Dec. 2015. Vol. 7, No.2 ISSN 2305-1493
International Journal of Scientific Knowledge
Computing and Information Technology

www.ijsk.org/ijsk

MULI AGENT SYSTEMS FOR INTEGRATING MULTIPLE SCHEDULING STRATEGIES

[1]ARWA IBRAHIM AHMED, [2]REDA A. AMMAR

[1]Asstt Prof., Department of Information Systems, Princes Noura University, KSA
[2]Prof., Department of Computer Science and Engineering, University of Connecticut USA
E-mail: [1] ariahmed@pnu.edu.sa, [2]reda.ammar@uconn.edu

ABSTRACT

In this paper, we use MAS technology to integrate several scheduling strategies together. Our motivation is seeking the highest performance due to using the best strategy for the given flow of requests to the computer system. In terms of contribution, the paper provides a short survey of the most popular scheduling strategies, and describes concepts and methodologies within the field of Multi-Agent Systems (MAS) that are appropriate for making the right selection among several scheduling strategies for a given data set. . Different categories of agents, their internal structures and functions are used in building the target system. Programming solutions to MAS framework including data interactions among agents and the scenario to implement the proposed plan are also developed. The plan has a feedback path to continuously monitoring and selecting the best scheduling strategy to apply for the given batch of requests.

Keywords: ***Multi Agent Systems (MAS), Artificial Intelligence, Scheduling strategies, System Performance. System Integration***

1. INTRODUCTION

Scheduling has been perhaps the most widely researched topic in the literature of Computer Science and Engineering and many other fields [1-8]. It is also a well-structured and conceptually-demanding problem. Improper or inefficient scheduling may degrade system performance and can offset the advantages of modern day high-speed processors, especially in real time applications where tasks need to complete before their deadlines. The need for efficient scheduler increases when there is a continuous stream of requests from statistically diverse sources to the web servers and data warehouses.

Flow size distribution often exhibits a high variability property. Most of the time they are short and a tiny fraction of the largest flows constitutes more than half of the total load [5]. Recent studies have revealed that network traffic exhibits multiple time scale traffic [7-8]. The study shows that the web and data servers receive a continuous stream of request from sources that are statistically vulnerable. In other words, execution times of requests vary over several orders of magnitude. In the light of these findings it is interesting to revisit the scheduling policies to see if one can improve the number of requests satisfied per second by a web server or the data warehouse controller.

Typically requests are stored in a buffer or a queue and then processed in the order of their arrivals using FCFS policy [5]. The FCFS scheduling model which is used in most of the existing web server and data warehouses does not provide high quality of services because the completion time of a request depends on how many requests are already in the queue.

FCFS policy has nothing to do other than serving requests in the order of job arrival. There is no scope of paying attention to job size variation criteria. Short jobs suffer from longer delays due to the presence of longer jobs in front. Instead of FCFS, other scheduling algorithms such as Round-Robin algorithm are used to improve system performance, whether the performance parameter is the system time (in non-real time case) or the

number of jobs meeting deadline (in real time system case).

Previous scheduling policies including our publications [2,7,8] focus on dynamically implementing one scheduling policy. In this paper, we compared the performance of static scheduling strategies [3-6] namely FCFS, Shortest Request First (SRF) and Round Robin (RR). These strategies were selected due to their simple implementation and demonstration for the purpose of this research study. However, our integration approach is very general to add more complex scheduling strategies for real-time application such as Residual Time Based (RSB) scheduling [7-8].

In this paper, we consider dynamic scheduling in which we use several scheduling algorithms instead of using a single one. To make this possible we are integrating these algorithms using a Multiple Agents System [9-12]. Two agents Information Agent (IA) and Evaluation Agent (EA) are responsible of one scheduling policy. Finally a quality agent considers the recommendation of different scheduling policies and select the choice that minimizes the overall response among them. While the selected request is being executed by the execution agent (XA), information agents and evaluation agents start assessing the response time via a look-ahead timing diagram..

In this study, we integrate three scheduling strategies, namely First Come First Served (FCFS), Round Robin (RR), and Earliest Deadline First (EDF) in scheduling jobs. Which one of these be implemented at a given point of time will be decided by a Multiple Agent System (MAS). Our objective is achieving a better overall average response time relative to using a single scheduling strategy alone.

