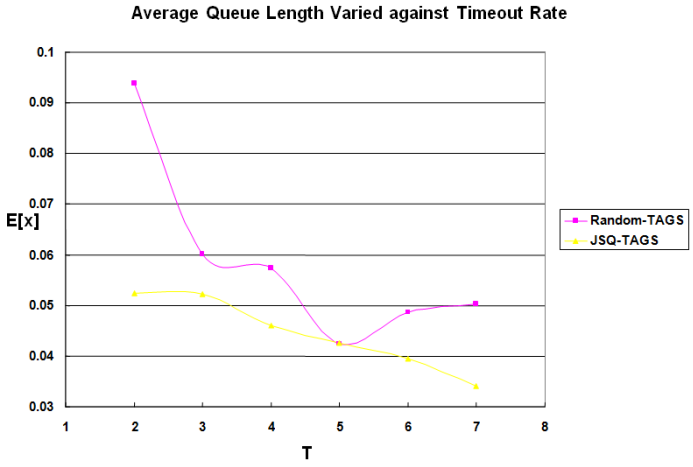

Figure 5.1. Average Queue Length Varied against Timeout Rate, λ=5, μ=10, k=1.05

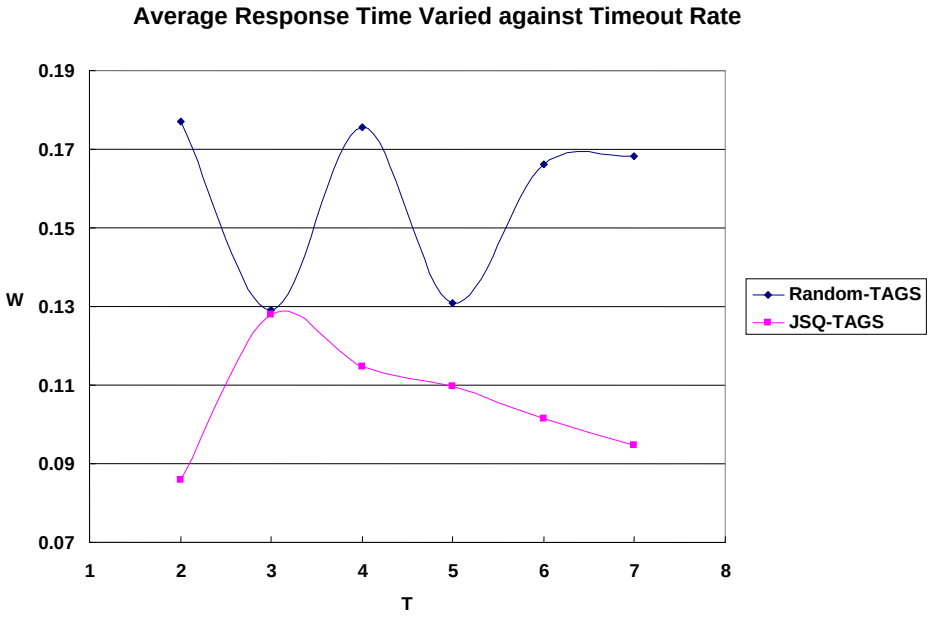

Figure 5.2. Average Response Time Varied against Timeout Rate, λ=5, μ=10, k=1.05

Figure 5.2 displays the average response time against timeout rate from 2 to 7. In this graph, the line standing for Random-TAGS scheme appears to be very interesting. The average response time of Random-TAGS is quite unstable, which fluctuates around 0.15. JSQ-TAGS scheme has less average response time than Random-TAGS scheme. When the timeout rate is 5, the average response time of JSQ-TAGS reaches the peak value roughly 0.13, then it tends to decrease with the increase of timeout rate.

Comparing with the lines in Figure 3.4 (the scheme with single policy and two server hosts), JSQ-TAGS scheme performs more efficiently than TAGS in Figure 3.4. However, Random-TAGS scheme shows similar performance comparing with TAGS in Figure 3.4. So, according the performance in average response time, JSQ-TAGS scheme is the first choice when multi-scheme is needed.

