J. Parallel Distrib. Comput. 106 (2017) 50–61

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

On the injection of hardware faults in virtualized multicore systems

Marcello Cinque, Antonio Pecchia [*]

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione, Universitá degli Studi di Napoli Federico II, Via Claudio 21, 80125, Naples, Italy

HIGHLIGHTS

- Emerging industrial parallel computing applications.
- A framework to inject hardware faults.
- Implementation based on widely-used technologies.
- Extensive fault injection campaigns.

ARTICLE INFO

Article history:
Received 22 August 2016
Received in revised form
3 February 2017
Accepted 6 March 2017
Available online 22 March 2017

Keywords:
Multicore
Virtualization
Fault injection
Machine check exception
Dependability

ABSTRACT

Virtualized multicore systems represent an emerging computing paradigm in the critical systems industry. Virtualization-based solutions leverage the different cores of the processor to run operating systems and applications within separate partitions, to support the development of parallel mixed-criticality systems, and to improve fault-tolerance by protecting and isolating the operating environments. The critical systems industry is subjected to international standards, which recommend fault injection as a mean to contribute with evidence to safety cases. This paper proposes a framework to inject hardware faults in virtualized multicore systems. Our proposal capitalizes on the error reporting architecture of modern processors and allows injecting faults both at hypervisor- and guest-OS-level. We implement the framework in the context of the widely used Intel Core i7 processor and Xen hypervisor. We demonstrate the use of the framework by means of about 60,000 injection experiments in a Linux-based virtualized multicore system installation.

1. Introduction

Over the past years the critical systems industry has started looking to **multicore processors** with growing interest across different domains, such as *avionic* [27], *automotive* [36] and *medical* [48]. Beside the recognized advantages in terms of performance, which was among the early concerns in industrial parallel computing applications [12,42], the recent *industrial driver* of the multicore technology is the chance to integrate many software functions on the same chip. Multicore processors reduce the number of hardware units and communication links across them, which represent a significant cause of system-wide failures. The multicore technology is fostering the migration from *federated* to *integrated* architectures, which consist of different tasks running on the same chip. Integrated Modular Avionics (IMA) [2] and AUTOSAR[1] clearly represent this industry trend in two distinct domains.

Let us present an example for the sake of clarity. The latest versions of AUTOSAR have introduced supports for multicore systems. The AUTOSAR OS specifies a number of capabilities, such as static task assignment to a given core and inter/intra-core coordination to access shared resources. Modern cars host up to 70+ Electronic Control Units (ECUs) connected by communication links: ECUs are responsible for executing tasks, such as gear control, engine control, and adaptive cruise control. In the multicore scenario, different ECUs can be deployed on the same processor. The mechanisms implemented by AUTOSAR to manage isolation and scheduling issues are described in [36].

Virtualization is often used on the top of multicore processors to achieve *isolation* among tasks [10]. It allows running different operating systems and applications on the same processor within *virtual machines*, which access the hardware through the

[*] Corresponding author.
E-mail addresses: macinque@unina.it (M. Cinque), antonio.pecchia@unina.it (A. Pecchia).

[1] http://www.autosar.org.

http://dx.doi.org/10.1016/j.jpdc.2017.03.004
0743-7315/© 2017 Elsevier Inc. All rights reserved.

hypervisor. Virtualization is strongly beneficial to the critical systems industry because it supports:

- hardware/software consolidation, i.e., different operating systems (e.g., a real time and not real time OS) and applications run on the same hardware unit [28];
- developing parallel mixed-criticality systems, i.e., safety and non-safety tasks are isolated and run on the same unit;
- gradually migrating to Off-The-Shelf (OTS) technologies and to new hardware devices [41];
- implementing fault-tolerance mechanisms by means of replication and/or diversity approaches [14];
- reducing certification costs by isolating certified software.

Overall these benefits have made **virtualized multicore systems** an emerging parallel computing paradigm within the critical systems industry. For example, the use of virtualization-based solutions has been investigated in the aerospace [15], and automotive [35] domains. Commercial solutions in the context of certified hypervisors and safety-related applications have been proposed by WindRiver and Intel [46].

The critical systems industry is subjected to international safety standards, such as ISO-26262, IEC-61508, NASA-STD-8719.13B and DO-178B, which guide system validation and certification. In this respect, **fault injection** is strongly recommended (if not even mandatory) to accomplish a variety of requirements imposed by the market, such as contributing with evidence to *safety cases*, supplementing the set of tests to achieve *fault coverage* and *code coverage* including the analysis of the error-handling mechanisms, or ganging insights into the system fault-tolerance in face of exceptional conditions [29]. For example, fault injection is an important constituent of the ISO-26262 [22] to supplement software unit and integration testing. This process aims to ensure the correctness of a given implementation in terms of specification, technical and functional safety requirements.

