Parallel steganography benchmark for ValiPar concurrent program testing tool

Silvia M. D. Diaz [1], **Paulo S. L. de Souza**[1]

[1]Institute of Computer and Mathematics Sciences - University of Sao Paulo (USP)
Sao Carlos - SP - Brazil

smdiazdiaz@usp.br, pssouza@icmc.usp.br

Abstract. *The validation and verification of concurrent programs is a major concern in software testing and parallel programming areas. Testing tools have been an essential support for V&V tasks, and there are practices that allow analysing and validating their different characteristics and limitations. Here is where benchmarks come up with a solution with the goal of evaluating their behavior. ValiPar is a tool that conducts structural testing for both shared and distributed memory concurrent programs, and there is a set of benchmarks that were developed to verify several aspects. However, it is required the evaluation of the tool in regards of scalability, and the purpose of this work is to implement a real problem with a greater size algorithm to be tested in ValiPar, using both memory access models and a larger number of interactions among processes. This paper presents the results of a parallel steganography program development and the results of its evaluation, finding that the tool achieved to instrument and evaluate the parallel code but showed some limitations in the execution of test cases.*

1. Introduction

Given the current computational necessities and researches in concurrent software testing, the tools that support it also require the implementation of strategies to improve their functioning. ValiPar is a tool developed at the ICMC-USP, and supports the execution of structural testing techniques for parallel programs in different programming languages and both shared and distributed memory paradigms. Meanwhile, benchmarks are programs that are implemented with the objective of studying the behavior of testing tools. For ValiPar, a set of benchmarks was developed considering critical characteristics of concurrent programs [Dourado 2015]; most of them are simple common computational problems, adapted and implemented with complex communication patterns. However, the size of this programs is not as big as they could be to test scalability.

2. Background

The goal of this paper is to develop of a parallel algorithm in Java that supports the verification of the scalability of ValiPar tool, and also contributes to a benchmark suite developed in [Dourado 2015]. Nevertheless, most of the developed benchmarks are relatively small-size programs of simple problems, whose main focus was the complexity of communications, the implementation for both paradigms and the utilization of different

communication patterns. It is necessary to develop concurrent programs with a larger size in terms of LOC, with greater input data and higher number of communications, leaving aside the complexity of parallel events.

Steganography is a method to hide messages into objects, such as images, audio files, videos, documents, etc. An image can be represented as a matrix of pixels, which contains the code color information in the Red, Blue and Green (RGB) color model for each pixel.There are different techniques for hiding a message into an image [Shelke et al. 2014], and one of the most utilized is the Least Significant Bit (LSB), which consists in making a substitution of the LSB of each RGB color byte with one bit from the message. The algorithms of this paper will focus on the encoding of a plain message using LSB image steganography and its decoding.

3. Proposal

This work presents the implementation of a parallel steganography algorithm using the LSB technique with both memory access models and a larger size in terms of LOC, processes and interactions, that can be evaluated with ValiPar tool by analyzing the results of the execution of each one of its modules. Given that some limitations of ValiPar have not been verified so far, and the fact that there are not many benchmarks that consider both concurrent programming paradigms (or memory access models), this benchmark has the objective of verifying the scalability of the tool, analyzing and evaluating its behavior when there is a larger number of interactions. The resulting parallel programs were compared to the main characteristics that a benchmark should accomplish [Dourado 2015]; this is important since it allows contrasting different programs by using standard features to evaluate. To develop the concurrent program, it was performed an analysis of the problem structure and applied a parallel design methodology (PCAM).

The LSB steganography method was analysed and a sequential algorithm was developed; based on this, it was achieved a parallel design following Foster's methodology [Foster 1995] and the implementation of the parallel algorithm. For the ValiPar evaluation, it was made a previous study of each module execution and scripts were generated to facilitate this task. Moreover, the analysis of results contemplated different executions and instrumentation possibilities in order to provide reliability in the conclusions.

4. Parallel design and development

Following the PCAM methodology [Foster 1995], the most suitable method for partitioning was the domain decomposition, focusing on the data handled by the algorithm. Both message string and pixel matrix were divided in equal n parts, which are the most accessed and larger data structures. The objective behind this scheme is that each task processes a part of the message with the corresponding matrix rows. For the agglomeration phase, few tasks were combined. In the mapping phase, a Client-Server model was established to characterize the interaction among processes. A static load-balancing strategy was implemented by dividing the data in the number of Client processes. Connectionless sockets (Java Datagram Sockets) were utilized for distributed memory and Java Threads for shared memory, as libraries for parallel applications. Other ValiPar limitations were contemplated in the development of the algorithms, such as in the implementation of libraries and loops. It is worth to say that these constrains complicate the utilization of any

