

# RISC-V Server Software Landscape

Dr Oliver Perks - Rivos Inc.  
[operks@rivosinc.com](mailto:operks@rivosinc.com)

# Introductions



## Who are Rivos?

A RISC-V silicon company founded in 2021  
Focus on power efficient server solutions

## CPU and Accelerator

Meeting demands of AI and Data Analytics  
Reuse existing models + Frameworks

## Rivos = RISC-V Open Source

Open software stack  
Contribute developments upstream



## What is RISE?

An industry consortium under the Linux Foundation  
Software ecosystem readiness for RISC-V  
Mandate for Open Source

## RISE Vs. RISC-V International (RVI)

RVI focuses on hardware standards  
RISE is for practical development of software

## RISE is not specifically HPC/AI/ML

New architectures need a lot of system software  
Foundation to build specific stacks upon

## Premier Members



## General Members



北京开源芯片研究院  
BEIJING INSTITUTE OF OPEN SOURCE CHIP



tenstorrent

Join RISE

Apply for membership (€40k + 2 FTE)  
Governing board currently full



Additional Information: <https://riseproject.dev/>




# RISC-V CPU Hardware State of the Nation



**SpacemiT - K1**  
**8 Core @ 1.6 GHz**  
**RVV 1.0 - 256 bit**  
**2.0 TOPS AI**



**T-Head C910 / C920**  
**4 Core clusters @ 2GHz**  
**RVV 0.7.1 - 128 bit**  
**120W TDP (64 core SG2042)**

	SBC	Vectorization	Specs
 <p>Milk-V Pioneer</p>	✗	✓	<ul style="list-style-type: none"> <li>• 64 Cores SG2042 - C920</li> <li>• 128GB DDR4</li> <li>• PCIe, SATA, M.2</li> <li>• Fedora, Ubuntu</li> </ul>
 <p>Banana Pi BPI3-F3</p>	✓	✓	<ul style="list-style-type: none"> <li>• 8 Cores K1</li> <li>• 16GB LPDDR4</li> <li>• GPIO, MicroSD, PCIe, M.2</li> <li>• Bianbu (Debian based)</li> </ul>
 <p>Lichee Pi 4A</p>	✓	✓	<ul style="list-style-type: none"> <li>• 4 Cores TH1520 C910</li> <li>• 16GB LPDDR4</li> <li>• GPIO, Ethernet</li> <li>• Debian / OpenWRT / Android</li> </ul>

# What's Missing for Server

## Hardware features

- **Virtualization**
- **More hardware with RVV 1.0**
- **Vector Crypto (Zvbb, Zvbc, Zvkg, Zvkn, ..)**
- **Bitmanip (Zba, Zbb, Zbs)**

## Memory

- **Many builds take a lot of memory**
- **GCC / LLVM parallel builds**

## Cores

- **Too slow / too few for parallel builds**
- **Sadly emulation is often faster**

**Analogy: Developing enterprise software for AWS  
Graviton 4 based on Raspberry Pi**

## Lots of [pre-]silicon companies

- **Ventana, Codasip, SiFive, Rivos**

## Expect a Cambrian explosion - Very soon

- **Server class CPUs**
- **RISC-V based accelerators**

## Robust enterprise software support

- **Many already in the pipeline**

## Hardware / Software chicken and egg

- **Break the cycle to kickstart adoption**

# Ecosystem from the Ground Up

## Mainstream RISC-V

- Needs foundational software
- Built from the ground up
- Operating Systems -> Applications

## General Purpose not SDK

- Adapt to changing user requirements
- Standard tools and libraries

## So many different use cases

- Server is just a subset for RISC-V



<https://landscape.riscv.org/>

# How to Develop an Open Ecosystem for Server

## RISC-V faces same challenges AArch64

- **Multiple vendor implementations**
- **Heavy lifting of software development**
- **Move from X86 code (intrinsics / ASM)**