Rest of the paper is organized as follows: in section 2, we describe briefly the three scheduling strategies along with deriving the overall average performance if this strategy has been selected. In section 3, we briefly describe the Multiple Agent System. In section 4, we describe the integration algorithm in details with numerical examples. We conclude the paper in section 5.

2. SCHEDULING STRATEGIES

2.1 Task model

Tasks are assumed to be independent, identically distributed, and preemptable. They are characterized by the following:

- task arrival time, A_i
- task execution time, E_i
- task deadline, D_i
- task start time, S_i

Task, i is ready to execute as soon as it arrives, regardless of processor availability. Tasks may have to wait for some time, W_i, before they start executing for the first time due to scheduling decision. As a result, we have:

$$S_i = A_i + W_i, \quad \text{where} \quad W_i \geq 0$$

Turnaround time is calculated as time of submission of a request to the time of completion of the request.

Selecting of a scheduling strategy is a decision process that follows many metrics. In this paper, we plan to use the average waiting time and the average turnaround time.

2.2. Scheduling strategies

There are many scheduling strategies available in the literature [1-8] We are not introducing a new strategy in this paper but we rather use multiple of them simultaneously and switching among as needed to reduce the average waiting time and/or the turnaround time. We consider three strategies: First Come First Serve, Shortest Request First and Round Robin. Below is a brief definition of each.

2.2.1. First Come First Serve (FCFS) Scheduling Strategy

The tasks are allowed to join the arrival pool as soon as they arrive. The scheduler selects the first request in the queue and executes it until completion. This policy does not need any estimation of execution time for each request or its probability distribution. It only requires arrival rate λ, and the average execution time, $\overline{E}$, and coefficient of variation, C^2 to use Pollaczek-Khinchin (P-K) formula in calculating the average response time. It worth to note that $\overline{E}$ and C^2 are assumed to stay with no change at any time. The average turnaround time is $\overline{R}$ and the average waiting time is $\overline{R} - \overline{E}$ Both values can be calculated on short term basis using Gantt Chart.

2.2.2. Shortest Request First

The scheduler starts with the shortest request first. It arranges the requests with the least execution time in head of the queue and longest execution time in tail of the queue. This requires advanced knowledge or estimations about the time required for a request to complete [5]. This algorithm is designed for maximum throughput in most scenarios.

The scheduler keeps the queue sorted using the estimated execution time. It insert each new arrival into the right place in the queue to keep it ordered according the shortest request first. It also initiated a Gantt chart at the beginning and update it with the arrival of each new request.

2.2.3. Round Robin (RR):

The scheduler selects the first request in the queue and executes it for a given time-slice (time-quantum). At the end of the time slice the request is appended at the back of the queue. Thus, the requests share the processor time in a Round Robin fashion.

The scheduler initiates a Gantt chart at the beginning and updates it with the arrival of each new request. According to the round robin strategy. The scheduler builds the Gantt chart virtually for a complete cycle. It uses this Gantt chart to calculate the average waiting time and the average turnaround time.

3 MULTIPLE AGENT SYSTEM

An agent is a computer system/component situated in some environment, and is capable of autonomous action in order to meet its required objectives. It is usually internally motivated, embedded, and adaptive with inferential capability, communication ability and mobility.

3.1. MAS Framework

MAS establishes a framework for continuous process improvement. Architecture of the system is composed of multi-layers to specify the current performance and future direction, understanding, control and improvement. Each layer governed by type of agent with different capability; environment and knowledge from agents in the other layers. The number of agent in each layer depends on the size of the system, process and needs.

The first layer establishes when the first type of agent called info agent (IA) is attached to different and homogenies environments in the organizations to collect all the system process activities, information and performance. There is a separate IA affiliated with each one of the scheduling strategy listed above. Moreover, it records the result in the measurement repository as an output of the first layer.

The Second layer is the backbone of the architecture is govern by Evaluation agent (EA), it has sub agents. Each subagent is affiliated with one of the scheduling strategy to calculate different metrics listed above including the average waiting time and the average turnaround time.

EAs produces evaluation report as an output of the second layer, and send it to quality agent (QA).

The third layer is managed by "Quality agent" which responsible for determining which scheduling strategies is implemented and send its decision to all IAs and EAs.

The forth layer is composed from execution/implementation agents. Each scheduling strategy has its own XA. Each XA receives the decision from the QA and implement it if its strategy is the selected one.

