7th China Open Source Annual Conference

(COSCon'22, online)

A Swiss Army Knife for Distributed Computing & AI

Feng Li (李枫)
hkli2013@126.com
Oct 29, 2022

Agenda

I. Background

- Tech Stack
- Testbed

II. Ray 2.0

- An overview of Project Ray
- What's new in 2.0

III. Ray on ARM

- Installation
- Clustering

IV. Speeding up Ray

- Accelerating Python
- Accelerating messaging and RPC

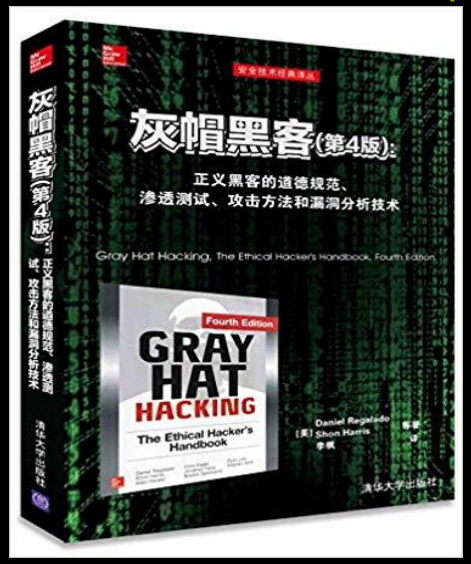
V. Ray.Rust

- Rust-based alternatives
- A new design

VI. Wrap-up

Who Am I

- An indie developer from China
- The main translator of the book «Gray Hat Hacking The Ethical Hacker's Handbook, Fourth Edition» (ISBN：9787302428671) & «Linux Hardening in Hostile Networks, First Edition» (ISBN: 9787115544384)



- Pure SW dev for ~15 years(~11 years on Mobile dev)
- An enthusiast for the Open Source Community
- https://github.com/XianBeiTuoBaFeng2015/MySlides/tree/master/Conf
  https://github.com/XianBeiTuoBaFeng2015/MySlides/tree/master/LTS
- Recently, focus on infrastructure of Cloud/Edge Computing，AI, Virtualization, Program Runtimes, Network, 5G, RISC-V, EDA…

I. Background

1) Tech Stack

1.1 Distributed computing

- https://en.wikipedia.org/wiki/Distributed_computing

Distributed computing is a field of computer science that studies distributed systems. A *distributed system* is a system whose components are located on different networked computers, which communicate and coordinate their actions by passing messages to one another from any system.[1][2] The components interact with one another in order to achieve a common goal. Three significant challenges of distributed systems are: maintaining concurrency of components, overcoming the lack of a global clock, and managing the independent failure of components.[1] When a component of one system fails, the entire system does not fail.[3] Examples of distributed systems vary from SOA-based systems to massively multiplayer online games to peer-to-peer applications.

A computer program that runs within a distributed system is called a **distributed program**,[4] and *distributed programming* is the process of writing such programs.[5] There are many different types of implementations for the message passing mechanism, including pure HTTP, RPC-like connectors and message queues.[6]

Distributed computing also refers to the use of distributed systems to solve computational problems. In *distributed computing*, a problem is divided into many tasks, each of which is solved by one or more computers,[7] which communicate with each other via message passing.[8]

Introduction [edit]

The word *distributed* in terms such as "distributed system", "distributed programming", and "distributed algorithm" originally referred to computer networks where individual computers were physically distributed within some geographical area.[9] The terms are nowadays used in a much wider sense, even referring to autonomous processes that run on the same physical computer and interact with each other by message passing.[8]

While there is no single definition of a distributed system,[10] the following defining properties are commonly used as:

- There are several autonomous computational entities (*computers* or *nodes*), each of which has its own local memory.[11]
- The entities communicate with each other by message passing.[12]

A distributed system may have a common goal, such as solving a large computational problem;[13] the user then perceives the collection of autonomous processors as a unit. Alternatively, each computer may have its own user with individual needs, and the purpose of the distributed system is to coordinate the use of shared resources or provide communication services to the users.[14]

Other typical properties of distributed systems include the following:

- The system has to tolerate failures in individual computers.[15]
- The structure of the system (network topology, network latency, number of computers) is not known in advance, the system may consist of different kinds of computers and network links, and the system may change during the execution of a distributed program.[16]
- Each computer has only a limited, incomplete view of the system. Each computer may know only one part of the input.[17]

Architectures [edit]

Various hardware and software architectures are used for distributed computing. At a lower level, it is necessary to interconnect multiple CPUs with some sort of network, regardless of whether that network is printed onto a circuit board or made up of loosely coupled devices and cables. At a higher level, it is necessary to interconnect processes running on those CPUs with some sort of communication system.[29]

Distributed programming typically falls into one of several basic architectures: client–server, three-tier, *n*-tier, or peer-to-peer; or categories: loose coupling, or tight coupling.[30]

- Client–server: architectures where smart clients contact the server for data then format and display it to the users. Input at the client is committed back to the server when it represents a permanent change.
- Three-tier: architectures that move the client intelligence to a middle tier so that stateless clients can be used. This simplifies application deployment. Most web applications are three-tier.
- *n*-tier: architectures that refer typically to web applications which further forward their requests to other enterprise services. This type of application is the one most responsible for the success of application servers.
- Peer-to-peer: architectures where there are no special machines that provide a service or manage the network resources.[31]:227 Instead all responsibilities are uniformly divided among all machines, known as peers. Peers can serve both as clients and as servers.[32] Examples of this architecture include BitTorrent and the bitcoin network.

Another basic aspect of distributed computing architecture is the method of communicating and coordinating work among concurrent processes. Through various message passing protocols, processes may communicate directly with one another, typically in a master/slave relationship. Alternatively, a "database-centric" architecture can enable distributed computing to be done without any form of direct inter-process communication, by utilizing a shared database.[33] Database-centric architecture in particular provides relational processing analytics in a schematic architecture allowing for live environment relay. This enables distributed computing functions both within and beyond the parameters of a networked database.[34]

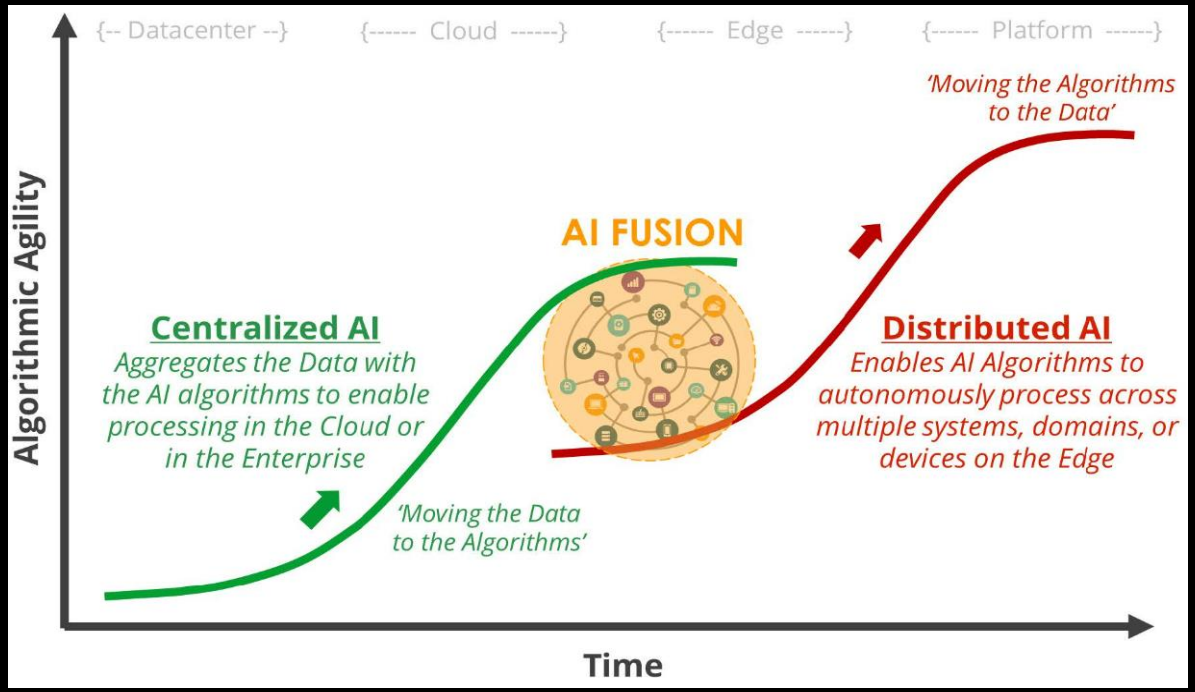
- https://en.wikipedia.org/wiki/CAP_theorem
- https://www.spiceworks.com/tech/cloud/articles/what-is-distributed-computing/
- https://www.freecodecamp.org/news/a-thorough-introduction-to-distributed-systems-3b91562c9b3c/
- …

Distributed AI

- https://en.wikipedia.org/wiki/Distributed_artificial_intelligence

Distributed Artificial Intelligence (DAI) also called Decentralized Artificial Intelligence[1] is a subfield of artificial intelligence research dedicated to the development of distributed solutions for problems. DAI is closely related to and a predecessor of the field of multi-agent systems.

- https://www.xenonstack.com/blog/distributed-ai-latest-trends
- https://developer.ibm.com/learningpaths/get-started-distributed-ai-apis/what-is-distributed-ai/
- https://engineering.cmu.edu/accelerator/news/2021/03/03-ai-fusion.html



AI Fusion

AI Frameworks & Algorithms

- AI Optimized Models
- Distributed Algorithmic Processing
- Autonomous Data Discovery
- Amorphous Learning & Prediction

AI Fabric & Abstraction Layer

- Dynamic AI Resource Management & Prioritization
- Algorithmic orchestration and ensembling between AI in the Enterprise and AI embedded in platforms and sensors at the Edge

AI Microelectronics & AI-infused Systems

- Edge AI Hardware
- AI Optimized Micro-Electronics
- Cognitive Sensors & Synthesis
- AI-Ready ICs supporting Algorithmic Multi-Threading

AI Engineering & Security

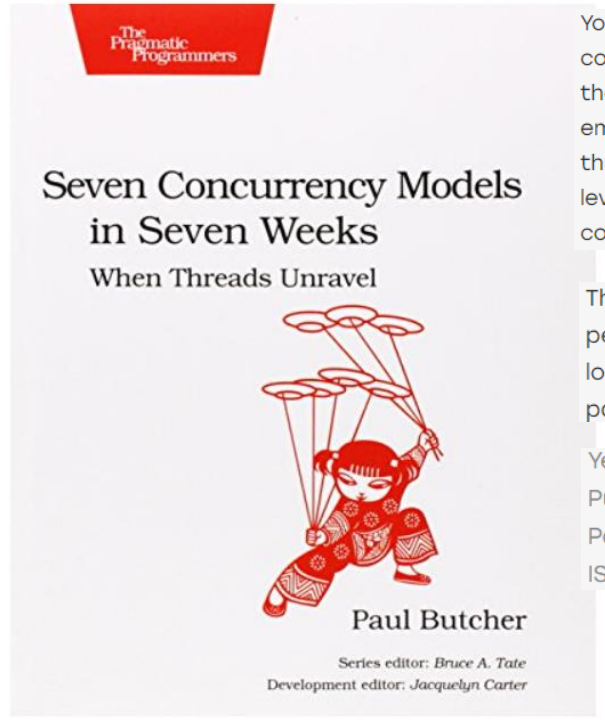
- AI Verification and Validation
- AI System Architecture and Integration
- AI Deployment and Monitoring *(AI DevOps)*
- Data / Model Management & Coherence
- AI Infrastructure Orchestration
- AI Policy Mechanisms *(Ethical Principles)*

1.2 Concurrent programming

- https://en.wikipedia.org/wiki/Concurrent_computing
- https://en.wikipedia.org/wiki/Concurrency_(computer_science)
- https://en.wikipedia.org/wiki/Parallel_computing

Concurrency Models

- Pragmatic Programmers



Your software needs to leverage multiple cores, handle thousands of users and terabytes of data, and continue working in the face of both hardware and software failure. Concurrency and parallelism are the keys, and *Seven Concurrency Models in Seven Weeks* equips you for this new world. See how emerging technologies such as actors and functional programming address issues with traditional threads and locks development. Learn how to exploit the parallelism in your computer's GPU and leverage clusters of machines with MapReduce and Stream Processing. And do it all with the confidence that comes from using tools that help you write crystal clear, high-quality code.

This book will show you how to exploit different parallel architectures to improve your code's performance, scalability, and resilience. You'll learn about seven concurrency models: threads and locks, functional programming, separating identity and state, actors, sequential processes, data parallelism, and the lambda architecture.

Year:	2014	Edition:	1
Publisher:	Pragmatic Bookshelf	Language:	english
Pages:	300	ISBN 10:	1937785653
ISBN 13:	9781937785659	Series:	The Pragmatic Programmers

- **Erlang**
  https://en.wikipedia.org/wiki/Erlang_(programming_language)
  Inspired by Actor Model and CSP(Communicating Sequential Processes).
- https://en.wikipedia.org/wiki/List_of_concurrent_and_parallel_programming_languages
- https://en.wikipedia.org/wiki/Actor_model

Name	Status	Latest release	License	Languages
...				
Vert.x	Active	2018-02-13	Apache 2.0	Java, Groovy, Javascript, Ruby, Scala, Kotlin, Ceylon
ActorFx	Inactive	2013-11-13	Apache 2.0	.NET
Akka (toolkit)	Active	2019-05-21[51]	Commercial[52] (Apache 2.0 up to 2.6.19)	Java and Scala
Akka.NET	Active	2020-08-20[53]	Apache 2.0	.NET
...				
rotor	Active	2022-04-23[80]	MIT License	C++17
Orleans	Active	2022-08-15[81]	MIT License	C#/.NET
Skynet	Active	2020-12-10	MIT License	C/Lua
Reactors.IO	Active	2016-06-14	BSD License	Java/Scala
libagents	Active	2020-03-08	Free software license	C++11
Proto.Actor	Active	2021-01-05	Free software license	Go, C#, Python, JavaScript, Kotlin
FunctionalJava	Active	2018-08-18[82]	BSD 3-Clause	Java
Riker	Active	2019-01-04	MIT License	Rust
Comedy	Active	2019-03-09	EPL 1.0	JavaScript
VLINGO XOOM Actors	Active	2017-12-20	Mozilla Public License 2.0	Java, Kotlin, JVM languages, C# .NET
wasmCloud	Active	2021-03-23	Apache 2.0	WebAssembly (Rust, TinyGo, Zig, AssemblyScript)
ray	Active	2020-08-27	Apache 2.0	Python
cell	Active	2012-08-02	New BSD License	Python

1.3 Rust is booming

1.3.1 What's new in Rust

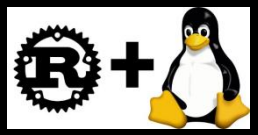
- https://foundation.rust-lang.org/

- https://www.infoworld.com/article/3267624/whats-new-in-the-rust-language.html
- https://github.com/RalfJung/minirust
  A precise specification for "Rust lite / MIR plus".
- https://github.com/rust-lang/miri/
  An interpreter for Rust's mid-level intermediate representation.
- https://lwn.net/Articles/900721/ //Rust frontend approved for GCC
- https://foundation.rust-lang.org/news/2022-09-13-rust-foundation-establishes-security-team/
- …

Roadmap

- https://blog.rust-lang.org/inside-rust/2022/04/04/lang-roadmap-2024.html
- https://blog.rust-lang.org/inside-rust/2022/02/22/compiler-team-ambitions-2022.html
- https://www.ncameron.org/blog/ten-challenges-for-rust/

1.3.2 Rust heads into Linux Kernel



- **What's happening!**

  https://lwn.net/Articles/909095/ //BPF as a safer kernel programming env
  https://lwn.net/Articles/889924/ //Rustaceans at the border
  ~~https://lwn.net/Articles/853423/ //Rust heads into the kernel?~~
  https://lwn.net/Articles/852704/ //Rust in the Linux kernel
  https://lwn.net/Articles/849849/ //Rust support hits linux-next
  https://lwn.net/Articles/829858/ //Supporting Linux kernel development in Rust
  …

- **https://github.com/Rust-for-Linux**

  The goal of this project is to add support for the Rust language to the Linux kernel. This repository contains the work that will be eventually submitted for review to the LKML.

  Feel free to contribute! To start, take a look at `Documentation/rust`.

- **Rust for Linux patch 10**

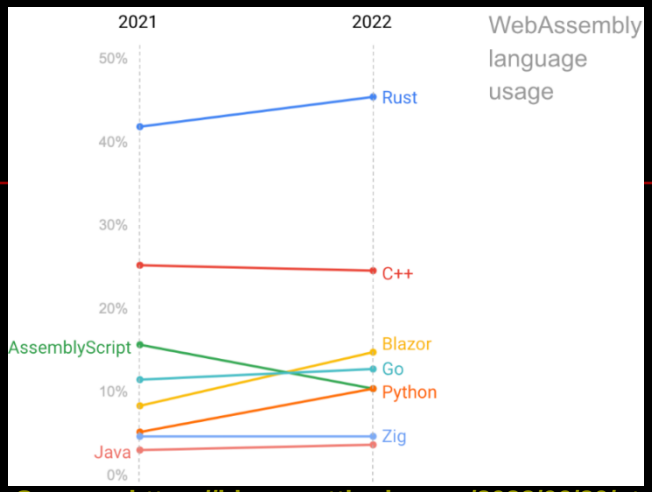
  https://www.phoronix.com/news/Rust-v10-Linux-Patches
  https://lore.kernel.org/lkml/20220927131518.30000-1-ojeda@kernel.org/

  ```
  89 files changed, 12552 insertions(+), 51 deletions(-)
  ```

Get Rust into Linux kernel 6.1

- https://events.linuxfoundation.org/linux-kernel-maintainer-summit/
- https://www.phoronix.com/news/Rust-Is-Merged-Linux-6.1

1.3.3 Rust for Wasm

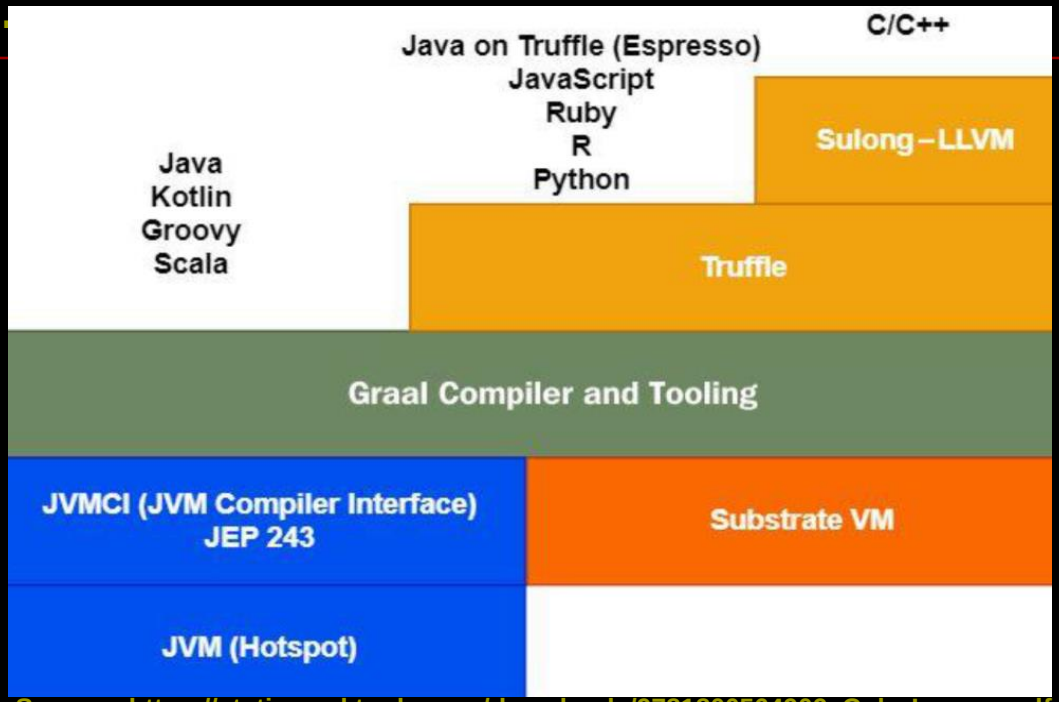


Source: https://blog.scottlogic.com/2022/06/20/state-of-wasm-2022.html

- https://www.rust-lang.org/what/wasm
- https://rustwasm.github.io/book
- https://github.com/rustwasm
- https://github.com/bytecodealliance/wasmtime
- https://github.com/wasmerio/wasmer
- https://github.com/wasmCloud/wasmCloud
- https://github.com/paritytech/wasmi
- https://github.com/yewstack/yew
- https://github.com/denoland/deno
- …

1.4 GraalVM

- https://en.wikipedia.org/wiki/GraalVM
- https://www.graalvm.org/
-

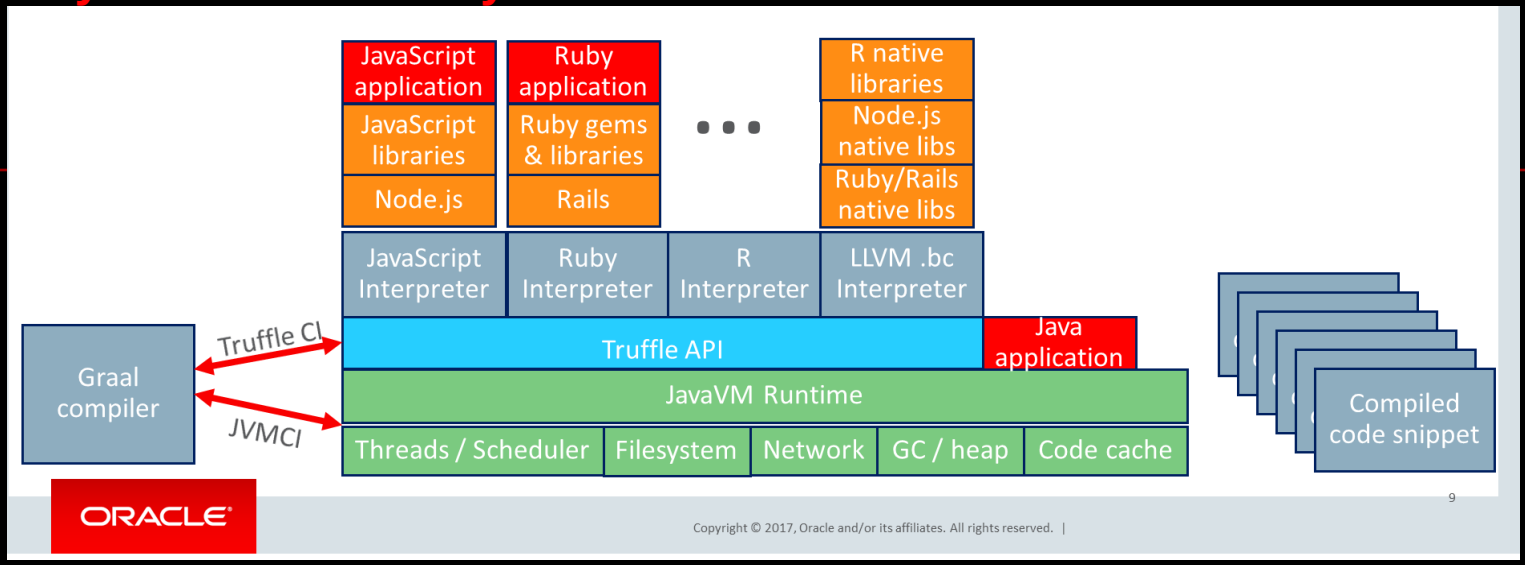


Source: https://static.packt-cdn.com/downloads/9781800564909_ColorImages.pdf

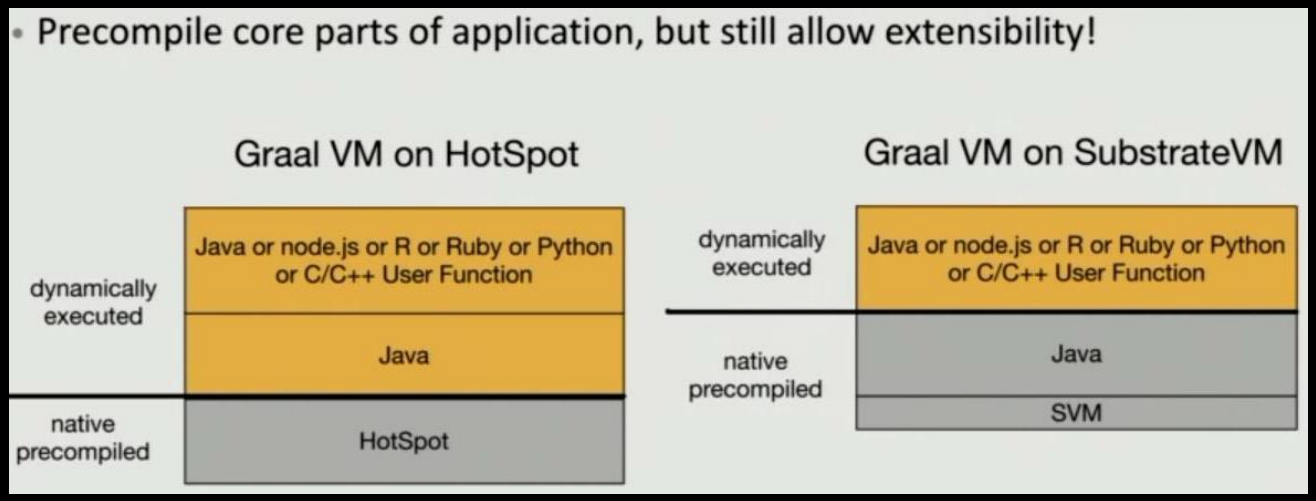
- A Universal High-Performance Polyglot VM
- A meta-runtime for Language-Level Virtualization
- Base on OpenJDK 8, 11, 16, 17 and 19 with JVMCI support.
- https://github.com/graalvm
- https://github.com/oracle/graal

Architecture & design

- A hybrid of static & dynamic runtimes

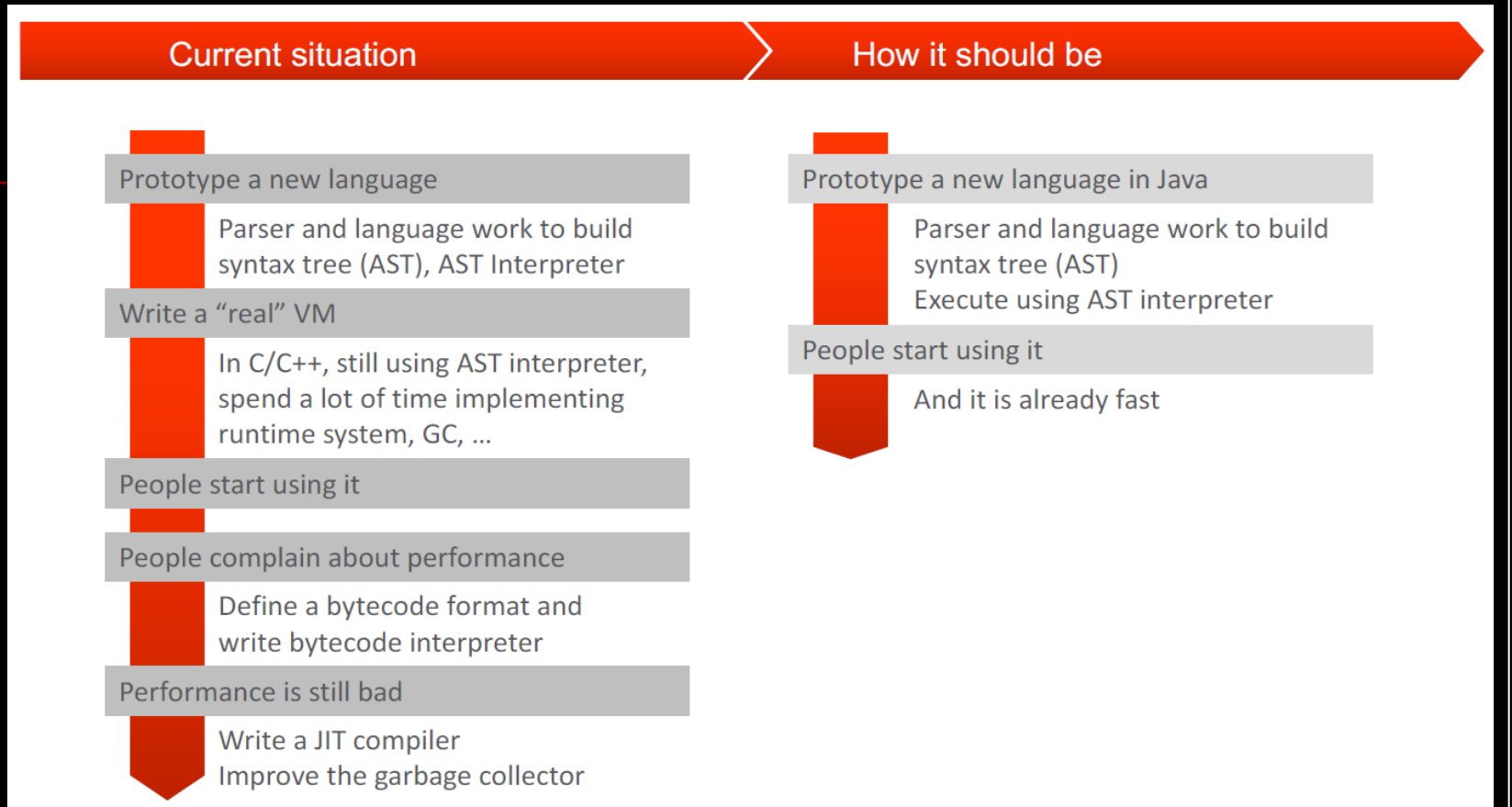


Source: https://ics.psu.edu/wp-content/uploads/2017/02/GraalVM-PSU.pptx



Source: "Adopting Java for the Serverless world", Vadym Kazulkin, JUG London 2020.

Implement a new language runtime



Source: "Turning the JVM into a Polyglot VM with Graal", Chris Seaton, Oracle Labs.

https://www.graalvm.org/22.0/graalvm-as-a-platform/language-implementation-framework/Languages/

1.4.1 Current release and roadmap

- https://medium.com/graalvm/announcing-the-graalvm-community-roadmap-b8d77201b497

GraalVM 22.2

- https://medium.com/graalvm/graalvm-22-2-smaller-jdk-size-improved-memory-usage-better-library-support-and-more-cb34b5b68ec0

  Smaller JDK size, improved memory usage, better library support, and more!

GraalVM version and platform	JDK size in 22.1	JDK size in 22.2	Difference
GraalVM Community / Java 17 / Linux AMD64	431 MB	251 MB	-42%
GraalVM Community / Java 17 / Darwin AMD64	425 MB	247 MB	-42%
GraalVM Enterprise / Java 17 / Darwin AMD64	495 MB	271 MB	-45%

...

JDK 19

- https://openjdk.java.net/projects/jdk/19/

  Features

405: Record Patterns (Preview)
422: Linux/RISC-V Port
424: Foreign Function & Memory API (Preview)
425: Virtual Threads (Preview)
426: Vector API (Fourth Incubator)
427: Pattern Matching for switch (Third Preview)
428: Structured Concurrency (Incubator)

- https://jdk.java.net/19/release-notes

1.5 eBPF

- https:https://en.wikipedia.org/wiki/Berkeley_Packet_Filter

Extensions and optimizations [edit]

Some projects use BPF instruction sets or execution techniques different from the originals.

Some platforms, including FreeBSD, NetBSD, and WinPcap, use a just-in-time (JIT) compiler to convert BPF instructions into native code in order to improve performance. Linux includes a BPF JIT compiler which is disabled by default.

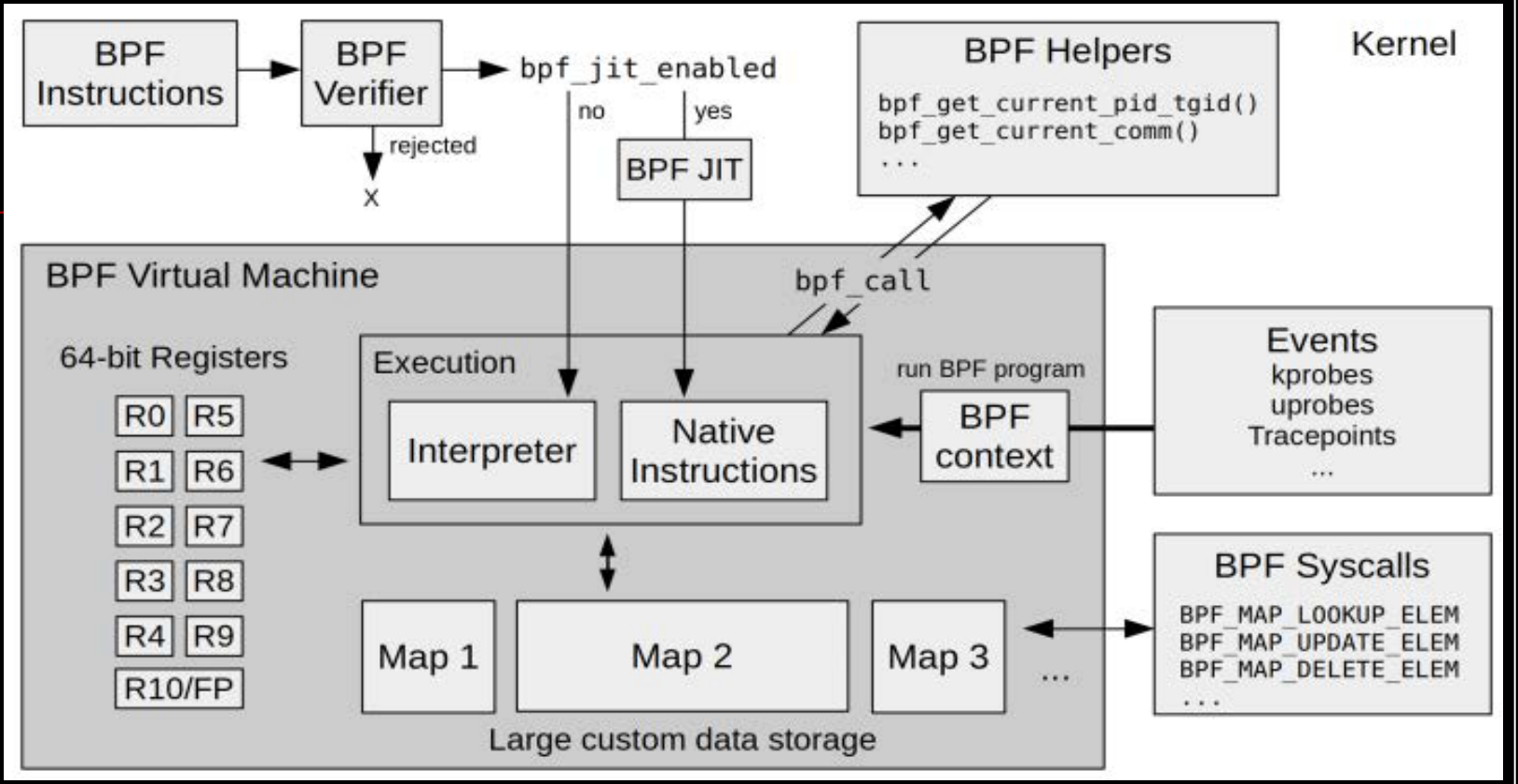
Kernel-mode interpreters for that same virtual machine language are used in raw data link layer mechanisms in other operating systems, such as Tru64 Unix, and for socket filters in the Linux kernel and in the WinPcap and Npcap packet capture mechanism.

Since version 3.18, the Linux kernel includes an extended BPF virtual machine with ten 64-bit registers, termed **extended BPF** (**eBPF**). It can be used for non-networking purposes, such as for attaching eBPF programs to various tracepoints.[5][6][7] Since kernel version 3.19, eBPF filters can be attached to sockets,[8][9] and, since kernel version 4.1, to traffic control classifiers for the ingress and egress networking data path.[10][11] The original and obsolete version has been retroactively renamed to **classic BPF** (**cBPF**). Nowadays, the Linux kernel runs eBPF only and loaded cBPF bytecode is transparently translated into an eBPF representation in the kernel before program execution.[12] All bytecode is verified before running to prevent denial-of-service attacks. Until Linux 5.3, the verifier prohibited the use of loops.[13]

A user-mode interpreter for BPF is provided with the libpcap/WinPcap/Npcap implementation of the pcap API, so that, when capturing packets on systems without kernel-mode support for that filtering mechanism, packets can be filtered in user mode; code using the pcap API will work on both types of systems, although, on systems where the filtering is done in user mode, all packets, including those that will be filtered out, are copied from the kernel to user space. That interpreter can also be used when reading a file containing packets captured using pcap.