## Vendors working together

- **No one company can do it alone**
- **But all dependent on same packages**
- **De-duplicate and share efforts**

## Common interfaces + tools

- **Community wont port to each chip**
- **Need commonality and reuse**

## RISC-V is built on standards

- **ISA is open, but well defined**
  - Extensions must be ratified
- **RVA23 Profile**

## Communication is key

- **RVI hosts multiple SIGs**
- **Focused efforts to share activities**
- **Friendly and open**

**All vendors need a robust ecosystem to thrive**

**“A rising tide lifts all boats”**

# RVA23 Profile

## What is RVA23 Profile?

- **Standardizing critical features needed for performance-intensive computing**
- **List of extensions included**
- **Establishes a scalable, secure, and future-proof foundation**

## Critical for software development

- **Standardizes support for advanced OS stacks and rich applications**
- **Binary compatibility between**


[Profile Link](#)

### RVA23U64 Mandatory Extensions

The following mandatory extensions were present in RVA22U64.

- M Integer multiplication and division.
- A Atomic instructions.
- F Single-precision floating-point instructions.
- D Double-precision floating-point instructions.
- C Compressed instructions.
- B Bit-manipulation instructions.
- Zicsr CSR instructions. These are implied by presence of F.
- Zicntr Base counters and timers.
- Zihpm Hardware performance counters.
- Ziccif Main memory regions with both the cacheability and coherence PMAs must support instruction fetch, and any instruction fetches of naturally aligned power-of-2 sizes up to  $\min(\text{ILEN}, \text{XLEN})$  (i.e., 32 bits for RVA23) are atomic.
- Ziccrse Main memory regions with both the cacheability and coherence PMAs must support RsvrEventual.
- Ziccamao Main memory regions with both the cacheability and coherence PMAs must support all atomics in A.
- Zicclsm Misaligned loads and stores to main memory regions with both the cacheability and coherence PMAs must be supported.
- Za64rs Reservation sets are contiguous, naturally aligned, and a maximum of 64 bytes.
- Zihintpause Pause hint.
- Zic64b Cache blocks must be 64 bytes in size, naturally aligned in the address space.
- Zicbom Cache-block management instructions.
- Zicbop Cache-block prefetch instructions.
- Zicboz Cache-Block Zero Instructions.
- Zfhmin Half-precision floating-point.
- Zkt Data-independent execution latency.

The following mandatory extensions are new in RVA23U64:

- V Vector extension.
-  *V was optional in RVA22U64.*
- Zvfhmin Vector minimal half-precision floating-point.
- Zvbb Vector basic bit-manipulation instructions.
- Zvkt Vector data-independent execution latency.
- Zihintntl Non-temporal locality hints.
- Zicond Integer conditional operations.
- Zimop may-be-operations.
- Zemop Compressed may-be-operations.
- Zcb Additional compressed instructions.
- Zfa Additional floating-Point instructions.
- Zawrs Wait-on-reservation-set instructions.
- Supm Pointer masking, with the execution environment providing a means to select PMLen=0 and PMLen=7 at minimum.



# Server Software Readiness

<b>Compilers &amp; Toolchains</b>	LLVM, GCC
<b>System Libraries</b>	Boringssl, OpenSSL, OpenBLAS, SLEEP
<b>Kernel &amp; Virtualization</b>	Docker, Kubernetes
<b>Language Runtimes</b>	C, C++, Fortran, Python, Rust, Java, Go, V8 and Node.js
<b>Debug &amp; Profiling Tools</b>	Valgrind, Perf, GDB
<b>Simulator/Emulators</b>	QEMU, Gem5, Spike
<b>Operating Systems</b>	Ubuntu, Fedora
<b>System Software/FW</b>	UEFI, ACPI, TianoCore, EDK2
<b>Dev &amp; Infra</b>	Linux Kernel CI, GCC CI, OpenJDK CI, GCC Fuzz CI



