arm

Rust and rust-vmm

Insights for friends in the Linaro community

Robin Randhawa

Technical Director

System Software Architecture – Open Source Safety

January 2021

Rust and rust-vmm

Introduction

About me..

- **Background**
  - Productization using open-source software
  - Linux, FreeBSD kernel development
  - Custom OS kernel development
  - Firmware, toolchain development
- **Primary focus areas**
  - Safety oriented programming languages
  - Safety oriented OS and kernel architecture
- **At Arm I work on**
  - H/W and S/W architecture for Autonomous Vehicle Control
  - Automotive and IoT team architect

- Long time friend of Linaro since the beginning
  - Involved with Linaro's power management team as the Tech Lead of Arm's OS power-perf team for over 6 years

I  Linaro

Rust and AArch64: An announcement!

My mission in 2020

Announcing Rust 1.49.0

Dec. 31, 2020 · The Rust Release Team

What's in 1.49.0 stable

For this release, we have some new targets and an improvement to the test framework. See the detailed release notes to learn about other changes not covered by this post.

64-bit ARM Linux reaches Tier 1

The Rust compiler supports a wide variety of targets, but the Rust Team can't provide the same level of support for all of them. To clearly mark how supported each target is, we use a tiering system:

- Tier 3 targets are technically supported by the compiler, but we don't check whether their code build or passes the tests, and we don't provide any prebuilt binaries as part of our releases.
- Tier 2 targets are guaranteed to build and we provide prebuilt binaries, but we don't execute the test suite on those platforms: the produced binaries might not work or might have bugs.
- Tier 1 targets provide the highest support guarantee, and we run the full suite on those platforms for every change merged in the compiler. Prebuilt binaries are also available.

Rust 1.49.0 promotes the `aarch64-unknown-linux-gnu` target to Tier 1 support, bringing our highest guarantees to users of 64-bit ARM systems running Linux! We expect this change to benefit workloads spanning from embedded to desktops and servers.

This is an important milestone for the project, since it's the first time a non-x86 target has reached Tier 1 support: we hope this will pave the way for more targets to reach our highest tier in the future.

The longer story....

Rust and AArch64: An announcement!

My mission in 2020

Ann

Dec. 31, 202

What's in 1.49.0 stable

For this release, we have some new targets and an improvement to the test framework. See the detailed release notes to learn about other changes not covered by this post.

64-bit ARM Linux reaches Tier 1

The Rust compiler supports a wide variety of targets, but the Rust Team can't provide the same level of support for all of them. To clearly mark how supported each target is, we use a tiering system:

- Tier 3 targets are technically supported by the compiler, but we don't check whether their code build or passes the tests, and we don't provide any prebuilt binaries as part of our releases.
- Tier 2 targets are guaranteed to build and we provide prebuilt binaries, but we don't execute the test suite on those platforms: the produced binaries might not work or might have bugs.
- Tier 1 targets provide the highest support guarantee, and we run the full suite on those platforms for every change merged in the compiler. Prebuilt binaries are also available.

Rust 1.49.0 promotes the `aarch64-unknown-linux-gnu` target to Tier 1 support, bringing our highest guarantees to users of 64-bit ARM systems running Linux! We expect this change to benefit workloads spanning from embedded to desktops and servers.

This is an important milestone for the project, since it's the first time a non-x86 target has reached Tier 1 support: we hope this will pave the way for more targets to reach our highest tier in the future.

arm

Rust

What lead me to Rust ?

- Arm asked me to explore the overlaps between Open Source Software and Safety Critical Domains
  - That lead me to exploring microkernels
- System and Processor architects at Arm asked me to look at Safety themed programming languages
  - That lead me to exploring Erlang and Ada
- I built an understanding of what a language needs, from Arm's PoV, to be a focal point of open and safe software construction
  - Rust ticked most boxes

arm

Rust

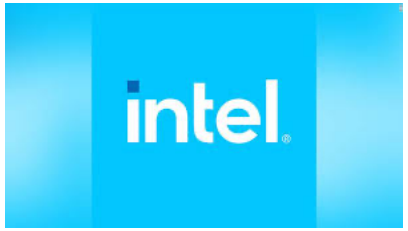
What is Rust ?

