Sapling: Quantifying and Measuring the Maturity of the RISC-V Software Ecosystem

Yuhang Liu*
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
liuyuhang@ict.ac.cn

Chenchen Ji*
Institute of Software, Chinese Academy of Sciences
Beijing, China
jichenchen@iscas.ac.cn

Haoquan Li
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
lihaoquan24s@ict.ac.cn

Jiageng Yu
The Institute of Software, Chinese Academy of Sciences
Beijing, China
jiageng08@iscas.ac.cn

Mingyu Chen
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
cmy@ict.ac.cn

Yanjun Wu
Institute of Software, Chinese Academy of Sciences
Beijing, China
yanjun@iscas.ac.cn

Yungang Bao
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
baoyg@ict.ac.cn

Abstract

Open-source technology has enabled RISC-V to emerge as a cost-effective alternative for processor manufacturers by reducing intellectual property expenses. A key factor influencing its broader application is the maturity of the RISC-V software ecosystem. However, quantifying this maturity remains challenging due to the complexity of the ecosystem. In this study, we introduce Sapling, an automated evaluation system designed to systematically evaluate the maturity of the RISC-V software ecosystem. Our framework categorizes the ecosystem into *five* scenarios comprising *thirty* software components, each assessed across *six* dimensions. These evaluations are then aggregated across multiple dimensions within each scenario and further synthesized to produce an overall result through a cross-scenario evaluation. Our findings indicate that while the RISC-V software ecosystem remains in its developmental phase, it is steadily advancing, demonstrating strong potential for sustained growth and broader industry application. Furthermore, *thirteen* key insights and *five* strategic recommendations are presented to advance the maturity of the RISC-V software ecosystem. We hope that this study contributes to a deeper understanding of the maturity of the RISC-V software ecosystem and offers valuable guidance for its effective enhancement.

Keywords

RISC-V, Software Ecosystem Maturity, Quantitative Evaluation

ACM Reference Format:
Yuhang Liu, Chenchen Ji, Haoquan Li, Jiageng Yu, Mingyu Chen, Yanjun Wu, and Yungang Bao. 2026. Sapling: Quantifying and Measuring the Maturity of the RISC-V Software Ecosystem. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3744916.3773168

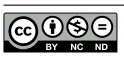
1 Introduction

The instruction set architecture (ISA) plays a crucial role in processor manufacturing, as it defines the interface between software and hardware in computer systems. RISC-V, an open-standard ISA that enables license-free, modular, and extensible processor design, has emerged as a cost-effective alternative to proprietary architectures such as x86 and ARM.

A key factor influencing the broader adoption of an ISA is the maturity of its software ecosystem. In this study, *software ecosystem maturity* is defined as the comprehensive performance of an emerging ISA software ecosystem across multiple dimensions, benchmarked against established ISAs. The RISC-V software ecosystem has garnered significant attention from the community. For example, the RISE Project [30] aims to accelerate the development of open-source software for RISC-V, while the Jiachen Initiative [3] outlines a strategic roadmap to establish a comprehensive RISC-V ecosystem by 2036, targeting open-standard systems and software stacks with maturity comparable to or surpassing mainstream architectures. Mezger et al. [26] emphasize the importance of unified testing standards as a critical step toward enhancing the reliability and scalability of the RISC-V ecosystem.

An ISA's software ecosystem comprises the applications, developer tooling, and community resources organized around the

*Yuhang Liu is the corresponding author. Both Yuhang Liu and Chenchen Ji contributed equally to this research.



This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License.
ICSE '26, Rio de Janeiro, Brazil
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2025-3/26/04
https://doi.org/10.1145/3744916.3773168

architecture. The maturity of this ecosystem is inherently multi-dimensional—covering technical capability, tooling readiness, and community vitality (with specifics detailed later). These dimensions interact and co-evolve, collectively shaping ecosystem robustness, lowering adoption friction, catalyzing developer participation, and ultimately influencing the architecture's market impact and evolutionary trajectory.

Effectively quantifying and measuring the maturity of the RISC-V software ecosystem is essential to accelerate its evolution. However, this ecosystem encompasses a broad spectrum of factors, each associated with multiple indicators of varying significance, necessitating a quantitative, objective, and comprehensive evaluation approach. Therefore, a rigorous and well-defined methodology is required to systematically assess the maturity of the RISC-V software ecosystem.

To this end, we present *Sapling*, a comprehensive system that quantifies the maturity of the RISC-V software ecosystem by integrating vertical and horizontal assessments. The name—evoking a young tree—reflects an ecosystem that is early-stage yet rapidly developing. Sapling models the ecosystem as a five-scenario hierarchy, each scenario comprising multiple software components. Evaluation proceeds in three steps: (i) vertically assess each component along defined dimensions; (ii) aggregate results to obtain scenario-level vertical scores; and (iii) perform a horizontal, cross-scenario synthesis to derive overall ecosystem maturity.