**MEMBER**

# What does RISE offer?

## Documentation

- **Optimization Guide**
- **Troubleshooting**

## Continuous Integration

- **Cloud based build farm**
- **Kernels, compilers and libraries**

## Python Packages

- **Build and host binary wheels**
- **Upstream tracking**

## RFPs

- **Funded development work**
- **Further the ecosystem**

## Dev Appreciation Grants

- **GitHub Sponsorship for OSS dev work**
- **Specifically `linux-riscv64`**

# RISC-V Optimization Guide

## Designed as a vendor agnostic

- Examples of how to program to Spec
- Some implementation specific
  - E.g. LMUL length

## Establish best practices for high performance cores

- How to do things 'correctly'
- Results in portable solutions

## Heavy focus on vectorization

- Assembly examples



Zero can be folded into any instruction with a register operand. There's no need to initialize a temporary register with 0 for the sole purpose of using that register in a subsequent instruction. The following table identifies cases where a temporary register can be eliminated by prudent use of x0.

Do	Don't
<code>fmv.d.x f0,x0</code>	<code>li x5,0</code> <code>fmv.d.x f0,x5</code>
<code>amoswap.w.aqr1 a0,x0,(x10)</code>	<code>li x5,0</code> <code>amoswap.w.aqr1 x6,x5,(x10)</code>
<code>sb x0,0(x5)</code>	<code>li x6,0</code> <code>sb x6,0(x5)</code>
<code>bltu x0,x7,1f</code>	<code>li x5,0</code> <code>bltu x5,x7,1f</code>

# Case Study - Porting SLEEF to RISC-V

## SIMD Library for Evaluating Elementary Functions (SLEEF)

- **Library for vectorised maths routines (libm + DFT)**
- **Abstraction layer for different vector implementations**
- **Strong community in AArch64 and X86**

## RISC-V Port

- **Challenges:**
  - Porting to a spec not an implementation (X-Compile, QEMU)
  - Needs to work on everyone's (RVV 1.0) hardware
  - Vector length agnostic (similar to SVE)
  - Work with existing architectures to ensure conformance
- **Success:**
  - Collaboration with Arm (maintainers)
  - Multiple RISC-V orgs (Rivos, SiFive) working together
  - Enables PyTorch, OpenJDK + many more
  - Makes adopting RVV easier
  - [RISC-V Summit Presentation](#)

		Linux		
		Linux Build & Test <span>passing</span>		
Arch.	Vector Extensions	gcc	llvm	icc
x86_64	SSE2, SSE4, AVX, AVX2, AVX512F	●	●	●
x86 32bit (i386)	SSE	●	●	●
AArch64 (arm)	Neon, SVE	●	●	N/A
AArch32 (armhf)	NEON	●	●	N/A
PowerPC (ppc64el)	VSX, VSX3	●	●	N/A
IBM/Z (s390x)	VXE, VXE2	●	●	N/A
RISC-V (riscv64)	RVV1, RVV2	●	●	N/A