- A general-purpose programming language
  - Used in everything from embedded systems to Web Apps
  - Ideal for systems software development
- That emphasizes reliability with zero ambiguity
  - Top-class reference manual
  - Clear static typing rules
  - No type width ambiguity
  - Clear pointer rules
  - You can’t have a buffer overflow/underflow
  - You can’t have a data race
- **Safety checks are done at *Compile Time***
- Emphasises
  - Easy document-ability
  - Cross target development
  - Developer friendly tooling
  - Modularity
- Provides a well-designed separation between **safe** and **unsafe** code
  - If the compiler says the code is safe, then all the safety guarantees hold true!
  - If you need to do something unsafe for good reasons, you have a choice to do that
  - Easy to audit code for non-safety

Rust

Who’s using it ?

[Comprehensive list of companies using Rust in production]



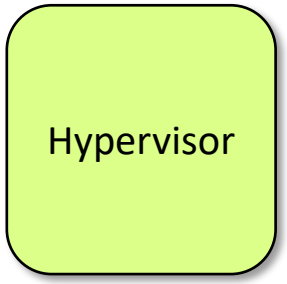
rust-vmm

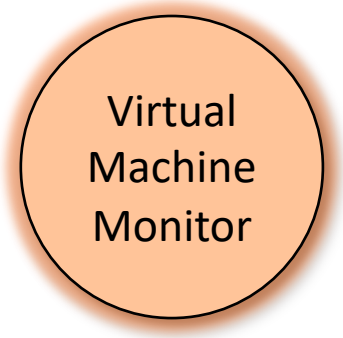
What it's *not!*

- rust-vmm is *NOT* a hypervisor

rust-vmm

What it *is*





A collection of Rust ‘crates’ (~libraries)

Dependent on a Hypervisor

Enables building of VMMs

rust-vmm is a toolkit for building Virtual Machine Monitors

arm

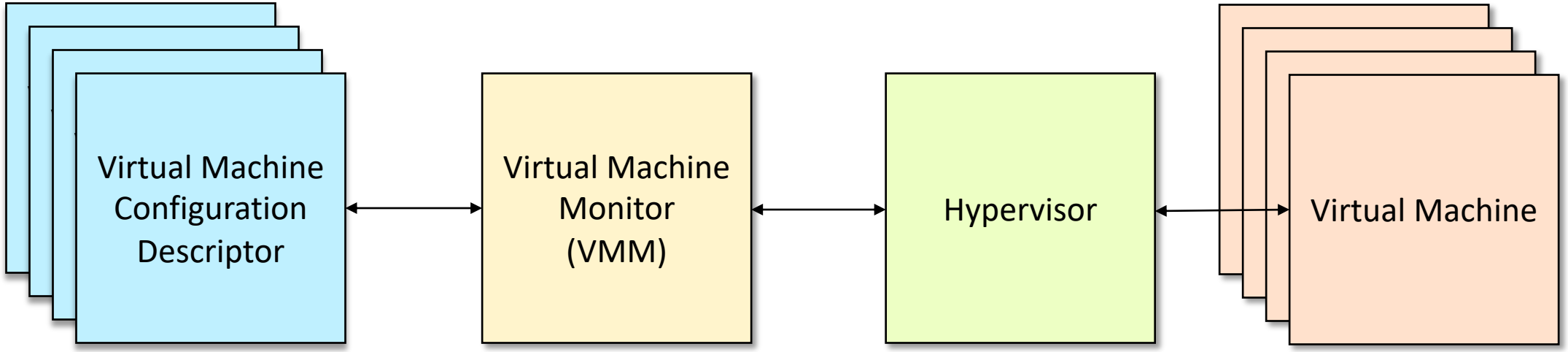
rust-vmm

Again!

“rust-vmm is
an open-source collection
of building blocks written in Rust
to build VMMs
***in any language*”**

rust-vmm

What is a VMM ?