In the evaluation dimension layer, our framework examines six key dimensions.

From the vertical perspective, the comprehensive score for each scenario is calculated as the average of its software components' scores, while the overall ecosystem maturity is derived from the weighted average of all scenario scores from the horizontal perspective.

Our evaluation system is designed with the following criteria: *objectivity*, ensuring that all data originate from actual measurements or well-founded mathematical models; *transparency*, making all weight-setting rules publicly available; *comprehensiveness*, covering diverse fields and scenarios; *automation*, allowing minimal human intervention in the evaluation process; and *timeliness*, allowing data to be regularly updated as needed.

We conducted a comparative evaluation using two hardware platforms: one based on the RISC-V instruction set and the other on the ARM instruction set. Both systems ran the same operating system to ensure a consistent software environment. Our evaluation spanned multiple applications across various scenarios, measuring key performance metrics, including wall-clock execution time, CPU instruction count, cache miss rates, functional integrity, heterogeneous computing support, and ecosystem vitality. Analysis of the collected data revealed several key insights. To facilitate reproducibility, we provide a fully functional replication package, accessible at [https://github.com/everjcc/sapling-riscv-ecosystem], with a detailed description presented in the Appendix.

The main contributions of this work are as follows.

- We introduce Sapling, a novel methodology for quantifying the maturity of the RISC-V software ecosystem. To the best of our knowledge, this is the first systematic study to measure and evaluate the maturity of an ISA software ecosystem.
- We conduct comprehensive experiments and compile comparative datasets from a diverse range of software applications in multiple scenarios to validate our methodology.
- Our study provides a comprehensive analysis of the RISC-V software ecosystem, offering *thirteen* key insights and *five* strategic recommendations to guide its development.

The remainder of the paper is organized as follows: Section 2 quantifies ecosystem maturity in a single dimension; Section 3 generalizes to a multidimensional evaluation; Section 4 details the experimental setup; Section 5 reports comparative benchmarks and statistical validation; Section 6 provides cross-dimensional analysis; Section 7 offers predictive insights; Section 8 positions our work relative to the state of the art; and Section 9 concludes.

2 Single-dimensional Evaluation Methodology

We model the ISA software ecosystem as a set of scenarios $\{S_i\}$, where each scenario S_i comprises multiple software components $\{S_{ij}\}$; here S_{ij} denotes the j-th component within scenario S_i.

Our study focuses on the application software of the RISC-V ISA in datacenter environments. The official RISC-V International taxonomy [25] is not tailored for classifying datacenter application software, which leads to a significant portion being grouped into "Other" (23.42%). We therefore adopt a domain-specific taxonomy aligned with datacenter practice.

Figure 1 depicts the hierarchy of the RISC-V software ecosystem. We select five representative scenarios that reflect where RISC-V is already seeing meaningful deployment, and where ISA extensions (e.g., C for compressed, F/D for single/double-precision FP, V for vector) are directly relevant [14]. Specifically, S_1 denotes the language runtime scenario, S_2 the media & AI computing scenario, S_3 big-data processing scenario, S_4 virtualization & containerization scenario, and S_5 cloud computing and cloud-native scenario.

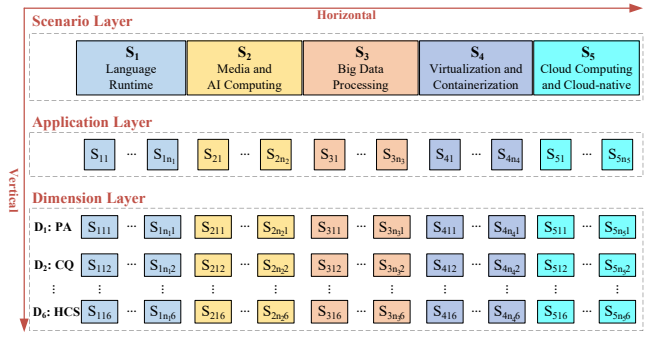


Figure 1: Hierarchy of the RISC-V Software Ecosystem

Within our evaluation framework, the above five scenarios and their associated software sets form a statistically representative sample of RISC-V's datacenter application space. The sample size and composition can be adjusted to match specific environments or time budgets without changing the methodology.

We adopt a three-stage procedure: (i) *component-level profiling*—measure each component on defined metrics under a fixed toolchain; (ii) *scenario-level analysis*—aggregate component scores

within each scenario S_i using a consistent rule; and (iii) *cross-scenario end-to-end validation*—rerun the pipeline across all S_i to verify reproducibility, parity, and robustness.