# SLEEF: Keeping it up to Date

## Porting once is not sufficient

- Projects need to be maintained
- CI Infrastructure and human resources

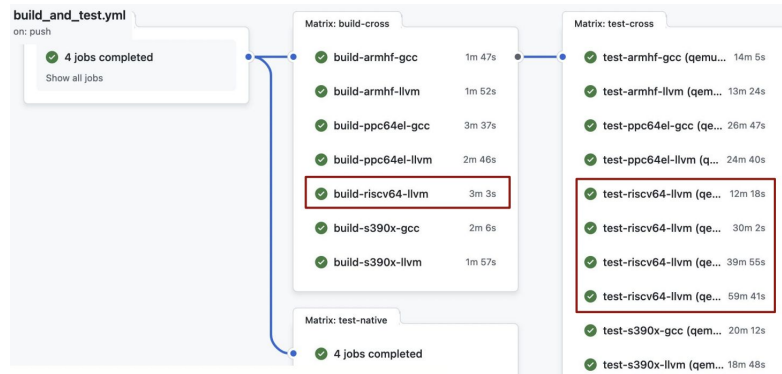
## Automated Testing

- Using QEMU to test different configs
- Using the existing SLEEF ctest

## GitHub Actions

- Build RISC-V pipelines on GitHub
- Helps prevent bitrotting port
- Add real machine when available

```
$> export QEMU_CPU="rv64,zba=true,zbb=true,zbs=true,v=true,vlen=512"
$> cd _build-riscv64 && ctest -j$(nproc)
Test project _build-riscv64
Start 60: qiutrvm2
Start 61: qiutirvmm2
Start 10: iutrvm2nofma
Start 11: iutyrvvm2nofma
1/62 Test #10: iutrvm2nofma ..... Passed 167.53 sec
Start 12: iutirvmm2nofma
2/62 Test #11: iutyrvvm2nofma ..... Passed 180.74 sec
[...]
61/62 Test #52: roundtriptest2dsp_5_15 ..... Passed 9.91 sec
62/62 Test #53: tester3printf ..... Passed 732.17 sec
100% tests passed, 0 tests failed out of 62
```



# Case Study: Python Packaging

## Why is Python Packaging needed?:

- **Many packages aren't actually in Python**
  - Just an API to C code
  - Needs to be compiled before use
  - Pure Python packages just need interpreter
- **Don't want the user building itself**
  - Specific dependencies
  - Build tools + package dependencies
- **Binary wheels built by vendors**
  - Uploaded to PyPI
- **Build Matrix**
  - Python version, OS, Arch
  - Numpy 2.1.3 = 54 wheels (no RISC-V)

## Past:

- **RISC-V not recognised architecture**
  - In build/install tools
  - pip, uv, manylinux, auditwheel, maturin
- **GitHub runners timeout with QEMU user**
  - No self hosted runners for RISC-V hardware

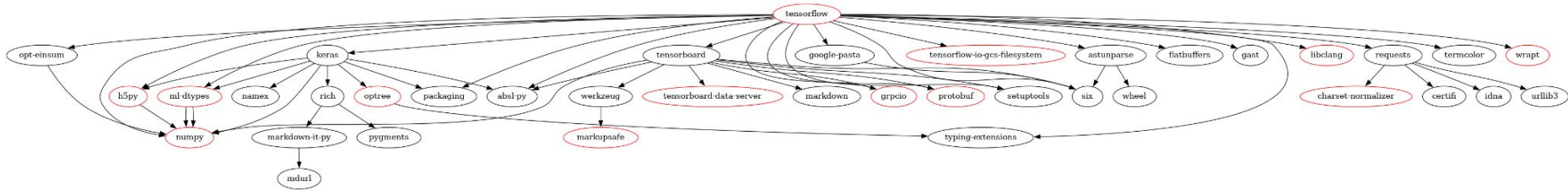
## Current:

- **Most build tools now working**
  - Pip, uv, auditwheel, maturin
  - Manylinux - Alma has no RISC-V support

## Target:

- **PyPI support it as a native platform**
  - Also CIBuildWheel

# Tensorflow Dependency Graph



13

Binary distributed package

25

Pure Python packages

# Python Packaging: RISE

[https://gitlab.com/riseproject/python/wheel\\_builder](https://gitlab.com/riseproject/python/wheel_builder)

## RISE Building Packages for you

- **Python packaging infrastructure**
  - GitLab - Repo to host wheels
  - Dedicated runner
    - Based on QEMU
    - No timeout
  - Own version of CIBuildWheel + manylinux
    - RISC-V patches
    - Based on Ubuntu
- **Building wheels for select packages**
  - Currently ~25
- **Active effort to expand package list**

