Computational Intelligence in Security for Information Systems
16th International – 5th-7th September 2023

Fuzzing Robotic Software using HPC

Francisco Borja Garnelo Del Río, Francisco J. Rodríguez Lera

Camino Fernández Llamas, Vicente Matellán Olivera

Universidad de León - Campus de Vegazana s/n, 24071 León (Spain)

infbgd01@estudiantes.unileon.es, {fjrodl, cferll, vmato}@unileon.es

Table of Contents

Introduction

Fuzzing is the automation of generating and testing malformed inputs in software in order to find unexpected behavior in the software.

Robotic systems are systems that interact with their environment, including people, through the use of numerous sensors, actuators, and user interfaces to give intelligent services and information.

High Performance Computing (HPC) is a technique that processes huge multidimensional information and solves complex problems at incredibly fast rates by using clusters of powerful processors operating in parallel.

Introduction - Goals and Hypotheses

Research question

What are the implications of using HPC systems when fuzzing ROS applications?

This question arises from the following hypothesis and questions:

- **H1:** Distributing the fuzzing workload across multiple processing units can significantly increase overall fuzzer performance.
- **Q1:** What are the key concepts associated with Fuzzing in HPC?
- **Q2:** What is the performance of the fuzzer when running in a personal computer or HPC computer?

Introduction - State of the art

Over the past two years, fuzzing tests have evolved quickly in the evaluation of robotics software.

- **PHYS-FUZZ** (Woodlief et al. 2021): integrates traditional fuzzing with physical constrains and hazards associated to mobile robots.
- **ROZZ** (Xie 2022):Multi-dimensional generation method to generate effective test cases for ROS programs.
- **RoboFuzz** (Seulbae 2022): Effectiveness fuzzing tool, tested in a case study based on a robotic manipulator system and compares its performance with other state-of-the-art fuzz testing tools.

Introduction - State of the art

Several researchers have been studying how to introduce HPC in the field of robotics.

- High-Performance Robotic concept, use HPC in different robotic environments (Camargo-Forero 2018).
- Position paper overviewing the use of HPC for exploring computation in cloud systems with cybersecurity and explainability (Matellán et al. 2021).
- Set of software components deployed within Singularity containers that run a Robot Operating System (ROS) supported on Message Passing Interface (MPI) with the idea of benchmarking HPC performance with multiple vehicle simulations (Brewer et al. 2022).

Materials and Methods - Infrastructures

Each experiment was tested on two different infrastructures:

- **Standalone** SDO, virtual machine with 8GB ram and 6 x Intel Xeon E3-12xx v2 vcpu (virtual cpu) and linux kernel *5.3.11-100.x86_64 (x86_64)*. Local storage on mechanical hard disks.
- **High-Performance Computing** HPC, computing cluster with Haswell nodes in bare-metal with 48GB of ram and 2 x Intel Xeon E5-2630 v3 @ 3.20GHz with a total of 16 cores and a Linux kernel *3.10.0-1062.9.1.x86_64 (x86_64)* . Network storage and cache on solid disks.

NOTE: When using containers the distro is indifferent.

Materials and Methods - Software

The experiments made use of the following software:

- **RoboFuzz**, an autonomous fuzz testing tool for robotic systems.
- **SLURM** (Simple Linux Utility for Resource Management) a widely used open-source workload manager and job scheduler for Linux and Unix-based clusters and supercomputers.
- **Singularity** a container solution created to run complex applications on HPC clusters in a simple, portable, and reproducible way.
- **Docker**, an open container platform for developing, shipping, and running applications.

Materials and Methods - Experiments

- **MoveIt 2** MI2, consisted of testing the MoveIt 2 library, which is a robotic manipulation library for ROS that implements fundamental robotic manipulation concepts.
- **Turtlebot 3 TB3**, tested a differential wheeled mobile robot equipped with a LiDAR sensor.
- **Phoronix Test Suite.** a comprehensive software for Linux testing and benchmarking performance. The following suites were used:
  - pts/sysbench-1.1.0 General CPU and memory tests.
  - pts/byte Pure compute CPU tests.
  - pts/fs-mark Filesystem tests.