3.2 Internal architecture and function of each agent category

3.2.1 Information agents (IA)

Info agents works in the first layer. Our system has three IAs Info agents install themselves and a collect an image of the queue of requests. It monitors its queue and report all information needed to the Evaluation Agent affiliated with its scheduling strategy. It sends it to the measurements repository.

3.2.2 Evaluation agents

Evaluation agent is the heart of the system; worked in the second layer, it is proactive with intelligent features to answer the critical question of "what if the affiliated scheduling strategy is implemented". It evaluates the values $\overline{E}$ and C^2 and $\overline{R}$

3.2.3 Quality agent (QA)

Quality agent works in the third layer. It follows the MAS key process area to identify issues that must be addressed to achieve a performance level. The numbers of quality agent (QA) depend on the situation of the system and the ability to produce many suggestions to improve the performance levels. In this study we use one QA only. It

compares the $\overline{R}$ values (and/or the turnaround time) of different scheduling strategies as reported by EAs, decides which strategy achieves the best system performance at this point of operation and broadcasts the decision to XAs.

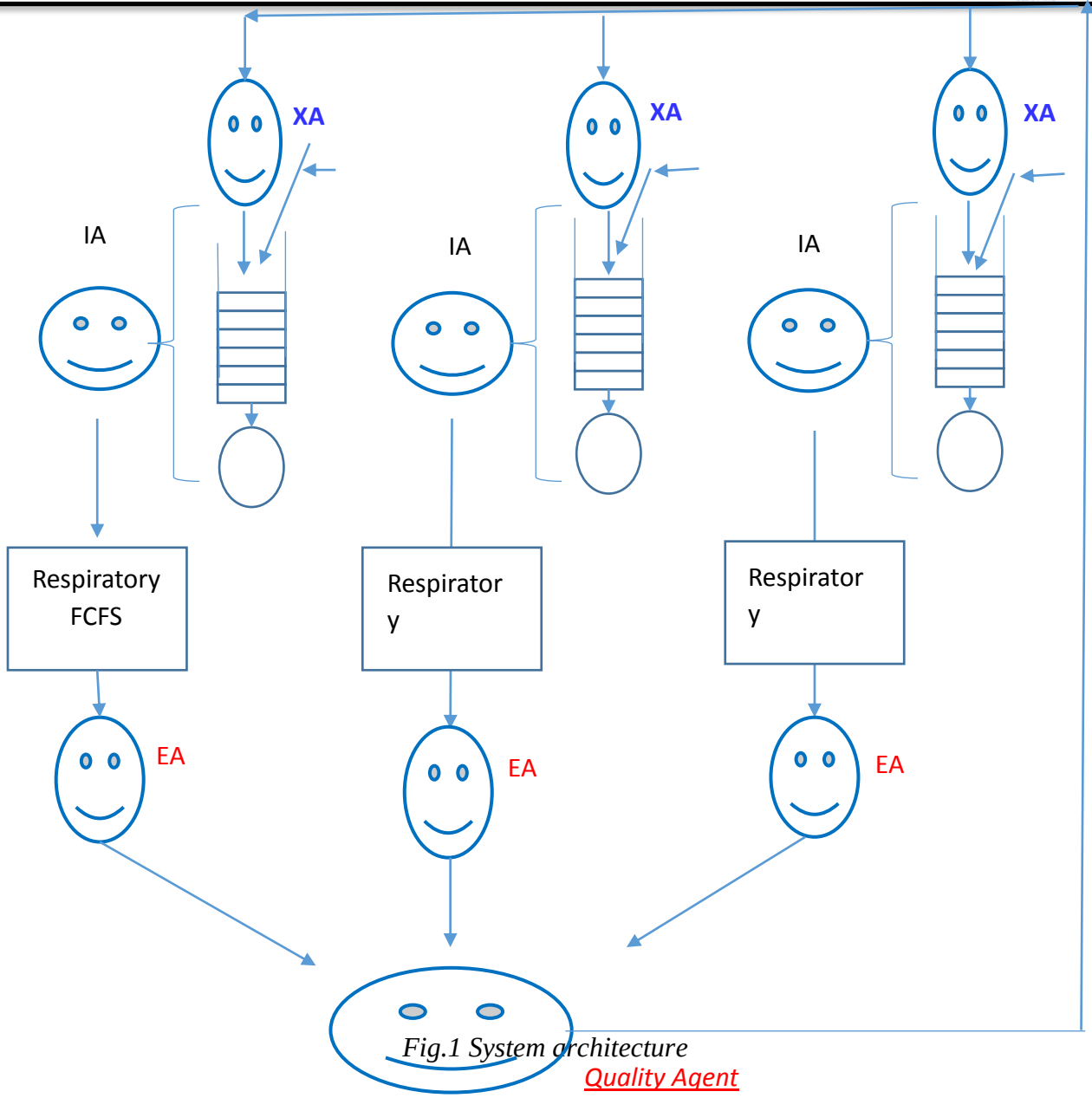


Fig.1 System architecture

Dec. 2015. Vol. 7, No.2 ISSN 2305-1493
International Journal of Scientific Knowledge
Computing and Information Technology

www.ijsk.org/ijsk

3.2.4. Execution Agent (XA)

Execution agent works in the fourth layer. There is one XA for each scheduling strategy. It follows the MAS key process areas to implement the recommended decision; namely the selected scheduling strategy as follows:

1) In case of FCFS, the first request in the queue proceeds to the processor, be removed from all other queues.
2) In case of SRF, the shortest job is removed from all of the queues.
3) In case of RR, the execution time of the selected request is updated in each copy of the queues of requests. If the new time is less than or equal zero, the request is removed from all queues. .

3.3. Activities due to new arrivals

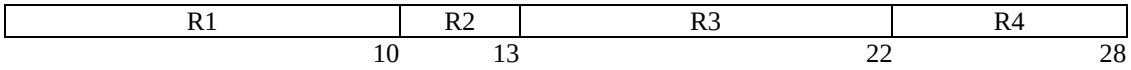