- Aims at being a universal in-kernel virtual machine
- A simple way to extend the functionality of Kernel at runtime
- "DTrace for Linux"
- $KERNEL_SRC/Documentation/networking/filter.txt
- https://lwn.net/Articles/740157/ //A thorough introduction to eBPF
- https://github.com/iovisor/bcc/blob/master/docs/kernel-versions.md
- https://docs.cilium.io/en/stable/bpf/
- https://brendangregg.com/bpf-performance-tools-book.html
- https://github.com/dthaler/ebpf-docs/blob/update/isa/kernel.org/instruction-set.rst

BPF runtime internals



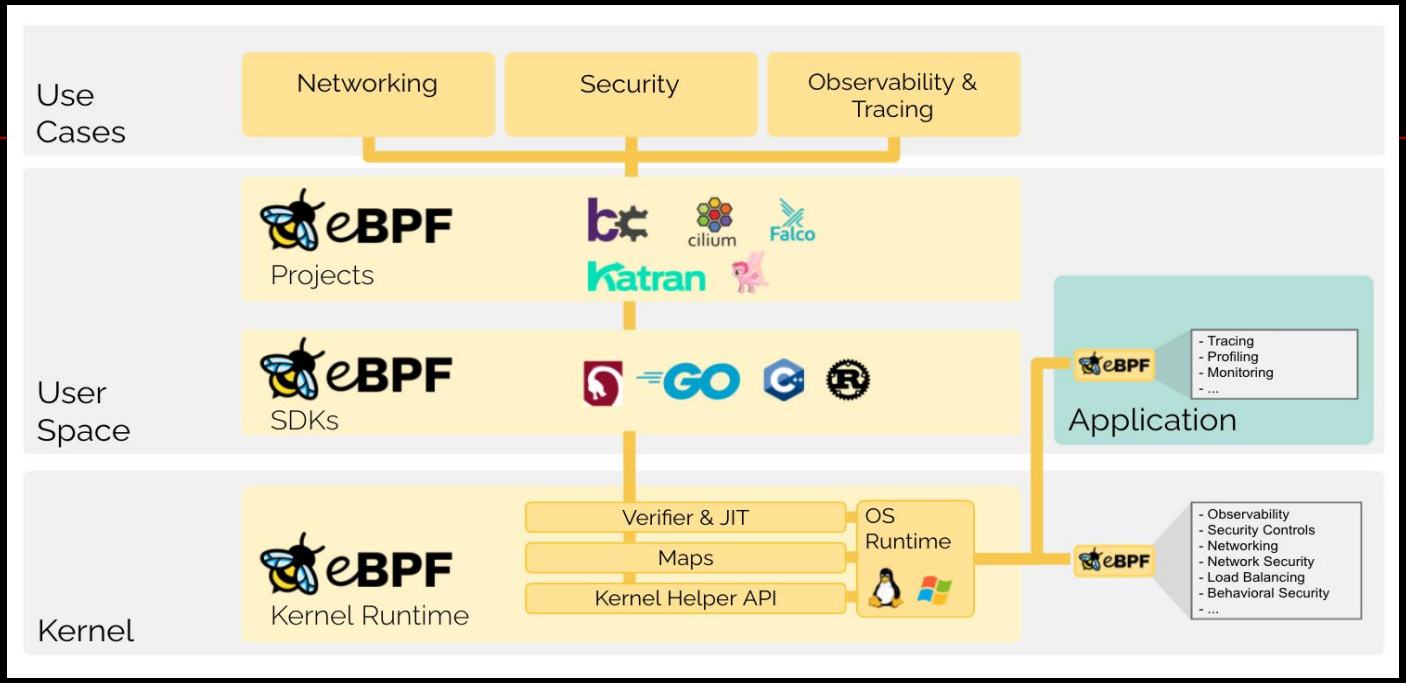
Source: https://brendangregg.com/bpf-performance-tools-book.html

Kernel configuration

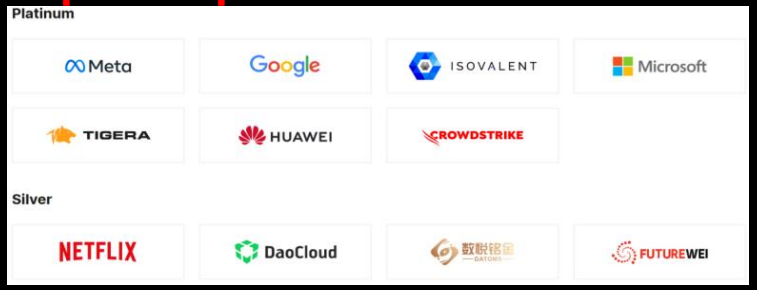
```
[mydev@fedora ~]$ cat /boot/config-5.19.14-200.fc36.aarch64 |grep -i BPF
CONFIG_BPF=y
CONFIG_HAVE_EBPF_JIT=y
CONFIG_ARCH_WANT_DEFAULT_BPF_JIT=y
# BPF subsystem
CONFIG_BPF_SYSCALL=y
CONFIG_BPF_JIT=y
CONFIG_BPF_JIT_ALWAYS_ON=y
CONFIG_BPF_JIT_DEFAULT_ON=y
CONFIG_BPF_UNPRIV_DEFAULT_OFF=y
CONFIG_BPF_PRELOAD=y
CONFIG_BPF_PRELOAD_UMD=m
CONFIG_BPF_LSM=y
# end of BPF subsystem
CONFIG_CGROUP_BPF=y
CONFIG_IPV6_SEG6_BPF=y
CONFIG_NETFILTER_XT_MATCH_BPF=m
# CONFIG_BPFILTER is not set
CONFIG_NET_CLS_BPF=m
CONFIG_NET_ACT_BPF=m
CONFIG_BPF_STREAM_PARSER=y
CONFIG_LWTUNNEL_BPF=y
CONFIG_BPF_LIRC_MODE2=y
CONFIG_LSM="lockdown,yama,integrity,selinux,bpf,landlock"
CONFIG_BPF_EVENTS=y
# CONFIG_BPF_KPROBE_OVERRIDE is not set
CONFIG_TEST_BPF=m
[mydev@fedora ~]$
```

1.5.1 Official Site and Foundation

- https://ebpf.io/



- https://ebpf.io/foundation/



https://www.linuxfoundation.org/press-release/facebook-google-isovalent-microsoft-and-netflix-launch-ebpf-foundation-as-part-of-the-linux-foundation/

1.5.2 Rust for eBPF

Overview

- https://kbknapp.dev/ebpf-part-i/
- https://kbknapp.dev/ebpf-part-ii/
- https://kbknapp.dev/ebpf-part-iii/
- https://kbknapp.dev/ebpf-part-iv/
- …

1.5.2.1 Aya

- https://github.com/aya-rs/aya

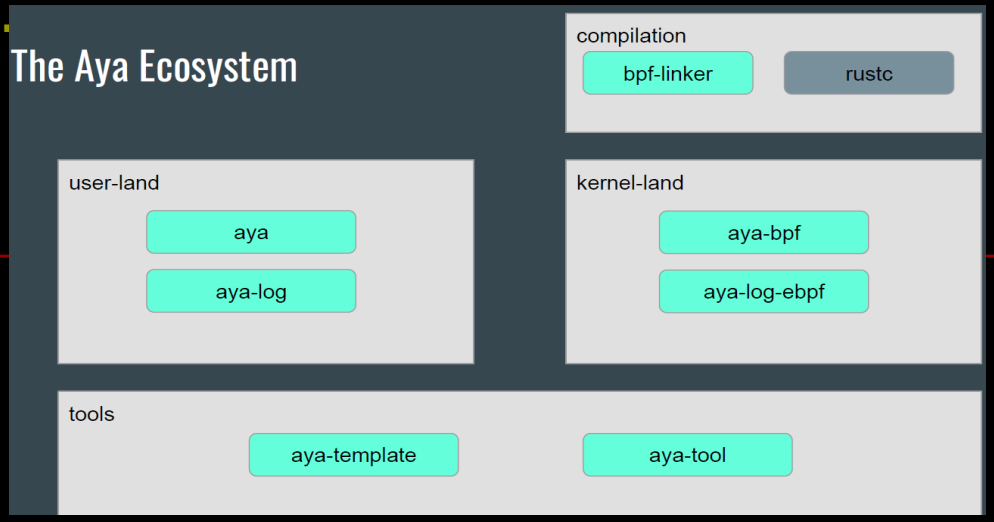
  An eBPF library for the Rust programming language, built with a focus on developer experience and operability.

- **Features**

Aya is an eBPF library built with a focus on operability and developer experience. It does not rely on libbpf nor bcc - it's built from the ground up purely in Rust, using only the libc crate to execute syscalls. With BTF support and when linked with musl, it offers a true compile once, run everywhere solution, where a single self-contained binary can be deployed on many linux distributions and kernel versions.

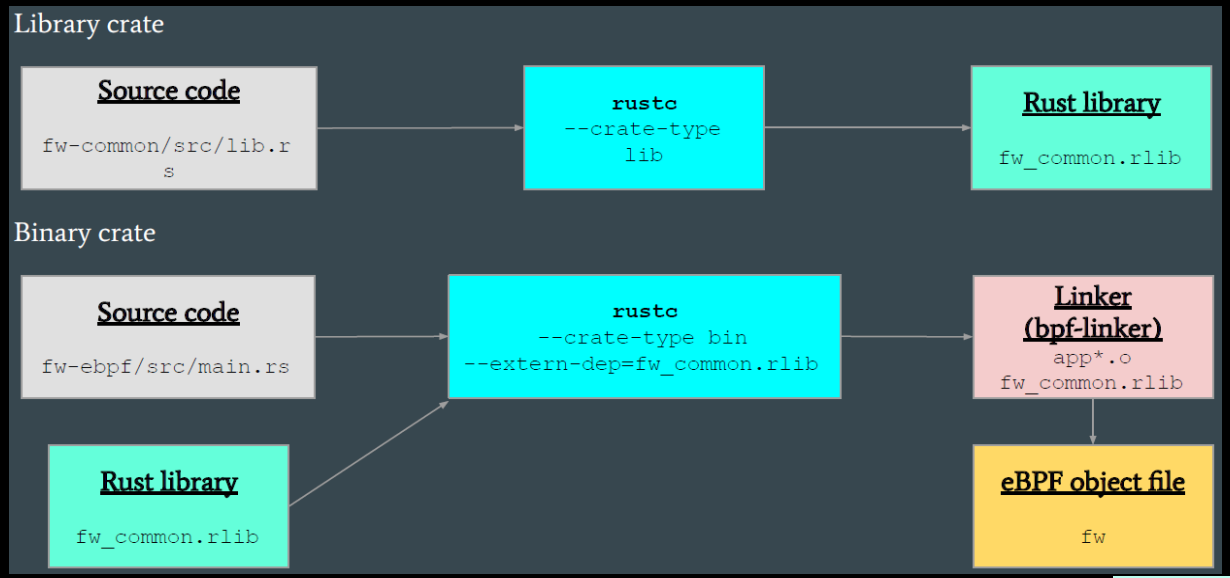
Some of the major features provided include:

- Support for the **BPF Type Format** (BTF), which is transparently enabled when supported by the target kernel. This allows eBPF programs compiled against one kernel version to run on different kernel versions without the need to recompile.
- Support for function call relocation and global data maps, which allows eBPF programs to make **function calls** and use **global variables and initializers**.
- **Async support** with both tokio and async-std.
- Easy to deploy and fast to build: aya doesn't require a kernel build or compiled headers, and not even a C toolchain; a release build completes in a matter of seconds.



Source: "Rust in the Kernel (via eBPF)", Dave Tucker etc, LPC 2022.

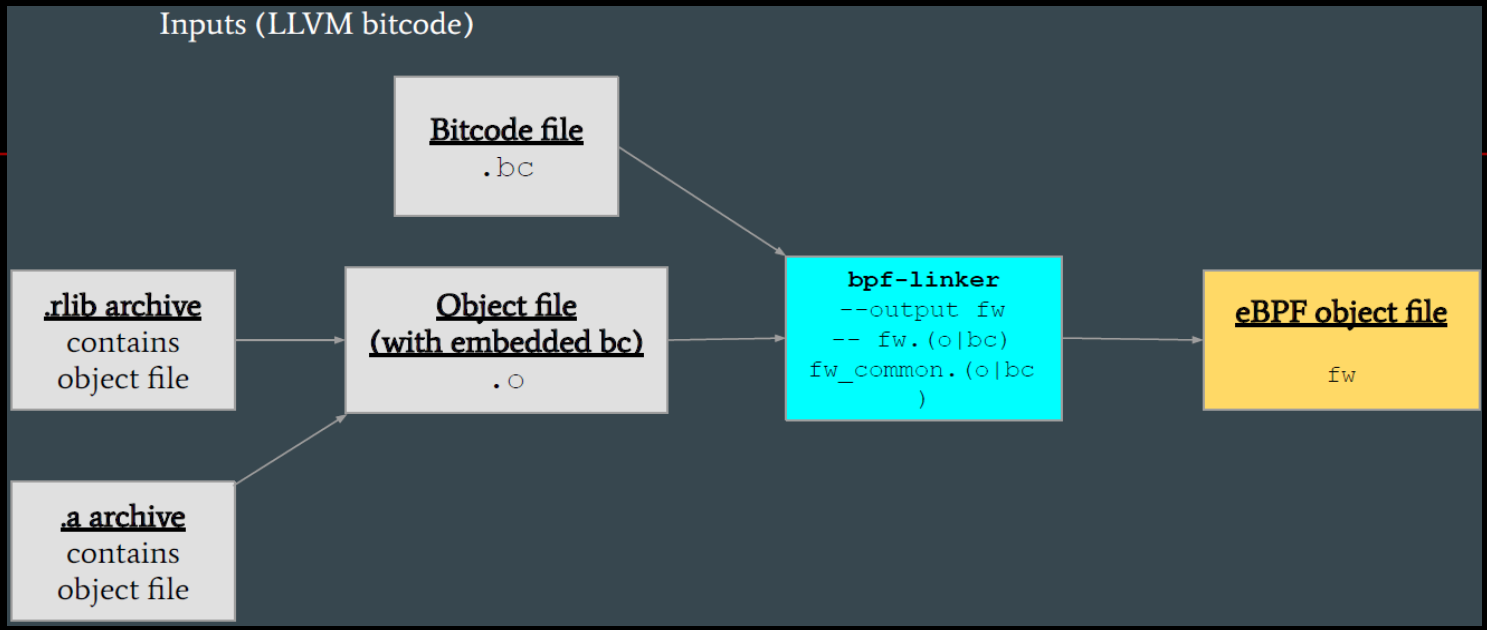
Rust -> eBPF compilation



Source: "Rust in the Kernel (via eBPF)", Dave Tucker etc, LPC 2022.

- **bpf-linker**

  https://github.com/aya-rs/bpf-linker)



Source: "Rust in the Kernel (via eBPF)", Dave Tucker etc, LPC 2022.

- **https://lwn.net/Articles/859784/ //Aya: writing BPF in Rust**
- **https://aya-rs.dev/book/**
- **https://github.com/aya-rs/awesome-aya**
- **https://deepfence.io/aya-your-trusty-ebpf-companion/**
- **…**

1.6 Arrow

- https://en.wikipedia.org/wiki/Apache_Arrow

Apache Arrow is a language-agnostic software framework for developing data analytics applications that process columnar data. It contains a standardized column-oriented memory format that is able to represent flat and hierarchical data for efficient analytic operations on modern CPU and GPU hardware.[3][4][5][6][7] This reduces or eliminates factors that limit the feasibility of working with large sets of data, such as the cost, volatility, or physical constraints of dynamic random-access memory.[8]

Interoperability [edit]

Arrow can be used with Apache Parquet, Apache Spark, NumPy, PySpark, pandas and other data processing libraries. The project includes native software libraries written in C, C++, C#, Go, Java, JavaScript, Julia, MATLAB, Python, R, Ruby, and Rust. Arrow allows for zero-copy reads and fast data access and interchange without serialization overhead between these languages and systems.[3]

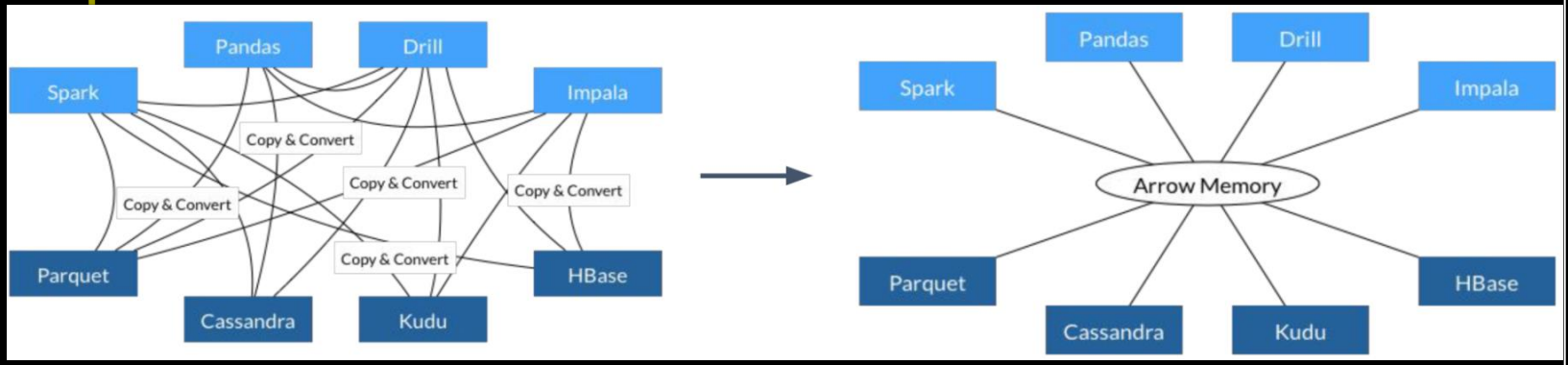
Applications [edit]

Arrow has been used in diverse domains, including analytics,[9] genomics,[10][8] and cloud computing.[11]

Comparison to Apache Parquet and ORC [edit]

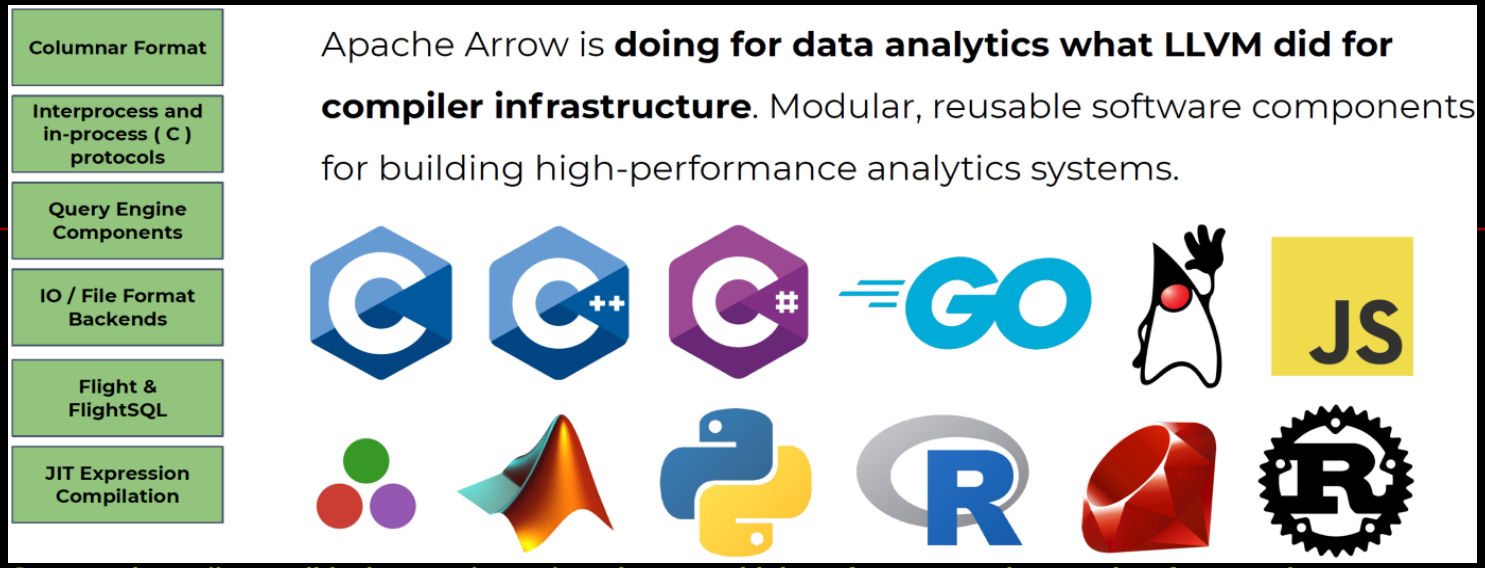
Apache Parquet and Apache ORC are popular examples of on-disk columnar data formats. Arrow is designed as a complement to these formats for processing data in-memory.[12] The hardware resource engineering trade-offs for in-memory processing vary from those associated with on-disk storage.[13] The Arrow and Parquet projects includes libraries that allow for reading and writing data between the two formats.[14]

- https://arrow.apache.org/
- Simplified data access with the Arrow columnar format



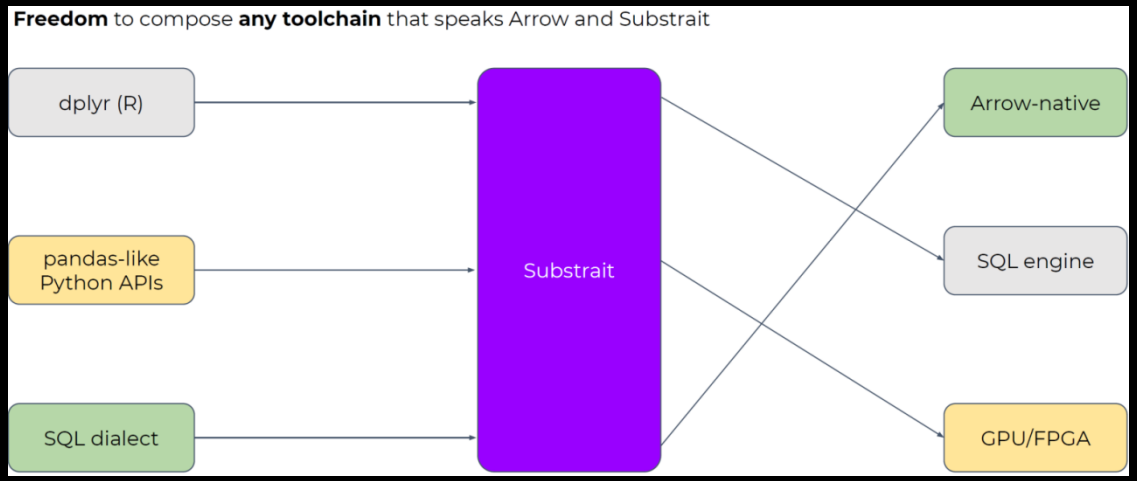
Source: https://www.slideshare.net/wesm/apache-arrow-high-performance-columnar-data-framework

Features



Source: https://www.slideshare.net/wesm/apache-arrow-high-performance-columnar-data-framework

LEGO for data ecosystem



Source: https://www.slideshare.net/wesm/apache-arrow-high-performance-columnar-data-framework

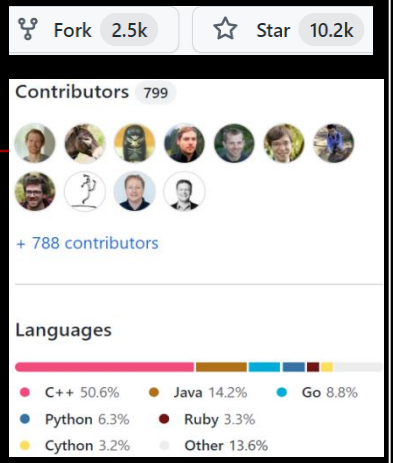
- - -

Src

- https://github.com/apache/arrow

Major components of the project include:

- The Arrow Columnar In-Memory Format: a standard and efficient in-memory representation of various datatypes, plain or nested
- The Arrow IPC Format: an efficient serialization of the Arrow format and associated metadata, for communication between processes and heterogeneous environments
- The Arrow Flight RPC protocol: based on the Arrow IPC format, a building block for remote services exchanging Arrow data with application-defined semantics (for example a storage server or a database)
- C++ libraries
- C bindings using GLib
- C# .NET libraries
- Gandiva: an LLVM-based Arrow expression compiler, part of the C++ codebase
- Go libraries
- Java libraries
- JavaScript libraries
- Plasma Object Store: a shared-memory blob store, part of the C++ codebase
- Python libraries
- R libraries
- Ruby libraries
- Rust libraries



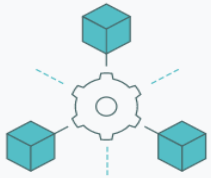
1.7 gRPC

- https://grpc.io/

A high performance, open source universal RPC framework.

Simple service definition

Define your service using Protocol Buffers, a powerful binary serialization toolset and language

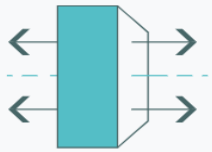


Start quickly and scale

Install runtime and dev environments with a single line and also scale to millions of RPCs per second with the framework

Works across languages and platforms

Automatically generate idiomatic client and server stubs for your service in a variety of languages and platforms



Bi-directional streaming and integrated auth

Bi-directional streaming and fully integrated pluggable authentication with HTTP/2-based transport

- **Why is it**

gRPC is a modern open source high performance Remote Procedure Call (RPC) framework that can run in any environment. It can efficiently connect services in and across data centers with pluggable support for load balancing, tracing, health checking and authentication. It is also applicable in last mile of distributed computing to connect devices, mobile applications and browsers to backend services.

Src

- https://github.com/grpc/grpc

Fork 9.4k Star 35.6k

For instructions on how to use the language-specific gRPC runtime for a project, please refer to these documents

- C++: follow the instructions under the `src/cpp` directory
- C#: NuGet package `Grpc`
- Dart: pub package `grpc`
- Go: `go get google.golang.org/grpc`
- Java: Use JARs from Maven Central Repository
- Kotlin: Use JARs from Maven Central Repository
- Node: `npm install grpc`
- Objective-C: Add `gRPC-ProtoRPC` dependency to podspec
- PHP: `pecl install grpc`
- Python: `pip install grpcio`
- Ruby: `gem install grpc`
- WebJS: follow the grpc-web instructions

Contributors 748

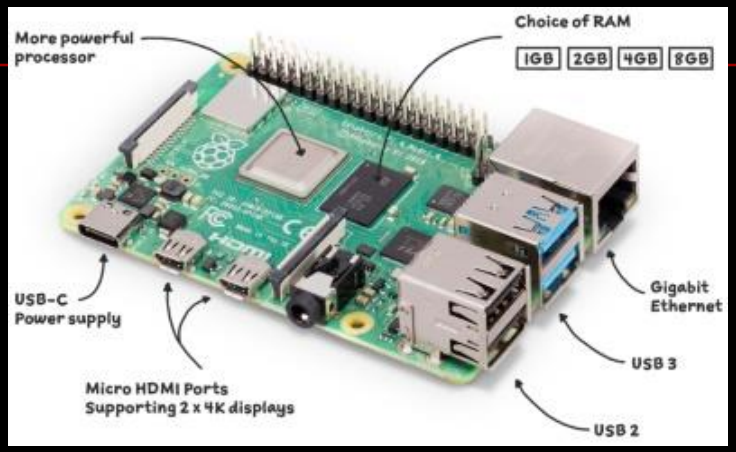
+ 737 contributors

Languages

2) Testbed

2.1 Raspberry Pi 4(8GB LPDDR4) with Fedora 36

- HW env



SW env

```
[mydev@fedora ~]$ uname -a
Linux fedora 5.19.14-200.fc36.aarch64 #1 SMP PREEMPT_DYNAMIC Wed Oct 5 21:10:47 UTC 2022 aarch64 aarch64 aarch64 GNU/Linux
[mydev@fedora ~]$
[mydev@fedora ~]$ free -m
               total        used        free      shared  buff/cache   available
Mem:            7827         446        5330          13        2051        7115
Swap:           7826           0        7826
[mydev@fedora ~]$
```

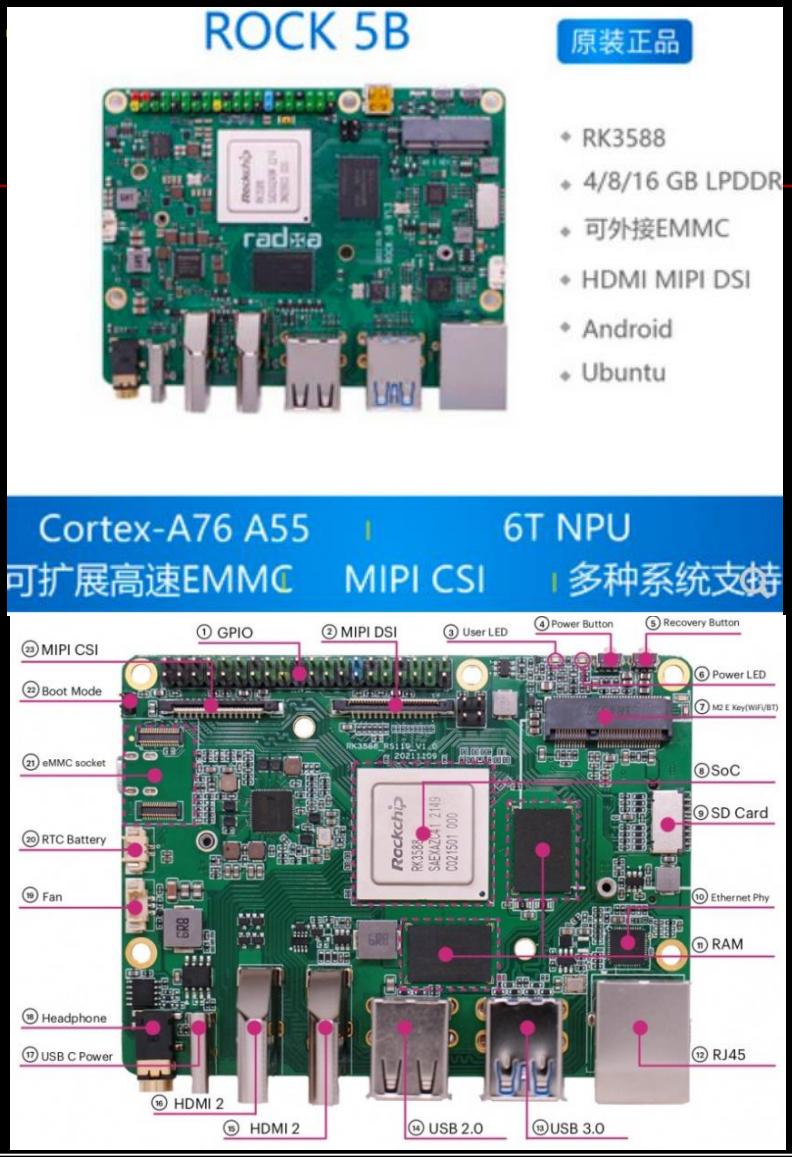
```
[mydev@fedora ~]$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/aarch64-redhat-linux/12/lto-wrapper
Target: aarch64-redhat-linux
Configured with: ../configure --enable-bootstrap --enable-languages=c,c++,fortran,objc,obj-c++,ada,go,d,lto --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --enable-shared --enable-threads=posix --enable-checking=release --enable-multilib --with-system-zlib --enable-__cxa_atexit --disable-libunwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-version-only --enable-libstdcxx-backtrace --with-linker-hash-style=gnu --enable-plugin --enable-initfini-array --with-isl=/builddir/build/BUILD/gcc-12.2.1-20220819/obj-aarch64-redhat-linux/isl-install --enable-gnu-indirect-function --build=aarch64-redhat-linux --with-build-config=bootstrap-lto --enable-link-serialization=1
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 12.2.1 20220819 (Red Hat 12.2.1-2) (GCC)
[mydev@fedora ~]$
[mydev@fedora ~]$ clang -v
clang version 14.0.5 (Fedora 14.0.5-1.fc36)
Target: aarch64-redhat-linux-gnu
Thread model: posix
InstalledDir: /usr/bin
Found candidate GCC installation: /usr/bin/../lib/gcc/aarch64-redhat-linux/12
Selected GCC installation: /usr/bin/../lib/gcc/aarch64-redhat-linux/12
Candidate multilib: .;@m64
Selected multilib: .;@m64
[mydev@fedora ~]$
```

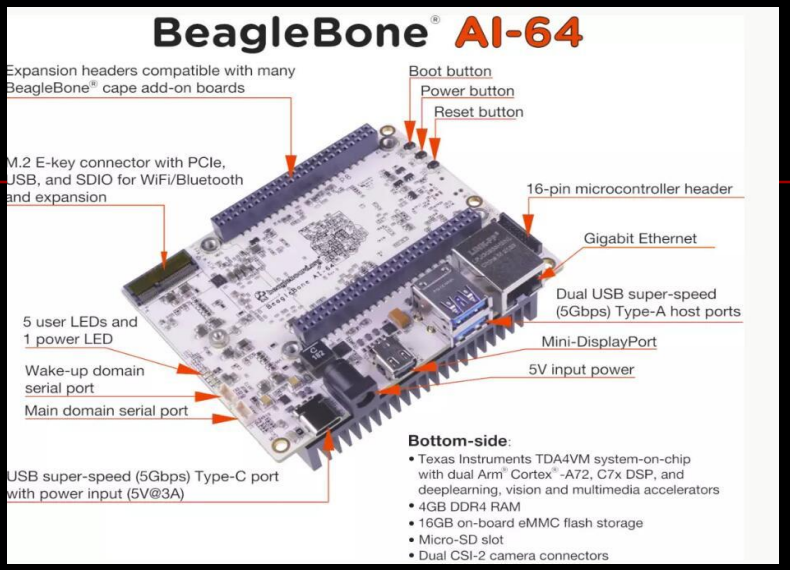
```
[mydev@fedora ~]$ python -V
GraalVM Python 3.8.5 (GraalVM CE Native 22.3.0-dev)
[mydev@fedora ~]$
[mydev@fedora ~]$ java --version
openjdk 19 2022-09-20
OpenJDK Runtime Environment GraalVM CE 22.3.0-dev (build 19+36-jvmci-22.3-b06)
OpenJDK 64-Bit Server VM GraalVM CE 22.3.0-dev (build 19+36-jvmci-22.3-b06, mixed mode, sharing)
[mydev@fedora ~]$
[mydev@fedora ~]$ gu list
ComponentId              Version             Component name                Stability                     Origin
------------------------------------------------------------------------------------------------------------------
graalvm                  22.3.0-dev          GraalVM Core                  Experimental
js                       22.3.0-dev          Graal.js                      Experimental
llvm                     22.3.0-dev          LLVM Runtime Core             Experimental
llvm-toolchain           22.3.0-dev          LLVM.org toolchain            Experimental
native-image             22.3.0-dev          Native Image                  Experimental
native-image-llvm-backend22.3.0-dev          Native Image LLVM Backend     Experimental
python                   22.3.0-dev          GraalVM Python                Experimental
visualvm                 22.3.0-dev          VisualVM                      Experimental
wasm                     22.3.0-dev          GraalWasm                     Experimental
[mydev@fedora ~]$
```

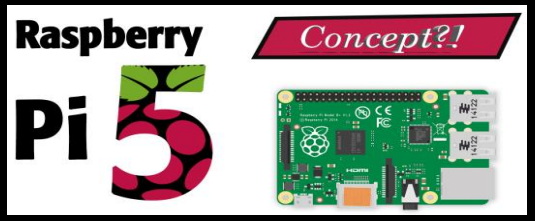
```
[mydev@fedora ~]$ go version
go version go1.19.1 linux/arm64
[mydev@fedora ~]$
[mydev@fedora ~]$ rustc -V
rustc 1.64.0 (a55dd71d5 2022-09-19)
[mydev@fedora ~]$
[mydev@fedora ~]$ cargo -V
cargo 1.64.0 (387270bc7 2022-09-16)
[mydev@fedora ~]$
[mydev@fedora ~]$ rustup -V
rustup 1.25.1 (bb60b1e89 2022-07-12)
info: This is the version for the rustup toolchain manager, not the rustc compiler.
info: The currently active `rustc` version is `rustc 1.64.0 (a55dd71d5 2022-09-19)`
[mydev@fedora ~]$
[mydev@fedora ~]$ dotnet --version
7.0.100-rc.1.22431.12
[mydev@fedora ~]$ _
```

```
[mydev@fedora ~]$ cmake --version
cmake version 3.22.2

CMake suite maintained and supported by Kitware (kitware.com/cmake).
[mydev@fedora ~]$
[mydev@fedora ~]$ ninja --version
1.10.2
[mydev@fedora ~]$
[mydev@fedora ~]$ make --version
GNU Make 4.3
Built for aarch64-redhat-linux-gnu
Copyright (C) 1988-2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
[mydev@fedora ~]$
[mydev@fedora ~]$ mvn --version
Apache Maven 3.8.4 (Red Hat 3.8.4-3)
Maven home: /usr/share/maven
Java version: 19, vendor: GraalVM Community, runtime: /opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.19.14-200.fc36.aarch64", arch: "aarch64", family: "unix"
[mydev@fedora ~]$
[mydev@fedora ~]$ bazel version
WARNING: Invoking Bazel in batch mode since it is not invoked from within a workspace (below a directory having a WORKSPACE file).
Build label: 5.3.1
Build target: bazel-out/aarch64-opt/bin/src/main/java/com/google/devtools/build/lib/bazel/BazelServer_deploy.jar
Build time: Mon Sep 19 17:28:14 2022 (1663608494)
Build timestamp: 1663608494
Build timestamp as int: 1663608494
[mydev@fedora ~]$
[mydev@fedora ~]$ gradle --version

------------------------------------------------------------
Gradle 7.5.1
------------------------------------------------------------

Build time:   2022-08-05 21:17:56 UTC
Revision:     d1daa0cbf1a0103000b71484e1dbfe096e095918

Kotlin:       1.6.21
Groovy:       3.0.10
Ant:          Apache Ant(TM) version 1.10.11 compiled on July 10 2021
JVM:          19 (GraalVM Community 19+36-jvmci-22.3-b06)
OS:           Linux 5.19.14-200.fc36.aarch64 aarch64

[mydev@fedora ~]$
```

2.2 Switching to more powerful development boards







II. Ray 2.0

1) An overview of Project Ray

1.1 What is it

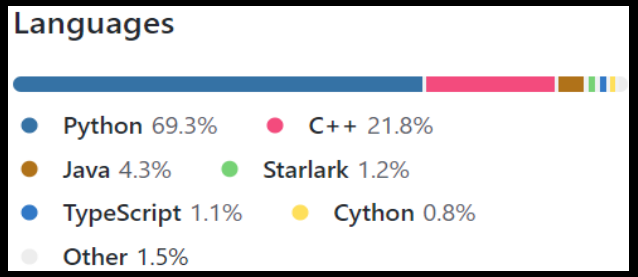
- **https://www.ray.io/**

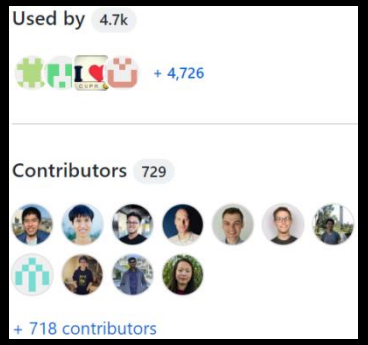
  Effortlessly scale your most complex workloads

  Ray is an open-source unified compute framework that makes it easy to scale AI and Python workloads — from reinforcement learning to deep learning to tuning, and model serving. Learn more about Ray's rich set of libraries and integrations.

- **https://github.com/ray-project/ray/**

  A unified framework for scaling AI and Python applications. Ray consists of a core distributed runtime and a toolkit of libraries (Ray AIR) for accelerating ML workloads.