We assess software ecosystem maturity along six orthogonal dimensions: **PA** (Performance Advancement: application-level efficiency under representative workloads), **CQ** (Compilation Quality: code-generation effectiveness and correctness across toolchains), **MP** (Memory Performance: bandwidth/latency/locality behavior and memory-efficiency), **FI** (Functional Integrity: ISA/ABI conformance and stability of core implementations and runtimes), **ODA** (Open-Source Developer Activity: contributor breadth, release cadence, issue/PR throughput), and **HCS** (Heterogeneous Computing Support: first-class enablement and interoperability of GPUs/NPUs/FPGAs and their software stacks).

Rather than relying on distal indicators such as market share or patent counts, we focus on these six dimensions because they are (i) *proximal* to ISA competitiveness (capturing intrinsic architectural and ecosystem capabilities), (ii) *specific* to an open-source, community-driven ISA like RISC-V (reflecting the health and sustainability of the upstream), and (iii) *holistic* across hardware efficiency, software-stack robustness, and forward compatibility with datacenter-class workloads.

Together, these six dimensions constitute a concise and actionable evaluation framework for assessing RISC-V software ecosystem maturity, spanning technical efficiency, community health, and forward compatibility.

2.1 Dimension 1: PA

The Performance Advancement (PA) metric evaluates architectural efficiency by comparing wall-clock execution times across ISAs. We take ARM as the reference (**ISA 1**) and RISC-V as the target (**ISA 2**). Each ISA is evaluated on q test cases. For a given test case k, the *time-efficiency ratio of the target ISA* is defined from the wall-clock time T as in Equation (1).

$$RatioT_k = \frac{T_{k,\text{reference}}}{T_{k,\text{target}}} \tag{1}$$

The geometric mean of the time efficiency ratios across all test cases, as defined in Equation (2), quantifies the overall system's PA.

$$S_T = \left(\prod_{k=1}^{q} RatioT_k\right)^{\frac{1}{q}} \tag{2}$$

In our vertical maturity assessment framework, the PA dimension is assigned a normalized weighting coefficient of $w_1 = 20$, contributing proportionally to the overall maturity score of the scenario, as formulated in Equation (3).

$$PA = w_1 \cdot \min\left(S_T, 1\right) \tag{3}$$

2.2 Dimension 2: CQ

The Compilation Quality (CQ) metric evaluates architectural efficiency by analyzing instruction count discrepancies between the target architecture (RISC-V) and the reference architecture (ARM). For each test case k, the instruction count ratio is defined as follows.

$$\text{RatioN}_k = \frac{N_{k,\text{reference}}}{N_{k,\text{target}}} \tag{4}$$

where $N_{k,\,x}$ represents the dynamic instruction count when executing the test case k on ISA x. This ratio reflects compiler effectiveness; values greater than 1 indicate fewer instructions on RISC-V (advantage), whereas values below 1 indicate the opposite.

To aggregate the ratios across q test cases while reducing the influence of outliers, we employ the geometric mean aggregation method:

$$S_N = \left(\prod_{k=1}^{q} \text{RatioN}_k\right)^{\frac{1}{q}} \tag{5}$$

The contribution of the CQ dimension to the total maturity score is calculated using Equation (6), with $w_2 = 20$.

$$CQ = w_2 \cdot \min\left(S_N, 1\right) \tag{6}$$

2.3 Dimension 3: MP

The Memory Performance (MP) metric assesses the effectiveness of the cache hierarchy by analyzing L1 data cache miss behaviors. For each test case k, the cache miss ratio is calculated using Equation (7).

$$RatioM_k = \frac{M_{k,\text{reference}}}{M_{k,\text{target}}} \tag{7}$$

where $M_{k,x}$ denotes the number of L1 data-cache load misses when running test case k on ISA x. Larger values of the (ARM/RISC-V) ratio indicate fewer misses on RISC-V; smaller values imply higher cache pressure on RISC-V, suggesting further optimization of memory-access patterns or prefetching. For a comprehensive assessment of architectural adaptation, we apply geometric-mean normalization over all q test cases, as shown in Equation (8).

$$S_M = \left(\prod_{k=1}^{q} RatioM_k\right)^{\frac{1}{q}} \tag{8}$$

To incorporate MP into the overall maturity assessment, we assign it a normalized weight of w_3=20. The score representing memory performance is given by Equation (9).

$$MP = w_3 \cdot \min\left(S_M, 1\right) \tag{9}$$

2.4 Dimension 4: FI

The Functional Integrity (FI) metric measures architectural compatibility through cross-architectural validation suites. Let $a_1, a_2 \in [0, 1]$ denote the pass rates of identical functional test cases executed on ARM and RISC-V, respectively. The maturity score is then computed using Equation (10), where w_4=20 represents its 20% contribution to the total maturity score.

$$FI = \begin{cases} w_4 & \text{if } a_1 = 0 \\ w_4 \cdot \min\left(\frac{a_2}{a_1}, 1\right) & \text{if } a_1 \neq 0 \end{cases} \tag{10}$$

2.5 Dimension 5: ODA

The Open-source Development Activity (ODA) metric is measured by the number of commits within a given period. In software development and version control, when developers complete a task—such as fixing a bug or adding a new feature—they submit these changes to a version control system (e.g., Git), each associated with a unique commit identifier.