Step-by-step documentation at Prjoct GitHub

Materials and Methods - Implemented cases

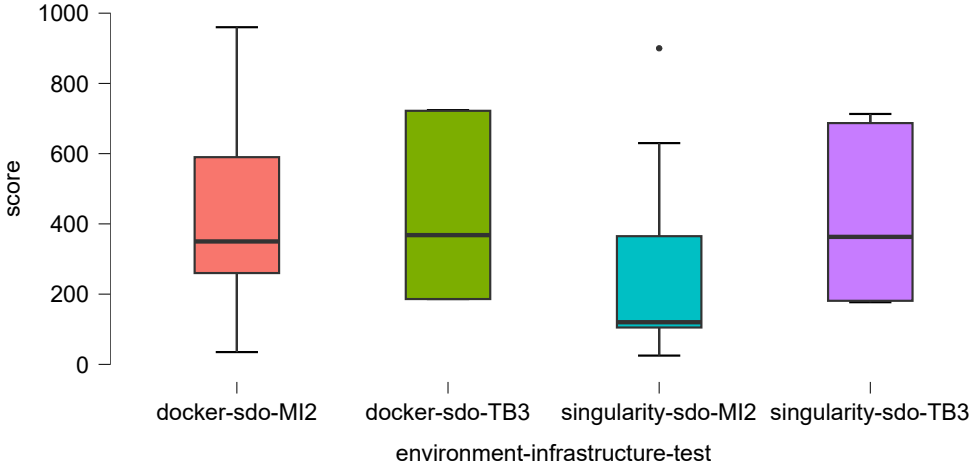
- **ROBOFUZZ**, main tests used in the validation of Robofuzz by his author.
  - check_install.sh
  - run_PX4_quadcopter_micrortps_agent.sh
  - test_Move_It_2_plus_PANDA_manipulator.sh
  - test_PX4_quadcopter_mutating_parameter.sh
  - test_PX4_quadcopter_offboard_mode.sh
  - test_PX4_quadcopter_remote_control.sh
  - test_TurtleBot3_Burger.sh
  - test_Turtlesim.sh

Materials and Methods - Implemented cases

- **ROS 2**, demos of Data Distribution Service (DDS).
  - demo-Cyclone_listener.sh
  - demo-Cyclone_talker.sh
  - demo-FastRTPS_listener.sh
  - demo-FastRTPS_talker.sh
- **BENCHMARK**, experiments of this paper.
  - docker-phoronix-test-suite.sh
  - docker-robofuzz-MI2.sh
  - docker-robofuzz-TB3.sh
  - singularity-phoronix-test-suite.sh
  - singularity-robofuzz-MI2.sh
  - singularity-robofuzz-TB3.sh

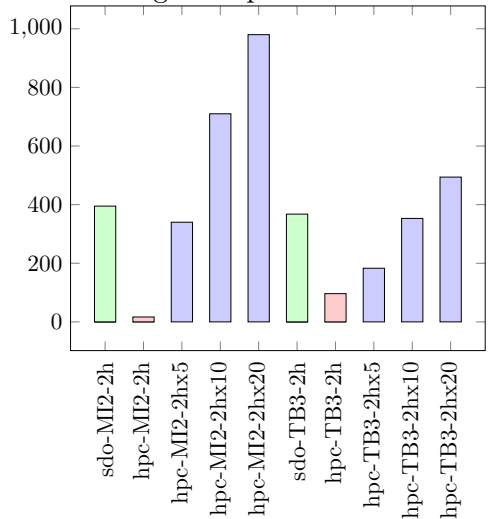
Results

Docker vs singularity performance comparison.



Results

Performance comparison of single vs. parallel tasks.



Conclusions

- High-Performance Computing improves fuzzing testing by providing computational resources, parallel processing, and advanced algorithms for complex systems.
- Recent kernels outperform older ones due to container management improvements and processor vulnerability patches; parallelization reduces task completion time.

Conclusions

These are this work's three main contributions:

1. Empirical evaluations that allow us to answer the hypothesis above.
2. A set of publicly available Singularity containers deploying a ROS 2 fuzzer.
3. Lessons learned of using a SOTA fuzzer in an HPC environment.

Next steps

- Adapt completely HOUSE framework to robotic environments using ROS for decoupled deployment in HPC, improving fuzzying processes in multi-variable contexts.

Demo time

Thank you for your attention.
Do you have any questions?