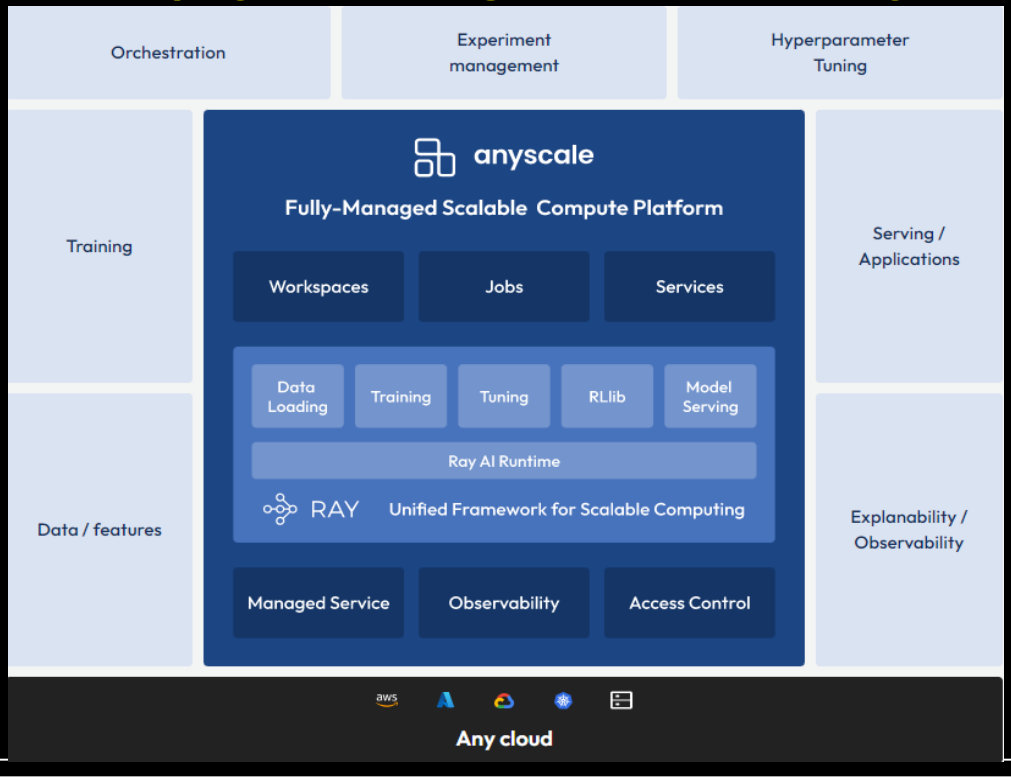


- **https://rise.cs.berkeley.edu/projects/ray/**

Ray is a high-performance distributed execution framework targeted at large-scale machine learning and reinforcement learning applications. It achieves scalability and fault tolerance by abstracting the control state of the system in a global control store and keeping all other components stateless. It uses a shared-memory distributed object store to efficiently handle large data through shared memory, and it uses a bottom-up hierarchical scheduling architecture to achieve low-latency and high-throughput scheduling. It uses a lightweight API based on dynamic task graphs and actors to express a wide range of applications in a flexible manner.

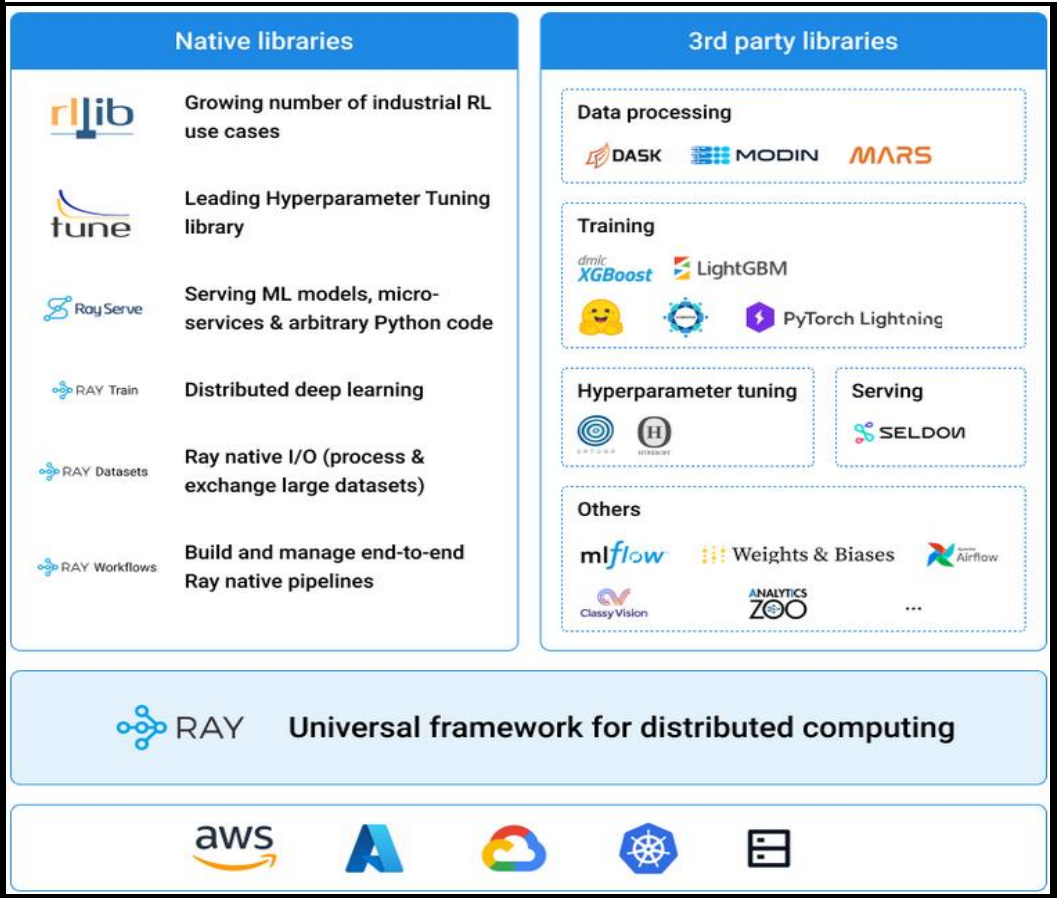
- **https://www.anyscale.com/**

Anyscale is a fully managed, unified compute platform that makes it easy to build, deploy, and manage scalable AI and Python applications on Ray.



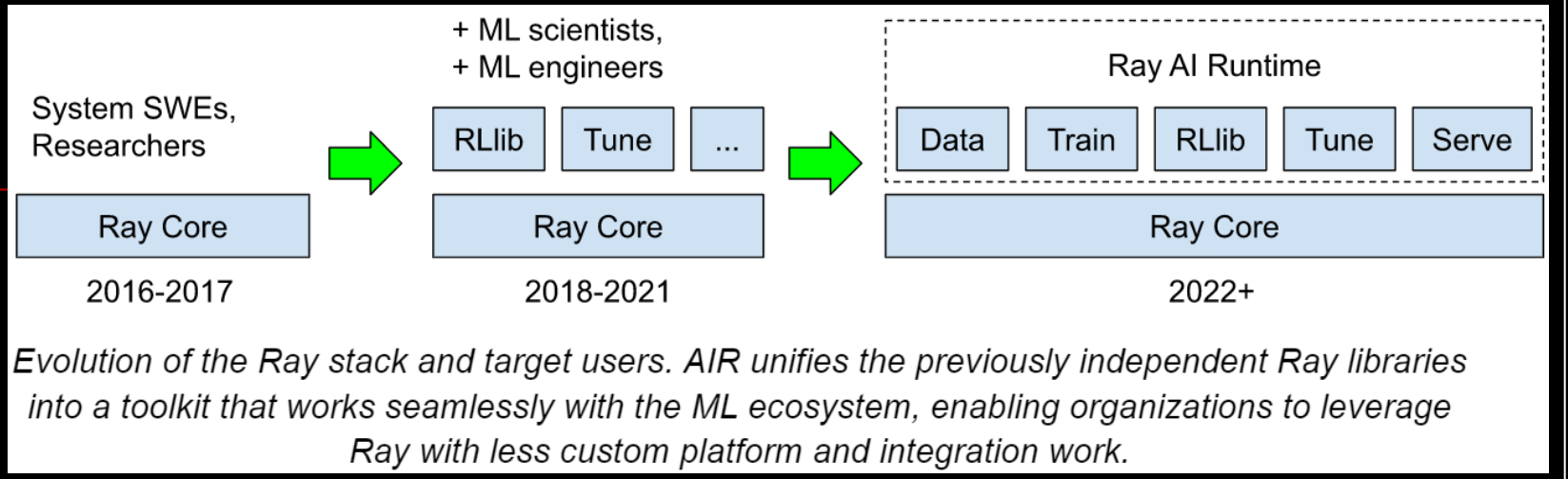
- **https://docs.ray.io/en/latest/index.html**
  Ray 1.x

Ray is an open-source project developed at UC Berkeley RISE Lab. As a general-purpose and universal distributed compute framework, you can flexibly run any compute-intensive Python workload — from distributed training or hyperparameter tuning to deep reinforcement learning and production model serving.



- ...

History



Evolution of the Ray stack and target users. AIR unifies the previously independent Ray libraries into a toolkit that works seamlessly with the ML ecosystem, enabling organizations to leverage Ray with less custom platform and integration work.

Source: Ray AIR Technical Whitepaper

- **https://en.wikipedia.org/wiki/AMPLab**
- **https://news.berkeley.edu/story_jump/berkeley-launches-riselab-enabling-computers-to-make-intelligent-real-time-decisions/**
- **https://rise.cs.berkeley.edu/**

Architecture & design

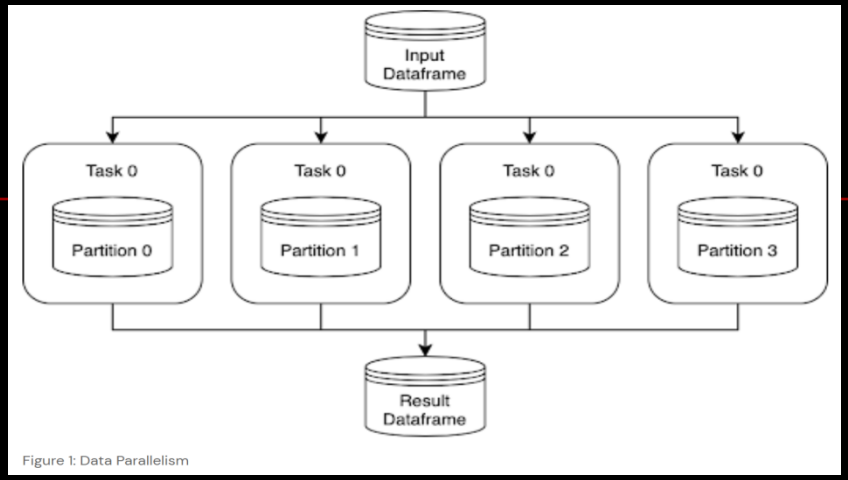


Figure 1: Data Parallelism

VS

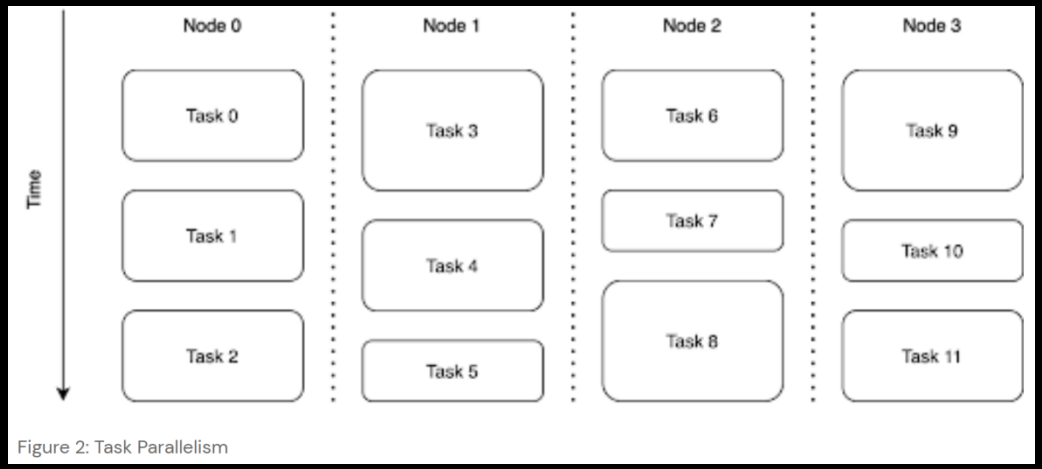
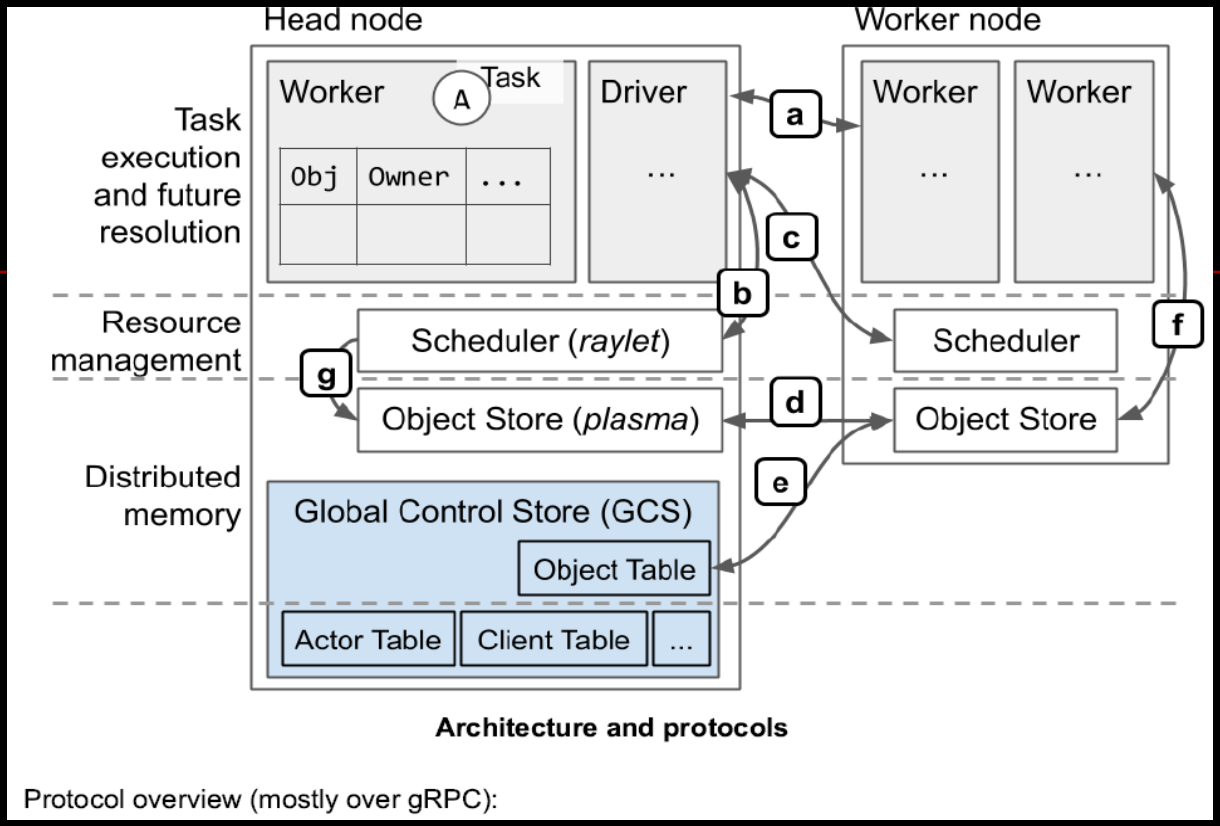


Figure 2: Task Parallelism

Source: https://www.databricks.com/blog/2021/11/19/ray-on-databricks.html



Architecture and protocols

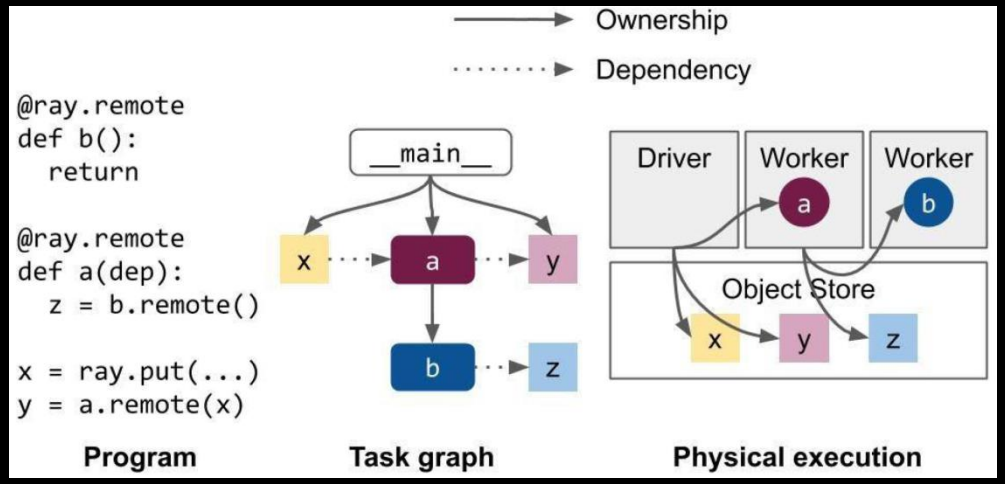
Protocol overview (mostly over gRPC):

a. Task execution, object reference counting.
b. Local resource management.
c. Remote/distributed resource management.
d. Distributed object transfer.
e. Location lookup for objects stored in distributed object store.
f. Storage and retrieval of large objects. Retrieval is via `ray.get` or during task execution, when replacing a task's ObjectID argument with the object's value.
g. Scheduler fetches objects from remote nodes to fulfill dependencies of locally queued tasks.

Source: Ray Architecture whitepaper 1.0

- *Task* - A single function invocation that executes on a process different from the caller. A task can be stateless (a `@ray.remote` function) or stateful (a method of a `@ray.remote` class - see *Actor* below). A task is executed asynchronously with the caller: the `.remote()` call immediately returns an `ObjectRef` that can be used to retrieve the return value.
- *Object* - An application value. This may be returned by a task or created through `ray.put`. Objects are **immutable**: they cannot be modified once created. A worker can refer to an object using an `ObjectRef`.
- *Actor* - a stateful worker process (an instance of a `@ray.remote` class). Actor tasks must be submitted with a *handle*, or a Python reference to a specific instance of an actor.
- *Driver* - The program root. This is the code that runs `ray.init()`.
- *Job* - The collection of tasks, objects, and actors originating (recursively) from the same driver.

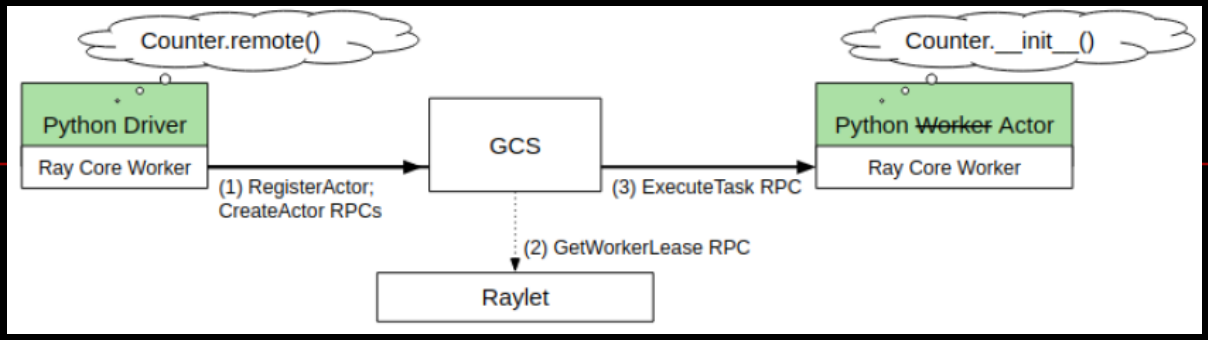
Some system metadata is managed by the GCS, a server backed by a pluggable data store. This metadata is also cached locally by the workers, e.g., the address of an actor. The GCS manages metadata that is less frequently accessed but likely to be used by most or all workers in the cluster, e.g., the current node membership of the cluster. This is to ensure that GCS performance is not critical to application performance.



Source: Ray Architecture whitepaper 1.0

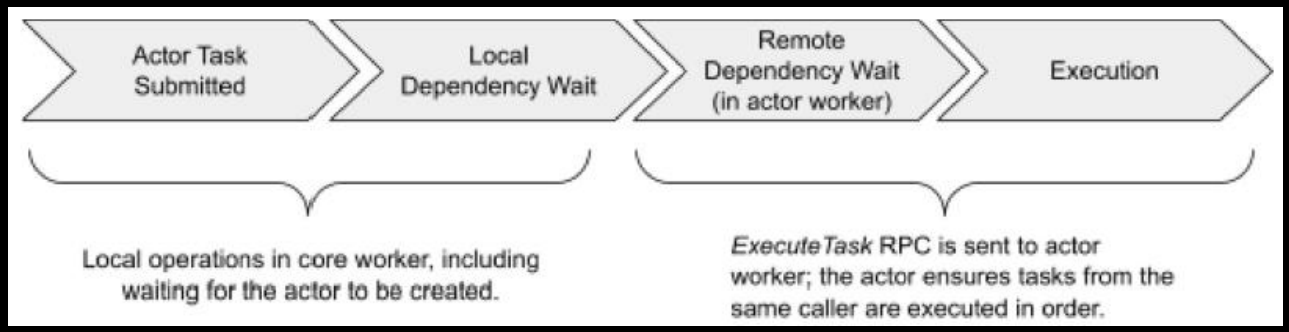
- **Actor management**

Actor creation



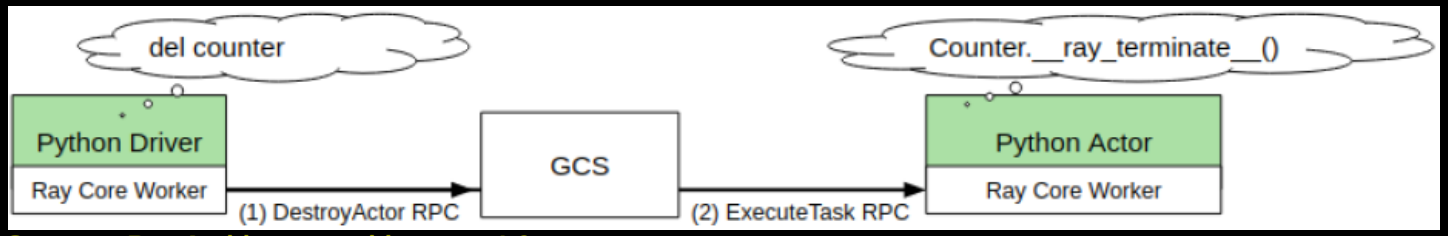
Source: Ray Architecture whitepaper 1.0

Actor task execution



Source: Ray Architecture whitepaper 1.0

Actor death



Source: Ray Architecture whitepaper 1.0

- **Ray Design Patterns 1.0**

Demo

- https://www.databricks.com/notebooks/raydemo.html

```
# First, decorate your function with @ray.remote to declare that you want to run this function remotely.
# Lastly, call that function with .remote() instead of calling it normally.
# This remote call yields a future, or ObjectRef that you can then fetch with ray.get.

@ray.remote
def f(x):
    return x * x

futures = [f.remote(i) for i in range(4)]
print(ray.get(futures)) # [0, 1, 4, 9]

[0, 1, 4, 9]
```

```
# Ray provides actors to allow you to parallelize an instance of a class in Python.
# When you instantiate a class that is a Ray actor, Ray will start a remote instance of that class in the cluster.
# This actor can then execute remote method calls and maintain its own internal state.

@ray.remote
class Counter(object):
    def __init__(self):
        self.n = 0

    def increment(self):
        self.n += 1

    def read(self):
        return self.n

counters = [Counter.remote() for i in range(4)]
[c.increment.remote() for c in counters]
futures = [c.read.remote() for c in counters]
print(ray.get(futures)) # [1, 1, 1, 1]

[1, 1, 1, 1]
```

...

Ecosystem

Source: https://www.anyscale.com/blog/four-reasons-why-leading-companies-are-betting-on-ray

- https://www.anyscale.com/blog/why-you-should-build-your-ai-applications-with-ray
- https://www.anyscale.com/blog/the-ideal-foundation-for-a-general-purpose-serverless-platform
- …

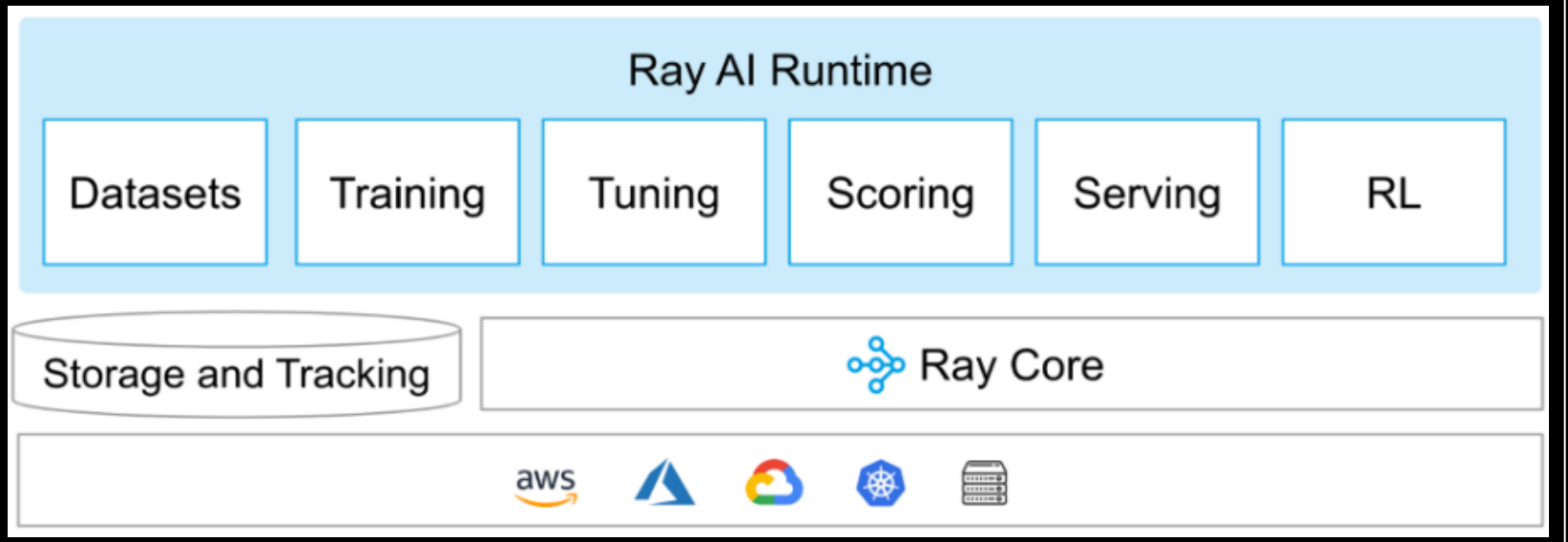
1.2 What's new in 2.0

- https://www.anyscale.com/blog/announcing-ray-2-0

Unifying the ML Ecosystem

- **Ray AI Runtime — a scalable and unified toolkit for ML applications**

Since its inception, Ray has been used to accelerate and scale compute-heavy machine learning workloads. The Ray AI Runtime (AIR), released as beta, is an effort to incorporate and synthesize lessons learned: by talking to community users and taking into account their use cases both large and small. This deliberate effort has produced a simple, unified toolkit for the Ray community.



With Ray AIR, we're aligning Ray's existing native ML libraries to work seamlessly together and integrate easily with popular ML frameworks in the community. Our goal with Ray AIR is to make it easy to run any kind of ML workload in just a few lines of Python code, letting Ray do the heavy lifting of coordinating computations at scale.

ray.data

```
dataset = ray.data.read_datasource(...)
train_ds, valid_ds = train_test_split(
    dataset, test_size=0.3)
test_ds = valid_ds.drop_columns(["target"])
preprocessor = StandardScaler(columns=["mean radius"])
```

ray.train

```
trainer = ray.train.xgboost.XGBoostTrainer(
   scaling_config=ScalingConfig(num_workers=128),
   label_column="target",
   datasets=dict(train=train_ds, valid=valid_ds},
   preprocessor=preprocessor)
result = trainer.fit()
```

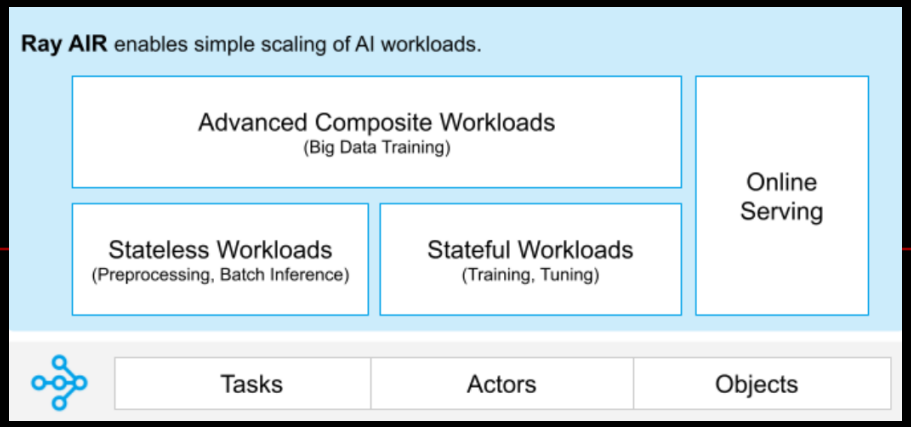
ray.tune

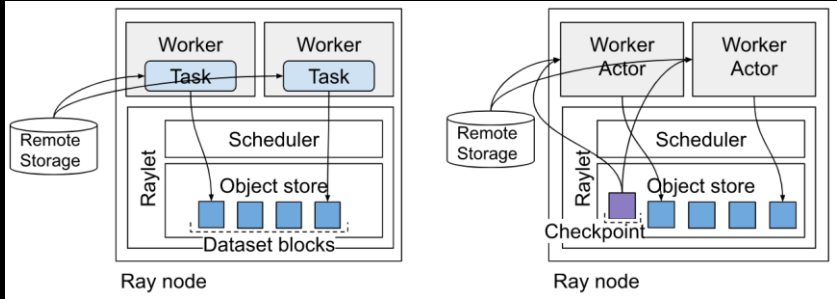
```
tuner = ray.tune.Tuner(
   trainer,
   param_space={"params": {"depth": tune.randint(1, 9)}},
   tune_config=TuneConfig(
      num_samples=5, metric="logloss", mode="min"))
checkpoint = tuner.fit().get_best_result().checkpoint
```

```
batch_predictor = BatchPredictor.from_checkpoint(
    checkpoint, XGBoostPredictor)
predicted_probabilities = batch_predictor.predict(test_ds)
predicted_probabilities.show()
```

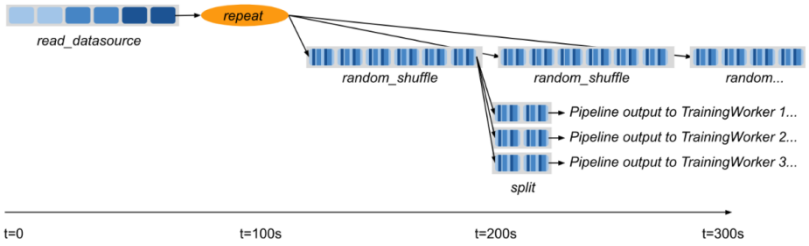
ray.serve

```
deployment = PredictorDeployment.options(name="XGB_Serve")
deployment.deploy(XGBoostPredictor, checkpoint, ...)
print(deployment.url)
```





Stateless transformations can involve only tasks (left), as well as worker actors containing soft model state (right), or a combination of both (e.g., CPU read tasks to load data, followed by GPU inference actors to do the inference (not shown)).

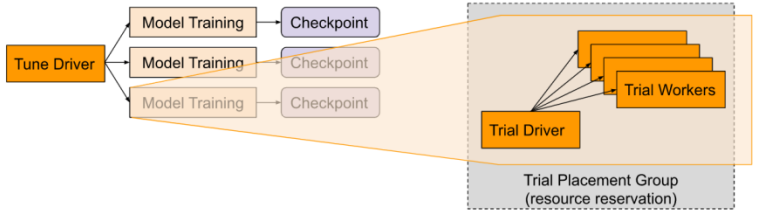


Visual depiction of the example we overview here. Per-epoch data shuffles execute concurrently with the distributed training. Diagram from Pipelining Compute Guide.

Stateful Compute

Distributed training and hyperparameter tuning are stateful workloads. These workloads create actors that hold the current state of the training / tuning job, and placement groups are used for gang-scheduling of these actors.

If a distributed training job is run under Tune, a tree of actors may be created, consisting of one "driver" actor for each tuning trial, and sub-actors that implement the parallel training:



A tree of actors implements a nested Tuning / Training job. Each model training trial consists of a driver actor and multiple worker actors held in a placement group. The Tune driver coordinates the overall computation, creating and destroying these trials to run the experiment.

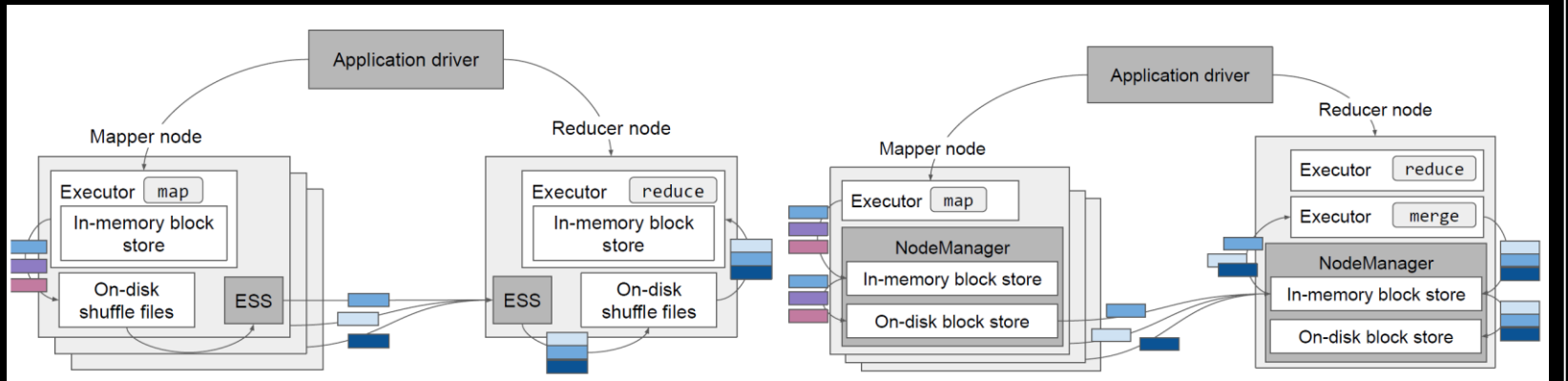
Online Serving

Finally, AIR makes it easy to go from training and scoring to online serving using the same infrastructure. This comes for free since AIR checkpoints work out of the box with Ray Serve. Under the hood, Serve uses Ray actors to serve online RPC requests. Data flow between the actors is handled via the Ray object store and actor calls.

Source: Ray AIR Technical Whitepaper

Large-Scale Shuffle for ML Workloads

Many of our ML users tell us that if Ray supported shuffling big data, they could simplify their infrastructure to standardize more on Ray. In Ray 2.0, this is now a reality: Ray now supports natively shuffling 100TB or more of data with the Ray Datasets library. Check out the docs for more information on how to get started, and refer to our technical paper on how distributed shuffling was implemented in Ray.



(a) Example of a monolithic shuffle architecture. (b) Exoshuffle.

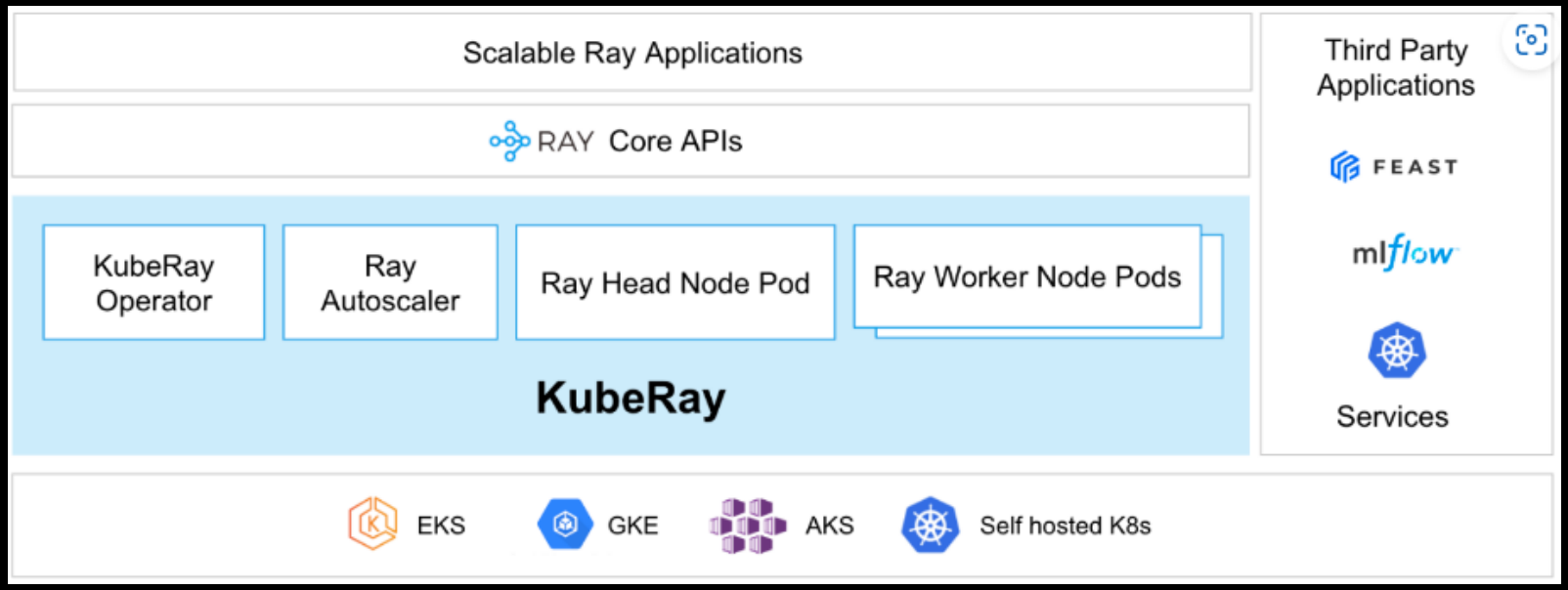
Figure 4: Comparing a monolithic vs. application-level shuffle architecture. (a) implements all coordination and block management through an external shuffle service (ESS) on each node, in this case implementing the Magnet shuffle strategy [24]. (b) shows the same shuffle strategy but implemented as an application on a generic distributed futures system.

Source: https://arxiv.org/pdf/2203.05072.pdf

Production and Scale

- **Production support for Kubernetes via KubeRay**

Developed in collaboration with ByteDance and Microsoft, KubeRay makes it easier than ever to deploy Ray-based jobs and services on Kubernetes and replaces the legacy Python-based Ray operator. KubeRay, currently in beta, is the officially recommended way of running Ray on K8s. Learn more about KubeRay here.



High-Availability for Large-Scale Ray Serve Deployments

With KubeRay, you can easily deploy Ray Serve clusters in fault-tolerant and HA modes, eliminating the previous operational work necessary to set this up. You can deploy services in fail-over mode, or Ray clusters in fully HA mode (alpha), which removes any single point of failure from the Ray cluster. Learn more about deploying Serve in KubeRay here.