Table 1. Parallel Steganography benchmark characteristics

Characteristic	Accomplishment	Values
Easy understanding	ⓧⓧⓧⓧ◯	Yes
Interactions among processes	ⓧⓧⓧⓧ◯	Uses, associations > 3.000.000
Test cases	ⓧⓧⓧⓧⓧ	Yes
Cyclomatic complexity	ⓧⓧⓧ◯◯	McCabe scale > 20
Program classification	ⓧⓧⓧ◯◯	Real problem
Primitives scope	ⓧⓧⓧⓧⓧ	Communication and synchronization primitives
Event based vision	ⓧⓧⓧⓧ◯	Yes
Both paradigms	ⓧⓧⓧⓧⓧ	Distributed and shared memory
Communication patterns	ⓧⓧ◯◯◯	Point to point, Client-Server
Non-determinism	ⓧⓧⓧ◯◯	Impractical
Pseudo-code	ⓧⓧⓧⓧ◯	Yes
Standard documentation	ⓧⓧⓧⓧ◯	Documented code
LOC	ⓧⓧⓧⓧ◯	Encode and Decode > 1000

parallel program in ValiPar, since the non consideration of its limitations will make the tool not be able to instrument the code and thus evaluate its coverage.

The encoding algorithm has a total of 5 processes and 16 threads; one Server and four Clients (each with four threads). Server process reads the image and the message and gets all of the Clients ports and addresses, so that it is able to fragment the data into portions and send it to the corresponding Client. The sent data is a *n* size message portion (*n* = total message length / number of Client processes) and a *n x 3* pixel matrix, so each Client receives the piece of data to hide and exactly the number of rows of the matrix to do so; the complete image matrix is not sent since it would increase overheads by message size. Each Client process converts the message to binary and creates four threads to process the received data portion (data is shared among threads, but each treated a different sub-portion). When all threads finished the modification of data, the Client processes send it back to the Server process, which assembles the received data portion and inserts the new pixels into the image and writes it to the specified directory.

The decoding algorithm works in accordance to the encoding algorithm communication basis, except for the Client processes, which do not create threads and extract the message portion themselves. Server process concatenates the received message ensuring that it is printed in order.

5. Benchmark characteristics

Table 1 presents the evaluation of the main characteristics that a benchmark program should accomplish [Dourado 2015]. The column *Accomplishment* presents the level of fulfillment of the feature and *Value* shows the resulting values of such aspects. Security and complexity in the message hiding process were not considered in this benchmark, since the main concern are the quantity of communications among processes and the size of transferred data to validate scalability. The interactions among processes is high, which reflects in the number of uses, associations and synchronization edges found by ValiPar and also by the size of the Parallel Control Flow Graph (PCFG); on the other hand, the number of LOC is greater compared to the existing benchmarks.

Despite non-determinism was present in the sent and receive executions, this situation had to be controlled in behalf of coupling the received packets in Server processes, since pixel rows could not be saved in a different position from the original; as well as in the decoding program, where the message could not be printed in other order.

6. ValiPar evaluation

Both coding and encoding algorithms were evaluated in ValiPar, as well as other similar test programs with different communication patterns that were not shown in this paper. A correct instrumentation of the two programs was verified by generating the PCFG and it resulted in an extensive and limited-comprehensible graph, that made difficult the analysis of the interactions among processes. Required elements and test cases were successfully generated but when executing the ValiExec module, the program locked and could not complete the test case execution. Hence, the evaluation of the covered elements could not be accomplished since there was not information from the resulting executions.

Analysing the results, ValiPar got to instrument the programs and generate required elements, so it is able to interpret parallel programs with larger process interactions and greater number of processes. However, there was found a limitation regarding the execution of the test cases, presumably by the great quantity of interactions evidenced by the PCFG. The ValiPar limitations were studied and adapted to the programs, and instrumentation and test case generation were verified several times in order to provide reliability to the concluding results.

7. Conclusion

It was developed a functional parallel steganography program (encoding and decoding) that was successfully instrumented and evaluated in ValiPar tool; this allowed the analysis of its behavior when there are a larger number of interactions among a greater number of processes, using both memory access models. It also relates to a real problem and involves a larger number of LOC. The parallel programs fulfilled the requirements of ValiPar tool and allowed the generation of a PCFG that represented the quantity of interactions and evidenced the great number of synchronization edges in the graph. The characteristics of this benchmark were evaluated regarding the specifications in [Dourado 2015], and a description document was developed as well. A limitation in the scalability of ValiPar was found, given that the tool failed to execute the test cases for the programs; this is a major finding and contribution to the current research, since it relies in the evaluation of concurrent programs criteria that considers the identification of defects on dynamic aspects. However, ValiPar should be evaluated in other environments and conditions to verify the cause of the execution results, clarifying whether the ValiExec limitation was in the benchmark development or the tool implementation, or if it was related to the machine resources.

References

Dourado, G. G. M. (2015). Contribuindo para a Avaliação do Teste de Programas Concorrentes: uma abordagem usando benchmarks. *Master thesis, Instituto de Ciências Matem´aticas e de Computação da Universidade de São Paulo*.

Foster, I. (1995). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Shelke, F. M., Dongre, A. A., and Soni, P. D. (2014). Comparison of different techniques for Steganography in images. 3(2):171–176.