A Virtual Machine Monitor

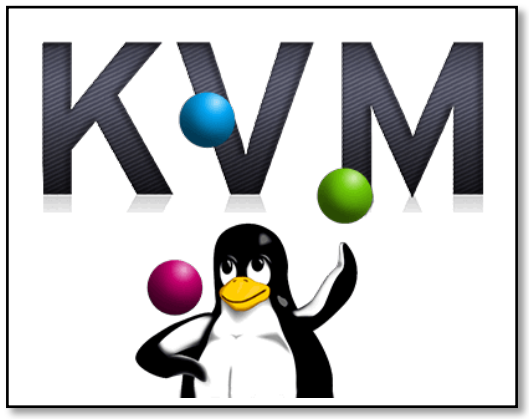
- Acts on requests to configure, create and operate Virtual Machines
- Relies on a Hypervisor for vCPU and system configuration and operation
- Can implement device emulation to support Hypervisors

arm

rust-vmm

Key Insight

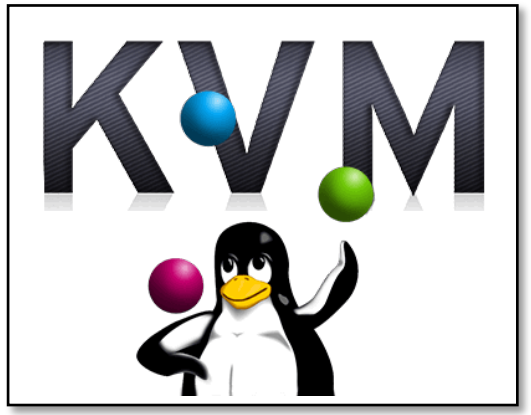
rust-vmm is primarily designed to work with KVM



arm

rust-vmm

Examples of VMMs built using rust-vmm: Crosvm by Google



Google

A collection of Rust ‘crates’ (~libraries)

rust-vmm is ***primarily*** designed for use with KVM

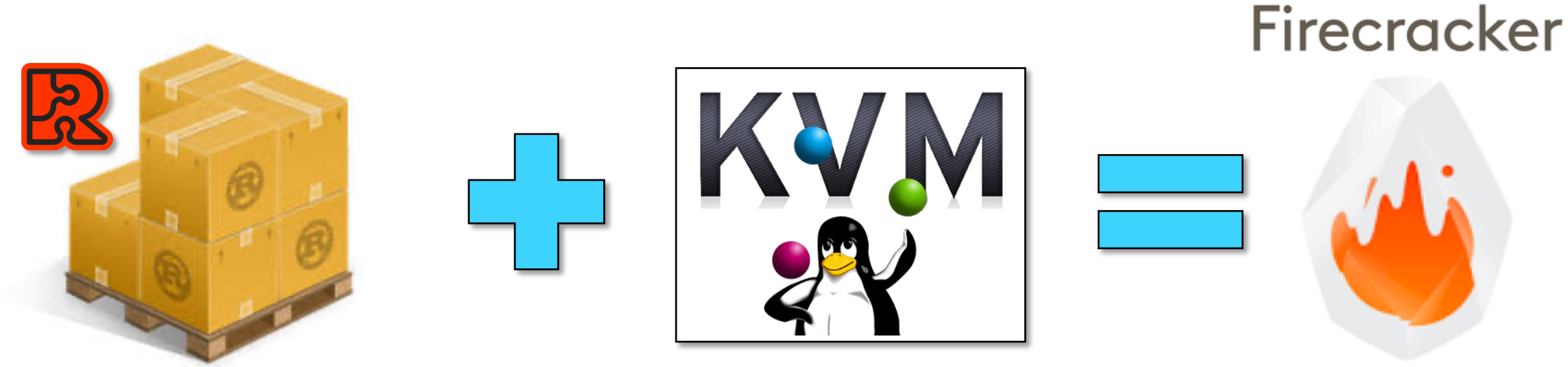
- Lightweight VMM in Rust
- For app isolation
- In ChromiumOS

rust-vmm is a toolkit for building Virtual Machine Monitors

arm

rust-vmm

Examples of VMMs built using rust-vmm: Firecracker by Amazon AWS



A collection of Rust ‘crates’ (~libraries)

