Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

# Dynamic translation for virtual machine based traffic simulation

Bin Yu [a,*], Michael Zhang [b], Zhongren Wang [c], Dong Bian [d]

[a] Key Laboratory of Road and Traffic Engineering of the Ministry of Education, Tongji University, Shanghai, PR China
[b] Department of Civil and Environmental Engineering, University of California, Davis, USA
[c] California Department of Transportation, Sacramento, USA
[d] School of Transportation Engineering, Tongji University, Shanghai, PR China

ARTICLE INFO

*Article history:*
Received 13 December 2013
Received in revised form 24 June 2014
Accepted 25 June 2014
Available online 21 July 2014

*Keywords:*
Dynamic translation
Scripting language
Traffic simulation
Virtual machine

ABSTRACT

For virtual machine based traffic simulation platforms, the paper proposes a software framework that performs trace-based dynamic translation. Through monitoring the run-time execution status of bytecodes and translating frequently executed bytecodes, also known as hot spots, into equivalent native machine codes, the framework considerably improves the performance of virtual machine based traffic simulation platforms up to ten times or more, as the experiments showed. For the first time, the presented work clearly exhibits that a seamless combination of the two technologies – dynamic translation and virtual machine could lead to a new generation of applicable traffic simulation platforms. Such a platform not only offers high flexibility in terms of traffic model simulation, but also preserves the ability of conducting numerical computation-intensive simulations generally found in real-life industrial projects.

## 1. Introduction

To simulate various traffic behaviors, traditional simulation platforms will select a particular set of models, also named built-in models, and embed logic of the particularly selected models into their underlying computer implementations. For instance, consider simulation of the car following behavior. PARAMICS chooses a model adopted from the one proposed by Fritzsche [1]. The one used in VISSIM is a psycho physical car following model proposed by Wiedemann [2,3]. INTEGRATION used a model proposed by Van Aerde [4] and Van Aerde and Rakha [5], which combines the Pipes model and the Greenshields model together. And the car following model used in MITSIMLab [6–8] is the fifth generation model developed by General Motors researchers. Through closely embedding model logic, traditional platforms achieve better computation performance. However, this would also lead to traditional platforms' inflexibility when simulating models other than the embedded ones, as usually tweaking the internal codebase may be required. To overcome the issue, traditional platforms come up with solutions such as Application Programming Interface (API), Common Object Model (COM), and so on. But all require considerable amount of programming knowledge, which is typically beyond the reach of average transportation researchers. Thus, for research application such as comparison of traffic models, traditional platforms are limited. A unified platform with less coding requirement is more desirable.

* Corresponding author. Tel.: +86 216 958 4687.
*E-mail addresses:* by@tongji.edu.cn (B. Yu), hmzhang@ucdavis.edu (M. Zhang), zhongren_wang@dot.ca.gov (Z. Wang), biandong_tj@qq.com (D. Bian).

http://dx.doi.org/10.1016/j.simpat.2014.06.013
1569-190X/© 2014 Published by Elsevier B.V.

The term of *virtual machine* first appeared as an operating system concept in the 1960s [9]. In early era, virtual machine was mainly applied in low level sub-disciplines, i.e. hardware, operating system, and so on. Smith [10] discussed the architecture of virtual machine used in both low level and high level. For high level programming languages, application of virtual machine could allow them to achieve better cross-platform. The languages include Java, Lua, Perl, Python, Ruby, to name a few, among which probably the most famous one is Java, first introduced in the year of 1991 by James Gosling, Mike Sheridan, and Patrick Naughton. In addition to better cross-platform, application of virtual machine has another attracting value of allowing construction of a well customized software environment upon which a secondary development would be easier and faster. That is why a high level virtual machine is often applied in many specialized sub-disciplines. For instance, in the numerical computation sub-discipline, MATLAB [11] from Mathworks is a successful example. The R software platform [12] for statistical computing is a more recent instance.

Unfortunately, for the traffic simulation sub-discipline, no similar research effort was reported yet. Thus, one of the authors, Bin Yu, started an experimental project to develop a virtual machine based traffic simulation prototype system from scratch in 2005. Based on the ideas from Lua [13,14], the prototype system has two major components that are a special-purpose dynamic data type scripting language oriented to traffic simulation and a virtual machine. The special-purpose scripting language was designed and implemented from the ground up. In general, microscopic traffic simulation has its own particular architecture. In the language level, the existing computer languages lack the mechanisms in supporting the particular architecture. Consequently, it is difficult for end users to use the languages to program math logic of concrete traffic models to be simulated. Correspondingly, the virtual machine capable of executing bytecodes compiled from source scripts written in the special-purpose scripting language was implemented from scratch too. Using the prototype system, Yu et al. [15] verified the flexibility of a virtual machine based traffic platform in terms of model simulation and showed that the prototype system can virtually simulate any microscopic traffic model. However, in the meantime, it was also observed that the prototype system's computation performance is slower.

For a computer system, application of virtual machine technology can bring merits, but it also introduces computation overheads. After all virtual machine is a layer of software. The situation becomes even worse for high level virtual machine. As bytecodes are generally interpreted, high level virtual machine's performance worsens. For instance, for the early Java virtual machine, Tyma [16] wrote "*Java is not just slow, it's really slow, surprisingly slow*". Considerable research efforts are spent to seek techniques improving performance of virtual machine. So far, probably the most effective and predominant methodology is dynamic translation, also known as just-in-time compilation.

The concept of dynamic translation first appeared in the seminal paper written by McCarthy [17]. Since then, researchers studied dynamic translation in different systems, for instance Mitchell [18], Deutsch and Schiffman [19]. More recently, just-in-time compilation has blossomed with the success of Java which is shipped with a HotSpot just-in-time module [20,21]. Just-in-time compilation is a mature technique. For a rather complete review of just-in-time, readers can refer to Aycock [22] who categorized just-in-time compilation into four generations and claimed the most recent and predominant fourth generation is trace-based, by writing "*...fourth-generation simulators expand upon the third-generation by dynamically translating paths, or traces.*" [22].

Although Yu et al. [15] discussed the feasibility of virtual machine technology in the next generation of traffic simulation platforms, their system was based upon a pure interpreting virtual machine. Thus, for real life applications, availability of such a system is questionable, given its slower computation speed. For this purpose, a trace-based dynamic translation framework belonging to the fourth-generation category according to Aycock [22] is proposed in the article.

### *1.1. Work significance and paper organization*

The presented work demonstrated for the first time how a seamless integration of a dynamic translation module into a virtual machine based traffic simulation platform would affect computation performance. In principle, all fourth-generation dynamic translation technologies are similar to some extent. The main difference among them is the way how the dynamic translation technology is integrated into the underlying computer system. The more closely the dynamic translation technology is integrated, the better the performance, though a close integration in general may mean an overhaul of the system. For example, for the prototype system [15], LibJIT from the DotGNU project [23] was ever imported as a standalone module to add just-in-time support, but the outcome was unsatisfactory. Only a marginal improvement was achieved.

The rest of the paper is organized as follows. In Section 2, the proposed dynamic translation framework is presented. Performance benchmarks are executed and numerical results are illustrated in Section 3, through implementing and seamlessly integrating the proposed framework into the prototype system. Summary and proposal for further studies is presented in Section 4.

## 2. Framework

### *2.1. Simulation architecture*

First we briefly describe the simulation architecture of a virtual machine based traffic simulation, which is quite different from traditional traffic simulation platforms'. Interested readers can refer to Yu et al. [15] for more details.