ARTICLE IN PRESS

+ MODEL

Available online at www.sciencedirect.com

ScienceDirect

Karbala International Journal of Modern Science xx (2017) 1–18

http://www.journals.elsevier.com/karbala-international-journal-of-modern-science/

Thread scheduling using ant colony optimization: An intelligent scheduling approach towards minimal information leakage

Kushal Anjaria*, Arun Mishra

Department of Computer Science & Engineering, DIAT, Pune, India

Received 21 February 2017; revised 17 June 2017; accepted 10 August 2017

Abstract

In multithreaded programs, scheduler controls the execution of threads. As a result, the scheduler may arrange the execution sequence of threads in such a way that multithreaded programs may violate the non-interference confidentiality policy. Due to a violation of non-interference, multithreaded programs may leak security sensitive information. In the proposed work, Ant Colony Optimization (ACO) based intelligent scheduling policy has been proposed in the form of algorithms to schedule threads in multithreading environment in such a way that the execution sequence leads to minimal information leakage. In the present work, the proposed scheduler also deals with conflicting scheduling parameters and provides the algorithmic solution which can handle all conflicting entities of scheduling like throughput, delay, security-privacy and fairness. In this work, dynamic creation and deletion of threads are also handled during the scheduling. Although the focus of this work is on the scheduling of threads, the proposed policy can be used as a general purpose scheduling policy in many computing fields.

Keywords: Multithreading; Ant colony optimization; Swarm computing; Scheduling; Intelligent computing

1. Introduction

In simple terminology, Multithreading is the ability of the processor with a single core or multiple cores to execute more than one thread using the scheduler in such a way that it appears to the user that threads are being executed concurrently. As a result, in multithreading, scheduler is the major factor as it can affect information flow of the multithreaded program [1].

In general, commonly used schedulers are First Come First Serve (FCFS), Round Robin (RR), Time Division Multiple Access (TDMA), Priority schedulers and Shortest Job First (SJF) schedulers. The multithreading and the concurrency are ubiquitous in modern computer programming. In multithreading also, the aforementioned schedulers are used. The details of these schedulers are incorporated in Table 1:

In multithreaded programs, the scheduler will control the execution of threads. In other words, the scheduler will be shared by the threads of multithreaded programs. As suggested by Kadloor et al. in [2], shared resources among the entities always lead to information leakage. The shared scheduler in the

* Corresponding author.
E-mail address: kushal.anjaria@gmail.com (K. Anjaria).
Peer review under responsibility of University of Kerbala.

http://dx.doi.org/10.1016/j.kijoms.2017.08.003

Table 1
The details of commonly used scheduler in multithreading.

	Name	Characteristics
1	FCFS	This scheduler serves jobs in the order in which it receives the job.
2	RR	One job is served in succession from each queued up sources.
3	TDMA	Each source is assigned a predefined time in which the scheduler serve the job assigned from the particular source.
4	SJF	The jobs are selected by the scheduler based on the processing time. The job with shortest processing time is served first by this scheduler.

multithreaded environment also leads to information leakage. An example of shared scheduler leaking information was provided by Russo and Sabelfeld, in [1]. Consider the example:

Th_1: high: = 0; low: = high.

Th_2: high: = secret_content.

In the example, two threads Th_1 and Th_2 are being executed. This multithreaded program is divided into two portions i.e. low-security sensitive content and high-security sensitive content. In the program, the variable 'low' is low-security sensitive variable and the variables 'high' and 'secret_content' are high-security sensitive variables. Both the portions are interfering with each other. If both the threads are executed in isolation, then the program is secure, but if both the threads are being executed under the effect of scheduler then they may leak information. In isolation, the outcome of thread execution will not change i.e. output of thread Th_1 will always 0. Thus, the attacker of the program will not gain anything. But if the same program is executed under the effect of the scheduler then it is possible that the thread Th_1 may be executed first by the scheduler. So the statement high: = 0 will be executed first. After that, the thread switch may occur and the thread Th_2 may be executed. After that again the thread switch may occur and statement low: = high may be executed. If the statement low: = high is executed last then the program may leak the secret content. Thus, the problem of information flow in the multithreaded program, affected by the shared scheduler still remains an open research challenge [3]. The present work tries to address this challenge.

The characteristics required by the schedulers in multithreaded programs for information flow security are:

- **Permissiveness:** Presence of scheduler in the multithreaded program enables new attacks which are not possible in the sequential environment. In presence of shared scheduler information leakage through the covert channel and side channel attacks is also possible. Consider the example provided by Kadloor et al. in [1]. The example was general and not exactly related to multithreading. "Alice and Bob are executing their jobs on the shared resource. Bob is facing the delay in executing his task. That means he can assume that Alice is executing her task on the shared resource. Bob can know the job execution time of Alice and based on this execution time, he can craft the attack on Alice's job". This example leads to the timing covert channel and traffic analysis. The attacks that can be crafted based on the traffic analysis are information from the keystroke [4], knowledge on the visited website [5] and words on VoIP [6]. These attacks can also be crafted to gain the secret content from a multithreaded program. Thus, the scheduler should permit thread execution in such a way that information leakage can be mitigated or minimized.
- **Scheduler independence:** The scheduler independence property means the security of multithreaded program execution should be made independent of the scheduler. In other words, it can be said that this characteristic conveys to separate scheduling policy and security policy from each other.
- **Practical enforcement:** The design of the scheduling policy should not be complex. Because of the complex scheduling policy, the delay in job processing will be higher and it can become impractical to use.

Besides, the aforementioned scheduler characteristics, a scheduler or a scheduling policy is chosen based on different performance matrix parameters such as throughput, average delay, fairness, security, and privacy. The scheduler performance matrix parameters are described in the Table 2:

The study of scheduler characteristics and performance matrix parameters shows that they are conflicting with each other. The shared scheduler that leads to the better throughput will lead to more privacy breaches and information leakage. For example, as suggested by Kadloor et al., in [1], the FCFS and TDMA are two extreme scheduling policies. The FCFS maintains least information privacy while the TDMA maintains the highest privacy. But on the other hand, performance or throughput of FCFS is very high compared to TDMA policy. Thus, the system designer should calculate the trade-offs and then select the

Please cite this article in press as: K. Anjaria, A. Mishra, Thread scheduling using ant colony optimization: An intelligent scheduling approach towards minimal information leakage, Karbala International Journal of Modern Science (2017), http://dx.doi.org/10.1016/j.kijoms.2017.08.003