Let ISA 1 denote ARM and ISA 2 denote RISC-V. For ISA $i \in \{1, 2\}$, define b_{i1} as the cumulative number of patches submitted to date, and b_{i2} as the number of patches submitted in the most recent year (past 12 months). This dual–scale measure captures both long-run contribution volume and near-term development momentum.

With a dimension weight of $w_5 = 10$, the ODA score in Equation (11) integrates sustained development (ODA_1) with a weight of 30%, as defined in Equation (12), and current momentum (ODA_2) with a weight of 70%, as defined in Equation (13).

$$ODA = ODA_1 + ODA_2 \tag{11}$$

$$ODA_1 = \begin{cases} 0.3 \cdot w_5 & \text{if } b_{11} = 0 \\ 0.3 \cdot w_5 \cdot \min\left(\frac{b_{21}}{b_{11}}, 1\right) & \text{if } b_{11} \neq 0 \end{cases} \tag{12}$$

$$ODA_2 = \begin{cases} 0.7 \cdot w_5 & \text{if } b_{12} = 0 \\ 0.7 \cdot w_5 \cdot \min\left(\frac{b_{22}}{b_{12}}, 1\right) & \text{if } b_{12} \neq 0 \end{cases} \tag{13}$$

2.6 Dimension 6: HCS

To assess maturity in the Heterogeneous Computing Support (HCS) dimension, we compute, for each software component, the support rate of heterogeneous accelerators (e.g., GPUs, NPUs) and SIMD extensions under each ISA. Let $c_1, c_2 \in [0, 1]$ denote the support rates for ARM and RISC-V, respectively (reported as $c_i \times 100\%$). With weight $w_6 = 10$, the HCS score of the target ISA RISC-V is given in Equation (14).

$$HCS = \begin{cases} w_6 & \text{if } c_1 = 0 \\ w_6 \cdot \min\left(\frac{c_2}{c_1}, 1\right) & \text{if } c_1 \neq 0 \end{cases} \tag{14}$$

3 Multi-dimensional Evaluation Methodology

In this section, we extend the evaluation methodology in Section 2 to a multi-dimensional framework.

3.1 Software Component Maturity

For a specific software component j in scenario S_i, the comprehensive score $P_{S_{ij}}$ is calculated as shown in Equation (15).

$$P_{S_{ij}} = PA_{ij} + CQ_{ij} + MP_{ij} + FI_{ij} + ODA_{ij} + HCS_{ij} \tag{15}$$

3.2 Single Scenario Maturity

The scenario-wise maturity metric P_{S_i} aggregates the constituent indicators using an arithmetic mean composition, as formalized in Equation (16).

$$P_{S_i} = \frac{1}{n_i} \sum_{j=1}^{n_i} P_{S_{ij}} \tag{16}$$

This design assumes equal influence across all n_i software components within scenario i, ensuring an unbiased and more credible assessment of the scenario.

3.3 Overall Ecosystem Maturity

Having established scenario-specific maturity metrics through multidimensional evaluation, we now synthesize cross-scenario interactions using holistic weighting.

For a software ecosystem that has m scenarios, let $W_i \in [0, 1]$ denote the architectural significance weight assigned to scenario i, with the constraint that $\sum_{i=1}^{m} W_i = 1$. The global maturity score P is calculated as shown in Equation (17).

$$P = \sum_{i=1}^{m} W_i \cdot P_{S_i} \tag{17}$$

This weighted aggregation model captures horizontal dependencies between the five scenarios in the ISA ecosystem, where the scenario weights W_i demonstrate temporal adaptability and objective-driven reconfigurability.

The cross-scenario synthesis conducted via Equation (17) provides more than just a static score; it offers a dynamic evaluation tool for stakeholders with varying needs. Given that the architectural significance weights W_i possess temporal adaptability and objective-driven reconfigurability, evaluators can redistribute weights based on actual datacenter business priorities—such as prioritizing S_5 (Cloud Computing and Cloud-native) or S_3 (Big Data Processing). This weighted aggregation model effectively captures the horizontal dependencies within the ISA ecosystem, ensuring that the final global maturity score P truly reflects RISC-V's market competitiveness and technical readiness at a specific evolutionary stage.