New Ray Observability Tooling

https://docs.ray.io/en/latest/ray-observability/index.html

Observability, Debugging, and Profiling

- Ray Dashboard
  - Getting Started
  - Views
  - Advanced Usage
  - References
- Monitoring Ray States
  - Getting Started
  - Key Concepts
  - Summary
  - List
  - Get
  - Logs
  - Failure Semantics
  - API Reference
- Ray Debugger
  - Getting Started
  - Running on a Cluster
  - Debugger Commands
  - Stepping between Ray tasks
  - Post Mortem Debugging
  - Debugging APIs
- Logging
  - Driver logs
  - Worker logs
  - Disabling logging to the driver
  - Customizing Actor logs prefixes
  - How to set up loggers
  - How to use structured logging
  - Redirecting Ray logs to stderr
- Exporting Metrics
  - Getting Started (Single Node)
  - Getting Started (Multi-nodes)
  - Getting Started (Cluster Launcher)
  - Prometheus Service Discovery Support
  - Application-level Metrics
- Tracing
  - Getting Started
  - Tracing Startup Hook
- Debugging (internal)
  - Starting processes in a debugger
  - Backend logging
  - Backend event stats
  - Callback latency injection
- Profiling (internal)
  - Getting a stack trace of Ray C++ processes
  - Installation
  - Launching the to-profile binary
  - Memory Profiling
  - Visualizing the CPU profile
  - Running Microbenchmarks
  - References

Simpler, More Powerful Library APIs

Ray Serve Deployment Graph — multi-step, multi-model inference simplified

Combining or chaining multiple models to make a single prediction has emerged as a common pattern in modern ML applications. Ray Serve's flexible execution model has always made it a popular choice for these use cases.

Building on these strengths, we are excited to introduce Ray Serve's **Deployment Graph API** — a new and easier way to build, test, and deploy an inference graph of deployments. Deployment Graphs are built on Ray's intuitive task and actor APIs to make inference pipelines that mix model and business logic accessible to all ML practitioners.

RLlib - Updated Developer APIs and Offline Reinforcement Learning Improvements

In Ray 2.0, the RLlib team has refactored the implementation of all key RLlib algorithms to follow simpler, imperative patterns, making it easier to implement new algorithms or make adjustments to existing ones. Learn more about these new patterns here.

RLlib is also introducing a new Connectors API to make the preprocessing "glue" for environments and policy models more explicit and transparent to users, simplifying the deployment of RLlib models.

Finally, Ray 2.0 features newly added algorithms and functionality for helping run reinforcement learning workloads in production. RLlib now supports Critic Regularized Regression (CRR), a state-of-the-art algorithm for offline RL and Doubly Robust Off-Policy Estimator, an algorithm for policy evaluation. Learn more about RLlib's Offline RL support here.

1.2.1 Ray Core

- https://docs.ray.io/en/latest/ray-core/walkthrough.html#

Ray Core provides a small number of core primitives (i.e., tasks, actors, objects) for building and scaling distributed applications. Below we'll walk through simple examples that show you how to turn your functions and classes easily into Ray tasks and actors, and how to work with Ray objects.

1.2.1.1 Key concepts

- https://docs.ray.io/en/latest/ray-core/key-concepts.html

Tasks

Ray enables arbitrary functions to be executed asynchronously on separate Python workers. These asynchronous Ray functions are called "tasks". Ray enables tasks to specify their resource requirements in terms of CPUs, GPUs, and custom resources. These resource requests are used by the cluster scheduler to distribute tasks across the cluster for parallelized execution.

See the User Guide for Tasks.

Actors

Actors extend the Ray API from functions (tasks) to classes. An actor is essentially a stateful worker (or a service). When a new actor is instantiated, a new worker is created, and methods of the actor are scheduled on that specific worker and can access and mutate the state of that worker. Like tasks, actors support CPU, GPU, and custom resource requirements.

See the User Guide for Actors.

Objects

In Ray, tasks and actors create and compute on objects. We refer to these objects as *remote objects* because they can be stored anywhere in a Ray cluster, and we use *object refs* to refer to them. Remote objects are cached in Ray's distributed shared-memory *object store*, and there is one object store per node in the cluster. In the cluster setting, a remote object can live on one or many nodes, independent of who holds the object ref(s).

See the User Guide for Objects.

...

III. Ray on ARM

1) Package install

- https://docs.ray.io/en/latest/ray-overview/installation.html

Ray currently supports Linux, MacOS and Windows. Ray on Windows is currently in beta.

Official Releases

You can install the latest official version of Ray as follows.

```
# Install Ray with support for the dashboard + cluster launcher
pip install -U "ray[default]"

# Install Ray with minimal dependencies
# pip install -U ray
```

To install Ray libraries:

```
pip install -U "ray[air]" # installs Ray + dependencies for Ray AI Runtime
pip install -U "ray[tune]"  # installs Ray + dependencies for Ray Tune
pip install -U "ray[rllib]"  # installs Ray + dependencies for Ray RLlib
pip install -U "ray[serve]"  # installs Ray + dependencies for Ray Serve
```

Daily Releases (Nightlies)

You can install the nightly Ray wheels via the following links. These daily releases are tested via automated tests but do not go through the full release process. To install these wheels, use the following `pip` command and wheels:

```
# Clean removal of previous install
pip uninstall -y ray
# Install Ray with support for the dashboard + cluster launcher
pip install -U "ray[default] @ LINK_TO_WHEEL.whl"

# Install Ray with minimal dependencies
# pip install -U LINK_TO_WHEEL.whl
```

Linux	MacOS	Windows (beta)
Linux Python 3.10	MacOS Python 3.10	Windows Python 3.10
Linux Python 3.9	MacOS Python 3.9	Windows Python 3.9
Linux Python 3.8	MacOS Python 3.8	Windows Python 3.8
Linux Python 3.7	MacOS Python 3.7	Windows Python 3.7
Linux Python 3.6	MacOS Python 3.6	

Note

Python 3.10 support is currently experimental.

Note

On Windows, support for multi-node Ray clusters is currently experimental and untested. If you run into issues please file a report at https://github.com/ray-project/ray/issues.

Note

Usage stats collection is enabled by default (can be disabled) for nightly wheels including both local clusters started via `ray.init()` and remote clusters via cli.

- **Ray is mainly focus on X86**
-

M1 Mac (Apple Silicon) Support

Ray has experimental support for machines running Apple Silicon (such as M1 macs). To get started:

1. Install miniforge.
   - `wget https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-MacOSX-arm64.sh`
   - `bash Miniforge3-MacOSX-arm64.sh`
   - `rm https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-MacOSX-arm64.sh # Cleanup.`
2. Ensure you're using the miniforge environment (you should see (base) in your terminal).
   - `source ~/.bash_profile`
   - `conda activate`
3. Ensure that the `grpcio` package is installed via forge and **not pypi**. Grpcio currently requires special compilation flags, which pypi will _not_ correctly build with. Miniforge provides a prebuilt version of grpcio for M1 macs.
   - `pip uninstall grpcio; conda install grpcio=1.43.0`
4. Install Ray as you normally would.
   - `pip install ray`

Note

At this time, Apple Silicon ray wheels are being published for **releases only**. As support stabilizes, nightly wheels will be published in the future.

Try to install Ray with Miniforge on RPi4

https://github.com/conda-forge/miniforge/releases/

Miniforge3-Linux-aarch64.sh 47.7 MB

```
[mydev@fedora MiniForge]$ ll
total 48804
drwxr-xr-x. 1 mydev mydev      140 Oct 19 08:25 ./
drwxr-xr-x. 1 mydev mydev       18 Oct 19 08:22 ../
-rw-r--r--. 1 mydev mydev      105 Oct 19 08:23 Miniforge3-4.14.0-1-Linux-aarch64.sh.sha256
-rwxr--r--. 1 mydev mydev 49968120 Oct 19 08:25 Miniforge3-Linux-aarch64.sh*
[mydev@fedora MiniForge]$
[mydev@fedora MiniForge]$ bash ./Miniforge3-Linux-aarch64.sh

Welcome to Miniforge3 4.14.0-1

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
Miniforge installer code uses BSD-3-Clause license as stated below.

Binary packages that come with it have their own licensing terms
```

...

```
Do you accept the license terms? [yes|no]
[no] >>> yes

Miniforge3 will now be installed into this location:
/home/mydev/miniforge3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/mydev/miniforge3] >>>
PREFIX=/home/mydev/miniforge3
Unpacking payload ...
Extracting python-3.10.6-h023d47c_0_cpython.tar.bz2
Extracting ca-certificates-2022.9.24-h4fd8a4c_0.tar.bz2
```

...

```
Extracting conda-4.14.0-py310h4c7bcd0_0.tar.bz2

conda-forge/linux-aarch64                          Using cache
conda-forge/noarch                                 Using cache

Transaction

  Prefix: /home/mydev/miniforge3

  Updating specs:

   - python==3.10.6=h023d47c_0_cpython
   - ca-certificates==2022.9.24=h4fd8a4c_0
```

```
  - conda==4.14.0=py310h4c7bcd0_0

Package                    Version  Build               Channel                    Size
Install:
+ _openmp_mutex              4.5    2_gnu               conda-forge/linux-aarch64  Cached
+ brotlipy                 0.7.0    py310h761cc84_1004  conda-forge/linux-aarch64  Cached
...
  + yaml                   0.2.5    hf897c2e_2          conda-forge/linux-aarch64  Cached

  Summary:

  Install: 41 packages

  Total download: 0 B

Transaction starting
Linking ca-certificates-2022.9.24-h4fd8a4c_0
Linking ld_impl_linux-aarch64-2.36.1-h02ad14f_2
...
Linking conda-4.14.0-py310h4c7bcd0_0
Transaction finished
installation finished.
Do you wish the installer to initialize Miniforge3
by running conda init? [yes|no]
[no] >>> yes
no change     /home/mydev/miniforge3/condabin/conda
no change     /home/mydev/miniforge3/bin/conda
no change     /home/mydev/miniforge3/bin/conda-env
no change     /home/mydev/miniforge3/bin/activate
no change     /home/mydev/miniforge3/bin/deactivate
no change     /home/mydev/miniforge3/etc/profile.d/conda.sh
no change     /home/mydev/miniforge3/etc/fish/conf.d/conda.fish
no change     /home/mydev/miniforge3/shell/condabin/Conda.psm1
no change     /home/mydev/miniforge3/shell/condabin/conda-hook.ps1
no change     /home/mydev/miniforge3/lib/python3.10/site-packages/xontrib/conda.xsh
no change     /home/mydev/miniforge3/etc/profile.d/conda.csh
modified      /home/mydev/.bashrc

==> For changes to take effect, close and re-open your current shell. <==

If you'd prefer that conda's base environment not be activated on startup,
   set the auto_activate_base parameter to false:

conda config --set auto_activate_base false

Thank you for installing Miniforge3!
[mydev@fedora MiniForge]$
```

```
[mydev@fedora MiniForge]$ conda config --set auto_activate_base false
/home/mydev/.local/lib/python3.10/site-packages/pkg_resources/__init__.py:123: PkgResourcesDeprecationWarning: git-archive.dev8b63d73a17 is an invalid version and will not be supported in a future release
  warnings.warn(
[mydev@fedora MiniForge]$
[mydev@fedora MiniForge]$
[mydev@fedora MiniForge]$
[mydev@fedora MiniForge]$ vim ~/.bashrc
[mydev@fedora MiniForge]$
[mydev@fedora MiniForge]$
[mydev@fedora MiniForge]$ cat ~/.bash_profile
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs
. "$HOME/.cargo/env"
[mydev@fedora MiniForge]$ _
```

start another shell

```
[mydev@fedora ~]$ source ~/.bash_profile
[mydev@fedora ~]$
[mydev@fedora ~]$ which conda
conda ()
{
    \local cmd="${1-__missing__}";
    case "$cmd" in
        activate | deactivate)
            __conda_activate "$@"
        ;;
        install | update | upgrade | remove | uninstall)
            __conda_exe "$@" || \return;
            __conda_reactivate
        ;;
        *)
            __conda_exe "$@"
        ;;
    esac
}
[mydev@fedora ~]$ conda activate
(base) [mydev@fedora ~]$ conda install grpcio=1.43.0
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /home/mydev/miniforge3

  added / updated specs:
    - grpcio=1.43.0


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    c-ares-1.18.1              |       hf897c2e_0         118 KB  conda-forge
    conda-22.9.0               |  py310h4c7bcd0_1         979 KB  conda-forge
    grpcio-1.43.0              |  py310h126c23a_0        48.0 MB  conda-forge
    libstdcxx-ng-12.2.0        |     hc13a102_18         4.2 MB  conda-forge
    six-1.16.0                 |     pyh6c4a22f_0          14 KB  conda-forge
    zlib-1.2.12                |       h4e544f5_4          99 KB  conda-forge
    ------------------------------------------------------------
                                           Total:        53.4 MB
```

```
The following NEW packages will be INSTALLED:

  c-ares             conda-forge/linux-aarch64::c-ares-1.18.1-hf897c2e_0
  grpcio             conda-forge/linux-aarch64::grpcio-1.43.0-py310h126c23a_0
  libstdcxx-ng       conda-forge/linux-aarch64::libstdcxx-ng-12.2.0-hc13a102_18
  six                conda-forge/noarch::six-1.16.0-pyh6c4a22f_0
  zlib               conda-forge/linux-aarch64::zlib-1.2.12-h4e544f5_4

The following packages will be UPDATED:

  conda                               4.14.0-py310h4c7bcd0_0 --> 22.9.0-py310h4c7bcd0_1


Proceed ([y]/n)? y


Downloading and Extracting Packages
libstdcxx-ng-12.2.0  | 4.2 MB    | ########################################################################################################################################################################## | 100%
conda-22.9.0         | 979 KB    | ########################################################################################################################################################################## | 100%
zlib-1.2.12          | 99 KB     | ########################################################################################################################################################################## | 100%
grpcio-1.43.0        | 48.0 MB   | ########################################################################################################################################################################## | 100%
c-ares-1.18.1        | 118 KB    | ########################################################################################################################################################################## | 100%
six-1.16.0           | 14 KB     | ########################################################################################################################################################################## | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Retrieving notices: ...working... done
(base)
```

```
(base) conda install ray
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.
Collecting package metadata (repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible solve.

PackagesNotFoundError: The following packages are not available from current channels:

  - ray

Current channels:

  - https://conda.anaconda.org/conda-forge/linux-aarch64
  - https://conda.anaconda.org/conda-forge/noarch

To search for alternate channels that may provide the conda package you're
looking for, navigate to

    https://anaconda.org

and use the search bar at the top of the page.


(base)
(base) pip install ray
ERROR: Could not find a version that satisfies the requirement ray (from versions: none)
ERROR: No matching distribution found for ray
WARNING: There was an error checking the latest version of pip.
(base) pip install ray==2.0.0
ERROR: Could not find a version that satisfies the requirement ray==2.0.0 (from versions: none)
ERROR: No matching distribution found for ray==2.0.0
WARNING: There was an error checking the latest version of pip.
```

```
[mydev@fedora /]$ which pip
/usr/bin/pip
[mydev@fedora /]$ pip install ray
Defaulting to user installation because normal site-packages is not writeable
ERROR: Could not find a version that satisfies the requirement ray (from versions: none)
ERROR: No matching distribution found for ray
WARNING: There was an error checking the latest version of pip.
[mydev@fedora /]$
[mydev@fedora /]$ pip install ray==1.13.0
Defaulting to user installation because normal site-packages is not writeable
ERROR: Could not find a version that satisfies the requirement ray==1.13.0 (from versions: none)
ERROR: No matching distribution found for ray==1.13.0
WARNING: There was an error checking the latest version of pip.
[mydev@fedora /]$
```

2) Build from source

- **Official guide**

  https://docs.ray.io/en/latest/ray-contribute/development.html#building-ray

build dashboard:

Make sure you have a local clone of Ray's git repository as explained above. You will also need to install NodeJS to build the dashboard.

Enter into the project directory, for example:

```
cd ray
```

Now you can build the dashboard. From inside of your local Ray project directory enter into the dashboard client directory:

```
cd dashboard/client
```

Then you can install the dependencies and build the dashboard:

```
npm install
npm run build
```

build ray:

- Clone the repository
- Prepare the Python environment
- Building Ray (Python Only)
- Preparing to build Ray on Linux
- Preparing to build Ray on MacOS
- Building Ray on Linux & MacOS (full)
- Building Ray on Windows (full)
- Environment variables that influence builds
- Installing additional dependencies for development
- Fast, Debug, and Optimized Builds
- Building the Docs
- Using a local repository for dependencies
- Troubleshooting

Building Ray on Linux(full)

Now let's build Ray for Python. Make sure you activate any Python virtual (or conda) environment you could be using as described above.

Enter into the `python/` directory inside of the Ray project directory and install the project with `pip`:

```
# Install Ray.
cd python/
pip install -e . --verbose  # Add --user if you see a permission denied error.
```

The `-e` means "editable", so changes you make to files in the Ray directory will take effect without reinstalling the package.

Warning

if you run `python setup.py install`, files will be copied from the Ray directory to a directory of Python packages (`/lib/python3.6/site-packages/ray`). This means that changes you make to files in the Ray directory will not have any effect.

Tip

If your machine is running out of memory during the build or the build is causing other programs to crash, try adding the following line to `~/.bazelrc`:

```
build --local_ram_resources=HOST_RAM*.5 --local_cpu_resources=4
```

The `build --disk_cache=~/bazel-cache` option can be useful to speed up repeated builds too.

Building Ray (Python Only)

> **Note**
>
> Unless otherwise stated, directory and file paths are relative to the project root directory.

RLlib, Tune, Autoscaler, and most Python files do not require you to build and compile Ray. Follow these instructions to develop Ray's Python files locally without building Ray.

1. Make sure you have a clone of Ray's git repository as explained above.
2. Make sure you activate the Python (virtual) environment as described above.
3. Pip install the **latest Ray wheels.** See Daily Releases (Nightlies) for instructions.

```
# For example, for Python 3.8:
pip install -U https://s3-us-west-2.amazonaws.com/ray-wheels/latest/ray-3.0.0.dev0-cp38-cp38-manylinux2014_x86_64.whl
```

4. Replace Python files in the installed package with your local editable copy. We provide a simple script to help you do this: `python python/ray/setup-dev.py`. Running the script will remove the `ray/tune`, `ray/rllib`, `ray/autoscaler` dir (among other directories) bundled with the `ray` pip package, and replace them with links to your local code. This way, changing files in your git clone will directly affect the behavior of your installed Ray.

```
# This replaces `<package path>/site-packages/ray/<package>`
# with your local `ray/python/ray/<package>`.
python python/ray/setup-dev.py
```

> **Note**
>
> [Advanced] You can also optionally skip creating symbolic link for directories of your choice.

```
# This links all folders except "_private" and "dashboard" without user prompt.
python setup-dev.py -y --skip _private dashboard
```

> **Warning**
>
> Do not run `pip uninstall ray` or `pip install -U` (for Ray or Ray wheels) if setting up your environment this way. To uninstall or upgrade, you must first `rm -rf` the pip-installation site (usually a directory at the `site-packages/ray` location), then do a pip reinstall (see the command above), and finally run the above `setup-dev.py` script again.

```
# To uninstall, delete the symlinks first.
rm -rf <package path>/site-packages/ray # Path will be in the output of `setup-dev.py`.
pip uninstall ray # or `pip install -U <wheel>`
```

- **Build & Test**

1. **Manually install Bazel on RPi4**

   https://github.com/bazelbuild/bazel/releases

2. **Go to $SRC-RAY/python and execute "pip install -e . --verbose –user"**

   master branch, last commit a47adb9d7722acf1e480db139acffc71f1f30a4f (~Sep 23), ~3 hours on my RPi4, depends on networking speed):

```
Using pip 22.2.2 from /home/mydev/.local/lib/python3.10/site-packages/pip (python 3.10)
Obtaining file:///opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python
  Preparing metadata (setup.py): started
  Running command python setup.py egg_info
  running egg_info
  creating /tmp/pip-pip-egg-info-73klgzhc/ray.egg-info
  writing /tmp/pip-pip-egg-info-73klgzhc/ray.egg-info/PKG-INFO
  writing dependency_links to /tmp/pip-pip-egg-info-73klgzhc/ray.egg-info/dependency_links.txt
  writing entry points to /tmp/pip-pip-egg-info-73klgzhc/ray.egg-info/entry_points.txt
  writing requirements to /tmp/pip-pip-egg-info-73klgzhc/ray.egg-info/requires.txt
  writing top-level names to /tmp/pip-pip-egg-info-73klgzhc/ray.egg-info/top_level.txt
  writing manifest file '/tmp/pip-pip-egg-info-73klgzhc/ray.egg-info/SOURCES.txt'
  reading manifest file '/tmp/pip-pip-egg-info-73klgzhc/ray.egg-info/SOURCES.txt'
  reading manifest template 'MANIFEST.in'
  writing manifest file '/tmp/pip-pip-egg-info-73klgzhc/ray.egg-info/SOURCES.txt'
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: attrs in /usr/lib/python3.10/site-packages (from ray==3.0.0.dev0) (21.4.0)
Requirement already satisfied: click≤8.0.4,≥7.0 in /usr/lib/python3.10/site-packages (from ray==3.0.0.dev0) (8.0.4)
Requirement already satisfied: filelock in /home/mydev/.local/lib/python3.10/site-packages (from ray==3.0.0.dev0) (3.7.0)
Requirement already satisfied: jsonschema in /home/mydev/.local/lib/python3.10/site-packages (from ray==3.0.0.dev0) (4.5.1)
Requirement already satisfied: msgpack<2.0.0,≥1.0.0 in /usr/lib64/python3.10/site-packages (from ray==3.0.0.dev0) (1.0.3)
Requirement already satisfied: protobuf<4.0.0,≥3.15.3 in /usr/lib64/python3.10/site-packages (from ray==3.0.0.dev0) (3.19.
4)
Requirement already satisfied: pyyaml in /usr/lib64/python3.10/site-packages (from ray==3.0.0.dev0) (6.0)
Requirement already satisfied: aiosignal in /usr/lib/python3.10/site-packages (from ray==3.0.0.dev0) (1.2.0)
Requirement already satisfied: frozenlist in /usr/lib64/python3.10/site-packages (from ray==3.0.0.dev0) (1.3.1)
Requirement already satisfied: requests in /usr/lib/python3.10/site-packages (from ray==3.0.0.dev0) (2.27.1)
Requirement already satisfied: virtualenv≥20.0.24 in /home/mydev/.local/lib/python3.10/site-packages (from ray==3.0.0.dev0
) (20.14.1)
...
```

```
Requirement already satisfied: urllib3<1.27,≥1.21.1 in /home/mydev/.local/lib/python3.10/site-packages (from requests→ray==3.0.0.dev0) (1.26.9)
Installing collected packages: grpcio, ray
  Running setup.py develop for ray
    Running command python setup.py develop
    running develop
    /home/mydev/.local/lib/python3.10/site-packages/setuptools/command/easy_install.py:144: EasyInstallDeprecationWarning: easy_install command is deprecated. Use build and pip and other standards-based tools
.
      warnings.warn(
    /home/mydev/.local/lib/python3.10/site-packages/setuptools/command/install.py:34: SetuptoolsDeprecationWarning: setup.py install is deprecated. Use build and pip and other standards-based tools.
      warnings.warn(
    /home/mydev/.local/lib/python3.10/site-packages/pkg_resources/__init__.py:123: PkgResourcesDeprecationWarning: git-archive.dev8b63d73a17 is an invalid version and will not be supported in a future release
      warnings.warn(
    running egg_info
    creating ray.egg-info
    writing ray.egg-info/PKG-INFO
    writing dependency_links to ray.egg-info/dependency_links.txt
    writing entry points to ray.egg-info/entry_points.txt
    writing requirements to ray.egg-info/requires.txt
    writing top-level names to ray.egg-info/top_level.txt
    writing manifest file 'ray.egg-info/SOURCES.txt'
    reading manifest file 'ray.egg-info/SOURCES.txt'
    reading manifest template 'MANIFEST.in'
    writing manifest file 'ray.egg-info/SOURCES.txt'
    running build_ext
    ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
    mu-editor 1.0.3 requires pyserial<3.5,≥3.0, but you have pyserial 3.5 which is incompatible.
    Starting local Bazel server and connecting to it...
    WARNING: Option 'strict_java_deps' is deprecated: Use --experimental_strict_java_deps instead
    Loading:
    Loading: 0 packages loaded
    Loading: 0 packages loaded
    DEBUG: /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/bazel/ray_deps_setup.bzl:67:14: No implicit mirrors used because urls were explicitly provided
    Loading: 0 packages loaded
```

```
    INFO: Analyzed 2 targets (160 packages loaded, 15649 targets configured).
    INFO: Found 2 targets...
    [0 / 36] [Prepa] Symlinking virtual headers for //cpp:ray_cpp_lib
    [842 / 2,695] Compiling src/google/protobuf/util/internal/datapiece.cc; 8s linux-sandbox ... (4 actions, 3 running)
    [912 / 2,695] Compiling src/google/protobuf/extension_set.cc; 12s linux-sandbox ... (4 actions, 3 running)
    INFO: From Compiling src/google/protobuf/message_lite.cc:
    In file included from /usr/include/string.h:535,
                     from external/com_google_protobuf/src/google/protobuf/stubs/port.h:39,
                     from external/com_google_protobuf/src/google/protobuf/stubs/common.h:48,
                     from external/com_google_protobuf/src/google/protobuf/message_lite.h:45,
                     from external/com_google_protobuf/src/google/protobuf/message_lite.cc:36:
    In function 'void* memcpy(void*, const void*, size_t)',
        inlined from 'uint8_t* google::protobuf::io::EpsCopyOutputStream::WriteRaw(const void*, int, uint8_t*)' at external/com_google_protobuf/src/google/protobuf/io/coded_stream.h:706:16,
        inlined from 'virtual uint8_t* google::protobuf::internal::ImplicitWeakMessage::_InternalSerialize(uint8_t*, google::protobuf::io::EpsCopyOutputStream*) const' at external/com_google_protobuf/src/goog
le/protobuf/implicit_weak_message.h:84:28,
        inlined from 'bool google::protobuf::MessageLite::SerializePartialToZeroCopyStream(google::protobuf::io::ZeroCopyOutputStream*) const' at external/com_google_protobuf/src/google/protobuf/message_lite.
cc:412:30:
    /usr/include/bits/string_fortified.h:29:33: warning: 'void* __builtin___memcpy_chk(void*, const void*, long unsigned int, long unsigned int)' specified size between 18446744071562067968 and 18446744073709
551615 exceeds maximum object size 9223372036854775807 [-Wstringop-overflow=]
       29 |   return __builtin___memcpy_chk (__dest, __src, __len,
          |          ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
       30 |                                  __glibc_objsize0 (__dest));
          |                                  ~~~~~~~~~~~~~~~~~~~~~~~~~~
    INFO: From Compiling src/google/protobuf/message_lite.cc [for host]:
```

```
    INFO: From Generating Descriptor Set proto_library @com_github_cncf_udpa//xds/type/v3:pkg:
    xds/type/v3/typed_struct.proto:10:1: warning: Import validate/validate.proto is unused.
    [1,773 / 3,439] Compiling src/idl_parser.cpp [for host]; 50s linux-sandbox ... (4 actions, 3 running)
    [1,918 / 3,439] Compiling cpp/src/ray/runtime/task/task_executor.cc; 40s linux-sandbox ... (4 actions, 3 running)
    [2,082 / 3,439] Compiling src/ray/gcs/gcs_client/accessor.cc; 11s linux-sandbox ... (4 actions, 3 running)
    [2,305 / 3,439] Compiling src/ray/core_worker/transport/direct_actor_transport.cc; 73s linux-sandbox ... (4 actions, 3
running)
    [2,461 / 3,439] Compiling src/ray/core_worker/core_worker.cc; 194s linux-sandbox ... (4 actions, 3 running)
    [2,696 / 3,439] Compiling src/ray/common/ray_object.cc; 11s linux-sandbox ... (4 actions, 3 running)
    [3,065 / 3,439] Compiling googlemock/src/gmock-spec-builders.cc; 9s linux-sandbox ... (4 actions, 3 running)
    [3,431 / 3,494] Compiling src/ray/core_worker/future_resolver.cc; 74s linux-sandbox ... (4 actions, 3 running)
    [3,701 / 3,735] Compiling src/ray/gcs/gcs_server/gcs_server.cc; 163s linux-sandbox ... (4 actions, 3 running)
    [3,822 / 3,838] Compiling python/ray/_raylet.cpp; 219s linux-sandbox ... (4 actions, 3 running)
    INFO: Elapsed time: 8020.008s, Critical Path: 453.93s
    INFO: 3985 processes: 1196 internal, 2779 linux-sandbox, 10 local.
    INFO: Build completed successfully, 3985 total actions
    INFO: Build completed successfully, 3985 total actions
    # of files copied to build/lib.linux-aarch64-cpython-310: 159
    Creating /home/mydev/.local/lib/python3.10/site-packages/ray.egg-link (link to .)
    Adding ray 3.0.0.dev0 to easy-install.pth file
    Installing ray script to /home/mydev/.local/bin
    Installing ray-operator script to /home/mydev/.local/bin
    Installing rllib script to /home/mydev/.local/bin
    Installing serve script to /home/mydev/.local/bin
    Installing tune script to /home/mydev/.local/bin

    Installed /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python
Successfully installed grpcio-1.43.0 ray
```

3. Testing

```
[mydev@fedora /]$ which python
/usr/bin/python
[mydev@fedora /]$ python -V
Python 3.10.7
[mydev@fedora /]$ which ray
~/.local/bin/ray
[mydev@fedora /]$ ll ~/.local/lib/python3.10/site-packages |grep -i ray
-rw-r--r--. 1 mydev mydev      61 Sep 23 08:29 ray.egg-link
[mydev@fedora /]$
[mydev@fedora /]$ cat ~/.local/lib/python3.10/site-packages/ray.egg-link
/opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python
.[mydev@fedora /]$
[mydev@fedora /]$ ray --help
Usage: ray [OPTIONS] COMMAND [ARGS]...

Options:
  --logging-level TEXT   The logging level threshold, choices=['debug',
                         'info', 'warning', 'error', 'critical'],
                         default='info'
  --logging-format TEXT  The logging format. default='%(asctime)s
                         %(levelname)s %(filename)s:%(lineno)s -- %(message)s'
  --version              Show the version and exit.
  --help                 Show this message and exit.

Commands:
  attach               Create or attach to a SSH session to a Ray cluster.
  cluster-dump         Get log data from one or more nodes.
  cpp                  Show the cpp library path and generate the bazel ...
  dashboard            Port-forward a Ray cluster's dashboard to the ...
  debug                Show all active breakpoints and exceptions in the ...
  disable-usage-stats  Disable usage stats collection.
  down                 Tear down a Ray cluster.
  enable-usage-stats   Enable usage stats collection.
  exec                 Execute a command via SSH on a Ray cluster.
  get                  Get a state of a given resource by ID.
  get-head-ip          Return the head node IP of a Ray cluster.
  get-worker-ips       Return the list of worker IPs of a Ray cluster.
  install-nightly      Install the latest wheels for Ray.
  list                 List all states of a given resource.
  logs                 Print the log file that matches the GLOB_FILTER.
  memory               Print object references held in a Ray cluster.
  microbenchmark       Run a local Ray microbenchmark on the current ...
  monitor              Tails the autoscaler logs of a Ray cluster.
  rsync-down           Download specific files from a Ray cluster.
  rsync-up             Upload specific files to a Ray cluster.
  stack                Take a stack dump of all Python workers on the ...
  start                Start Ray processes manually on the local machine.
  status               Print cluster status, including autoscaling info.
  stop                 Stop Ray processes manually on the local machine.
  submit               Uploads and runs a script on the specified cluster.
  summary              Return the summarized information of a given ...
  timeline             Take a Chrome tracing timeline for a Ray cluster.
  up                   Create or update a Ray cluster.
[mydev@fedora /]$
```

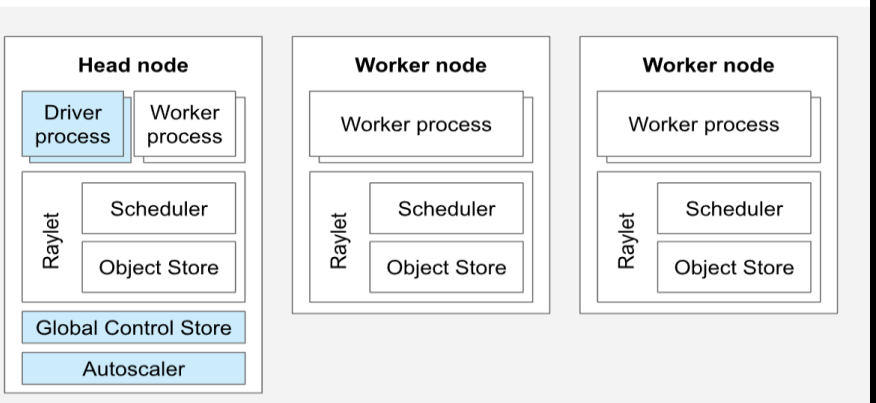
```
[mydev@fedora /]$ python -m pytest -v /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/tests/test_mini.py
============================= test session starts ==============================
platform linux -- Python 3.10.7, pytest-6.2.5, py-1.11.0, pluggy-1.0.0 -- /usr/bin/python
cachedir: .pytest_cache
rootdir: /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python
plugins: anyio-3.6.1
collected 3 items

opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/tests/test_mini.py::test_basic_task_api PASSED [ 33%]
opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/tests/test_mini.py::test_put_api PASSED [ 66%]
opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/tests/test_mini.py::test_actor_api ERROR [100%]

==================================== ERRORS ====================================
_____________________ ERROR at setup of test_actor_api _____________________

request = <SubRequest 'ray_start_regular' for <Function test_actor_api>>, maybe_external_redis = None

    @pytest.fixture
    def ray_start_regular(request, maybe_external_redis):
        param = getattr(request, "param", {})
>       with _ray_start(**param) as res:

opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/tests/conftest.py:246:
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
usr/lib64/python3.10/contextlib.py:135: in __enter__
    return next(self.gen)
opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/tests/conftest.py:216: in _ray_start
    address_info = ray.init("local", **init_kwargs)
opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/client_mode_hook.py:105: in wrapper
    return func(*args, **kwargs)
opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/worker.py:1419: in init
    _global_node = ray._private.node.Node(
opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/node.py:271: in __init__
    self.start_head_processes()
opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/node.py:1060: in start_head_processes
    self._write_cluster_info_to_kv()
opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/node.py:1044: in _write_cluster_info_to_kv
    ray_usage_lib.put_cluster_metadata(self.get_gcs_client())
opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/usage/usage_lib.py:541: in put_cluster_metadata
    gcs_client.internal_kv_put(
opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/gcs_utils.py:177: in wrapper
    return f(self, *args, **kwargs)
opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/gcs_utils.py:296: in internal_kv_put
    reply = self._kv_stub.InternalKVPut(req, timeout=timeout)
home/mydev/.local/lib/python3.10/site-packages/grpc/_channel.py:946: in __call__
    return _end_unary_response_blocking(state, call, False, None)
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
state = <grpc._channel._RPCState object at 0xffff90661030>, call = <grpc._cython.cygrpc.SegregatedCall object at 0xffff906f15c0>, with_call = False, deadline = None

    def _end_unary_response_blocking(state, call, with_call, deadline):
        if state.code is grpc.StatusCode.OK:
            if with_call:
                rendezvous = _MultiThreadedRendezvous(state, call, None, deadline)
                return state.response, rendezvous
            else:
                return state.response
        else:
>           raise _InactiveRpcError(state)
E           grpc._channel._InactiveRpcError: <_InactiveRpcError of RPC that terminated with:
E                status = StatusCode.UNAVAILABLE
E                details = "failed to connect to all addresses"
E                debug_error_string = "{"created":"@1666261938.142375504","description":"Failed to pick subchannel","file":"src/core/ext/filters/client_channel/client_channel.cc","file_line":3134,"referenced_errors":[{"created":"@1666261938.142323597","description":"failed to connect to all addresses","file":"src/core/lib/transport/error_utils.cc","file_line":163,"grpc_status":14}]}"
E           >

home/mydev/.local/lib/python3.10/site-packages/grpc/_channel.py:849: _InactiveRpcError
=========================== short test summary info ============================
ERROR opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/tests/test_mini.py::test_actor_api - grpc._channel._InactiveRpcError: <_InactiveRpcError of RPC that terminated with:
========================== 2 passed, 1 error in 34.58s ==========================
[mydev@fedora /]$
```

2) Clustering

2.1 Overview

- https://docs.ray.io/en/latest/cluster/vms/user-guides/launching-clusters/on-premises.html#manually-set-up-a-ray-cluster

A Ray cluster consists of a single head node and any number of connected worker nodes:



A Ray cluster with two worker nodes. Each node runs Ray helper processes to facilitate distributed scheduling and memory management. The head node runs additional control processes (highlighted in blue).

The number of worker nodes may be *autoscaled* with application demand as specified by your Ray cluster configuration. The head node runs the autoscaler.

Note

Ray nodes are implemented as pods when running on Kubernetes.

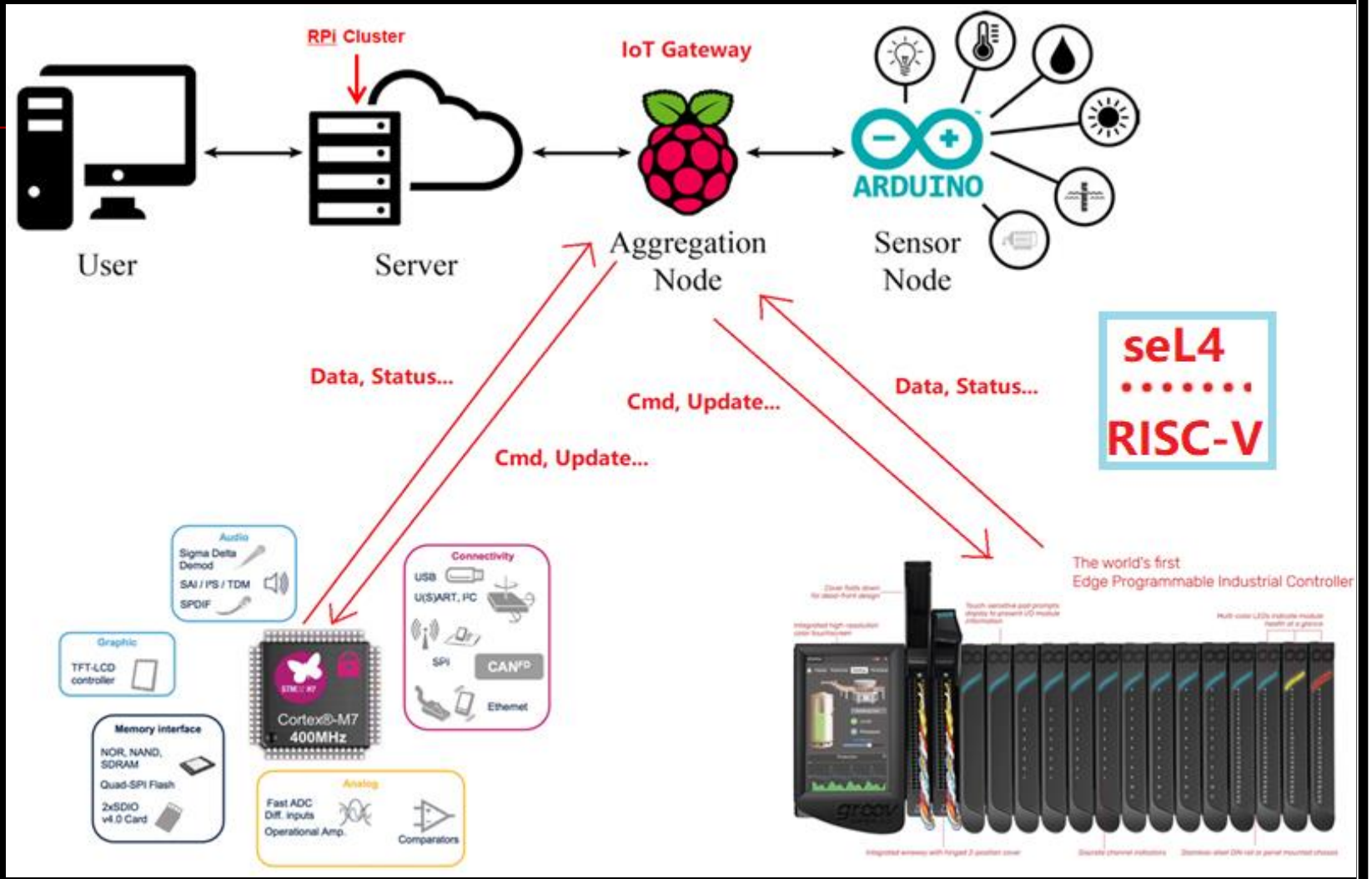
Users can submit jobs for execution on the Ray cluster, or can interactively use the cluster by connecting to the head node and running `ray.init`. See Ray Jobs for more information.