## End Goal

- **None of this is needed**
- **RISC-V is mainstream architecture**

```
# uname -a
Linux 8059e28be96c 5.15.0-88-generic #98-Ubuntu SMP Mon Oct 2 15:18:56
UTC 2023 riscv64 riscv64 riscv64 GNU/Linux

# python -m pip install scipy
Collecting scipy
  Downloading scipy-1.14.1.tar.gz (58.6 MB)

Installing build dependencies ... error
error: subprocess-exited-with-error

[...]
```

```
Collecting ninja>=1.5
  Using cached ninja-1.11.1.2.tar.gz (129 kB)
ERROR: Exception:
```

```
# python -m pip install scipy --index-url
https://gitlab.com/api/v4/projects/riseproject%2Fpython%2Fwheel_builder/p
ackages/pypi/simple
Looking in indexes:
https://gitlab.com/api/v4/projects/riseproject%2Fpython%2Fwheel_builder/p
ackages/pypi/simple
Collecting scipy
  Using cached
https://gitlab.com/api/v4/projects/56254198/packages/pypi/files/2f6d46a87
1ce9f9755227c05c08d60567f35f54f34579c965784d5287c11adc3/scipy-1.12.0-cp31
0-cp310-linux_riscv64.whl (29.5 MB)
Collecting numpy<1.29.0,>=1.22.4
  Using cached
https://gitlab.com/api/v4/projects/56254198/packages/pypi/files/c221d82e0
a3e29d517a2a0542a63679a147elb4b192db1418a42069b1e2e21c8/numpy-1.26.4-cp31
0-cp310-linux_riscv64.whl (10.0 MB)
Installing collected packages: numpy, scipy
Successfully installed numpy-1.26.4 scipy-1.12.0
```



# Software Development Challenges and Opportunities

## Rise of SBCs helps porting

- Access to real hardware helps
- However often not powerful enough
- Need more CI infrastructure
- QEMU is insufficient for performance

## Platform specific optimisation

- How to develop for an architecture
- Conform to (ratified) standards
- Prevents multiple vendor conflict

## Architectural Challenges

- So many architectural extension
- RVV - Vector length agnostic
- Weak memory model

## RISE Funding Opportunities

### RISE RFPs

- Propose a software porting effort
- Bids are evaluated and funded
- Software at all layers of the stack
- [Link](#)

### RISE RISC-V Developer Appreciation

- Reward for prior porting efforts
- Existing projects to RISC-V
- €500 - €3000
- [Link](#)

## Join various work groups

# Conclusions

## Ecosystem bringup on a new arch is hard

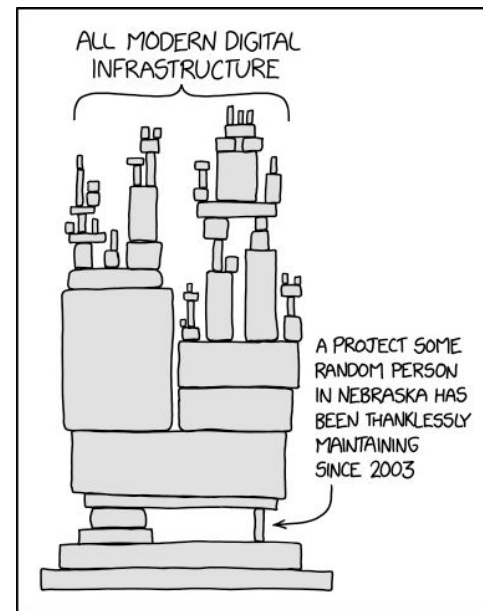
- Software dependencies are hard to track
- Hard to know what's broken until you try

## RISE has been kickstarting the effort

- Putting the infrastructure in place
- Funding work and rewarding developers

## Rivos is taking a leading position

- Commitment for open source
- Participation in consortium activities
- Heavy lifting pre-silicon means robust ecosystem



<https://xkcd.com/2347/>