In spite of the number of techniques and tools, the use of fault injection for the assessment of multicore-based systems is still recent. There are several challenges in injecting hardware faults in virtualized multicore systems. First, the availability of hardware/software interfaces to support the injection step. Injection of hardware faults was originally accomplished by tampering with the physical device through special-purpose equipment. This approach is costly and might damage the target; as such, it can be hardly pursued by industry, which is subjected to strict *budget* and *time-to-market* constraints. When hardware faults are reproduced by software, the challenge is conceiving a representative *fault model*, i.e., the specification of faults the system is actually expected to encounter in reality. Multicore systems encompass errors and hardware units, which cannot be addressed through traditional *bit-flip* and *stuck-at* models. To the best of our knowledge, no fault models or established frameworks are currently available in the context of virtualized multicore systems.

This paper proposes a **framework** to inject hardware faults in virtualized multicore systems. The proposal capitalizes on the use of the **error reporting architecture** implemented by modern processors, which encompasses the mechanisms and registers for *detecting* and *reporting* internal hardware errors. The architecture notifies either (i) corrected errors of the hardware units or (ii) errors that cannot be fully handled at hardware level, which cause the processor to interrupt the current program. The architecture provides detailed amount of diagnostic data to upper system levels through a number of **dedicated registers**. For example, the occurrence of a *cache hierarchy* error usually comes with information such as the cache level, type, and action performed at the time the error occurred (e.g., data read, instruction fetch, snoop).

The diagnostic data generated by the error reporting architecture is extremely valuable to the software running on the top of the processor. In this respect, the error-handling complexity is shifted to the software, which is expected to correctly interpret a given error notification and to implement suitable recovery mechanisms to address the error. Software will play an increasingly key role at implementing proper fault-tolerance/recovery mechanisms and mitigation actions: assessing its behavior against the occurrence of hardware faults is a key concern in critical systems.

Our framework injects **hardware faults** by writing into the registers of the error reporting architecture. This technique supports low-cost, time-effective, and controllable experimental campaigns, which traditionally represented the major challenges to the practicability of fault injection. We implement the framework in the context of widely used technologies, i.e., the Intel Core i7 2670QM processor and the Xen hypervisor. Our solution allows injecting faults both at hypervisor- and guest-OS-level. We demonstrate the use of the framework for the assessment the error-handling mechanisms of a virtualized multicore system; about 60,000 injection experiments have been run in a Linux-based installation, paving the way for fault-injection-based safety cases in parallel critical applications. Experiments made it possible to gain insights into the behavior of Xen at forwarding error notifications to guest OSs and revealed a potential bug of the error-handling procedure in the Linux OS.

The rest of the paper is organized as follows. Section 2 introduces the background and related work in the area. Section 3 provides an overview on the error reporting architecture of the Intel Core i7 and discusses the representativeness of the fault model adopted in this study. Section 4 describes the fault injection framework and the technicalities underlying the proposed solution. Section 5 presents the case study on the assessment of error-handling mechanisms, while Section 6 concludes the work and discusses future work directions.

2. Related work

2.1. Hardware fault injection

Several techniques have been proposed in order to inject hardware faults. **Hardware-based techniques** insert hardware faults in the target system (i) by means of special-purpose and architecture-dependent equipment or (ii) by interfering with the physical unit (e.g., lowering the device voltage, increasing the temperature, inducing electromagnetic interferences) [16]. These techniques have the advantage of reproducing *real* hardware faults; however, they are costly and risky to implement, as they might disrupt the target of the injection. Moreover, intrusiveness and interferences caused by the injectors make it hard to observe the effect of faults.

For these reasons, **software-implemented fault injection** (SWIFI) techniques have gained popularity. SWIFI consists in reproducing *by software* the effects of hardware faults. Injection is accomplished either at (i) compile time by inserting the effects of hardware faults into the target or (ii) at run time using timeouts, exceptions, code insertion or altering the state of the target in order to trigger faults. Several SWIFI tools have been proposed so far by the literature:

- FIAT [5] corrupts the data area of the binary according to three fault models, namely, zero-a-byte, set-a-byte, and two-bit-compensating. The zero-a-byte and set-a-byte fault resets and sets eight bits of a 32-bit word, while the two-bit-compensating flips two bits.
- FERRARI [24] injects permanent/transient faults, and control flow errors, bus errors, memory errors, and processor control line errors into systems based on SPARC processors from Sun Microsystems. FERRARI uses software traps to inject faults and has five fault models: XORing a bit, resetting a bit, setting a bit, setting a byte and resetting a byte.