2.2 ARM cluster

- For more details, you may refer to another presentation "Hosting FOSS EDA tools for RISC-V cross-platform development on RPi4" at COSCon 2022(Online) and the upcoming follow-ups.

Usage scenarios

- General case



2.3 Set up a Ray cluster on RPi4 Cluster

- **Official guide**
  https://docs.ray.io/en/latest/cluster/vms/user-guides/launching-clusters/on-premises.html#manually-set-up-a-ray-cluster

Manually Set up a Ray Cluster

This section assumes that you have a list of machines and that the nodes in the cluster share the same network. It also assumes that Ray is installed on each machine. You can use pip to install the ray command line tool with cluster launcher support. Follow the Ray installation instructions for more details.

```
# install ray
pip install -U "ray[default]"
```

Start the Head Node

Choose any node to be the head node and run the following. If the `--port` argument is omitted, Ray will first choose port 6379, and then fall back to a random port if in 6379 is in use.

```
ray start --head --port=6379
```

The command will print out the Ray cluster address, which can be passed to `ray start` on other machines to start the worker nodes (see below). If you receive a ConnectionError, check your firewall settings and network configuration.

Start Worker Nodes

Then on each of the other nodes, run the following command to connect to the head node you just created.

```
ray start --address=<head-node-address:port>
```

Make sure to replace `head-node-address:port` with the value printed by the command on the head node (it should look something like 123.45.67.89:6379).

Note that if your compute nodes are on their own subnetwork with Network Address Translation, the address printed by the head node will not work if connecting from a machine outside that subnetwork. You will need to use a head node address reachable from the remote machine. If the head node has a domain address like compute04.berkeley.edu, you can simply use that in place of an IP address and rely on DNS.

Ray autodetects the resources (e.g., CPU) available on each node, but you can also manually override this by passing custom resources to the `ray start` command. For example, if you wish to specify that a machine has 10 CPUs and 1 GPU available for use by Ray, you can do this with the flags `--num-cpus=10` and `--num-gpus=1`. See the Configuration page for more information.

...

test on a single RPi4 board

```
[mydev@fedora /]$ which python
/usr/bin/python
[mydev@fedora /]$
[mydev@fedora /]$ ray start --head --port=6379
Enable usage stats collection? This prompt will auto-proceed in 10 seconds to avoid blocking cluster startup. Confirm [Y/n]:
Usage stats collection is enabled. To disable this, add `--disable-usage-stats` to the command that starts the cluster, or run the following command: `ray disable-usage-stats` before starting the cluster. See
 https://docs.ray.io/en/master/cluster/usage-stats.html for more details.

Local node IP: 192.168.0.106

--------------------
Ray runtime started.
--------------------

Next steps
  To connect to this Ray runtime from another node, run
    ray start --address='192.168.0.106:6379'

  Alternatively, use the following Python code:
    import ray
    ray.init(address='auto')

  To connect to this Ray runtime from outside of the cluster, for example to
  connect to a remote cluster from your laptop directly, use the following
  Python code:
    import ray
    ray.init(address='ray://<head_node_ip_address>:10001')

  If connection fails, check your firewall settings and network configuration.

  To terminate the Ray runtime, run
    ray stop
[mydev@fedora /]$
```

```
[mydev@fedora /]$ ps aux |grep -i ray
mydev      200272  4.1  0.4 216152 34868 pts/0    Sl   03:58   0:02 /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/core/src/ray/gcs/gcs_server --log_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --config_list=eyJvYmplY3Rfc3BpbGxpbmdfY29uZmlnIjogIntcInR5cGVcIjogXCJmaWxlc3lzdGVtXCIsIFwicGFyYW1zXCI6IHtcImRpcmVjdG9yeV9wYXRoXCI6IFwiL3RtcC9yYXkvc2Vzc2lvbl8yMDIyLTEwLTIwXzAzLTU4LTUzXzIyMDM1NF8yMDAxOTNcIn19IiwgImlzX2V4dGVybmFsX3N0b3JhZ2VfdHlwZV9mcyI6IHRydWV9 --gcs_server_port=6379 --metrics-agent-port=61905 --node-ip-address=192.168.0.106 --redis_password=5241590000000000
mydev      200296  4.2  1.0 452604 84252 pts/0    Sl   03:58   0:02 /usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/autoscaler/_private/monitor.py --logs-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 --redis-password=5241590000000000 --monitor-ip=192.168.0.106
mydev      200303  7.5  1.0 433716 83332 pts/0    Sl   03:58   0:04 /usr/bin/python3 -m ray.util.client.server --address=192.168.0.106:6379 --host=0.0.0.0 --port=10001 --mode=proxy --redis-password=5241590000000000 --metrics-agent-port=61905
mydev      200310  7.4  1.1 442296 94352 pts/0    Sl   03:58   0:04 /usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/dashboard/dashboard.py --host=localhost --port=8265 --port-retries=0 --temp-dir=/tmp/ray --log-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --session-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 --minimal
mydev      200361  3.1  0.4 2461104 33396 pts/0   Sl   03:58   0:01 /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/core/src/ray/raylet/raylet --raylet_socket_name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet --store_socket_name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store --object_manager_port=0 --min_worker_port=10002 --max_worker_port=19999 --node_manager_port=0 --node_ip_address=192.168.0.106 --maximum_startup_concurrency=4 --static_resource_list=node:192.168.0.106,1.0,CPU,4,memory,4397877659,object_store_memory,2198938828 --python_worker_command=/usr/bin/python3 /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/workers/setup_worker.py /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/workers/default_worker.py --node-ip-address=192.168.0.106 --node-manager-port=RAY_NODE_MANAGER_PORT_PLACEHOLDER --object-store-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store --raylet-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet --redis-address=None --storage=None --temp-dir=/tmp/ray --metrics-agent-port=61905 --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 RAY_WORKER_DYNAMIC_OPTION_PLACEHOLDER --redis-password=5241590000000000 --java_worker_command= --cpp_worker_command=/usr/bin/python3 /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/workers/setup_worker.py /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/cpp/default_worker --ray_plasma_store_socket_name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store --ray_raylet_socket_name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet --ray_node_manager_port=RAY_NODE_MANAGER_PORT_PLACEHOLDER --ray_address=192.168.0.106:6379 --ray_redis_password=5241590000000000 --ray_session_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --ray_logs_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --ray_node_ip_address=192.168.0.106 RAY_WORKER_DYNAMIC_OPTION_PLACEHOLDER --native_library_path=/opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/cpp/lib --redis_password=5241590000000000 --temp_dir=/tmp/ray --session_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --log_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --resource_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/runtime_resources --metrics-agent-port=61905 --metrics_export_port=64385 --object_store_memory=2198938828 --plasma_directory=/dev/shm --ray-debugger-external=0 --gcs-address=192.168.0.106:6379 --agent_command=/usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/dashboard/agent.py --node-ip-address=192.168.0.106 --metrics-export-port=64385 --dashboard-agent-port=61905 --listen-port=52365 --node-manager-port=RAY_NODE_MANAGER_PORT_PLACEHOLDER --object-store-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store --raylet-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet --temp-dir=/tmp/ray --session-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --runtime-env-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/runtime_resources --log-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 --minimal
mydev      200368  6.3  1.0 414132 83080 pts/0    Sl   03:58   0:03 /usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/log_monitor.py --logs-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --gcs-address=192.168.0.106:6379 --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5
mydev      200408  7.2  1.1 436088 95492 pts/0    Sl   03:58   0:03 /usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/dashboard/agent.py --node-ip-address=192.168.0.106 --metrics-export-port=64385 --dashboard-agent-port=61905 --listen-port=52365 --node-manager-port=35059 --object-store-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store --raylet-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet --temp-dir=/tmp/ray --session-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --runtime-env-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/runtime_resources --log-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 --minimal --agent-id 424238335
mydev      200725  0.0  0.0 236020  3572 pts/0    R+   03:59   0:00 grep --color=auto -i ray
[mydev@fedora /]$
```

```
[mydev@fedora /]$ ray start --address='192.168.0.106:6379'
Local node IP: 192.168.0.106

--------------------
Ray runtime started.
--------------------

To terminate the Ray runtime, run
  ray stop
[mydev@fedora /]$
[mydev@fedora /]$ ps aux |grep -i ray
mydev      200272  4.5  0.4 216152 36264 pts/0    Sl   03:58   0:05 /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/core/src/ray/gcs/gcs_server --log_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --config_list=eyJvYmplY3Rfc3BpbGxpbmdfY29uZmlnIjogIntcInR5cGVcIjogXCJmaWxlc3lzdGVtXCIsIFwicGFyYW1zXCI6IHtcImRpcmVjdG9yeVwiOiBcIi9tcC9yYXkvc2Vzc2lvbl8yMDIyLTEwLTIwXzAzLTU4LTUzXzIyMDM1NF8yMDAxOTNcIn19IiwgImlzX2V4dGVybmFsX3N0b3JhZ2VfdHlwZV9mcyI6IHRydWV9 --gcs_server_port=6379 --metrics-agent-port=61905 --node-ip-address=192.168.0.106 --redis_password=5241590000000000
mydev      200296  2.1  1.0 452604 84244 pts/0    Sl   03:58   0:02 /usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/autoscaler/_private/monitor.py --logs-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 --redis-password=5241590000000000 --monitor-ip=192.168.0.106
mydev      200303  3.6  1.0 433716 83320 pts/0    Sl   03:58   0:04 /usr/bin/python3 -m ray.util.client.server --address=192.168.0.106:6379 --host=0.0.0.0 --port=10001 --mode=proxy --redis-password=5241590000000000 --metrics-agent-port=61905
mydev      200310  3.6  1.1 455236 96156 pts/0    Sl   03:58   0:04 /usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/dashboard/dashboard.py --host=localhost --port=8265 --port-retries=0 --temp-dir=/tmp/ray --log-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --session-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 --minimal
mydev      200361  3.0  0.4 2463664 35192 pts/0   Sl   03:58   0:03 /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/core/src/ray/raylet/raylet --raylet_socket_name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet --store_socket_name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store --object_manager_port=0 --min_worker_port=10002 --max_worker_port=19999 --node_manager_port=0 --node_ip_address=192.168.0.106 --maximum_startup_concurrency=4 --static_resource_list=node:192.168.0.106,1.0,CPU,4,memory,4397877659,object_store_memory,2198938828 --python_worker_command=/usr/bin/python3 /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/workers/setup_worker.py /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/workers/default_worker.py --node-ip-address=192.168.0.106 --node-manager-port=RAY_NODE_MANAGER_PORT_PLACEHOLDER --object-store-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store --raylet-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet --redis-address=None --storage=None --temp-dir=/tmp/ray --metrics-agent-port=61905 --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 RAY_WORKER_DYNAMIC_OPTION_PLACEHOLDER --redis-password=5241590000000000 --java_worker_command= --cpp_worker_command=/usr/bin/python3 /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/workers/setup_worker.py /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/cpp/default_worker --ray_plasma_store_socket_name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store --ray_raylet_socket_name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet --ray_node_manager_port=RAY_NODE_MANAGER_PORT_PLACEHOLDER --ray_address=192.168.0.106:6379 --ray_redis_password=5241590000000000 --ray_session_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --ray_logs_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --ray_node_ip_address=192.168.0.106 RAY_WORKER_DYNAMIC_OPTION_PLACEHOLDER --native_library_path=/opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/cpp/lib --redis_password=5241590000000000 --temp_dir=/tmp/ray --session_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --log_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --resource_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/runtime_resources --metrics-agent-port=61905 --metrics_export_port=64385 --object_store_memory=2198938828 --plasma_directory=/dev/shm --ray-debugger-external=0 --gcs-address=192.168.0.106:6379 --agent_command=/usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/dashboard/agent.py --node-ip-address=192.168.0.106 --metrics-export-port=64385 --dashboard-agent-port=61905 --listen-port=52365 --node-manager-port=RAY_NODE_MANAGER_PORT_PLACEHOLDER --object-store-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store --raylet-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet --temp-dir=/tmp/ray --session-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --runtime-env-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/runtime_resources --log-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 --minimal
mydev      200368  4.3  1.0 414132 83076 pts/0    Sl   03:58   0:05 /usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/log_monitor.py --logs-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --gcs-address=192.168.0.106:6379 --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5
mydev      200408  3.3  1.1 436088 95044 pts/0    Sl   03:58   0:03 /usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/dashboard/agent.py --node-ip-address=192.168.0.106 --metrics-export-port=64385 --dashboard-agent-port=61905 --listen-port=52365 --node-manager-port=35059 --object-store-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store --raylet-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet --temp-dir=/tmp/ray --session-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --runtime-env-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/runtime_resources --log-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 --minimal --agent-id 424238335



mydev      201028  3.6  0.4 2415152 32532 pts/0   Sl   04:00   0:00 /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/core/src/ray/raylet/raylet --raylet_socket_name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet.1 --store_socket_name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store.1 --object_manager_port=0 --min_worker_port=10002 --max_worker_port=19999 --node_manager_port=0 --node_ip_address=192.168.0.106 --maximum_startup_concurrency=4 --static_resource_list=node:192.168.0.106,1.0,CPU,4,memory,5033217639,object_store_memory,2157093273 --python_worker_command=/usr/bin/python3 /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/workers/setup_worker.py /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/workers/default_worker.py --node-ip-address=192.168.0.106 --node-manager-port=RAY_NODE_MANAGER_PORT_PLACEHOLDER --object-store-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store.1 --raylet-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet.1 --redis-address=None --storage=None --temp-dir=/tmp/ray --metrics-agent-port=61410 --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 RAY_WORKER_DYNAMIC_OPTION_PLACEHOLDER --redis-password=5241590000000000 --java_worker_command= --cpp_worker_command=/usr/bin/python3 /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/workers/setup_worker.py /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/cpp/default_worker --ray_plasma_store_socket_name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store.1 --ray_raylet_socket_name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet.1 --ray_node_manager_port=RAY_NODE_MANAGER_PORT_PLACEHOLDER --ray_address=192.168.0.106:6379 --ray_redis_password=5241590000000000 --ray_session_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --ray_logs_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --ray_node_ip_address=192.168.0.106 RAY_WORKER_DYNAMIC_OPTION_PLACEHOLDER --native_library_path=/opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/cpp/lib --redis_password=5241590000000000 --temp_dir=/tmp/ray --session_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --log_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --resource_dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/runtime_resources --metrics-agent-port=61410 --metrics_export_port=45285 --object_store_memory=2157093273 --plasma_directory=/dev/shm --ray-debugger-external=0 --gcs-address=192.168.0.106:6379 --agent_command=/usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/dashboard/agent.py --node-ip-address=192.168.0.106 --metrics-export-port=45285 --dashboard-agent-port=61410 --listen-port=52365 --node-manager-port=RAY_NODE_MANAGER_PORT_PLACEHOLDER --object-store-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store.1 --raylet-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet.1 --temp-dir=/tmp/ray --session-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --runtime-env-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/runtime_resources --log-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 --minimal
mydev      201035 10.3  1.0 414132 83148 pts/0    Sl   04:00   0:02 /usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/log_monitor.py --logs-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --gcs-address=192.168.0.106:6379 --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5
mydev      201075 14.2  1.1 436088 94412 pts/0    Sl   04:00   0:03 /usr/bin/python3 -u /opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/dashboard/agent.py --node-ip-address=192.168.0.106 --metrics-export-port=45285 --dashboard-agent-port=61410 --listen-port=52365 --node-manager-port=40827 --object-store-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/plasma_store.1 --raylet-name=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/sockets/raylet.1 --temp-dir=/tmp/ray --session-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193 --runtime-env-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/runtime_resources --log-dir=/tmp/ray/session_2022-10-20_03-58-53_220354_200193/logs --logging-rotate-bytes=536870912 --logging-rotate-backup-count=5 --gcs-address=192.168.0.106:6379 --minimal --agent-id 424238335
mydev      201245  0.0  0.0 236020  3576 pts/0    R+   04:00   0:00 grep --color=auto -i ray
[mydev@fedora /]$
```

test on two RPi4 boards

1. “clone” the repo that Ray successfully built together with those installed to ~/.local from one RPi4 board to another, and make sure the cloned version is work;
2. test

on node 192.168.0.102

```
[mydev@fedora /]$ ray start --address=192.168.0.106:6379
Local node IP: 192.168.0.102
2022-10-20 23:33:19,619 WARNING utils.py:1331 -- Unable to connect to GCS at 192.168.0.106:6379. Check that (1) Ray GCS with matching version started successfully at the specified address, and (2) there is no
 firewall setting preventing access.
```

```
Traceback (most recent call last):
  File "/home/mydev/.local/bin/ray", line 33, in <module>
    sys.exit(load_entry_point('ray', 'console_scripts', 'ray')())
  File "/opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/scripts/scripts.py", line 2596, in main
    return cli()
  File "/usr/lib/python3.10/site-packages/click/core.py", line 1128, in __call__
    return self.main(*args, **kwargs)
  File "/usr/lib/python3.10/site-packages/click/core.py", line 1053, in main
    rv = self.invoke(ctx)
  File "/usr/lib/python3.10/site-packages/click/core.py", line 1659, in invoke
    return _process_result(sub_ctx.command.invoke(sub_ctx))
  File "/usr/lib/python3.10/site-packages/click/core.py", line 1395, in invoke
    return ctx.invoke(self.callback, **ctx.params)
  File "/usr/lib/python3.10/site-packages/click/core.py", line 754, in invoke
    return __callback(*args, **kwargs)
  File "/opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/autoscaler/_private/cli_logger.py", line 852, in wrapper
    return f(*args, **kwargs)
  File "/opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/scripts/scripts.py", line 885, in start
    node = ray._private.node.Node(
  File "/opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/node.py", line 182, in __init__
    session_name = ray._private.utils.internal_kv_get_with_retry(
  File "/opt/MyWorkSpace/MyProjs/HCI/Ray/Official/ray-master/python/ray/_private/utils.py", line 1347, in internal_kv_get_with_retry
    raise ConnectionError(
ConnectionError: Could not read 'session_name' from GCS. Did GCS start successfully?
[mydev@fedora /]$
```

on node 192.168.0.106

```
[mydev@fedora /]$ sudo firewall-cmd --query-port=6379/tcp
[sudo] password for mydev:
no
```

```
[mydev@fedora /]$ sudo firewall-cmd --add-port=6379/tcp --permanent
success
[mydev@fedora /]$
```

```
[mydev@fedora /]$ sudo firewall-cmd --reload
success
[mydev@fedora /]$ sudo firewall-cmd --query-port=6379/tcp
yes
```

reconnect from node 192.168.0.102

```
[mydev@fedora /]$ ray start --address=192.168.0.106:6379
Local node IP: 192.168.0.102
[2022-10-20 23:55:09,328 I 116110 116110] global_state_accessor.cc:357: This node has an IP address of 192.168.0.102, while we can not find the matched Raylet address. This maybe come from when you connect th
e Ray cluster with a different IP address or connect a container.

--------------------
Ray runtime started.
--------------------

To terminate the Ray runtime, run
  ray stop
[mydev@fedora /]$
```

```
[mydev@fedora /]$ ray status
======== Autoscaler status: 2022-10-21 00:30:18.673354 ========
Node status
---------------------------------------------------------------
Healthy:
 1 node_6d968409b7c248088c2a9c94b90114c76375654bf38c8f7ce52300dc
 1 node_c5d6da920dc923e98ed9b114ce5796d7239c85550116265dfa5ccb4a
 1 node_231339a529adf874e772b2403ee348074301293a819dbf1882666322
Pending:
 (no pending nodes)
Recent failures:
 (no failures)

Resources
---------------------------------------------------------------
Usage:
 0.0/8.0 CPU
 0.00/8.783 GiB memory
 0.00/4.057 GiB object_store_memory

Demands:
 (no resource demands)
[mydev@fedora /]$
```

More testing is needed…

2.4 Arkade

- https://github.com/alexellis/arkade
  Open Source Marketplace For Kubernetes.
- https://kccncna20.sched.com/event/ekAF/the-past-present-and-future-of-kubernetes-on-raspberry-pi-alex-ellis-openfaas-ltd

arkade is how developers install the latest versions of their favourite tools and Kubernetes apps.

With `arkade get`, you'll have `kubectl`, `kind`, `terraform`, and `jq` on your machine faster than you can type `apt-get install/brew update`.

Add Ray to Arkade for ARM64

- Will work on a .whl for it

IV. Speeding up Ray

1) Accelerating Python

- For more details, you may refer to my previous talk "A survey of current Python implementations" at PyCon China 2021(Online) and the upcoming follow-ups.

1.1 Make CPython faster

Bottlenecks of current CPython implementation

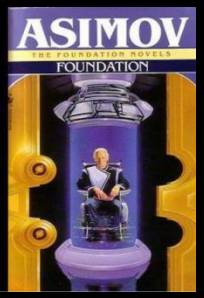
- No JIT
- GIL

  https://en.wikipedia.org/wiki/Global_interpreter_lock
- …

1.1.1 The “Shannon Plan”

- Plan from the Father of Python

  https://thenewstack.io/guido-van-rossums-ambitious-plans-for-improving-python-performance/
- • Posted to GitHub and python-dev last October
  - github.com/markshannon/faster-cpython
  - Based on experience with “HotPy”, “HoyPy 2”
  - Promises 5x in 4 years (1.5x per year)
  - Looking for funding



Source: https://github.com/markshannon/faster-cpython/blob/master/plan.md

- **https://pyfound.blogspot.com/2021/05/the-2021-python-language-summit-making.html**

Seven months ago, Guido van Rossum left a brief retirement to work at Microsoft. He was given the freedom to pick a project and decided to work on making CPython faster. Microsoft will be funding a small team consisting of Guido van Rossum, Mark Shannon, Eric Snow, and possibly others.

The team will:

- Collaborate fully and openly with CPython's core developers
- Make incremental changes to CPython
- Take care of maintenance and support
- Keep all project-specific repos open
- Have all discussions in trackers on open GitHub repos

The team will need to work within some constraints. They'll need to keep code maintainable, not break stable ABI compatibility, not break limited API compatibility, and not break or slow down extreme cases, such as pushing a million cases on the eval stack. Although they won't be able to change the data model, they will be able to change:

- The byte code
- The compiler
- The internals of a lot of objects

Who Will Benefit?

You'll benefit from the speed increase if you:

- Run CPU-intensive pure Python code
- Use tools or websites built in CPython

You might not benefit from the speed increase if you:

- Rewrote your code in C, Cython, C++, or similar to increase speed already (e.g. NumPy, TensorFlow)
- Have code that is mostly waiting for I/O
- Use multithreading
- Need to make your code more algorithmically efficient first

Stages

The stages to high performance

Stage 1 -- Python 3.10

The key improvement for 3.10 will be an adaptive, specializing interpreter. The interpreter will adapt to types and values during execution, exploiting type stability in the program, without needing runtime code generation.

Stage 2 -- Python 3.11

This stage will make many improvements to the runtime and key objects. Stage two will be characterized by lots of "tweaks", rather than any "headline" improvement. The planned improvements include:

- Improved performance for integers of less than one machine word.
- Improved peformance for binary operators.
- Faster calls and returns, through better handling of frames.
- Better object memory layout and reduced memory management overhead.
- Zero overhead exception handling.
- Further enhancements to the interpreter
- Other small enhancements.

Stage 3 -- Python 3.12 (requires runtime code generation)

Simple "JIT" compiler for small regions. Compile small regions of specialized code, using a relatively simple, fast compiler.

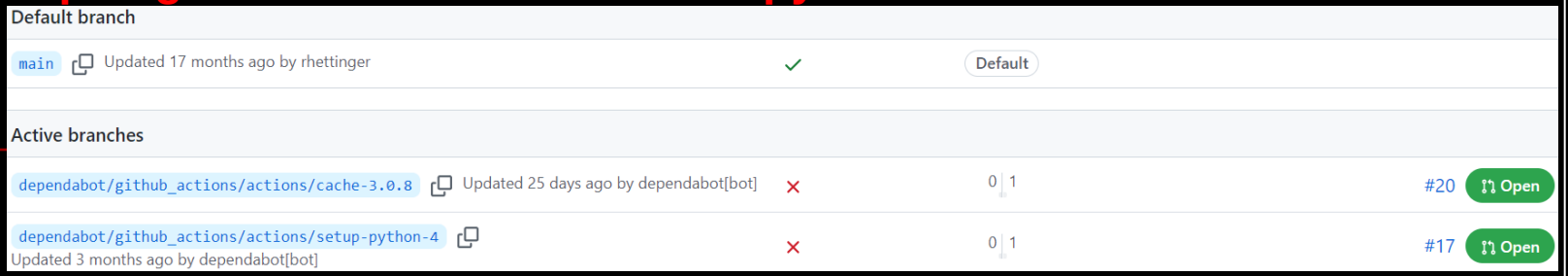
Stage 4 -- Python 3.13 (requires runtime code generation)

Extend regions for compilation. Enhance compiler to generate superior machine code.

Source: https://github.com/markshannon/faster-cpython/blob/master/plan.md

1.1.1.1 Current Status

- https://github.com/markshannon/cpython



…

https://github.com/faster-cpython/ideas

- https://pyfound.blogspot.com/2022/05/the-2022-python-language-summit_2.html

> Python 3.11, if you haven't heard, is fast. Over the past year, Microsoft has funded a team - led by core developers Mark Shannon and Guido van Rossum - to work full-time on making CPython faster. With additional funding from Bloomberg, and help from a wide range of other contributors from the community, the results have borne fruit. On the pyperformance benchmarks at the time of the beta release, Python 3.11 was around 1.25x faster than Python 3.10, a phenomenal achievement.

> The gains Shannon's team has achieved are hugely impressive, and likely to benefit the community as a whole in a profound way. But various problems lie on the horizon. Sam Gross's proposal for a version of CPython without the Global Interpreter Lock (the `nogil` fork) has potential for speeding up multithreaded Python code in very different ways to the Faster CPython team's work - but it could also be problematic for some of the optimisations that have already been implemented, many of which assume that the GIL exists. Eric Snow's dream of achieving multiple subinterpreters within a single process, meanwhile, will have a smaller performance impact on single-threaded code compared to `nogil`, but could still create some minor complications for Shannon's team.

- https://towardsdatascience.com/python-3-14-will-be-faster-than-c-a97edd01d65d

1.1.2 HPy

- https://hpyproject.org/
  A better C API for Python.
- Advantages

- **Zero overhead** on CPython: extensions written in HPy run at the same speed as "normal" extensions.
- **Much faster** on alternative implementations such as PyPy, GraalPython.
- **Universal binaries**: extensions built for the HPy Universal ABI can be loaded unmodified on CPython, PyPy, GraalPython, etc.
- **A migration path** for mixing legacy C-API calls with HPy API calls. Once all the code is migrated, the extension can be compiled as a universal binary that works on any CPython version, PyPy, or GraalPy.
- **Debug mode**: in debug mode, you can easily identify common problems such as memory leaks, invalid lifetime of objects, invalid usage of APIs. Have you ever forgot a `Py_INCREF` or `Py_DECREF`? The HPy debug mode can be activated at runtime to detect these mistakes for you on universal binaries.
- **Nicer API**: the standard Python/C API shows its age. HPy is designed to overcome some of its limitations, be more consistent, produce better quality extensions and to make it harder to introduce bugs.
- **Evolvability**: As nicely summarized in [PEP-620](https://peps.python.org/pep-0620/) the standard Python/C API exposes a lot of internal implementation details which makes it hard to evolve the C API. HPy doesn't have this problem because all internal implementation details are hidden.

- https://hpyproject.org/blog/posts/2022/06/hpy-0.0.4-third-public-release/
- https://medium.com/graalvm/hpy-better-python-c-api-in-practice-79328246e2f8

- https://medium.com/graalvm/hpy-binary-compatibility-and-api-evolution-with-kiwisolver-7f7a811ef7f9

Example

```c
#include "Python.h"

static PyObject *myabs(PyObject *self, PyObject *arg) {
    return Py_Absolute(arg);
}

static PyMethodDef methods[] = {
    {"myabs", myabs, METH_O, ""},
    {NULL, NULL, 0, NULL}
};

static struct PyModuleDef module = {
    .m_methods = methods
    // ...
};

```

```c
// ----------------------------------------
// The same with HPy:

#include "hpy.h"

HPyDef_METH(myabs, "myabs", myabs_impl, HPyFunc_O)
static HPy myabs_impl(HPyContext *ctx, HPy self, HPy arg) {
    return HPy_Absolute(ctx, arg);
}

static HPyDef *methods[] = { &myabs, NULL };

static HPyModuleDef module = {
    .defines = methods,
    // ...
};
```

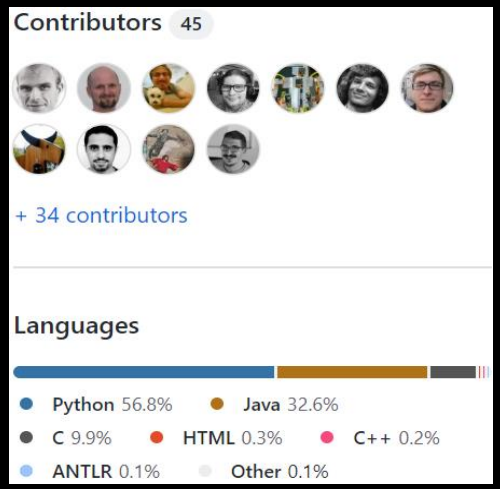
Source: https://medium.com/graalvm/hpy-better-python-c-api-in-practice-79328246e2f8

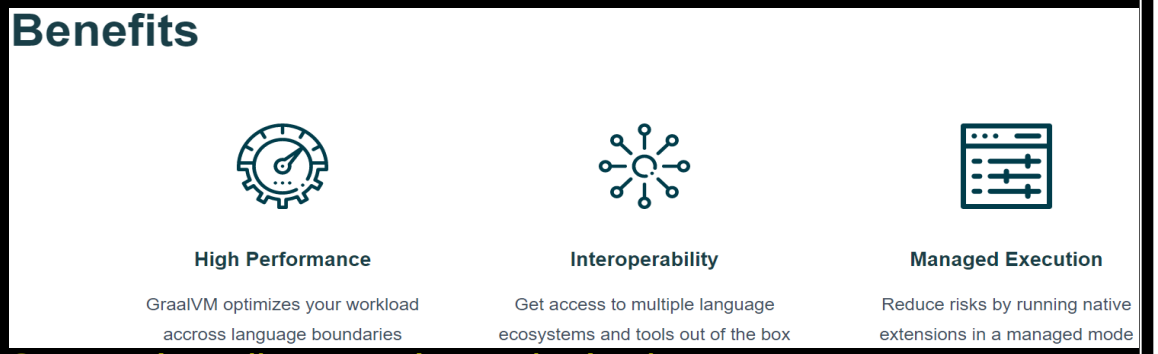
1.2 Best candidates for Ray

1.2.1 GraalPython

- https://github.com/oracle/graalpython

This is an early-stage experimental implementation of Python. A primary goal is to support SciPy and its constituent libraries. GraalPy can usually execute pure Python code faster than CPython (but not when C extensions are involved). GraalPy currently aims to be compatible with Python 3.8, but it is a long way from there, and it is very likely that any Python program that uses more features of standard library modules or external packages will hit something unsupported. At this point, the Python implementation is made available for experimentation and curious end-users.

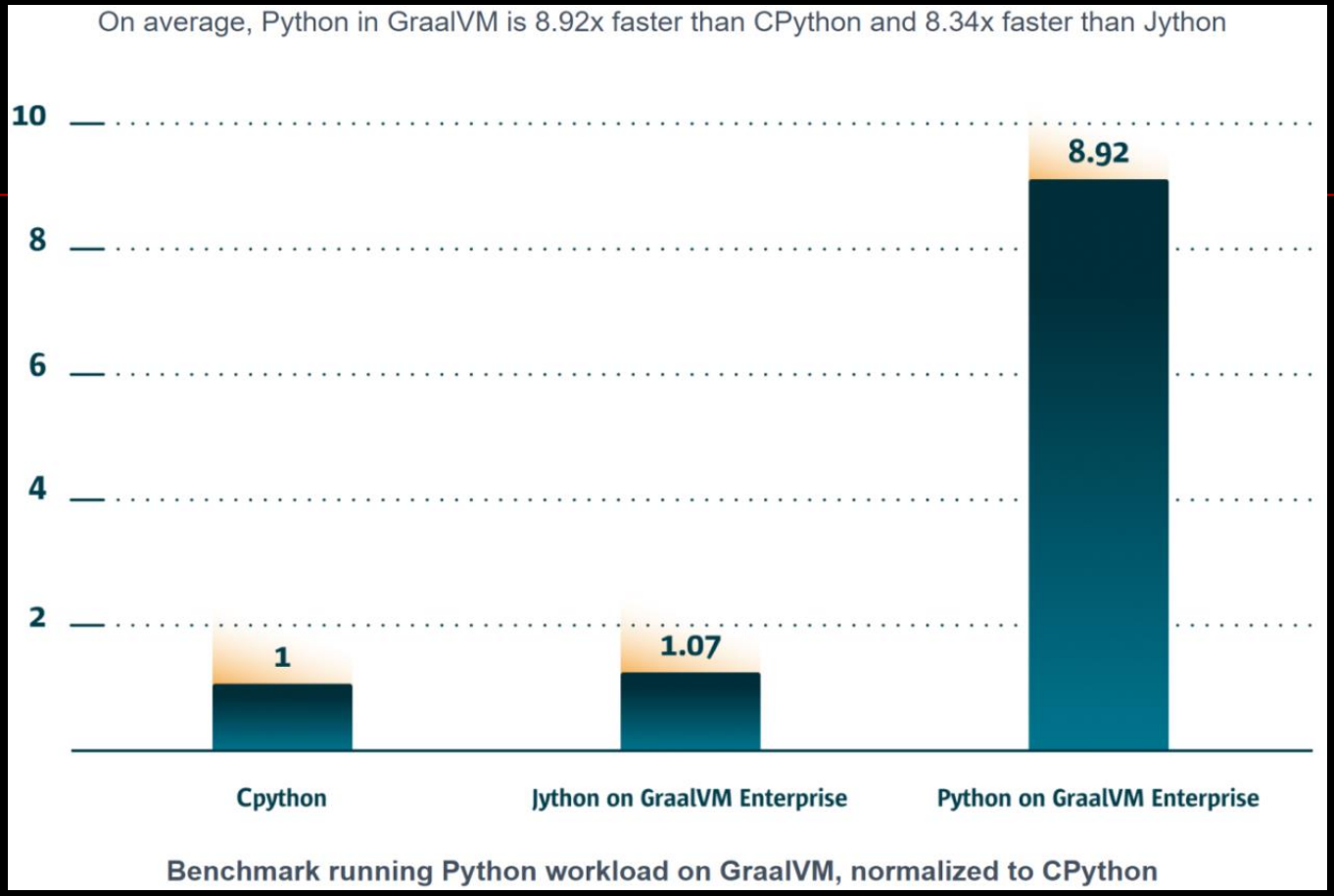




Source: https://www.graalvm.org/python/

- https://github.com/oracle/graalpython/blob/master/docs/user/Jython.md
- https://www.graalvm.org/python/
- …

Performance



Source: https://www.graalvm.org/python/

Supported architectures

- Mainly support X64, and that for AArch64 is experimental.

installable package for GraalVM CE 22.3.0-dev-20221004_1644:

plugin for graalvm

python-installable-svm-java11-darwin-aarch64-dev.jar	145 MB
python-installable-svm-java11-darwin-amd64-dev.jar	144 MB
python-installable-svm-java11-linux-aarch64-dev.jar	143 MB
python-installable-svm-java11-linux-amd64-dev.jar	145 MB
python-installable-svm-java17-darwin-aarch64-dev.jar	146 MB
python-installable-svm-java17-darwin-amd64-dev.jar	145 MB
python-installable-svm-java17-linux-aarch64-dev.jar	144 MB
python-installable-svm-java17-linux-amd64-dev.jar	146 MB
python-installable-svm-java19-darwin-aarch64-dev.jar	146 MB
python-installable-svm-java19-darwin-amd64-dev.jar	145 MB
python-installable-svm-java19-linux-aarch64-dev.jar	144 MB
python-installable-svm-java19-linux-amd64-dev.jar	146 MB

Source: https://github.com/graalvm/graalvm-ce-dev-builds/releases

or standalone version:

graalpython-dev-linux-aarch64.tar.gz	192 MB
graalpython-dev-linux-amd64.tar.gz	195 MB
graalpython-dev-macos-aarch64.tar.gz	199 MB
graalpython-dev-macos-amd64.tar.gz	202 MB

Source: https://github.com/graalvm/graalvm-ce-dev-builds/releases

Make it works on RPi4

```
[mydev@fedora /]$ which python
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/bin/python
[mydev@fedora /]$
[mydev@fedora /]$ tree -L 2 -d /opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python
├── bin
├── docs
│   ├── contributor
│   └── user
├── include
│   ├── cpython
│   └── hpy
├── lib
├── lib-graalpython
│   ├── include
│   ├── lib
│   ├── modules
│   └── patches
└── lib-python
    └── 3
```

```
[mydev@fedora /]$ ll /opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/bin |grep -i python
lrwxrwxrwx. 1 mydev mydev    31 Oct  6 13:32 graalpy → ../languages/python/bin/graalpy*
lrwxrwxrwx. 1 mydev mydev    30 Oct  6 13:32 python → ../languages/python/bin/python*
lrwxrwxrwx. 1 mydev mydev    31 Oct  6 13:32 python3 → ../languages/python/bin/python3*
[mydev@fedora /]$
[mydev@fedora /]$ tree /opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/bin
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/bin
├── graalpy
├── python
└── python3
```

...