4 Experimental Setup

Figure 2 outlines the detailed experiment procedure. To evaluate the maturity of RISC-V, this study adopts ARM as the primary reference architecture. Both RISC-V and ARM are of RISC style, sharing features such as fixed instruction lengths and simplified pipeline designs. These similarities result in comparable compiler optimization pathways and kernel implementation approaches. Furthermore, both architectures are increasingly employed in data centers and high-performance computing environments, where they engage in direct competition within low-power scenarios. Through a comparative analysis of RISC-V and ARM, this study aims to better assess the developmental stage of the emerging RISC-V architecture relative to a well-established RISC counterpart. Such a comparison facilitates the identification of RISC-V's position within its lifecycle and offers valuable guidance for future development trajectories.

We selected SOPHON SG2042 and Kunpeng 920-4826 as representative platforms to compare the performance between RISC-V and ARM architectures. The Kunpeng 920 processor is a server-class chip based on the ARMv8-A architecture and fabricated using a 7-nm manufacturing process. It offers high levels of on-chip integration, supporting configurations of 32, 48, and 64 cores per

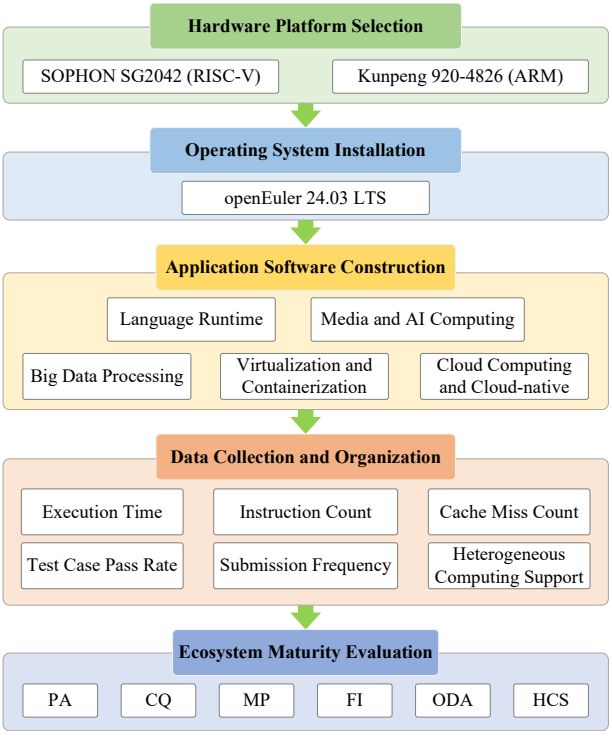


Figure 2: Experiment Procedure

CPU [1]. Built on a many-core NUMA architecture, the Kunpeng 920 provides superior single-node floating-point performance compared to the Intel Xeon E5-2680v2 and other domestically produced processors.

In SOPHON SG2042, the RISC-V CPU is housed within a Pioneer Box provided by Milk-V, and is equipped with 128GB of DDR4 RAM. As an open-source operating system, openEuler has achieved co-build compatibility for both ARM and RISC-V architectures [5]. In our experiments, the openEuler 24.03 LTS operating system is installed on the aforementioned hardware platforms [27], with specific configurations detailed in Table 1 [1, 39, 40].

Table 1: Experiment Platform

CPU	SOPHON SG2042	Kunpeng 920-4826
Processor Core	XuanTie C920	TaiShan V110
ISA	RV64GCV	ARMv8.2
CPU Core Count	64	64
CPU Frequency	2 GHz	2.6 GHz
L1 I-cache	64 KB	64 KB
L1 D-cache	64 KB	64 KB
L2 cache	1 MB shared per 4 cores	2 MB shared per 4 cores
L3 cache	128 MB	128 MB
Memory Capacity	128 GB	128 GB
OS Kernel	Linux 6.6	Linux 6.6
OS	openEuler 24.03	openEuler 24.03

In software testing, a test case is the basic unit that specifies conditions, inputs, procedures, and expected results to verify that a function or feature behaves as intended. A case typically includes input data, execution steps, and expected outcomes to assess system correctness and functional behavior. We compiled and built more than 30 software packages spanning multiple ISA-ecosystem scenarios on the Kunpeng 920 and SG2042 platforms and defined test cases for each package. The links to these cases and their counts appear in Table 2.

We evaluated the SOPHON SG2042 and Kunpeng 920-4826 platforms using the Perf Tool, which not only supports system-wide performance monitoring but also offers granular function-level data collection capabilities [10]. These evaluations were conducted through a series of application software test cases. As formalized in Section 2.4, FI quantification employs normalized test pass rates across the two architectures in comparison. We executed test cases of the application software on the SOPHON SG2042 and Kunpeng 920-4826, and collected the pass rates. The details are summarized in columns 5 and 6 of Table 2.