Independently from MAS, request arrivals will continue without any interference from all agents. Each EA adds the new arrivals, organizes the queue following the scheduling strategy and starts preparing for the next assessment cycle.

4 Numerical Example

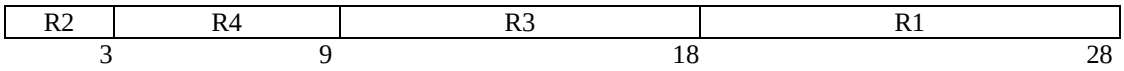
Consider the following set of requests along with their estimated execution time:

Request number	Estimated Execution time in ms
R1	10
R2	3
R3	9
R4	6

a) FCFS

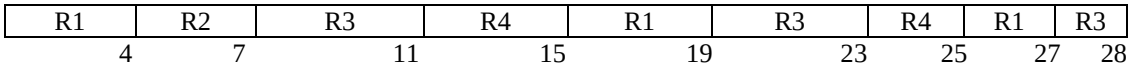


b) SRF



c) RR

Assign time quantum as 4 ms for each request



Average waiting time

	FCFS	*SRF*	*RR*
R1	0	18	25
R2	10	0	4
R3	13	9	27
R4	22	3	23
Average	11.25	7.5	19.75
deviation	9.069179	7.937254	10.62623

For the given data, the SRF gives the best average waiting time and the lowest deviation. RR achieves the lowest coefficient of variation.

Turnaround time

	FCFS	*SRF*	*RR*
R1	10	28	27
R2	13	3	7
R3	22	18	28
R4	28	9	25
Average	18.25	14.5	21.75
deviation	8.261356	10.90871	9.912114

For the given data, the SRF gives the best average waiting time. However it is the worst in the devotion. FCFS achieves lowest deviation. But RR achieves the smallest coefficient of variation. This gives flexibility to the system to use the appropriate scheduling strategy needed at a specific time and can switch from one to the other based on the target performance.

5. INTEGRATE SCHEDULING ALGORITHM FOR REAL_TIME APPLICATIONS

The integrated scheme described above is for static scheduling that have been developed for non-real-time applications. The new integrated scheduling combines them into a dynamic scheduling but for non-real-time applications. Fortunately, same approach can be developed for dynamic scheduling real-time applications. We can combine the following two new algorithms in this case:

5.1. EDF (Earliest Deadline First)

Earliest Deadline First (EDF) is a deadline driven algorithm, which can be applied for real time jobs. Jobs are arranged in a queue with a descending order of their deadline time. We have implemented EDF as a processor-sharing algorithm. The scheduler selects the first job in the queue and executes for a given time-slice (time-quantum). At the end of the time-slice the job is inserted in the queue so they remain sorted in a descending order of their deadline. The jobs keep on sharing the processor until they are done.