dynamic libraries:

```
[mydev@fedora /]$ find /opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19 -name "*.so" |grep -i graalpy
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/lib/graalpy-38-native-aarch64-linux/libbz2.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/lib/graalpy-38-native-aarch64-linux/liblzma.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/_bz2.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/_cpython_sre.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/_cpython_struct.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/_cpython_unicodedata.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/_ctypes_test.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/_mmap.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/_testcapi.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/_testmultiphase.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/libbz2support.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/libhpy.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/liblzmasupport.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/libposix.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/libpython.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/libzsupport.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/modules/pyexpat.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/libbz2support.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/libhpy.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/liblzmasupport.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/libposix.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/libpython.graalpy-38-native-aarch64-linux.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/libpythonjni.so
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/libzsupport.graalpy-38-native-aarch64-linux.so
[mydev@fedora /]$
```

failed to run:

```
[mydev@fedora /]$ python -V
ERROR: com.oracle.truffle.api.CompilerDirectives$ShouldNotReachHere: Unable to load native posix support library
org.graalvm.polyglot.PolyglotException: com.oracle.truffle.api.CompilerDirectives$ShouldNotReachHere: Unable to load native posix support library
        at org.graalvm.truffle/com.oracle.truffle.api.CompilerDirectives.shouldNotReachHere(CompilerDirectives.java:574)
        at com.oracle.graal.python.runtime.NFIPosixSupport$InvokeNativeFunction.loadLibrary(NFIPosixSupport.java:337)
        at com.oracle.graal.python.runtime.NFIPosixSupport$InvokeNativeFunction.call(NFIPosixSupport.java:277)
        at com.oracle.graal.python.runtime.NFIPosixSupport$InvokeNativeFunction.callLong(NFIPosixSupport.java:292)
        at com.oracle.graal.python.runtime.NFIPosixSupport.lseek(NFIPosixSupport.java:579)
        at com.oracle.graal.python.runtime.NFIPosixSupportGen$PosixSupportLibraryExports$Uncached.lseek(NFIPosixSupportGen.java:3511)
        at com.oracle.graal.python.runtime.ImageBuildtimePosixSupport.lseek(ImageBuildtimePosixSupport.java:257)
        at com.oracle.graal.python.runtime.ImageBuildtimePosixSupportGen$PosixSupportLibraryExports$Uncached.lseek(ImageBuildtimePosixSupportGen.java:1347)
        at com.oracle.graal.python.runtime.PosixSupportLibraryGen$UncachedDispatch.lseek(PosixSupportLibraryGen.java:6106)
        at com.oracle.graal.python.builtins.modules.io.FileIOBuiltins$SeekNode.internalSeek(FileIOBuiltins.java:803)
        at com.oracle.graal.python.builtins.modules.io.FileIOBuiltins$TellNode.internalTell(FileIOBuiltins.java:828)
        at com.oracle.graal.python.builtins.modules.io.AbstractBufferedIOBuiltins$BufferedInitNode.internalInit(AbstractBufferedIOBuiltins.java:119)
        at com.oracle.graal.python.builtins.modules.io.BufferedReaderBuiltins$BufferedReaderInit.internalInit(BufferedReaderBuiltins.java:96)
        at com.oracle.graal.python.builtins.modules.SysModuleBuiltins.initStd(SysModuleBuiltins.java:661)
        at com.oracle.graal.python.builtins.modules.SysModuleBuiltins.postInitialize(SysModuleBuiltins.java:644)
        at com.oracle.graal.python.builtins.Python3Core.postInitialize(Python3Core.java:932)
        at com.oracle.graal.python.runtime.PythonContext.patch(PythonContext.java:1338)
        at com.oracle.graal.python.PythonLanguage.patchContext(PythonLanguage.java:349)
        at com.oracle.graal.python.PythonLanguage.patchContext(PythonLanguage.java:134)
        at org.graalvm.truffle/com.oracle.truffle.api.LanguageAccessor$LanguageImpl.patchEnvContext(LanguageAccessor.java:418)
        at org.graalvm.truffle/com.oracle.truffle.polyglot.PolyglotLanguageContext.patch(PolyglotLanguageContext.java:784)
        at org.graalvm.truffle/com.oracle.truffle.polyglot.PolyglotContextImpl.patch(PolyglotContextImpl.java:3187)
        at org.graalvm.truffle/com.oracle.truffle.polyglot.PolyglotEngineImpl.loadPreinitializedContext(PolyglotEngineImpl.java:1864)
        at org.graalvm.truffle/com.oracle.truffle.polyglot.PolyglotEngineImpl.createContext(PolyglotEngineImpl.java:1736)
        at org.graalvm.truffle/com.oracle.truffle.polyglot.PolyglotEngineDispatch.createContext(PolyglotEngineDispatch.java:162)
        at org.graalvm.sdk/org.graalvm.polyglot.Context$Builder.build(Context.java:1861)
        at com.oracle.graal.python.shell.GraalPythonMain.launch(GraalPythonMain.java:648)
        at org.graalvm.launcher.AbstractLanguageLauncher.launch(AbstractLanguageLauncher.java:296)
        at org.graalvm.launcher.AbstractLanguageLauncher.launch(AbstractLanguageLauncher.java:121)
        at org.graalvm.launcher.AbstractLanguageLauncher.runLauncher(AbstractLanguageLauncher.java:168)
Caused by: java.lang.UnsatisfiedLinkError: libcrypt.so.1: cannot open shared object file: No such file or directory
Internal GraalVM error, please report at https://github.com/oracle/graal/issues/.
[mydev@fedora /]$
```

the libcrypt issue:

https://fedoraproject.org/wiki/Changes/Replace_glibc_libcrypt_with_libxcrypt

...

check /lib64:

```
[mydev@fedora /]$ cd /lib64
[mydev@fedora lib64]$
[mydev@fedora lib64]$ ll |grep -i crypt
drwxr-xr-x. 1 root  root        216 Jul 21 07:00 cryptsetup/
lrwxrwxrwx. 1 root  root         21 Sep 16 01:20 libbd_crypto.so.2 → libbd_crypto.so.2.0.0*
-rwxr-xr-x. 1 root  root      69464 Sep 16 01:20 libbd_crypto.so.2.0.0*
-rw-r--r--. 1 root  root     635614 Feb  1  2022 libcrypt.a
lrwxrwxrwx. 1 root  root         18 Jul  7 05:38 libcrypto.so → libcrypto.so.3.0.5*
lrwxrwxrwx. 1 root  root         19 Jul  7 05:37 libcrypto.so.1.1 → libcrypto.so.1.1.1q*
-rwxr-xr-x. 1 root  root    2896072 Jul  7 05:38 libcrypto.so.1.1.1q*
lrwxrwxrwx. 1 root  root         18 Jul  7 05:38 libcrypto.so.3 → libcrypto.so.3.0.5*
-rwxr-xr-x. 1 root  root    4239968 Jul  7 05:38 libcrypto.so.3.0.5*
lrwxrwxrwx. 1 root  root         23 Jan 19  2022 libcryptsetup.so.12 → libcryptsetup.so.12.7.0*
-rwxr-xr-x. 1 root  root     476784 Jan 19  2022 libcryptsetup.so.12.7.0*
lrwxrwxrwx. 1 root  root         17 Feb  1  2022 libcrypt.so → libcrypt.so.2.0.0*
lrwxrwxrwx. 1 root  root         17 Feb  1  2022 libcrypt.so.2 → libcrypt.so.2.0.0*
-rwxr-xr-x. 1 root  root     201520 Feb  1  2022 libcrypt.so.2.0.0*
lrwxrwxrwx. 1 root  root         19 May 31 03:00 libgcrypt.so.20 → libgcrypt.so.20.4.1*
-rwxr-xr-x. 1 root  root     947632 May 31 03:00 libgcrypt.so.20.4.1*
-rwxr-xr-x. 1 root  root      70480 Jan 24  2022 libHScryptohash-sha256-0.11.102.0-BIK7tUgUf6XGJKq7LBFHn0-ghc8.10.5.so*
-rwxr-xr-x. 1 root  root    8079264 Feb 13  2022 libHScryptonite-0.29-5r7lnD4aYWdIGrQffy1AVK-ghc8.10.5.so*
-rwxr-xr-x. 1 root  root     138344 Jan 24  2022 libHScryptonite-conduit-0.2.2-Et9uThN3dHs8OO3LRc8k88-ghc8.10.5.so*
lrwxrwxrwx. 1 root  root         18 Jun 15 23:17 libk5crypto.so → libk5crypto.so.3.1*
lrwxrwxrwx. 1 root  root         18 Jun 15 23:17 libk5crypto.so.3 → libk5crypto.so.3.1*
-rwxr-xr-x. 1 root  root     136984 Jun 15 23:17 libk5crypto.so.3.1*
lrwxrwxrwx. 1 root  root         20 Jan 20  2022 libtomcrypt.so.1 → libtomcrypt.so.1.0.1*
-rwxr-xr-x. 1 root  root     863136 Jan 20  2022 libtomcrypt.so.1.0.1*
[mydev@fedora lib64]$
```

```
[mydev@fedora lib64]$ strings libcrypt.so.2.0.0 |grep GLIBC
GLIBC_2.17
GLIBC_2.25
GA+GLIBCXX_ASSERTIONS
[mydev@fedora lib64]$
```

add soft link for libcrypt.so.1 against libcrypt.so.2.0.0:

```
[mydev@fedora lib64]$ sudo ln -s libcrypt.so.2.0.0 libcrypt.so.1
[sudo] password for mydev:
[mydev@fedora lib64]$
[mydev@fedora lib64]$ ll |grep -i crypt
drwxr-xr-x. 1 root  root        216 Jul 21 07:00 cryptsetup/
lrwxrwxrwx. 1 root  root         21 Sep 16 01:20 libbd_crypto.so.2 → libbd_crypto.so.2.0.0*
-rwxr-xr-x. 1 root  root      69464 Sep 16 01:20 libbd_crypto.so.2.0.0*
-rw-r--r--. 1 root  root     635614 Feb  1  2022 libcrypt.a
lrwxrwxrwx. 1 root  root         18 Jul  7 05:38 libcrypto.so → libcrypto.so.3.0.5*
lrwxrwxrwx. 1 root  root         19 Jul  7 05:37 libcrypto.so.1.1 → libcrypto.so.1.1.1q*
-rwxr-xr-x. 1 root  root    2896072 Jul  7 05:38 libcrypto.so.1.1.1q*
lrwxrwxrwx. 1 root  root         18 Jul  7 05:38 libcrypto.so.3 → libcrypto.so.3.0.5*
-rwxr-xr-x. 1 root  root    4239968 Jul  7 05:38 libcrypto.so.3.0.5*
lrwxrwxrwx. 1 root  root         23 Jan 19  2022 libcryptsetup.so.12 → libcryptsetup.so.12.7.0*
-rwxr-xr-x. 1 root  root     476784 Jan 19  2022 libcryptsetup.so.12.7.0*
lrwxrwxrwx. 1 root  root         17 Feb  1  2022 libcrypt.so → libcrypt.so.2.0.0*
lrwxrwxrwx. 1 root  root         17 Oct  6 14:31 libcrypt.so.1 → libcrypt.so.2.0.0*
lrwxrwxrwx. 1 root  root         17 Feb  1  2022 libcrypt.so.2 → libcrypt.so.2.0.0*
-rwxr-xr-x. 1 root  root     201520 Feb  1  2022 libcrypt.so.2.0.0*
lrwxrwxrwx. 1 root  root         19 May 31 03:00 libgcrypt.so.20 → libgcrypt.so.20.4.1*
-rwxr-xr-x. 1 root  root     947632 May 31 03:00 libgcrypt.so.20.4.1*
-rwxr-xr-x. 1 root  root      70480 Jan 24  2022 libHScryptohash-sha256-0.11.102.0-BIK7tUgUf6XGJKq7LBFHn0-ghc8.10.5.so*
-rwxr-xr-x. 1 root  root    8079264 Feb 13  2022 libHScryptonite-0.29-5r7lnD4aYWdIGrQffy1AVK-ghc8.10.5.so*
-rwxr-xr-x. 1 root  root     138344 Jan 24  2022 libHScryptonite-conduit-0.2.2-Et9uThN3dHs8OO3LRc8k88-ghc8.10.5.so*
lrwxrwxrwx. 1 root  root         18 Jun 15 23:17 libk5crypto.so → libk5crypto.so.3.1*
lrwxrwxrwx. 1 root  root         18 Jun 15 23:17 libk5crypto.so.3 → libk5crypto.so.3.1*
-rwxr-xr-x. 1 root  root     136984 Jun 15 23:17 libk5crypto.so.3.1*
lrwxrwxrwx. 1 root  root         20 Jan 20  2022 libtomcrypt.so.1 → libtomcrypt.so.1.0.1*
-rwxr-xr-x. 1 root  root     863136 Jan 20  2022 libtomcrypt.so.1.0.1*
[mydev@fedora lib64]$
```

but still got failed:

```
[mydev@fedora lib64]$ python -V
ERROR: com.oracle.truffle.api.CompilerDirectives$ShouldNotReachHere: Unable to load native posix support library
org.graalvm.polyglot.PolyglotException: com.oracle.truffle.api.CompilerDirectives$ShouldNotReachHere: Unable to load native posix support library
        at org.graalvm.truffle/com.oracle.truffle.api.CompilerDirectives.shouldNotReachHere(CompilerDirectives.java:574)
        at com.oracle.graal.python.runtime.NFIPosixSupport$InvokeNativeFunction.loadLibrary(NFIPosixSupport.java:337)
```

...

```
        at org.graalvm.launcher.AbstractLanguageLauncher.runLauncher(AbstractLanguageLauncher.java:168)
Caused by: java.lang.UnsatisfiedLinkError: /lib64/libcrypt.so.1: version `GLIBC_2.17' not found (required by /opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/lib-graalpython/libposix.graalpy-38-native-aarch64-linux.so)
Internal GraalVM error, please report at https://github.com/oracle/graal/issues/.
[mydev@fedora lib64]$
```

downloaded another one from the internet and retried:

https://centos.pkgs.org/7/centos-aarch64/glibc-2.17-317.el7.aarch64.rpm.html

```
[mydev@fedora lib64]$ strings libcrypt-2.17.so |grep GLIBC
GLIBC_2.17
GLIBC_PRIVATE
memcpy@@GLIBC_2.17
strtoul@@GLIBC_2.17
strlen@@GLIBC_2.17
errno@@GLIBC_PRIVATE
__cxa_finalize@@GLIBC_2.17
snprintf@@GLIBC_2.17
malloc@@GLIBC_2.17
strncmp@@GLIBC_2.17
memset@@GLIBC_2.17
realloc@@GLIBC_2.17
__stpncpy@@GLIBC_2.17
strtol@@GLIBC_2.17
free@@GLIBC_2.17
strncpy@@GLIBC_2.17
__libc_alloca_cutoff@@GLIBC_PRIVATE
[mydev@fedora lib64]$
```

```
[mydev@fedora lib64]$ sudo rm -f libcrypt.so.1
[sudo] password for mydev:
[mydev@fedora lib64]$ sudo ln -s libcrypt-2.17.so libcrypt.so.1
[mydev@fedora lib64]$
[mydev@fedora lib64]$ ll |grep -i crypt
drwxr-xr-x. 1 root  root        216 Jul 21 07:00 cryptsetup/
lrwxrwxrwx. 1 root  root         21 Sep 16 01:20 libbd_crypto.so.2 → libbd_crypto.so.2.0.0*
-rwxr-xr-x. 1 root  root      69464 Sep 16 01:20 libbd_crypto.so.2.0.0*
-rw-r--r--. 1 root  root      74976 Oct  6 14:40 libcrypt-2.17.so
-rw-r--r--. 1 root  root     635614 Feb  1  2022 libcrypt.a
lrwxrwxrwx. 1 root  root         18 Jul  7 05:38 libcrypto.so → libcrypto.so.3.0.5*
lrwxrwxrwx. 1 root  root         19 Jul  7 05:37 libcrypto.so.1.1 → libcrypto.so.1.1.1q*
-rwxr-xr-x. 1 root  root    2896072 Jul  7 05:38 libcrypto.so.1.1.1q*
lrwxrwxrwx. 1 root  root         18 Jul  7 05:38 libcrypto.so.3 → libcrypto.so.3.0.5*
-rwxr-xr-x. 1 root  root    4239968 Jul  7 05:38 libcrypto.so.3.0.5*
lrwxrwxrwx. 1 root  root         23 Jan 19  2022 libcryptsetup.so.12 → libcryptsetup.so.12.7.0*
-rwxr-xr-x. 1 root  root     476784 Jan 19  2022 libcryptsetup.so.12.7.0*
lrwxrwxrwx. 1 root  root         17 Feb  1  2022 libcrypt.so → libcrypt.so.2.0.0*
lrwxrwxrwx. 1 root  root         16 Oct  6 14:42 libcrypt.so.1 → libcrypt-2.17.so
lrwxrwxrwx. 1 root  root         17 Feb  1  2022 libcrypt.so.2 → libcrypt.so.2.0.0*
-rwxr-xr-x. 1 root  root     201520 Feb  1  2022 libcrypt.so.2.0.0*
lrwxrwxrwx. 1 root  root         19 May 31 03:00 libgcrypt.so.20 → libgcrypt.so.20.4.1*
-rwxr-xr-x. 1 root  root     947632 May 31 03:00 libgcrypt.so.20.4.1*
-rwxr-xr-x. 1 root  root      70480 Jan 24  2022 libHScryptohash-sha256-0.11.102.0-BIK7tUgUf6XGJKq7LBFHn0-ghc8.10.5.so*
-rwxr-xr-x. 1 root  root    8079264 Feb 13  2022 libHScryptonite-0.29-5r7lnD4aYWdIGrQffy1AVK-ghc8.10.5.so*
-rwxr-xr-x. 1 root  root     138344 Jan 24  2022 libHScryptonite-conduit-0.2.2-Et9uThN3dHs8OO3LRc8k88-ghc8.10.5.so*
lrwxrwxrwx. 1 root  root         18 Jun 15 23:17 libk5crypto.so → libk5crypto.so.3.1*
lrwxrwxrwx. 1 root  root         18 Jun 15 23:17 libk5crypto.so.3 → libk5crypto.so.3.1*
-rwxr-xr-x. 1 root  root     136984 Jun 15 23:17 libk5crypto.so.3.1*
lrwxrwxrwx. 1 root  root         20 Jan 20  2022 libtomcrypt.so.1 → libtomcrypt.so.1.0.1*
-rwxr-xr-x. 1 root  root     863136 Jan 20  2022 libtomcrypt.so.1.0.1*
[mydev@fedora lib64]$ _
```

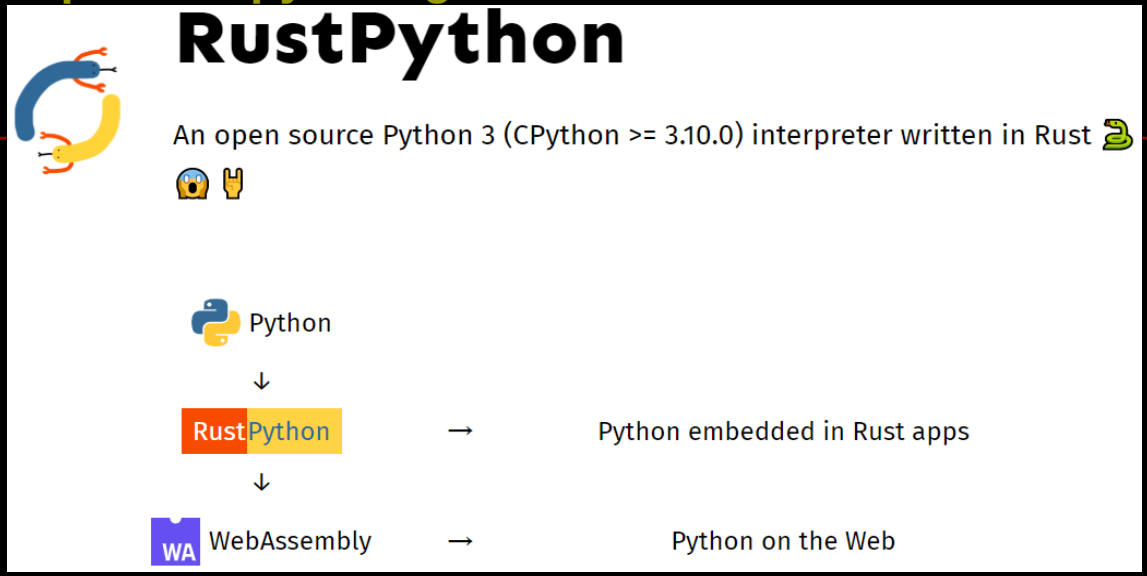
It works now!

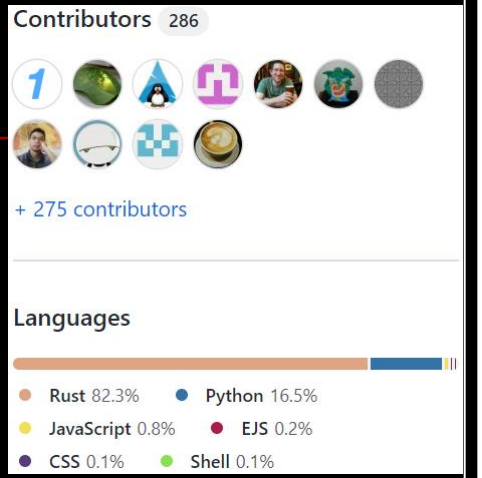
```
[mydev@fedora lib64]$ which python
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/bin/python
[mydev@fedora lib64]$
```

```
[mydev@fedora lib64]$ python -V
GraalVM Python 3.8.5 (GraalVM CE Native 22.3.0-dev)
[mydev@fedora lib64]$
[mydev@fedora lib64]$ which graalpy
/opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/bin/graalpy
[mydev@fedora lib64]$ graalpy -V
GraalVM Python 3.8.5 (GraalVM CE Native 22.3.0-dev)
[mydev@fedora lib64]$
[mydev@fedora lib64]$ ll /opt/MyWorkSpace/DevSW/Java/JDK/GraalVM/CE/v22.3.0-dev-20221004/Java19/languages/python/bin
total 264
drwxr-xr-x. 1 mydev mydev    40 Oct  5 09:16 ./
drwxr-xr-x. 1 mydev mydev   450 Oct  5 09:16 ../
-rwxr-xr-x. 1 mydev mydev 86176 Oct  5 09:16 graalpy*
-rwxr-xr-x. 1 mydev mydev 86176 Oct  5 09:16 python*
-rwxr-xr-x. 1 mydev mydev 86176 Oct  5 09:16 python3*
[mydev@fedora lib64]$
```

1.1.2 RustPython

- https://rustpython.github.io/





RustPython is a Python interpreter written in Rust. RustPython can be embedded into Rust programs to use Python as a scripting language for your application, or it can be compiled to WebAssembly in order to run Python in the browser. RustPython is free and open-source under the MIT license.

- Goals

Full Python 3 environment entirely in Rust (not CPython bindings), with a clean implementation and no compatiblity hacks.	Fast, reliable and secure implementation of Python that can be used from Rust or compiled to WebAssembly.

- **Benefits**

  Built-in support for any platform that support by Rust toolchain.

  Embedding:

  Interested in exposing Python scripting in an application written in Rust, perhaps to allow quickly tweaking logic where Rust's compile times would be inhibitive? Then `examples/hello_embed.rs` and `examples/mini_repl.rs` may be of some assistance.

WASI

You can compile RustPython to a standalone WebAssembly WASI module so it can run anywhere.

Build

```
$ cargo build --target wasm32-wasi --no-default-features --features freeze-stdlib,stdlib --release
```

Run by wasmer

```
$ wasmer run --dir . target/wasm32-wasi/release/rustpython.wasm extra_tests/snippets/stdlib_random.py
```

Run by wapm

```
$ wapm install rustpython
$ wapm run rustpython
>>>>> 2+2
4
```

Building the WASI file

You can build the WebAssembly WASI file with:

```
cargo build --release --target wasm32-wasi --features="freeze-stdlib"
```

> Note: we use the `freeze-stdlib` to include the standard library inside the binary. You also have to run once `rustup target add wasm32-wasi`.

JIT

RustPython has an **very** experimental JIT compiler that compile python functions into native code.

Building

By default the JIT compiler isn't enabled, it's enabled with the `jit` cargo feature.

```
$ cargo run --features jit
```

This requires autoconf, automake, libtool, and clang to be installed.

Using

To compile a function, call `__jit__()` on it.

```
def foo():
    a = 5
    return 10 + a

foo.__jit__()  # this will compile foo to native code and subsequent calls will execute that native code
assert foo() == 15
```

Build and test on RPi4

- **Dev env**

```
[mydev@fedora RustPython-main]$ cat ~/.cargo/config
[source.crates-io]
registry = "https://github.com/rust-lang/crates.io-index"
replace-with = 'ustc'

[source.ustc]
registry = "https://mirrors.ustc.edu.cn/crates.io-index"

[source.sjtu]
registry = "https://mirrors.sjtug.sjtu.edu.cn/git/crates.io-index/"

[source.tuna]
registry = "https://mirrors.tuna.tsinghua.edu.cn/git/crates.io-index.git"

[source.rustcc]
registry = "https://code.aliyun.com/rustcc/crates.io-index.git"
[mydev@fedora RustPython-main]$
```

```
[mydev@fedora RustPython-main]$ rustup -V
rustup 1.25.1 (bb60b1e89 2022-07-12)
info: This is the version for the rustup toolchain manager, not the rustc compiler.
info: The currently active `rustc` version is `rustc 1.64.0 (a55dd71d5 2022-09-19)`
[mydev@fedora RustPython-main]$
[mydev@fedora RustPython-main]$ rustup toolchain list
stable-aarch64-unknown-linux-gnu (default) (override)
[mydev@fedora RustPython-main]$
```

- **Build**

On branch main, last commit: 2a26ed00964b900d247d6a8478d8cd24b13cc096

```
[mydev@fedora RustPython-main]$ cargo build --release
    Updating `ustc` index
    Updating git repository `https://github.com/RustPython/__doc__`
  Downloaded ahash v0.7.6 (registry `ustc`)
  Downloaded log v0.4.16 (registry `ustc`)
```

```
...
  Downloaded sha3 v0.10.1 (registry `ustc`)
  Downloaded 64 crates (5.5 MB) in 4.93s (largest was `libz-sys` at 1.5 MB)
   Compiling autocfg v1.1.0
   Compiling unicode-xid v0.2.2
...
   Compiling toml v0.5.8
   Compiling rustpython-common v0.0.0 (/opt/MyWorkSpace/MyProjs/Languages/Python/Impl/Rust/RustPython/RustPython-main/commo
n)
   Compiling rustpython-compiler-core v0.1.2 (/opt/MyWorkSpace/MyProjs/Languages/Python/Impl/Rust/RustPython/RustPython-mai
n/compiler/core)
   Compiling proc-macro-crate v1.1.0
   Compiling num_enum_derive v0.5.7
   Compiling rustpython-ast v0.1.0 (/opt/MyWorkSpace/MyProjs/Languages/Python/Impl/Rust/RustPython/RustPython-main/compiler
/ast)
   Compiling rustpython-codegen v0.1.2 (/opt/MyWorkSpace/MyProjs/Languages/Python/Impl/Rust/RustPython/RustPython-main/comp
iler/codegen)
   Compiling num_enum v0.5.7
   Compiling sre-engine v0.4.1
   Compiling rustpython-parser v0.1.2 (/opt/MyWorkSpace/MyProjs/Languages/Python/Impl/Rust/RustPython/RustPython-main/compi
ler/parser)
   Compiling rustpython-compiler v0.1.2 (/opt/MyWorkSpace/MyProjs/Languages/Python/Impl/Rust/RustPython/RustPython-main/com
piler)
   Compiling rustpython-derive v0.1.2 (/opt/MyWorkSpace/MyProjs/Languages/Python/Impl/Rust/RustPython/RustPython-main/deriv
e)
   Compiling rustpython v0.1.2 (/opt/MyWorkSpace/MyProjs/Languages/Python/Impl/Rust/RustPython/RustPython-main)
    Finished release [optimized] target(s) in 20m 15s
[mydev@fedora RustPython-main]$
```

```
[mydev@fedora RustPython-main]$ ll target/release/
total 22764
drwxr-xr-x. 1 mydev mydev      212 Sep 24 12:59 ./
drwxr-xr-x. 1 mydev mydev       70 Oct 16  2021 ../
drwxr-xr-x. 1 mydev mydev     8680 Sep 24 12:39 build/
-rw-r--r--. 1 mydev mydev        0 Oct 16  2021 .cargo-lock
drwxr-xr-x. 1 mydev mydev    97948 Sep 24 12:59 deps/
drwxr-xr-x. 1 mydev mydev        0 Oct 16  2021 examples/
drwxr-xr-x. 1 mydev mydev    35592 Sep 24 12:39 .fingerprint/
drwxr-xr-x. 1 mydev mydev        0 Oct 16  2021 incremental/
-rw-r--r--. 1 mydev mydev    25560 Sep 24 12:59 librustpython.d
-rw-r--r--. 2 mydev mydev  1502094 Sep 24 12:50 librustpython.rlib
-rwxr-xr-x. 2 mydev mydev 21746256 Sep 24 12:59 rustpython*
-rw-r--r--. 1 mydev mydev    25643 Sep 24 12:59 rustpython.d
[mydev@fedora RustPython-main]$
[mydev@fedora RustPython-main]$ file target/release/rustpython
target/release/rustpython: ELF 64-bit LSB pie executable, ARM aarch64, version 1 (SYSV), dynamically linked, interpreter /l
ib/ld-linux-aarch64.so.1, BuildID[sha1]=bc1655a75608255e705fd4dc11eee17a3dd91c9b, for GNU/Linux 3.7.0, with debug_info, not
 stripped
[mydev@fedora RustPython-main]$
```

Test

```
[mydev@fedora RustPython-main]$ target/release/rustpython --help
RustPython 0.1.2
RustPython Team
Rust implementation of the Python language

USAGE:
    rustpython [OPTIONS] [-c CMD | -m MODULE | FILE] [PYARGS]...

FLAGS:
    -b              issue warnings about using bytes where strings are usually expected (-bb: issue errors)
    -d              Debug the parser.
    -B              don't write .pyc files on import
    -h, --help      Prints help information
    -E              Ignore environment variables PYTHON* such as PYTHONPATH
    -i              Inspect interactively after running the script.
    -I              isolate Python from the user's environment (implies -E and -s)
    -S              don't imply 'import site' on initialization
    -s              don't add user site directory to sys.path.
    -O              Optimize. Set __debug__ to false. Remove debug statements.
    -q              Be quiet at startup.
    -u              force the stdout and stderr streams to be unbuffered; this option has no effect on stdin; also
                    PYTHONUNBUFFERED=x
    -V, --version   Prints version information
    -v              Give the verbosity (can be applied multiple times)

OPTIONS:
    -c <cmd, args>...                                   run the given string as a program
        --check-hash-based-pycs <check-hash-based-pycs>
            always|default|never
            control how Python invalidates hash-based .pyc files [default: default]
    -X <implementation-option>...                       set implementation-specific option
        --install-pip <get-pip args>...
            install the pip package manager for rustpython; requires rustpython be build with the ssl feature enabled.

    -m <module, args>...                                run library module as script
    -W <warning-control>...                             warning control; arg is action:message:category:module:lineno

ARGS:
    <script, args>...
[mydev@fedora RustPython-main]$
```

```
running 1 test
Hello
Hello
test tests::test_run_script ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.44s

     Running unittests src/main.rs (target/debug/deps/rustpython-38f3c0b7cbdf3a30)

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

   Doc-tests rustpython

running 1 test
test src/lib.rs - (line 8) - compile ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.37s

[mydev@fedora RustPython-main]$
```

```
[mydev@fedora RustPython-main]$ target/release/rustpython -V
RustPython 0.1.2
[mydev@fedora RustPython-main]$ target/release/rustpython
Welcome to the magnificent Rust Python 0.1.2 interpreter
>>>>> 2+2
4
>>>>> print("Hello, world!")
Hello, world!
>>>>>
```

2) Accelerating Messaging and RPC

Goal

- A lightweight, efficient and HW-SW co-designed approach is preferred.

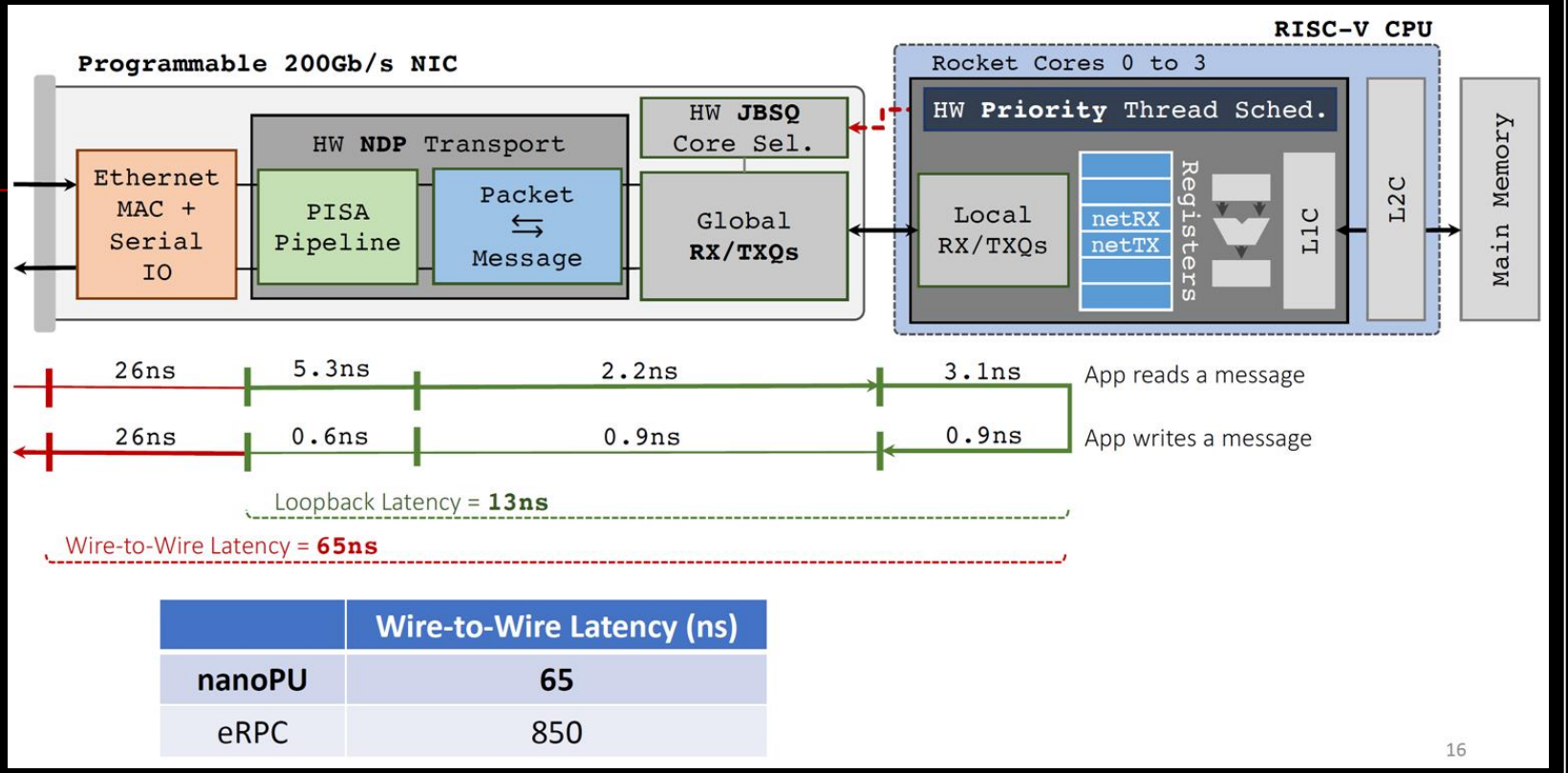
2.1 Customized Network Stack
2.1.1 nanoPU

- A nanosecond network stack for Datacenters
- Previous approaches to minimize RPC latency and software overheads

Approach	Limitation	Wire-to-Wire Latency	RPC Throughput
Dataplane operating systems (e.g. Shinjuku, Shenango)	Too coarse grained	Median: ~2-5µs Tail: 10-100µs	<10Mrps
Efficient RPC software libraries (e.g. eRPC)	Neglect tail latency optimizations	Median: 850ns Tail: 10-100µs	~10Mrps / core
Transport protocol offload (e.g. Tonic)	Only part of the solution	N/A	>100Mrps
RDMA NICs	Need low latency to remote *compute*, not *memory*	Median: ~700ns	N/A
Integrated NICs (e.g. NeBuLa)	Still room for improvement of latency and throughput	Median: ~100ns Tail: ~2-5µs	~20Mrps / core

Source: "The nanoPU: A Nanosecond Network Stack for Datacenters", Stephen Ibanez etc, OSDI 2021.

Microbenchmarks



	Wire-to-Wire Latency (ns)
nanoPU	**65**
eRPC	850

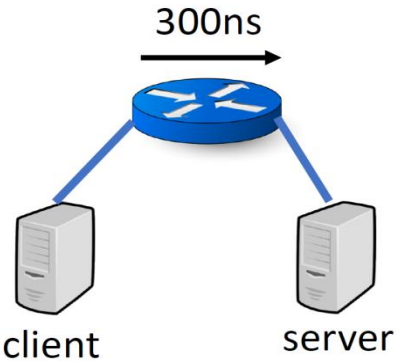
16

Source: "The nanoPU: A Nanosecond Network Stack for Datacenters", Stephen Ibanez etc, OSDI 2021

Comparison

State-of-the-art RDMA NIC

- Implemented in NIC HW
- End-to-end latency: ~2µs



The nanoPU

- Implemented in SW
- Can support arbitrary one-sided operations

One-sided RDMA	Latency (ns)	
	Median	90th %ile
Read	678	680
Write	679	686
Compare-and-Swap	687	690
Fetch-and-Add	688	692
Indirect Read	691	715

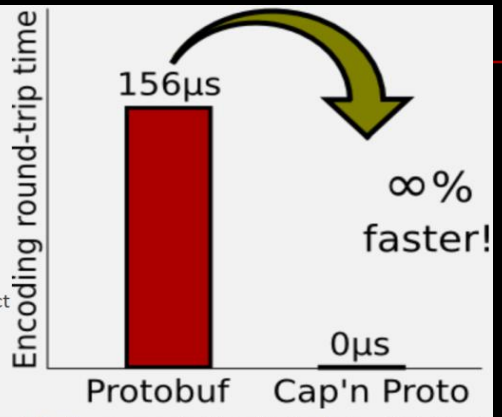
Source: "The nanoPU: A Nanosecond Network Stack for Datacenters", Stephen Ibanez etc, OSDI 2021

2.2 Lightweight Messaging & RPC
2.2.1 Cap'n Proto

- https://capnproto.org/

Cap'n Proto is an insanely fast data interchange format and capability-based RPC system. Think JSON, except binary. Or think Protocol Buffers, except faster. In fact, in benchmarks, Cap'n Proto is INFINITY TIMES faster than Protocol Buffers.

This benchmark is, of course, unfair. It is only measuring the time to encode and decode a message in memory. Cap'n Proto gets a perfect score because *there is no encoding/decoding step*. The Cap'n Proto encoding is appropriate both as a data interchange format and an in-memory representation, so once your structure is built, you can simply write the bytes straight out to disk!