As described in Section 2.5, the open-source contribution statistics focus on three key factors: code patches, feature enhancements, and bug fixes submitted to software communities. Contributions containing architecture-specific keywords, such as RISC-V and ARM, are systematically filtered. Table 2 presents: (1) cumulative contributions of RISC-V and ARM until November 15, 2024 (Columns 7–8) and (2) year-to-date submissions for both architectures during the observation period 2024 (Columns 9–10).

Regarding HCS, we select statistics covering the support for heterogeneous computing units, including GPU, NPU (MLU, TPU), and SIMD (RVV, NEON), across different architecture versions of the software. The specific details are shown in Table 3 below, where "Y" indicates support and "N" indicates no support.

Based on the empirical data above, a software maturity quantifying and measuring database is formed, enabling the computation of the RISC-V ecosystem maturity score through our comprehensive evaluation oracle, Sapling. The pseudocode of the data collection and processing procedure is shown in Algorithm 1.

5 Insights from the Perspective of Application Scenario

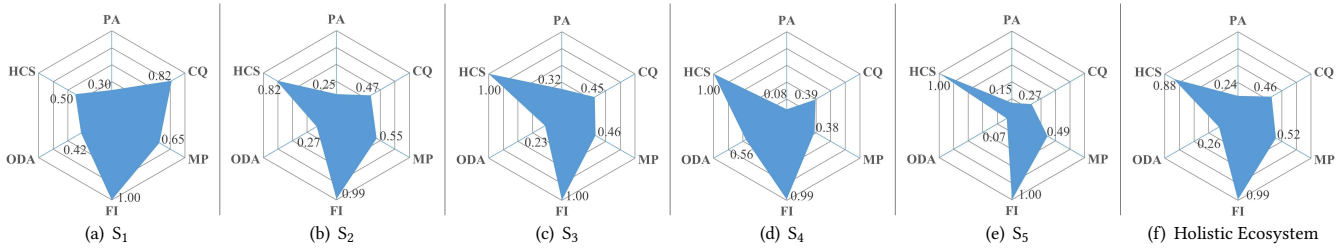
In this section, we systematically evaluate the software ecosystem maturity across five application scenarios (S_1–S_5) along six dimensions, with detailed scoring provided in Tables 4–8. To facilitate visual comparison of score distributions across dimensions, Figure 3 presents normalized radar charts for each domain. Normalization was achieved by scaling individual scores to their respective maximum values (20 or 10 points), thus ensuring a proportional representation of strengths and weaknesses.

5.1 Language Runtime Scenario

Within the S_1 scenario, Table 4 provides a systematic evaluation of software ecosystem maturity through six core dimensions. This stratified analysis reveals critical architectural limitations in language runtimes. Python demonstrates deficiencies in heterogeneous computing support across both RISC-V and ARM platforms. These limitations constrain its effectiveness in compute-intensive workloads. In contrast, OpenJDK exhibits RISC-V-specific gaps—its null score in heterogeneous support (e.g., lacking SIMD extensions analogous to ARM NEON) directly contributes to the low PA score of 4.70, significantly trailing behind other architectures. The domain achieves an aggregate maturity score of 64.66 through weighted multidimensional synthesis. While Python leads with 69.06 due to robust package management and debugging tool integration, its ODA metric of 0.50 highlights systemic community engagement challenges.