6. Conclusion

Through the analysis in previous sections, it is clear that when we use multi-scheduling scheme and add more server hosts in the system, the multi-scheme has better performance than the single scheduling scheme. Meanwhile, the whole system becomes much more powerful. However, according to the comparison between such two different multi-schemes, Random-TAGS scheme shows a quite unstable change in both average queue length and average response time; however, JSQ-TAGS scheme presents a stable and efficient performance. Consequently, in real-life computing environment, it is helpful to apply multi-scheduling schemes so as to optimize the system ability. On the other hand, it is necessary to choose the optimal multi-scheme with excellent performance in both stability and efficiency. Based on this research, in heavy-tail distributed environment, JSQ-TAGS multi-scheduling scheme is the best choice to optimize the system performance.

Acknowledgements

This paper is a revised and expanded version of a paper entitled "Performance Evaluation of Multi-Scheduling Schemes" presented at CST 2015, Suzhou, China, April

23-25, 2015. A special thank you goes to those who contributed to this paper: Dr. Nigel Thomas for his guidance and valuable comments. This work was supported by the Industrial Strategic Technology Development Program (10041740) funded by the Ministry of Trade, Industry and Energy (MOTIE) Korea, and the National Natural Science Foundation of China (61402234).

References

[1] M. Harchol-Balter, "Task Assignment with Unknown Duration", Journal of ACM, vol. 49, no. 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] M. Harchol-Balter and A. Downey, "Exploiting Process Lifetime Distributions for Dynamic Load Balancing", ACM Transactions on Computer Systems, **(1977)**.
[7] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web Traffic: Evidence and Possible Causes", IEEE/ACM Transaction on Networking, **(1977)**.
[8] M. E. Crovella, M. S. Taqqu and A. Bestavros, "Heavy-tailed Probability Distributions in the World Wide Web", Chapman & Hall, New York, **(1998)**.
[9] V. Paxson and S. Floyd, "Wide-area Traffic: The Failure of Poisson Modeling", IEEE/ACM Transaction on Networking, **(1995)** June.
[10] J. Wang, J.-U. Kim, L. Shu, Y. Niu and S. Lee, "A distance-based energy aware routing algorithm for wireless sensor networks", Sensors, vol. 10, no. 10, **(2010)**, pp. 9493-9511.
[11] J. Wang, Y. Yin, J. Zhang, S. Lee and R. S. Sherratt, "Mobility based energy efficient and multi-sink algorithms for consumer home networks", IEEE Transactions on Consumer Electronics, vol. 59, no. 1, **(2013)** February, pp. 77-84.
[12] R. Hong, J. Pan, S. Hao, M. Wang, F. Xue and X. Wu, "Image quality assessment based on matching pursuit", Inf. Sci., **(2014)**, pp.196-211.
[13] R. Hong, W. Cao, J. Pang and J. Jiang, "Directional projection based image fusion quality metric", Inf. Sci., **(2014)**, pp. 611-619.
[14] S. Kang and L. R. Lipsky, "Steady-State Solution for Join-the-Shortest-Queue Policy in General Server Systems", **(1996)**.
[15] W. E. Leland and T. J. Ott, "Load-balancing Heuristics and Process Behavior", In Proceedings of Performance and ACM Sigmetrics, **(1986)**.
[16] X. Chen and N. Thomas, "Performance of Novel Scheduling Strategies", UKPEW, **(2009)** July.

Authors

Xiao Chen, Dr Xiao Chen received his PHD degree in 2013 from Newcastle University in UK. His main research interests include performance modelling, smart environment and cloud computing.

Lei Jiang, MSC Lei Jiang received his Master's degree in 2014 from University of Electronic Science and Technology of China. His main research interests include cloud computing, embedded system and smart oilfield.