- Features

But doesn't that mean the encoding is platform-specific?

NO! The encoding is defined byte-for-byte independent of any platform. However, it is designed to be efficiently manipulated on common modern CPUs. Data is arranged like a compiler would arrange a struct – with fixed widths, fixed offsets, and proper alignment. Variable-sized elements are embedded as pointers. Pointers are offset-based rather than absolute so that messages are position-independent. Integers use little-endian byte order because most CPUs are little-endian, and even big-endian CPUs usually have instructions for reading little-endian data.

Doesn't that make backwards-compatibility hard?

Not at all! New fields are always added to the end of a struct (or replace padding space), so existing field positions are unchanged. The recipient simply needs to do a bounds check when reading each field. Fields are numbered in the order in which they were added, so Cap'n Proto always knows how to arrange them for backwards-compatibility.

Won't fixed-width integers, unset optional fields, and padding waste space on the wire?

Yes. However, since all these extra bytes are zeros, when bandwidth matters, we can apply an extremely fast Cap'n-Proto-specific compression scheme to remove them. Cap'n Proto calls this "packing" the message; it achieves similar (better, even) message sizes to protobuf encoding, and it's still faster.

When bandwidth really matters, you should apply general-purpose compression, like zlib or LZ4, regardless of your encoding format.

Isn't this all horribly insecure?

No no no! To be clear, we're NOT just casting a buffer pointer to a struct pointer and calling it a day.

Cap'n Proto generates classes with accessor methods that you use to traverse the message. These accessors validate pointers before following them. If a pointer is invalid (e.g. out-of-bounds), the library can throw an exception or simply replace the value with a default / empty object (your choice).

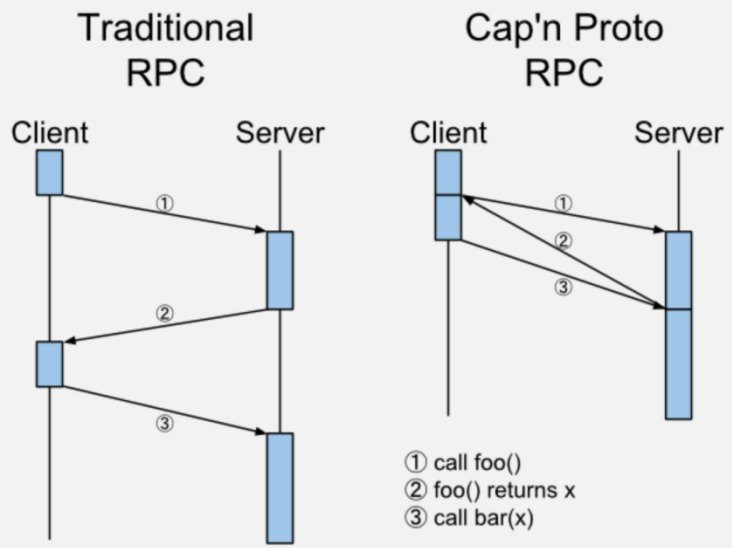
Thus, Cap'n Proto checks the structural integrity of the message just like any other serialization protocol would. And, just like any other protocol, it is up to the app to check the validity of the content.

Cap'n Proto was built to be used in Sandstorm.io, and is now heavily used in Cloudflare Workers, two environments where security is a major concern. Cap'n Proto has undergone fuzzing and expert security review. Our response to security issues was once described by security guru Ben Laurie as "the most awesome response I've ever had." (Please report all security issues to kenton@cloudflare.com.)

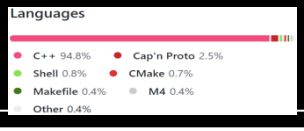
Other advantages

- **Incremental reads:** It is easy to start processing a Cap'n Proto message before you have received all of it since outer objects appear entirely before inner objects (as opposed to most encodings, where outer objects encompass inner objects).
- **Random access:** You can read just one field of a message without parsing the whole thing.
- **mmap:** Read a large Cap'n Proto file by memory-mapping it. The OS won't even read in the parts that you don't access.
- **Inter-language communication:** Calling C++ code from, say, Java or Python tends to be painful or slow. With Cap'n Proto, the two languages can easily operate on the same in-memory data structure.

- **Arena allocation:** Manipulating Protobuf objects tends to be bogged down by memory allocation, unless you are very careful about object reuse. Cap'n Proto objects are always allocated in an "arena" or "region" style, which is faster and promotes cache locality.
- **Tiny generated code:** Protobuf generates dedicated parsing and serialization code for every message type, and this code tends to be enormous. Cap'n Proto generated code is smaller by an order of magnitude or more. In fact, usually it's no more than some inline accessor methods!
- **Tiny runtime library:** Due to the simplicity of the Cap'n Proto format, the runtime library can be much smaller.
- **Time-traveling RPC:** Cap'n Proto features an RPC system that implements time travel such that call results are returned to the client before the request even arrives at the server!



- **RPC**
  https://capnproto.org/rpc.html
- **Src**
  https://github.com/capnproto/capnproto



2.3 Hardware-assisted

Overview

- …

2.3.1 Cost-effective SmartNIC for Edge Computing

Overview

- …

2.3.1.1 Our scheme

Current design

- A FPGA-based SmartNIC that fits inside the USB port;
- At least two interfaces:

  a USB port for host connection, and a RJ45 or Mini I/O connector for networking;
- The main logic in FPGA is to implement the desired functionalities of the SmartNIC, the conversion of I/O signals, and so on;
- Support for USB 4.0 is mostly preferred(so the host must also support USB 4.0).
- …

The main reason that this design do not use PCIe interfaced SmartNIC is base on the consideration of add more flexibility to our overall solution for Edge Computing.

2.4 HW-SW Co-design
2.4.1 nanoPU

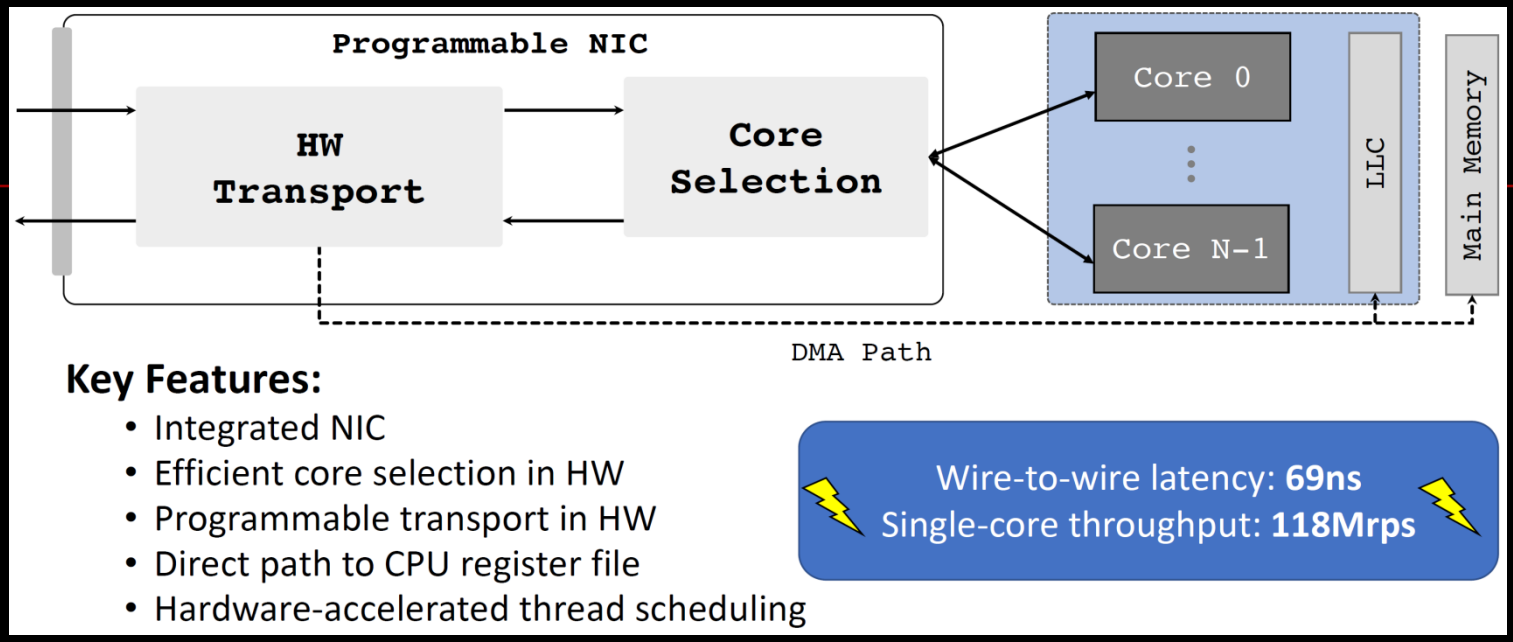
Overview

- **A nanosecond network stack for Datacenters**
- **Previous approaches to minimize RPC latency and software overheads**

Approach	Limitation	Wire-to-Wire Latency	RPC Throughput
Dataplane operating systems (e.g. Shinjuku, Shenango)	Too coarse grained	Median: ~2-5µs Tail: 10-100µs	<10Mrps
Efficient RPC software libraries (e.g. eRPC)	Neglect tail latency optimizations	Median: 850ns Tail: 10-100µs	~10Mrps / core
Transport protocol offload (e.g. Tonic)	Only part of the solution	N/A	>100Mrps
RDMA NICs	Need low latency to remote *compute*, not *memory*	Median: ~700ns	N/A
Integrated NICs (e.g. NeBuLa)	Still room for improvement of latency and throughput	Median: ~100ns Tail: ~2-5µs	~20Mrps / core

Source: "The nanoPU: A Nanosecond Network Stack for Datacenters", Stephen Ibanez etc, OSDI 2021.

Architecture & design



Key Features:

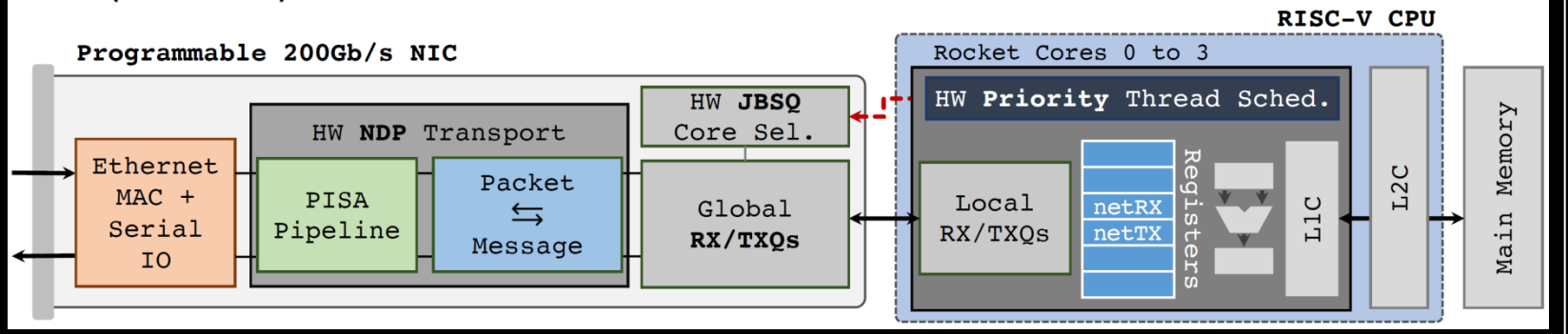
- Integrated NIC
- Efficient core selection in HW
- Programmable transport in HW
- Direct path to CPU register file
- Hardware-accelerated thread scheduling

Wire-to-wire latency: **69ns**
Single-core throughput: **118Mrps**

Source: "The nanoPU: A Nanosecond Network Stack for Datacenters", Stephen Ibanez etc, OSDI 2021

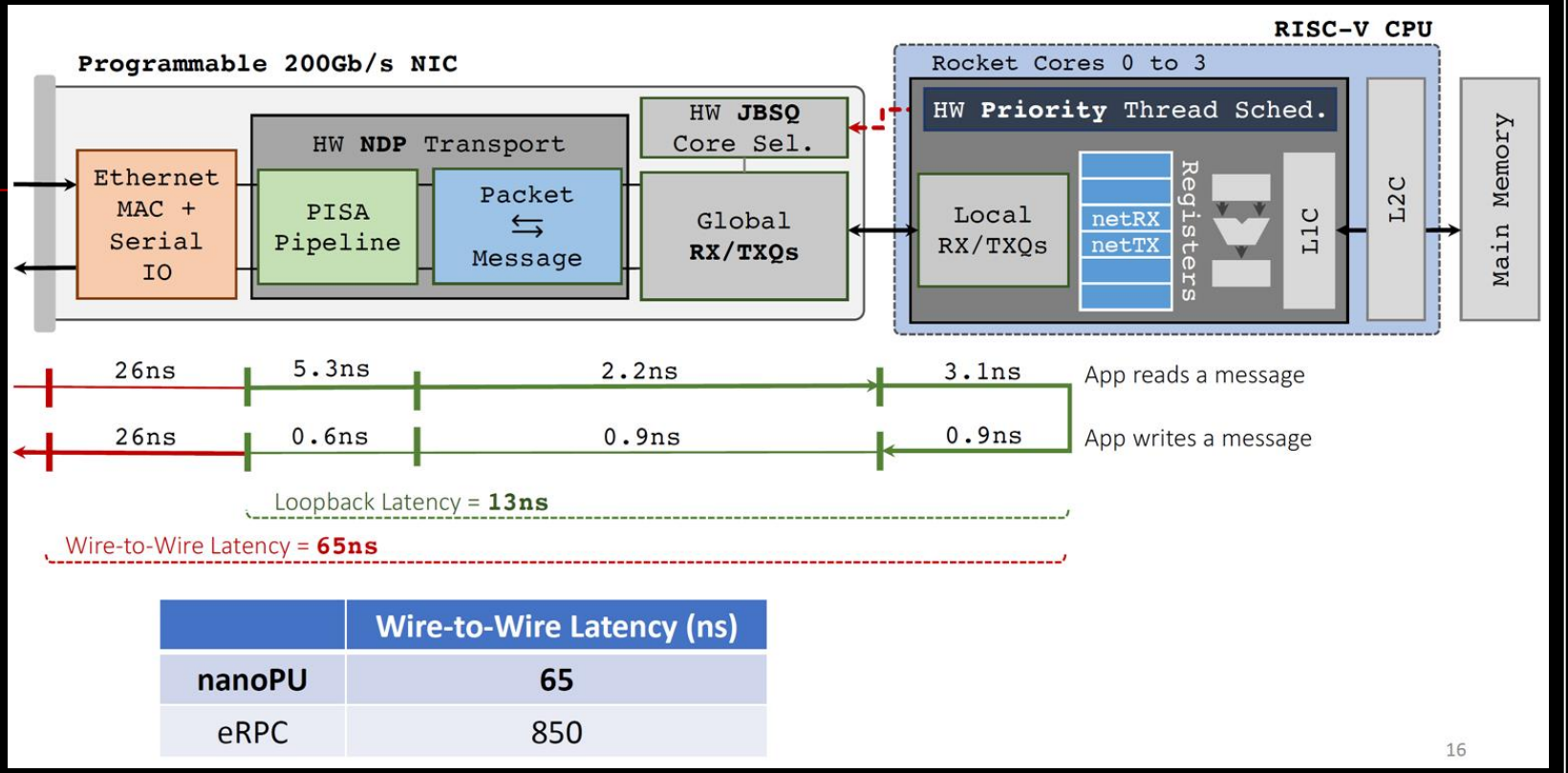
Prototyping

- Quad-core nanoPU based on open source RISC-V Rocket core
- 4,300 lines of Chisel code & 1,200 lines of C and RISC-V assembly for custom *nano*kernel
- Implements NDP transport
- Cycle-accurate simulations (3.2GHz) on AWS FPGAs using Firesim (ISCA '18)



Source: "The nanoPU: A Nanosecond Network Stack for Datacenters", Stephen Ibanez etc, OSDI 2021

Microbenchmarks



	Wire-to-Wire Latency (ns)
nanoPU	**65**
eRPC	850

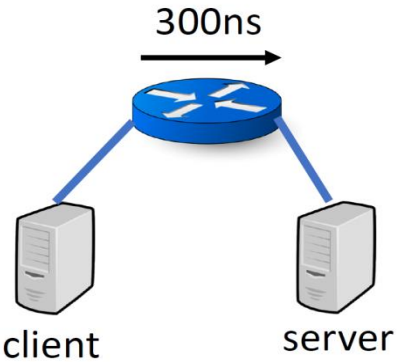
16

Source: "The nanoPU: A Nanosecond Network Stack for Datacenters", Stephen Ibanez etc, OSDI 2021

Comparison

State-of-the-art RDMA NIC

- Implemented in NIC HW
- End-to-end latency: ~2µs



The nanoPU

- Implemented in SW
- Can support arbitrary one-sided operations

One-sided RDMA	Latency (ns)	
	Median	90th %ile
Read	678	680
Write	679	686
Compare-and-Swap	687	690
Fetch-and-Add	688	692
Indirect Read	691	715

Source: "The nanoPU: A Nanosecond Network Stack for Datacenters", Stephen Ibanez etc, OSDI 2021

2.4.2 eBPF-based Novel Messaging and RPC

- For more details, you may look forward to our upcoming technical reports like "The fourth round discussion on eBPF-centric new approach for Hyper-Converged Infrastructure & Edge Computing" and "The third round discussion on GraalVM-based unified runtime for eBPF and Wasm".

V. Ray.Rust

1) Rust-based alternatives for Ray's dependencies

1.1 Dependencies of Ray

- https://github.com/ray-project/ray/blob/master/python/requirements.txt
  pyarrow…
- https://github.com/ray-project/ray/blob/master/bazel/ray_deps_setup.bzl
  Protobuf, Redis, Hiredis, Spdlog, Boost, Flatbuffers, Googletest, Gflags, Cython, OpenCensus, Abseil(C++), Prometheus(C++), gRPC, OpenSSL, MessagePack(C/C++), JSON(C++), RapidJSON, and many Bazel related rules and tools.
- …

1.2 Rust alternatives for Arrow

1.2.1 arrow-rs

Overview

- https://github.com/apache/arrow-rs

Official Rust implementation of Apache Arrow.

This repo contains the following main components:

Crate	Description	Documentation
arrow	Core functionality (memory layout, arrays, low level computations)	(README)
parquet	Support for Parquet columnar file format	(README)
arrow-flight	Support for Arrow-Flight IPC protocol	(README)

There are two related crates in a different repository

Crate	Description	Documentation
DataFusion	In-memory query engine with SQL support	(README)
Ballista	Distributed query execution	(README)

Collectively, these crates support a vast array of functionality for analytic computations in Rust.

For example, you can write an SQL query or a `DataFrame` (using the `datafusion` crate), run it against a parquet file (using the `parquet` crate), evaluate it in-memory using Arrow's columnar format (using the `arrow` crate), and send to another process (using the `arrow-flight` crate).

Generally speaking, the `arrow` crate offers functionality for using Arrow arrays, and `datafusion` offers most operations typically found in SQL, including `join`s and window functions.

You can find more details about each crate in their respective READMEs.

- ...

1.2.2 Arrow2

- https://github.com/jorgecarleitao/arrow2/

  Unofficial transmute-free Rust library to work with the Arrow format.

- Features

  - Most feature-complete implementation of Apache Arrow after the reference implementation (C++)
    - Float 16 unsupported (not a Rust native type)
    - Decimal 256 unsupported (not a Rust native type)
  - C data interface supported for all Arrow types (read and write)
  - C stream interface supported for all Arrow types (read and write)
  - Full interoperability with Rust's `Vec`
  - MutableArray API to work with bitmaps and arrays in-place
  - Full support for timestamps with timezones, including arithmetics that take timezones into account
  - Support to read from, and write to:
    - CSV
    - Apache Arrow IPC (all types)
    - Apache Arrow Flight (all types)
    - Apache Parquet (except deep nested types)
    - Apache Avro (all types)
    - NJSON
    - ODBC (some types)
  - Extensive suite of compute operations
    - aggregations
    - arithmetics
    - cast
    - comparison
    - sort and merge-sort
    - boolean (AND, OR, etc) and boolean kleene
    - filter, take
    - hash
    - if-then-else
    - nullif
    - temporal (day, month, week day, hour, etc.)
    - window
    - ... and more ...
  - Extensive set of cargo feature flags to reduce compilation time and binary size
  - Fully-decoupled IO between CPU-bounded and IO-bounded tasks, allowing this crate to both be used in `async` contexts without blocking and leverage parallelism
  - Fastest known implementation of Avro and Parquet (e.g. faster than the official C++ implementations)

- ...

Integration

- https://github.com/apache/arrow-datafusion/issues/1532

Is your feature request related to a problem or challenge? Please describe what you are trying to do.
Datafusion currently relies on the https://github.com/apache/arrow-rs implementation of Apache Arrow. This also means any project that is built on DataFusion is likely to end up using that implementation as well

There has been various talk / discussion / work on switching to arrow2 - https://github.com/jorgecarleitao/arrow2 from **@jorgecarleitao**

Describe the solution you'd like
A consensus on if we want to switch datafusion to using arrow2

Additional context

- Arrow2 milestone; https://github.com/apache/arrow-datafusion/milestone/3
- PR with more discussion on this issue: #68
- Code from **@houqp** and **@yjshen** https://github.com/houqp/arrow-datafusion/tree/arrow2_merge
- **@Igosuki** 's PR: https://github.com/Igosuki/arrow-datafusion/tree/arrow22r
- Roadmap: https://github.com/apache/arrow-datafusion/blob/0b8bffd6410ecdcfa29788f75fbc5ca15242a239/docs/source/specification/roadmap.md#runtime--infrastructure

- ...

1.2.3 DataFusion

- https://arrow.apache.org/datafusion/

  Apache Arrow DataFusion and Ballista query engines.

- https://github.com/apache/arrow-datafusion/

DataFusion is an extensible query execution framework, written in Rust, that uses Apache Arrow as its in-memory format.

DataFusion supports both an SQL and a DataFrame API for building logical query plans as well as a query optimizer and execution engine capable of parallel execution against partitioned data sources (CSV and Parquet) using threads.

DataFusion also supports distributed query execution via the Ballista crate.

Use Cases

DataFusion is used to create modern, fast and efficient data pipelines, ETL processes, and database systems, which need the performance of Rust and Apache Arrow and want to provide their users the convenience of an SQL interface or a DataFrame API.

Why DataFusion?

- *High Performance*: Leveraging Rust and Arrow's memory model, DataFusion achieves very high performance
- *Easy to Connect*: Being part of the Apache Arrow ecosystem (Arrow, Parquet and Flight), DataFusion works well with the rest of the big data ecosystem
- *Easy to Embed*: Allowing extension at almost any point in its design, DataFusion can be tailored for your specific usecase
- *High Quality*: Extensively tested, both by itself and with the rest of the Arrow ecosystem, DataFusion can be used as the foundation for production systems.

- https://arrow.apache.org/blog/2022/02/28/datafusion-7.0.0/
- ...

Ballista: Distributed Compute with Apache Arrow and DataFusion

https://github.com/apache/arrow-datafusion/blob/master/ballista/README.md

Ballista is a distributed compute platform primarily implemented in Rust, and powered by Apache Arrow and DataFusion. It is built on an architecture that allows other programming languages (such as Python, C++, and Java) to be supported as first-class citizens without paying a penalty for serialization costs.

The foundational technologies in Ballista are:

- Apache Arrow memory model and compute kernels for efficient processing of data.
- Apache Arrow Flight Protocol for efficient data transfer between processes.
- Google Protocol Buffers for serializing query plans.
- Docker for packaging up executors along with user-defined code.

Ballista can be deployed as a standalone cluster and also supports Kubernetes. In either case, the scheduler can be configured to use etcd as a backing store to (eventually) provide redundancy in the case of a scheduler failing.

Distributed Scheduler Overview

Ballista uses the DataFusion query execution framework to create a physical plan and then transforms it into a distributed physical plan by breaking the query down into stages whenever the partitioning scheme changes.

Specifically, any `RepartitionExec` operator is replaced with an `UnresolvedShuffleExec` and the child operator of the repartition operator is wrapped in a `ShuffleWriterExec` operator and scheduled for execution.

Each executor polls the scheduler for the next task to run. Tasks are currently always `ShuffleWriterExec` operators and each task represents one *input* partition that will be executed. The resulting batches are repartitioned according to the shuffle partitioning scheme and each *output* partition is streamed to disk in Arrow IPC format.

The scheduler will replace `UnresolvedShuffleExec` operators with `ShuffleReaderExec` operators once all shuffle tasks have completed. The `ShuffleReaderExec` operator connects to other executors as required using the Flight interface, and streams the shuffle IPC files.

How does this compare to Apache Spark?

Ballista implements a similar design to Apache Spark, but there are some key differences.

- The choice of Rust as the main execution language means that memory usage is deterministic and avoids the overhead of GC pauses.
- Ballista is designed from the ground up to use columnar data, enabling a number of efficiencies such as vectorized processing (SIMD and GPU) and efficient compression. Although Spark does have some columnar support, it is still largely row-based today.
- The combination of Rust and Arrow provides excellent memory efficiency and memory usage can be 5x - 10x lower than Apache Spark in some cases, which means that more processing can fit on a single node, reducing the overhead of distributed compute.
- The use of Apache Arrow as the memory model and network protocol means that data can be exchanged between executors in any programming language with minimal serialization overhead.

1.2.4 Polars

- https://www.pola.rs/

Lightning-fast DataFrame library for Rust and Python.

Familiar from the start

Knowing of data wrangling habits, Polars exposes a complete Python API, including the full set of features to manipulate DataFrames using an expression language that will empower you to create readable and performant code.

DataFrames to the Rust ecosystem

Polars is written in Rust, uncompromising in its choices to provide a feature-complete DataFrame API to the Rust ecosystem. Use it as a DataFrame library or as query engine backend for your data models.

On the shoulders of a giant

Polars is built upon the safe Arrow2 implementation of the Apache Arrow specification, enabling efficient resource use and processing performance. By doing so it also integrates seamlessly with other tools in the Arrow ecosystem.

- https://github.com/pola-rs/polars

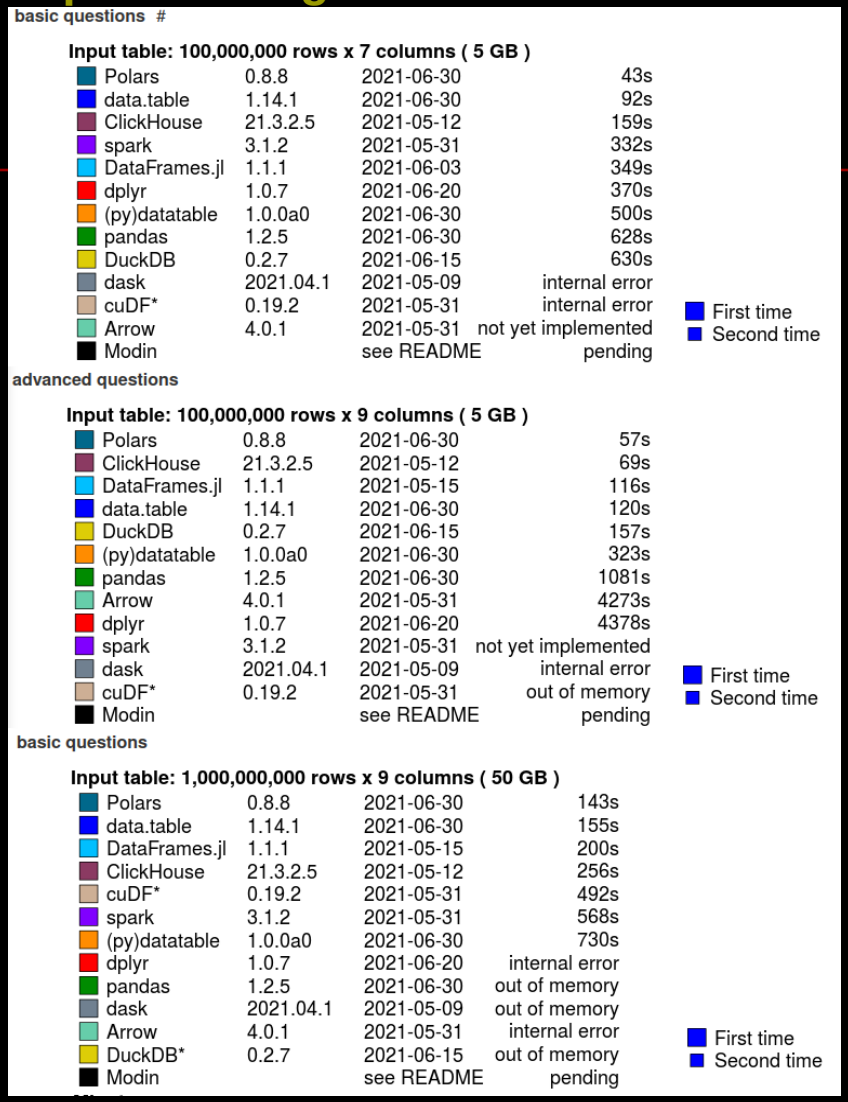
Blazingly fast DataFrames in Rust, Python & Node.js

Polars is a blazingly fast DataFrames library implemented in Rust using Apache Arrow Columnar Format as memory model.

- Lazy | eager execution
- Multi-threaded
- SIMD
- Query optimization
- Powerful expression API
- Rust | Python | ...

Performance

- https://h2oai.github.io/db-benchmark/



basic questions #

Input table: 100,000,000 rows x 7 columns (5 GB)

Polars	0.8.8	2021-06-30	43s
data.table	1.14.1	2021-06-30	92s
ClickHouse	21.3.2.5	2021-05-12	159s
spark	3.1.2	2021-05-31	332s
DataFrames.jl	1.1.1	2021-06-03	349s
dplyr	1.0.7	2021-06-20	370s
(py)datatable	1.0.0a0	2021-06-30	500s
pandas	1.2.5	2021-06-30	628s
DuckDB	0.2.7	2021-06-15	630s
dask	2021.04.1	2021-05-09	internal error
cuDF*	0.19.2	2021-05-31	internal error
Arrow	4.0.1	2021-05-31	not yet implemented
Modin		see README	pending

First time
Second time

advanced questions

Input table: 100,000,000 rows x 9 columns (5 GB)

Polars	0.8.8	2021-06-30	57s
ClickHouse	21.3.2.5	2021-05-12	69s
DataFrames.jl	1.1.1	2021-05-15	116s
data.table	1.14.1	2021-06-30	120s
DuckDB	0.2.7	2021-06-15	157s
(py)datatable	1.0.0a0	2021-06-30	323s
pandas	1.2.5	2021-06-30	1081s
Arrow	4.0.1	2021-05-31	4273s
dplyr	1.0.7	2021-06-20	4378s
spark	3.1.2	2021-05-31	not yet implemented
dask	2021.04.1	2021-05-09	internal error
cuDF*	0.19.2	2021-05-31	out of memory
Modin		see README	pending

First time
Second time

basic questions

Input table: 1,000,000,000 rows x 9 columns (50 GB)

Polars	0.8.8	2021-06-30	143s
data.table	1.14.1	2021-06-30	155s
DataFrames.jl	1.1.1	2021-05-15	200s
ClickHouse	21.3.2.5	2021-05-12	256s
cuDF*	0.19.2	2021-05-31	492s
spark	3.1.2	2021-05-31	568s
(py)datatable	1.0.0a0	2021-06-30	730s
dplyr	1.0.7	2021-06-20	internal error
pandas	1.2.5	2021-06-30	out of memory
dask	2021.04.1	2021-05-09	out of memory
Arrow	4.0.1	2021-05-31	internal error
DuckDB*	0.2.7	2021-06-15	out of memory
Modin		see README	pending

First time
Second time

- ...

1.3 Serialization in Rust

1.3.1 Serde

- https://serde.rs/

  A framework for *ser*ializing and *de*serializing Rust data structures efficiently and generically.

  The Serde ecosystem consists of data structures that know how to serialize and deserialize themselves along with data formats that know how to serialize and deserialize other things. Serde provides the layer by which these two groups interact with each other, allowing any supported data structure to be serialized and deserialized using any supported data format.

- **Design**

  Where many other languages rely on runtime reflection for serializing data, Serde is instead built on Rust's powerful trait system. A data structure that knows how to serialize and deserialize itself is one that implements Serde's `Serialize` and `Deserialize` traits (or uses Serde's derive attribute to automatically generate implementations at compile time). This avoids any overhead of reflection or runtime type information. In fact in many situations the interaction between data structure and data format can be completely optimized away by the Rust compiler, leaving Serde serialization to perform the same speed as a handwritten serializer for the specific selection of data structure and data format.

- https://github.com/serde-rs/serde

  Languages

  Rust 100.0%

Data formats

The following is a partial list of data formats that have been implemented for Serde by the community.

- JSON, the ubiquitous JavaScript Object Notation used by many HTTP APIs.
- Postcard, a no_std and embedded-systems friendly compact binary format.
- CBOR, a Concise Binary Object Representation designed for small message size without the need for version negotiation.
- YAML, a self-proclaimed human-friendly configuration language that ain't markup language.
- MessagePack, an efficient binary format that resembles a compact JSON.
- TOML, a minimal configuration format used by Cargo.
- Pickle, a format common in the Python world.
- RON, a Rusty Object Notation.
- BSON, the data storage and network transfer format used by MongoDB.
- Avro, a binary format used within Apache Hadoop, with support for schema definition.
- JSON5, a superset of JSON including some productions from ES5.
- URL query strings, in the x-www-form-urlencoded format.
- Envy, a way to deserialize environment variables into Rust structs. *(deserialization only)*
- Envy Store, a way to deserialize AWS Parameter Store parameters into Rust structs. *(deserialization only)*
- S-expressions, the textual representation of code and data used by the Lisp language family.
- D-Bus's binary wire format.
- FlexBuffers, the schemaless cousin of Google's FlatBuffers zero-copy serialization format.
- Bencode, a simple binary format used in the BitTorrent protocol.
- DynamoDB Items, the format used by rusoto_dynamodb to transfer data to and from DynamoDB.
- Hjson, a syntax extension to JSON designed around human reading and editing. *(deserialization only)*

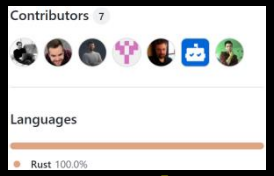
1.3.1.1 Pythonize

Overview

- https://github.com/davidhewitt/pythonize

This is an experimental serializer for Rust's serde ecosystem, which can convert Rust objects to Python values and back.

At the moment the Python structures it produces should be *very* similar to those which are produced by `serde_json`; i.e. calling Python's `json.loads()` on a value encoded by `serde_json` should produce an identical structure to that which is produced directly by `pythonize`.



Usage

This crate converts Rust types which implement the Serde serialization traits into Python objects using the PyO3 library.

Pythonize has two public APIs: `pythonize` and `depythonize`.

- Example

```rust
use serde::{Serialize, Deserialize};
use pyo3::Python;
use pythonize::{depythonize, pythonize};

#[derive(Debug, Serialize, Deserialize, PartialEq)]
struct Sample {
    foo: String,
    bar: Option<usize>
}

let gil = Python::acquire_gil();
let py = gil.python();

let sample = Sample {
    foo: "Foo".to_string(),
    bar: None
};

// Rust -> Python
let obj = pythonize(py, &sample).unwrap();

assert_eq!("{'foo': 'Foo', 'bar': None}", &format!("{}", obj.as_ref(py).repr().unwrap()));

// Python -> Rust
let new_sample: Sample = depythonize(obj.as_ref(py)).unwrap();

assert_eq!(new_sample, sample);
```

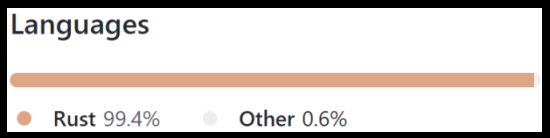
1.4 Rust-based alternatives for gRPC

1.4.1 Tonic

- https://github.com/hyperium/tonic

A rust implementation of gRPC, a high performance, open source, general RPC framework that puts mobile and HTTP/2 first.

`tonic` is a gRPC over HTTP/2 implementation focused on high performance, interoperability, and flexibility. This library was created to have first class support of async/await and to act as a core building block for production systems written in Rust.



- **Features**

- Bi-directional streaming
- High performance async io
- Interoperability
- TLS backed by `rustls`
- Load balancing
- Custom metadata
- Authentication
- Health Checking

1.4.2 Volo

Overview

- https://github.com/cloudwego/volo

  A high-performance and strong-extensibility Rust RPC framework that helps developers build microservices.

- Crates

Volo mainly consists of six crates:

1. The `volo` crate, which contains the common components of the framework.
2. The `volo-thrift` crate, which provides the Thrift RPC implementation.
3. The `volo-grpc` crate, which provides the gRPC implementation.
4. The `volo-build` crate, which generates thrift and protobuf code.
5. The `volo-cli` crate, which provides the CLI interface to bootstrap a new project and manages the idl files.
6. The `volo-macros` crate, which provides the macros for the framework.

- Features

Powered by GAT

Volo uses `Motore` as its middleware abstraction, which is powered by GAT.

Through GAT, we can avoid many unnecessary `Box` memory allocations, improve ease of use, and provide users with a more friendly programming interface and a more ergonomic programming paradigm.

https://github.com/cloudwego/motore

https://blog.rust-lang.org/2021/08/03/GATs-stabilization-push.html

…

开源社 kaiyuanshe

High Performance

Rust is known for its high performance and safety. We always take high performance as our goal in the design and implementation process, reduce the overhead of each place as much as possible, and improve the performance of each implementation.

First of all, it is very unfair to compare the performance with the Go framework, so we will not focus on comparing the performance of Volo and Kitex, and the data we give can only be used as a reference, I hope everyone can view it objectively; at the same time, due to the open source community has not found another mature Rust async version Thrift RPC framework, and performance comparison is always easy to lead to war, so we hope to weaken the comparison of performance data as much as possible, and we'll only publish our own QPS data.

Under the same test conditions as Kitex (limited to 4C), the Volo QPS is 350k; at the same time, we are internally verifying the version based on Monoio (CloudWeGo's open source Rust async runtime), and the QPS can reach 440k.

From the flame graph of our online business, thanks to Rust's static distribution and excellent compilation optimization, the overhead of the framework part is basically negligible (excluding syscall overhead).

Easy to Use

~~Rust is known for being hard to learn and hard to use~~, and we want to make it as easy as possible for users to use the Volo framework and write microservices in the Rust language, providing the most ergonomic and intuitive coding experience possible. Therefore, we make ease of use one of our most important goals.

For example, we provide the volo command line tool for bootstraping projects and managing idl files; at the same time, we split thrift and gRPC into two independent(but share some components) frameworks to provide programming paradigms that best conform to different protocol semantics and interface.

We also provide the `#[service]` macro (which can be understood as the `async_trait` that does not require `Box`) to enable users to write service middleware using async rust without psychological burden.

Strong Extensibility

Benefiting from Rust's powerful expression and abstraction capabilities, through the flexible middleware `Service` abstraction, developers can process RPC meta-information, requests and responses in a very unified form.