Table 2: Benchmark Suite and Performance Metrics for Maturity Evaluation

Scenario	Software	Test Cases (Link)	Count	Pass Rate		Cumulative Commits		Commits (2024)	
				RV	ARM	RV	ARM	RV	ARM
S_1	OpenJDK	https://www.spec.org/jvm2008/	20	100%	100%	357	1256	132	72
	Python	https://github.com/python/pyperformance	78	100%	100%	5	481	2	30
S_2	PyTorch	https://github.com/pytorch/examples	103	82%	84%	1	1637	1	160
	MXNet	https://github.com/apache/mxnet.git	152	97%	97%	0	50	0	0
	Caffe	https://github.com/BVLC/caffe.git	50	100%	100%	0	7	0	0
	OpenCV	https://github.com/opencv/opencv_zoo	115	95%	95%	57	338	17	32
	OpenBLAS	https://github.com/OpenMathLib/OpenBLAS	196	100%	100%	108	436	40	31
	MNN	https://github.com/alibaba/MNN/tree/2.9.3	342	98%	98%	0	75	0	2
	FFmpeg	https://github.com/FFmpeg/FFmpeg/tree/n7.0.1	146	95%	95%	111	1669	1	31
	scikit-learn	https://github.com/scikit-learn/scikit-learn/tree/1.5.1	382	95%	97%	0	236	0	37
	MLlib	https://github.com/apache/spark/tree/v3.5.1	266	89%	91%	1	834	0	21
	CNTK	https://github.com/microsoft/CNTK/tree/v2.6	69	61%	67%	0	45	0	0
	Mahout	https://github.com/apache/mahout/tree/mahout-14.1	61	100%	100%	0	7	0	0
	ONNX	https://github.com/onnx/models/tree/main	227	94%	95%	0	24	0	1
	TensorFlow	https://github.com/tensorflow/models	38	71%	74%	30	1015	0	66
	x264	https://ffmpeg.org//releases/ffmpeg-7.0.1.tar.gz	106	100%	100%	2	231	0	3
	x265	https://ffmpeg.org//releases/ffmpeg-7.0.1.tar.gz	175	100%	100%	0	197	0	98
S_3	Hadoop	https://github.com/apache/hadoop	69	86%	86%	0	256	0	14
	Kafka	https://github.com/apache/kafka	72	100%	100%	0	286	0	53
	MySQL	https://github.com/mysql/mysql-server	126	100%	100%	0	375	0	8
	openGauss	https://sourceforge.net/projects/benchmarksql	207	88%	88%	0	18	0	0
	Redis	https://github.com/redis/redis	454	98%	98%	1	85	0	3
	MariaDB	https://github.com/MariaDB/server	125	98%	98%	14	581	5	18
	Storm	https://github.com/apache/storm/tree/master/examples	72	100%	100%	0	6	0	0
S_4	iSulad	https://github.com/openeuler-mirror/iSulad	49	98%	100%	5	5	0	0
	Docker	https://github.com/moby/moby/tree/master	1375	99%	100%	24	1852	8	49
S_5	Grafana	https://github.com/grafana/grafana	2160	100%	100%	0	184	0	13
	Kubernetes	https://github.com/kubernetes/kubernetes	9198	100%	100%	3	938	1	33
	Helm	https://github.com/helm/helm	670	100%	100%	2	28	1	3
	Prometheus	https://github.com/prometheus/prometheus	1184	100%	100%	1	72	0	13



(a) S_1 (b) S_2 (c) S_3 (d) S_4 (e) S_5 (f) Holistic Ecosystem

Figure 3: Radar Chart Depicting the Maturity of the RISC-V Software Ecosystem

This disparity indicates over-reliance on commercial maintenance versus grassroots contributor participation. Our analysis, as illustrated in Figure 3(a), reveals that within the language runtime scenario RISC-V exhibits robust FI (1.00/1.00) and CQ (0.82/1.00), but shows significant deficiencies in PA (0.30/1.00) and ODA (0.42/1.00).

In-depth analysis of the S_1 scenario demonstrates that functional integrity does not necessarily equate to ecosystem maturity. Although RISC-V exhibits nearly perfect functional integrity (FI) in this scenario, proving its operational reliability, the extremely low Performance Advancement (PA) scores reveal that the software stack has yet to fully exploit the ISA's hardware potential. For instance, while Python benefits from robust package management, its execution efficiency remains constrained by scalar implementations due to a lack of upstream support for the RISC-V Vector (RVV) extension. Transitioning from generic implementations to ISA-aware deep optimizations will be the critical path for narrowing the performance gap between RISC-V and established architectures like ARM.

Table 3: Heterogeneous Computing Support (HCS)

	Software	RISC-V HCS				ARM HCS			
		GPU	MLU	TPU	RVV	GPU	MLU	TPU	NEON
S_1	OpenJDK	N	N	N	N	N	N	N	Y
	Python	N	N	N	N	N	N	N	N
S_2	PyTorch	Y	Y	Y	N	Y	Y	Y	Y
	MXNet	N	N	N	N	N	N	N	Y
	Caffe	Y	N	N	N	Y	N	N	N
	OpenCV	Y	Y	Y	Y	Y	Y	Y	Y
	OpenBLAS	N	N	N	N	N	N	N	Y
	MNN	Y	Y	Y	N	Y	Y	Y	Y
	FFmpeg	Y	N	N	Y	Y	N	N	Y
	scikit-learn	N	N	N	N	N	N	N	N
	MLlib	N	N	N	N	N	N	N	N
	CNTK	Y	N	N	N	Y	N	N	N
	Mahout	N	N	N	N	N	N	N	N
	ONNX	Y	Y	Y	N	Y	Y	Y	Y
	TensorFlow	Y	Y	Y	Y	Y	Y	Y	Y
	x264	N	N	N	Y	N	N	N	Y
	x265	N	N	N	Y	N	N	N	Y
S_3	Hadoop	N	N	N	N	N	N	N	N
	Kafka	N	N	N	N	N	N	N	N
	MySQL	N	N	N	N	N	N	N	N
	openGauss	N	N	N	N	N	N	N	N
	Redis	N	N	N	N	N	N	N	N
	MariaDB	N	N	N	N	N	N	N	N
	Storm	N	N	N	N	N	N	N	N
S_4	iSulad	N	N	N	N	N	N	N	N
	Docker	N	N	N	N	N	N	N	N
S_5	Grafana	N	N	N	N	N	N	N	N
	Kubernetes	N	N	N	N	N	N	N	N
	Helm	N	N	N	N	N	N	N	N
	Prometheus	N	N	N	N	N	N	N	N