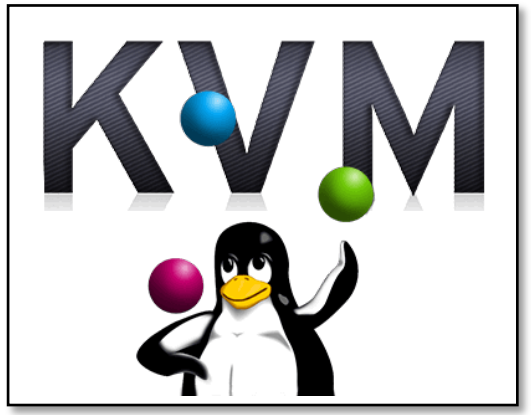
rust-vmm is ***primarily*** designed for use with KVM

- Lightweight VMM in Rust
- For short-lived workloads
- Multi-tenancy focus

rust-vmm is a toolkit for building Virtual Machine Monitors

rust-vmm

Examples of VMMs built using rust-vmm: Cloud Hypervisor by Intel



intel®

Cloud Hypervisor

A collection of Rust ‘crates’ (~libraries)

rust-vmm is ***primarily*** designed for use with KVM

- Lightweight VMM in Rust
- Strong focus on VirtIO
- Strong focus on machine-machine migration

rust-vmm is a toolkit for building Virtual Machine Monitors

arm

rust-vmm

History

Firecracker

rust-vmm

Cloud Hypervisor

Apr 2017 Oct 2017 Dec 2018 2019

arm

rust-vmm

Philosophy

- **Encourage re-use**
  - Try to find common design patterns in VMMs
  - Also in Hypervisor components
  - Implemented them once within rust-vmm
  - Get VMMs (and hypervisors) to use rust-vmm
- **Leverage Rust**
  - Rust's compile time memory safety is perfect for VMMs
  - Compile time Safety + Security: *even for concurrent and/or parallel code*
  - Rich standard library + Expressiveness from functional languages
  - Foreign Function Interface for interop with other languages
- **Leverage Modularisation**
  - Fuzz testing is easier with well designed modularization

rust-vmm

What does it contain ?

A collection of Rust ‘crates’
(~libraries)

- vmm-vcpu
- vm-device
- vm-virtio
- vmm-reference
- vhost
- kvm-bindings
- kvm-ioctls
- vfio-bindings
- virtio-bindings
- vm-memory
- vm-sysutil
- event-manager
- linux-loader
- vm-superio

At requirements gathering stage | Implementation underway | Production ready on crates.io

arm

rust-vmm

What does it contain ?

A collection of Rust ‘crates’
(~libraries)

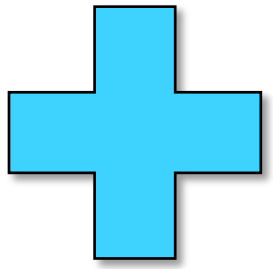
- vmm-vcpu
- vm-device
- vm-virtio
- vmm-reference
- vhost
- kvm-bindings
- kvm-ioctls
- vfio-bindings
- virtio-bindings
- vm-memory
- vm-sysutil
- event-manager
- linux-loader
- vm-superio

- A hypervisor-agnostic abstraction for Virtual CPUs
- A virtual machine device model crate.
- Common VirtIO device Traits and Implementations
- A reference VMM built using rust-vmm
- Support for vhost backend drivers for VirtIO devices
- Rust FFI bindings to KVM
- Safe wrappers over the KVM API
- Rust FFI bindings to use VFIO
- Rust FFI bindings to virtio Linux kernel headers
- Abstractions over a VM’s memory
- Helpers and utilities for building VMMs and Hypervisors
- Abstractions for building event based systems
- Parser and loader for vmlinux and bzImage + helpers
- Emulation for legacy devices

At requirements gathering stage | Implementation underway | Production ready on crates.io

rust-vmm

Areas of Interest



Resources

- Rust
- rust-vmm
- Linaro Virtual Connect
  - “How we got AArch64 Linux to become a Tier-1 Rust platform”

arm

Thank You
Danke
Merci
谢谢
ありがとう
Gracias
Kiitos
감사합니다
धन्यवाद
شكرًا
ধন্যবাদ
תודה