For example, service governance functions such as service discovery and load balancing can be implemented in the form of services without the need to implement Trait independently.

We have also created an organization `Volo-rs`, any contributions are welcome.

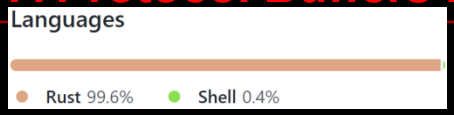
- **https://www.cloudwego.io/zh/docs/volo/guide/**

1.4.3 PROST

Overview

- https://github.com/tokio-rs/prost

A Protocol Buffers implementation for the Rust Language.




```
[submodule "prost-build/third-party/protobuf"]
        path = prost-build/third-party/protobuf
        url = git@github.com:protocolbuffers/protobuf
```

`prost` is a Protocol Buffers implementation for the Rust Language. `prost` generates simple, idiomatic Rust code from `proto2` and `proto3` files.

Compared to other Protocol Buffers implementations, `prost`

- Generates simple, idiomatic, and readable Rust types by taking advantage of Rust `derive` attributes.
- Retains comments from `.proto` files in generated Rust code.
- Allows existing Rust types (not generated from a `.proto`) to be serialized and deserialized by adding attributes.
- Uses the `bytes::{Buf, BufMut}` abstractions for serialization instead of `std::io::{Read, Write}`.
- Respects the Protobuf `package` specifier when organizing generated code into Rust modules.
- Preserves unknown enum values during deserialization.
- Does not include support for runtime reflection or message descriptors.

- **FAQ**

1. **Could `prost` be implemented as a serializer for Serde?**

Probably not, however I would like to hear from a Serde expert on the matter. There are two complications with trying to serialize Protobuf messages with Serde:

- Protobuf fields require a numbered tag, and currently there appears to be no mechanism suitable for this in `serde`.
- The mapping of Protobuf type to Rust type is not 1-to-1. As a result, trait-based approaches to dispatching don't work very well. Example: six different Protobuf field types correspond to a Rust `Vec<i32>`: `repeated int32`, `repeated sint32`, `repeated sfixed32`, and their packed counterparts.

But it is possible to place `serde` derive tags onto the generated types, so the same structure can support both `prost` and `Serde`.

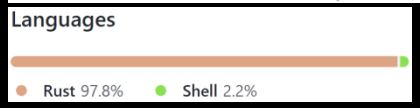
1.4.4 tarpc

Overview

- **https://github.com/google/tarpc**

Disclaimer: This is not an official Google product.

tarpc is an RPC framework for rust with a focus on ease of use. Defining a service can be done in just a few lines of code, and most of the boilerplate of writing a server is taken care of for you.



- **Features**

tarpc differentiates itself from other RPC frameworks by defining the schema in code, rather than in a separate language such as .proto. This means there's no separate compilation process, and no context switching between different languages.

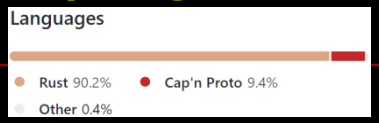
Some other features of tarpc:

- Pluggable transport: any type implementing `Stream<Item = Request> + Sink<Response>` can be used as a transport to connect the client and server.
- `Send + 'static` optional: if the transport doesn't require it, neither does tarpc!
- Cascading cancellation: dropping a request will send a cancellation message to the server. The server will cease any unfinished work on the request, subsequently cancelling any of its own requests, repeating for the entire chain of transitive dependencies.
- Configurable deadlines and deadline propagation: request deadlines default to 10s if unspecified. The server will automatically cease work when the deadline has passed. Any requests sent by the server that use the request context will propagate the request deadline. For example, if a server is handling a request with a 10s deadline, does 2s of work, then sends a request to another server, that server will see an 8s deadline.
- Distributed tracing: tarpc is instrumented with tracing primitives extended with OpenTelemetry traces. Using a compatible tracing subscriber like Jaeger, each RPC can be traced through the client, server, and other dependencies downstream of the server. Even for applications not connected to a distributed tracing collector, the instrumentation can also be ingested by regular loggers like env_logger.
- Serde serialization: enabling the `serde1` Cargo feature will make service requests and responses `Serialize + Deserialize`. It's entirely optional, though: in-memory transports can be used, as well, so the price of serialization doesn't have to be paid when it's not needed.

1.5 Serialization + RPC

1.5.1 capnproto-rust

- https://github.com/capnproto/capnproto-rust



- Features
  - tagged unions
  - generics
  - protocol evolvability
  - canonicalization
  - `Result`-based error handling
  - `no_std` support
- Crates

capnp	Runtime library for dealing with Cap'n Proto messages.	crates.io v0.14.10
capnpc	Rust code generator plugin, including support for hooking into a `build.rs` file in a `cargo` build.	crates.io v0.14.9
capnp-futures	Support for asynchronous reading and writing of Cap'n Proto messages.	crates.io v0.14.2
capnp-rpc	Object-capability remote procedure call system with "level 1" features.	crates.io v0.14.1

- https://docs.capnproto-rust.org/capnp/

1.6 Rust for Cloud Native

- https://github.com/awesome-rust-cloud-native/awesome-rust-cloud-native

Applications and Services

- apache/incubator-teaclave: open source universal secure computing platform, making computation on privacy-sensitive data safe and simple
- bottlerocket-os/bottlerocket: an operating system designed for hosting containers
- containers/krunvm: manage lightweight VMs created from OCI images
- containers/youki: a container runtime written in Rust
- datafuselabs/datafuse: A Modern Real-Time Data Processing & Analytics DBMS with Cloud-Native Architecture, built to make the Data Cloud easy
- firecracker-microvm/firecracker: secure and fast microVMs for serverless computing
- infinyon/fluvio: Cloud-native real-time data streaming platform with in-line computation capabilities
- krustlet/krustlet: Kubernetes Rust Kubelet
- kube-rs/controller-rs: a Kubernetes example controller
- kube-rs/version-rs: example Kubernetes reflector and web server
- kubewarden/policy-server: webhook server that evaluates WebAssembly policies to validate Kubernetes requests
- linkerd/linkerd2-proxy: a purpose-built proxy for the Linkerd service mesh
- openebs/mayastor: A cloud native declarative data plane in containers for containers
- rancher-sandbox/lockc: eBPF-based MAC security audit for container workloads
- tikv/tikv: distributed transactional key-value database
- tremor-rs/tremor-runtime: an event processing system that supporting complex workflows such as aggregation, rollups, an ETL language, and a query language
- valeriansaliou/sonic: fast, lightweight & schema-less search backend
- WasmEdge/WasmEdge: WasmEdge is a high-performance WebAssembly (Wasm) Virtual Machine (VM) runtime, which enables serverless functions to be embedded into any software platform; from cloud's edge to SaaS to automobiles

Libraries

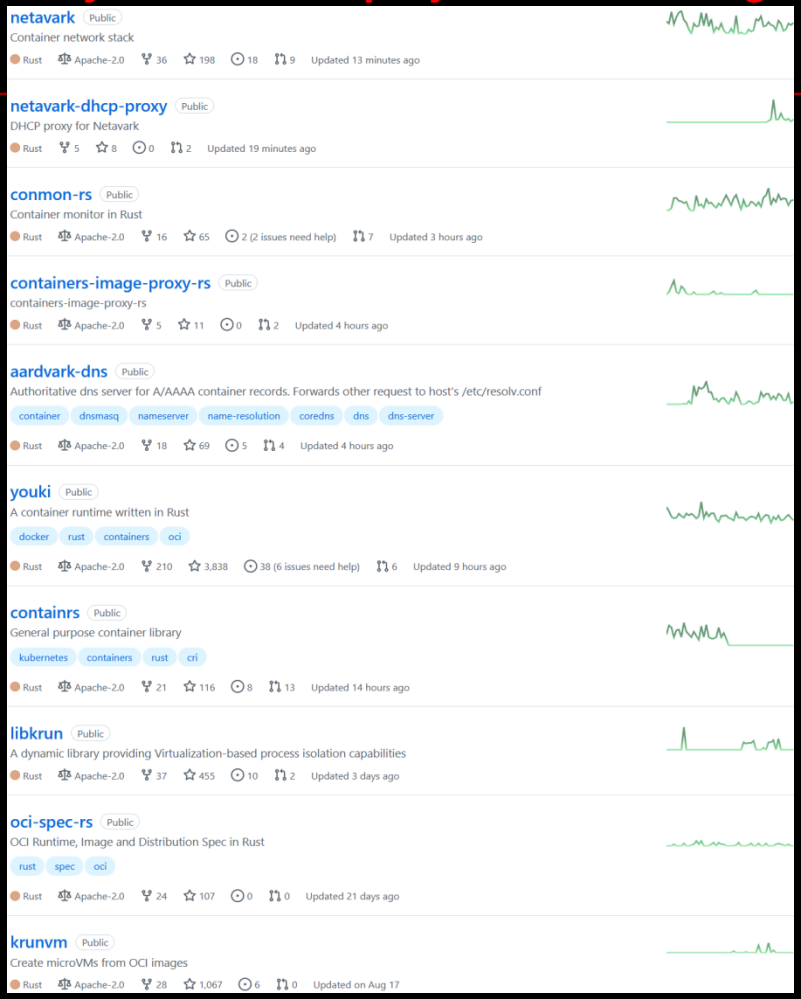
- CNI Plugins: crate/framework to write CNI (container networking) plugins in Rust (includes a few custom plugins as well)
- containers/libkrun: a dynamic library providing Virtualization-based process isolation capabilities
- kube-rs/kube-rs: Kubernetes Rust client and async controller runtime
- qovery/engine: Qovery Engine is an open-source abstraction layer library that turns easy apps deployment on AWS, GCP, Azure, and other Cloud providers in just a few minutes
- open-telemetry/opentelemetry-rust: OpenTelemetry is a set of APIs, SDKs, tooling and integrations that are designed for the creation and management of telemetry data such as traces, metrics, and logs.

- ...

Project Containers

- https://github.com/containers

Many Rust-based projects are coming:



netavark Public
Container network stack
Rust · Apache-2.0 · 36 · 198 · 18 · 9 · Updated 13 minutes ago

netavark-dhcp-proxy Public
DHCP proxy for Netavark
Rust · 5 · 8 · 0 · 2 · Updated 19 minutes ago

conmon-rs Public
Container monitor in Rust
Rust · Apache-2.0 · 16 · 65 · 2 (2 issues need help) · 7 · Updated 3 hours ago

containers-image-proxy-rs Public
containers-image-proxy-rs
Rust · Apache-2.0 · 5 · 11 · 0 · 2 · Updated 4 hours ago

aardvark-dns Public
Authoritative dns server for A/AAAA container records. Forwards other request to host's /etc/resolv.conf
container · dnsmasq · nameserver · name-resolution · coredns · dns · dns-server
Rust · Apache-2.0 · 18 · 69 · 5 · 4 · Updated 4 hours ago

youki Public
A container runtime written in Rust
docker · rust · containers · oci
Rust · Apache-2.0 · 210 · 3,838 · 38 (6 issues need help) · 6 · Updated 9 hours ago

containrs Public
General purpose container library
kubernetes · containers · rust · cri
Rust · Apache-2.0 · 21 · 116 · 8 · 13 · Updated 14 hours ago

libkrun Public
A dynamic library providing Virtualization-based process isolation capabilities
Rust · Apache-2.0 · 37 · 455 · 10 · 2 · Updated 3 days ago

oci-spec-rs Public
OCI Runtime, Image and Distribution Spec in Rust
rust · spec · oci
Rust · Apache-2.0 · 24 · 107 · 0 · 0 · Updated 21 days ago

krunvm Public
Create microVMs from OCI images
Rust · Apache-2.0 · 28 · 1,067 · 6 · 0 · Updated on Aug 17

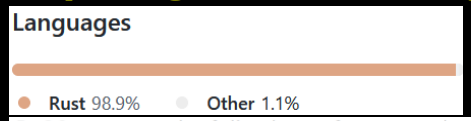
1.7 Integrating Rust into Python

1.7.1 PyO3

- https://pyo3.rs/

  Rust bindings for Python, including tools for creating native Python extension modules. Running and interacting with Python code from a Rust binary is also supported.

- https://github.com/PyO3/pyo3



PyO3 supports the following software versions:

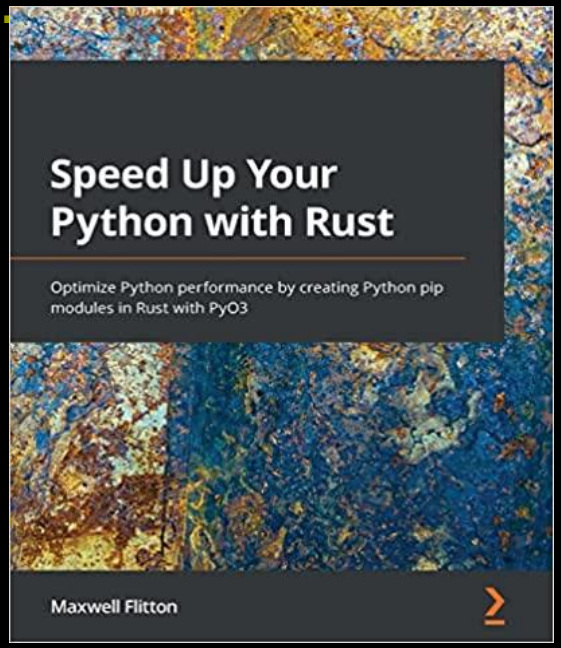
- Python 3.7 and up (CPython and PyPy)
- Rust 1.48 and up

You can use PyO3 to write a native Python module in Rust, or to embed Python in a Rust binary. The following sections explain each of these in turn.

- Tools and libraries

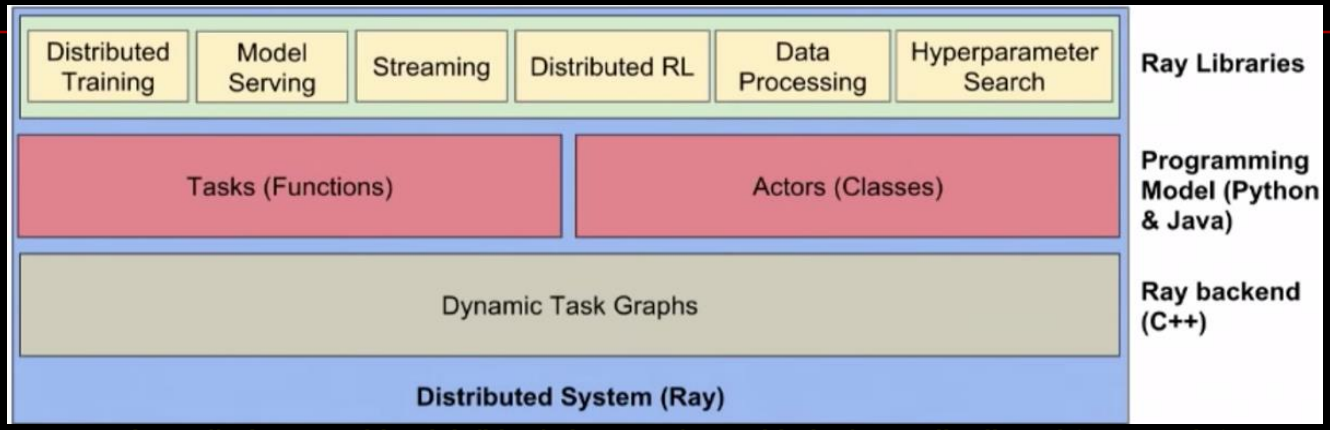
- maturin *Build and publish crates with pyo3, rust-cpython or cffi bindings as well as rust binaries as python packages*
- setuptools-rust *Setuptools plugin for Rust support.*
- pyo3-built *Simple macro to expose metadata obtained with the* `built` *crate as a* `PyDict`
- rust-numpy *Rust binding of NumPy C-API*
- dict-derive *Derive FromPyObject to automatically transform Python dicts into Rust structs*
- pyo3-log *Bridge from Rust to Python logging*
- pythonize *Serde serializer for converting Rust objects to JSON-compatible Python objects*
- pyo3-asyncio *Utilities for working with Python's Asyncio library and async functions*
- rustimport *Directly import Rust files or crates from Python, without manual compilation step. Provides pyo3 integration by default and generates pyo3 binding code automatically.*

Good Resources

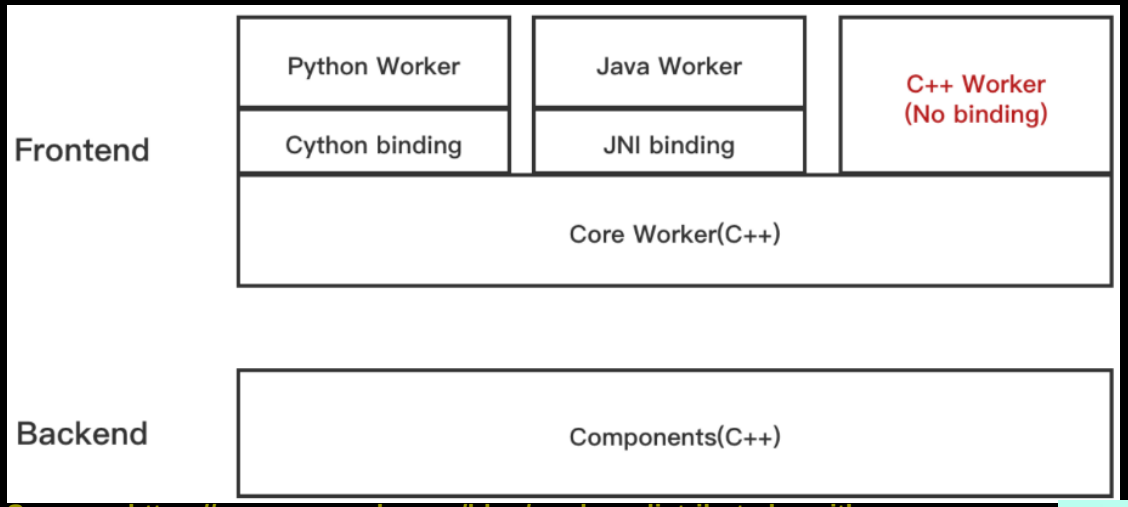
- https://github.com/ruwanego/RustPython-intro
- https://towardsdatascience.com/nine-rules-for-writing-python-extensions-in-rust-d35ea3a4ec29
- https://pythonspeed.com/articles/rust-cython-python-extensions/
- 

- …

2) A new design
2.1 Implementation of Ray
Software layers of Ray



Source: https://xzhu0027.gitbook.io/blog/ml-system/sys-ml-index/ray-a-distributed-framework-for-emerging-ai-applications



Source: https://www.anyscale.com/blog/modern-distributed-c-with-ray

Ray Core in C++

```
[mydev@fedora ray-master]$ tree -d src/ray
src/ray
├── common
│   ├── asio
│   ├── ray_syncer
│   ├── task
│   └── test
├── core_worker
│   ├── lib
│   │   └── java
│   ├── store_provider
│   │   └── memory_store
│   ├── test
│   └── transport
├── design_docs
├── gcs
│   ├── gcs_client
│   │   └── test
│   ├── gcs_server
│   │   └── test
│   ├── pubsub
│   ├── store_client
│   │   └── test
│   └── test
├── internal
├── object_manager
│   ├── plasma
│   │   └── test
│   └── test
├── protobuf
├── pubsub
│   └── test
├── raylet
│   ├── format
│   ├── scheduling
│   │   └── policy
│   └── test
├── raylet_client
├── rpc
│   ├── agent_manager
│   ├── gcs_server
│   ├── node_manager
│   ├── object_manager
│   ├── runtime_env
│   ├── test
│   └── worker
├── stats
├── thirdparty
└── util
```

```
[mydev@fedora ray-master]$ tokei src/ray
===============================================================================
 Language            Files        Lines         Code     Comments       Blanks
===============================================================================
 C                       3         6546         3828         2190          528
 C Header              230        42922        20774        15671         6477
 C++                   262        94618        72860        11217        10541
 FlatBuffers Schema      2          563          276          231           56
 LD Script               2           77           77            0            0
 Markdown                2          156            0          124           32
 Protocol Buffers       20         4439         2292         1614          533
 ReStructuredText        2          113           80            0           33
===============================================================================
 Total                 523       149434       100187        31047        18200
===============================================================================
[mydev@fedora ray-master]$
```

2.2 Our new scheme

2.2.1 Current design

Design goals of Ray.Rust:

- A reimplementation of Ray by Python + Rust to instead of Python + C++ in current implementation;
- The major runtime is RustPython;
- Switch the build system from Bazel to Meson;
- Make a self-contained implementation as far as possible;
- Keep the Python code in Ray AIR and its libraries reusable;
- A lightweight solution for much more DevOps friendly.

Technology selection

- Core
  Rewrite Ray Core in Rust [Actor Model framework]
- Data
  Arrow2 + DataFusion + Polars + …
- Object store
  arrow-rs
- KV store
  Redis with Rust API, or considering pure Rust-based database.

- **Messaging & RPC**
  capnproto-rust + Serde + [tarpc]
- **DevOps**
  A new container orchestration implementation(also base on Python + Rust) for Rust-based container runtime such like Kata Containers and Youki, which is Serverless architecture and Micro-VM friendly . In particular, add the support for xBPF and Wasm workload.
- **System-level acceleration**
  Customized network stack, in-kernel messaging and RPC, or a HW-SW co-designed one and more.
- **Interop**
  PyO3
- **Others**
  …

For more details, you may look forward to our new talk "First exploration of Ray.Rust".

Beyond Kubernetes

- Now Kubernetes is the de facto standard for today's production-grade Container orchestration, which is surrounded by an amazing huge and continuously growing ecosystem. It comes with the Container first design philosophy and is optimized for that.

- Kubernetes is really powerful, but it is getting more complex and accumulating more and more historical baggage at the same time…
- Lightweight Kubernetes distribution like K3S is suitable for Edge devices that have limited computing resources when compared with Cloud side, but may not still be a good fit for hardware platform with even less computing resources like Microcontrollers which is common on IoT devices.
- New workload like Wasm and eBPF is more lightweight than Container, and their ecosystem is booming.
- Current solutions like Krustlet or Kata Containers/WasmEdge supports the above new workload by extending Kubernetes or is implemented as OCI-compatible:
  the benefits are it can both support Container workload and the new workload, and make best of use the existed code and ecosystem of Kubernetes, Docker, and so on.
  the drawbacks are thus means it has to inherit some kind of "historical burden", and most of all, it is not burn for new workload like Wasm and eBPF.
- Get rid of Go is not a dream in the future.
- …
- Our new discussions on this topic will be coming soon…

2.3 Replacing Python plus C++ in AI frameworks with Python plus Rust?

- Today, most of the main stream AI frameworks like Tensorflow, PyTorch and MXNet embrace Python + C++ for their software layers design;
- While Rust is rapidly rising!

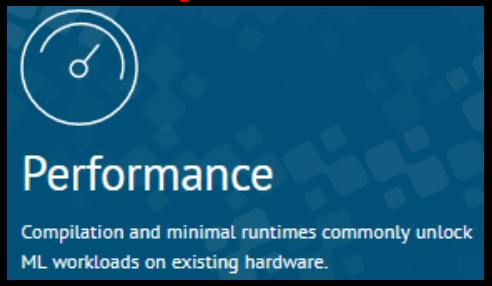
  https://vaaaaanquish.github.io/Awesome-Rust-MachineLearning/

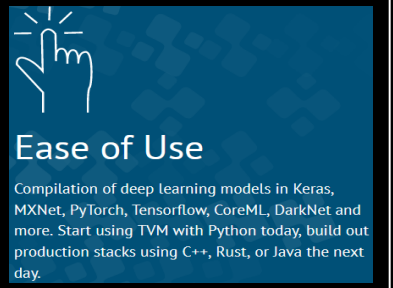
For more details, you may refer to our new talks "Rust for Edge AI" and "Rust for Edge Computing".

2.3.1 TVM

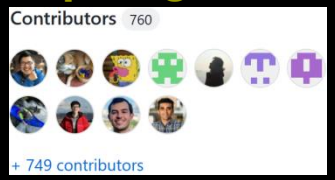
- https://tvm.apache.org/
  An End to End Machine Learning Compiler Framework for CPUs, GPUs and accelerators

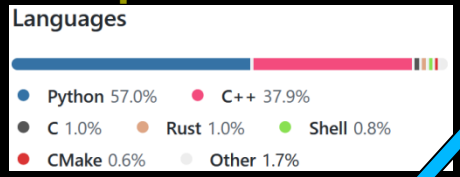
- Key Features & Capabilities





- https://github.com/apache/tvm






```
[submodule "dmlc-core"]
        path = 3rdparty/dmlc-core
        url = https://github.com/dmlc/dmlc-core.git
[submodule "dlpack"]
        path = 3rdparty/dlpack
        url = https://github.com/dmlc/dlpack.git
[submodule "3rdparty/rang"]
        path = 3rdparty/rang
        url = https://github.com/agauniyal/rang.git
[submodule "3rdparty/vta-hw"]
        path = 3rdparty/vta-hw
        url = https://github.com/apache/tvm-vta.git
[submodule "3rdparty/libbacktrace"]
        path = 3rdparty/libbacktrace
        url = https://github.com/tlc-pack/libbacktrace.git
[submodule "3rdparty/cutlass"]
        path = 3rdparty/cutlass
        url = https://github.com/NVIDIA/cutlass.git
```

https://github.com/dmlc/dmlc-core
A common bricks library for building scalable and portable distributed machine learning.

Src (main branch)

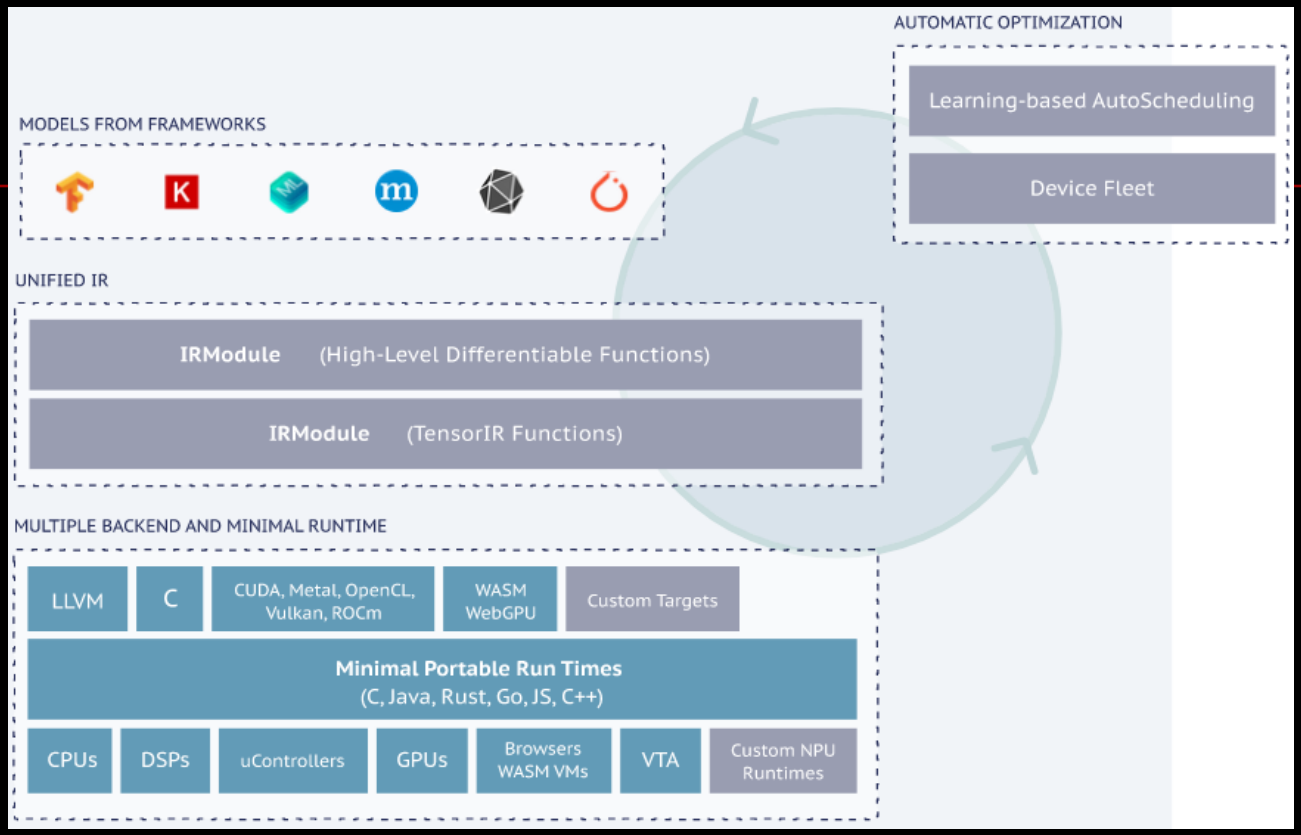
Stats with submodules (last commit: a4840e7de38c5a2000917f2101f3ec4a374bcd39)

```
[mydev@fedora tvm-main]$ tokei
```

Language	Files	Lines	Code	Comments	Blanks
Arduino C++	3	138	62	62	14
Autoconf	12	2966	2364	230	372
Automake	1	584	376	43	165
BASH	4	227	107	89	31
Batch	5	104	28	68	8
C	71	25795	18730	3612	3453
C Header	1167	370978	204921	109468	56589
CMake	157	11333	6639	3313	1381
Coq	10	1256	1108	0	148
C++	955	296392	215837	47202	33353
C++ Header	1	502	424	18	60
CSS	4	1849	1489	65	295
Dockerfile	1	27	16	5	6
Go	24	3788	2364	982	442
INI	1	33	13	16	4
Java	50	7031	4418	1890	723
JavaScript	170	6427	6004	294	129
JSON	3	505	505	0	0
Makefile	46	3563	2154	785	624
Meson	3	26	19	0	7
Objective-C	2	99	27	50	22
Objective-C++	10	2569	1850	430	289
Python	2282	638318	490018	65317	82983
ReStructuredText	101	17810	13232	0	4578
Scala	63	12548	8852	2292	1404
Shell	188	18513	11435	5099	1979
TCL	6	1723	1419	195	109
Plain Text	46	13463	0	12510	953
TOML	20	996	577	340	79
TypeScript	10	3110	2061	786	263
Visual Studio Pro|	1	165	165	0	0
Visual Studio Sol|	1	54	53	0	1
XML	37	2534	1765	628	141
YAML	8	433	331	75	27
HTML	2193	455426	433309	10103	12014
├ JavaScript	2192	13640	13637	1	2
(Total)		469066	446946	10104	12016
Markdown	97	12854	0	10337	2517
├ BASH	25	939	727	27	185
├ C++	11	1869	1176	332	361
├ Java	1	47	41	0	6
├ Makefile	2	30	13	7	10
├ Python	4	54	47	2	5
├ Rust	1	14	12	0	2
├ Shell	2	40	37	0	3
(Total)		15847	2053	10705	3089
Rust	92	11971	8590	1963	1418
├ Markdown	34	577	0	479	98
(Total)		12548	8590	2442	1516
Total	7845	1926110	1441262	278267	206581

```
[mydev@fedora tvm-main]$
```

Architecture & Design



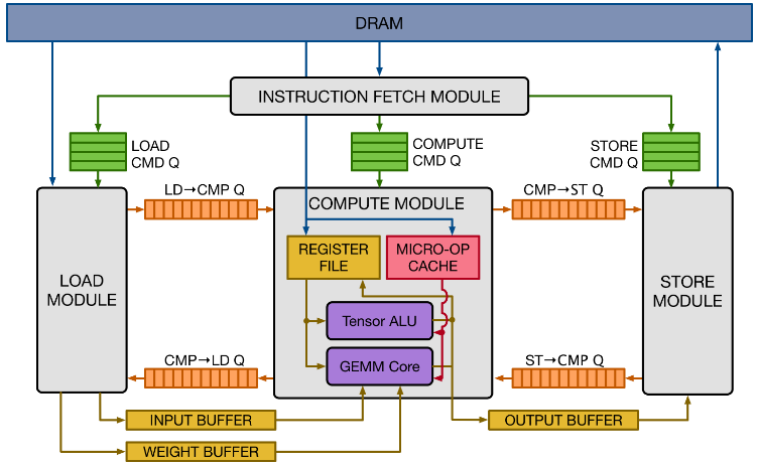
- **Python-first API**

VTA

- https://tvm.apache.org/docs/topic/vta/index.html

Versatile Tensor Accelerator

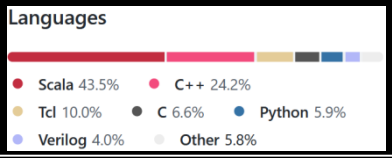
The Versatile Tensor Accelerator (VTA) is an open, generic, and customizable deep learning accelerator with a complete TVM-based compiler stack. We designed VTA to expose the most salient and common characteristics of mainstream deep learning accelerators. Together TVM and VTA form an end-to-end hardware-software deep learning system stack that includes hardware design, drivers, a JIT runtime, and an optimizing compiler stack based on TVM.



- Key Features

- Generic, modular, open-source hardware.
- Streamlined workflow to deploy to FPGAs.
- Simulator support to prototype compilation passes on regular workstations.
- Pynq-based driver and JIT runtime for both simulated and FPGA hardware back-end.
- End to end TVM stack integration.

- https://github.com/apache/tvm-vta



2.3.3.1 Rust for TVM

- https://github.com/apache/tvm/releases#rust-binding
- https://rustrepo.com/repo/ehsanmok-tvm-rust-rust-data-processing
- https://cdmana.com/2021/04/20210429180508145o.html
- https://rustmagazine.github.io/rust_magazine_2021/chapter_3/hw_rust_rvm_wasm_ai.html
- https://github.com/tinyms-ai/awesome-rusted-ai
- https://github.com/octoml/tvm-build

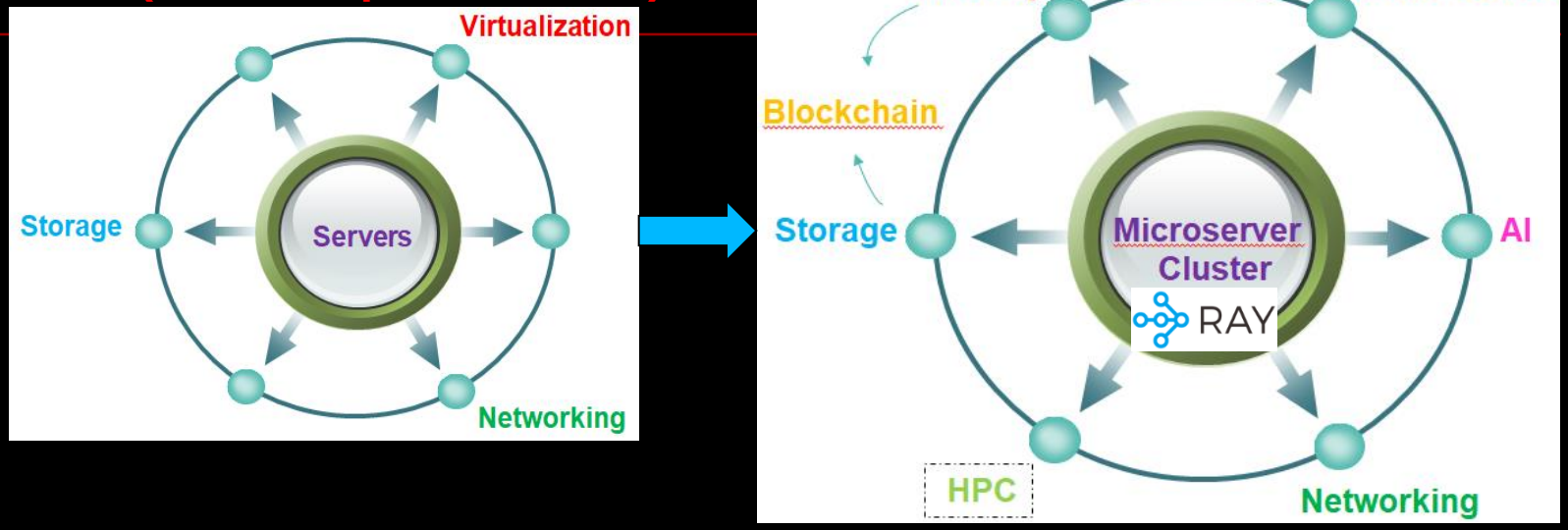
  A library for building TVM programmatically.
- …

How about rewrite TVM in Rust with Ray Core(also in Rust)…

VI. Wrap-up

Ray for Hyper-Converged Infrastructure

- Trend (from our point of view)



- HW/SW co-designed system is the future.

- You may look forward to our new talks "Ray as a universal infrastructure for distributed computing" and "The fourth round discussion on eBPF-centric new approach for Hyper-Converged Infrastructure & Edge Computing".

开源社
kaiyuanshe

THANK YOU

QUESTIONS?

微信公众号：开源社KAIYUANSHE
视频号：开源社KAIYUANSHE
新浪微博：开源社
B站：2020开源社

简书：开源社
头条：开源社
Facebook：KaiyuansheChina
Twitter：KAIYUANSHE

扫码关注开源社公众号

Reference

开源社 kaiyuanshe

Slides/materials from many and varied sources:

- http://en.wikipedia.org/wiki/
- http://www.slideshare.net/

- https://en.wikipedia.org/wiki/Parallel_computing
- https://en.wikipedia.org/wiki/Data_parallelism
- https://en.wikipedia.org/wiki/Task_parallelism
- https://www.anyscale.com/blog
- https://www.anyscale.com/ray-summit-2022/agenda
- https://www.graalvm.org/release-notes/version-roadmap/
- https://www.micahlerner.com/2021/06/27/ray-a-distributed-framework-for-emerging-ai-applications.html
- https://www.databricks.com/blog/2021/11/19/ray-on-databricks.html
- https://web.archive.org/web/20101223025529/http://www.stackless.com/
- https://github.com/nanomsg
- https://github.com/msgpack
- https://medium.com/geekculture/quick-start-to-grpc-using-rust-c655785fc6f4
- https://github.com/capnproto/capnproto-rust
- https://dev.to/kushalj/capn-proto-rpc-at-the-speed-of-rust-part-1-4joo

- https://capnproto.org/otherlang.html
- https://github.com/anowell/are-we-learning-yet
- https://github.com/rossmacarthur/serde
- https://docs.ray.io/en/latest/ray-overview/ray-libraries.html
- https://capnproto.org/otherlang.html
- https://www.raspberrypi.org/blog/experience-ai-deepmind-ai-education/
- https://www.zdnet.com/article/programming-languages-its-time-to-stop-using-c-and-c-for-new-projects-says-microsoft-azure-cto/
- https://cloud.google.com/blog/products/ai-machine-learning/build-a-ml-platform-with-kubeflow-and-ray-on-gke
- https://towardsdatascience.com/nine-rules-for-writing-python-extensions-in-rust-d35ea3a4ec29
- https://towardsdatascience.com/machine-learning-and-rust-part-4-neural-networks-in-torch-85ee623f87a
- https://bytexd.com/hardware/raspberry-pi-5-what-to-expect/
- …