5.2. Residual Time Based (RTB) algorithm

The algorithm uses the knowledge of the complete execution time and probability distribution. In the process of selecting the next job, RTB gives higher priority to those who are close to deadline as opposed to those who have already passed the deadline. This enables the approach to increase the probability of the number of jobs meeting deadlines

Same MAS architecture is used in both cases. But they are different in the collected data by the IA and definitely different in metrics calculated or monitored by the Evaluation Agent. The Quality Agent uses different associative rules to select the best algorithm that achieves the required deadline for most of the requests. This is different objective from the non-real-time case.

6. Managing the Integrated Scheduling Strategy

The objective of using several scheduling policies simultaneously is utilizing the differences in the statistical properties of the sources of the requests to be handled. The Quality Agent analyzes the content of the queue and selects the most appropriate scheduling strategy for this batch of requests.

The QA is functioning either periodically (every t time units) or when the queue has n requests. It is possible to mix the two approaches based on the application. The value or t and/or n should be selected with two goals in mind:

a) Reducing the number of switches from one scheduling strategy to another and hence have an acceptable level of overheads (extra time to be lost in resetting from one scheduling strategy to another).
b) Finding the best scheduling algorithm for each request in the queue and hence achieves the deadlines and/or maximize the throughput.

7. CONCLUSION

In this paper, we use MAS technology to integrate several scheduling strategies together. Our motivation is seeking the highest performance due to using the best strategy for the given flow of requests to the computer system. We succeeded to convert a set of static scheduling policies for non-real-time applications into a dynamic scheduling strategy that takes advantages from the variation in the statistical behavior of the stream of requests. This strategy is either implemented periodically

every t time units or whenever the general buffer is full.

Same approach can be also used to integrate dynamic scheduling of real-time applications. The objective in this case is not only increase the throughput of the system but achieves the deadlines in the majority of requests.

REFERENCES

1. Ankur Bhardwas, Rachhpal Singh, Gaurav, "Comparative Study of Scheduling Algorithm in Operating System" International Journal of Computer and Distributed Systems, Vol. No. 3, Issue I, April-May 2013.
2. Ahmed Hussien, Abeer Hamdy, Reda Ammar, " Efficient Processing Power Utilization to Improve scheduling Real Time Tasks", International journal of Computer Science Issues, IJCSI Volume 9, Issue 5, September 2012.
3. Lalit Kishor, et. al ""Optimized Scheduling Algorithm" International Journal of Computer Applications® (IJCA), 2011, PP 106-109.
4. Md. Mamunur Rashid and Md. Nasim Adhtar, " A New Multilevel CPU Scheduling Algorithm", Journals of Applied Sciences 6 (9): 20362039,2009
5. Michael L. Pinedo, Scheduling: Theory, Algorithms, and Systems 4th ed. Springer; 4th ed. 2012 edition (January 6, 2012)
6. Mr. Sindhu M, Mr. Rajkamal R and Mr. Vigneshwaran P, "An Optimum Multilevel CPU Scheduling Algorithm". 978-0-7695-4058-0/10 $26.00 © 2010 IEEE
7. Sarah Tasneem, Reda Ammar, Lester Lipsky, and Howard Sholl "Improvement of Real-Time Job Completion Using Residual Time-Based (RTB) Scheduling", International Journal of Computers and Applications, Vol. 17,no. 3 pp. 117-132, September 2010.
8. Sarah Tasneem, Lester Lipsky, Reda Ammar, and Howard Sholl "A Residual Time Based Scheduling: Performance Modeling in M/G/C Queueing Applications", Journal of Software Engineering and Applications, pp.746-755, August 2010.
9. Niazi, Muaz; Hussain, Amir (2011). "Agent-based Computing from Multi-agent Systems to Agent-Based Models: A Visual Survey". Scientometrics (Springer) 89 (2): 479–499. doi:10.1007/s11192-011-0468-9.
10. Kubera, Yoann; Mathieu, Philippe; Picault, Sébastien (2010), "Everything can be Agent!" (PDF), Proceedings of the ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'2010) (Toronto, Canada): 1547–1548
11. Fabio Luigi Bellifemine, Giovanni Caire, Dominic Greenwood, "Developing Multi-Agent Systems with JADE", Wiley 2007.
12. Yoav Shoham and Kevin Leyton-Brown, Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations Cambridge University Press, 2009