Table 4: Ecosystem Maturity of Language Runtime Software

Software	PA	CQ	MP	FI	ODA	HCS	Total Score
OpenJDK	4.70	12.86	14.84	20.00	7.85	0.00	60.25
Python	7.23	20.00	11.33	20.00	0.50	10.00	69.06
Avg.	5.97	16.43	13.09	20.00	4.18	5.00	64.66

Insight 1: In the language runtime scenario, RISC-V offers solid foundational support but lags in both performance and community adoption. The main limitation lies in the lack of RISC-V-specific optimizations in open-source patches, which directly impacts runtime efficiency. For example, *OpenJDK* lacks architecture-specific enhancements such as assembly-level support for ChaCha20, transcendental functions, memory operations, and immediate value handling; *Python* lacks architecture-aware acceleration and upstream support for key extensions, with no vectorized backend (e.g., *hacl*) utilizing the RISC-V Vector Extension (RVV), thus falling back to less efficient scalar C implementations. At the hardware level, optimizing and adopting the J-extension is also critical for narrowing the performance gap and achieving competitive runtime efficiency.

5.2 Media and AI Computing Scenario

S_2 refers to the scenario of media and AI computing. Table 5 presents the detailed ecological maturity scores of the software in this scenario.

The composite score for this scenario is 55.91, indicating that ecological maturity is at a moderate level, with significant variations in performance across the different software. Among the 15 open source software packages selected for testing, 60% of the software achieved a comprehensive score above 50, suggesting that the ecological maturity of the media and AI computing scenario is relatively comparable to ARM.

For a specific analysis, OpenCV has a comprehensive score of 76.82, demonstrating well-rounded performance across all dimensions, while OpenBLAS scored 29.52, with relatively low marks for compilation quality and memory performance, which negatively impacted the overall score for this scenario. In terms of functional integrity, the scores are nearly perfect and are comparable to those of the ARM ecosystem. However, there is room for improvement in performance, particularly in areas such as more efficient compiler optimization, instruction pipeline involvement, and the implementation of higher-performance processors to facilitate collaborative optimization of RISC-V hardware and software.

Figure 3(b) shows the ecological scoring radar chart for this scenario. RISC-V excels in HCS (0.82/1.00) and has nearly perfect FI (0.99/1.00), highlighting strong compiler support and robust functionality. However, it shows weaknesses in PA (0.25/1.00) and ODA (0.27/1.00). Moderate scores in CQ (0.47/1.00) and MP (0.55/1.00) suggest areas that need improvement to improve competitiveness and increase adoption.

Insight 2: In the scenario of media and AI computing, RISC-V faces three critical challenges that require coordinated hardware-software solutions. First, overcoming the dominance of NVIDIA's CUDA ecosystem necessitates the development of open-source acceleration stacks to reduce reliance on proprietary toolchains. Second, integrating specialized co-processors—such as Machine Learning Units (MLUs) and Tensor Processing Units (TPUs)—is vital for accelerating AI workloads and enabling efficient heterogeneous computing. Third, the current limitations of the RISC-V Vector Extension (RVV) in SIMD capabilities, particularly when compared to ARM's mature NEON-based multimedia optimizations, must be addressed.

5.3 Big Data Processing Scenario

S_3 refers to the big data processing scenario. Table 6 presents the detailed ecological maturity scores of the software components in this scenario. S_3 attains a composite maturity score of 56.86. Among the seven open source software packages selected for testing, more than half of the software achieved a comprehensive score above 50, demonstrating initial maturity in the big data processing domain comparable to ARM.

In terms of dimension-specific observations, FI and HCS reach full scores, indicating maturity in these dimensions. However, the HCS score is full due to the symmetrical lack of heterogeneous support on both RISC-V and ARM, rather than technological advantages, underscoring further enhancement for both architectures.

Algorithm 1 RISC-V Ecosystem Maturity Evaluation

Require: Benchmark suite BS, RISC-V platform, ARM platform, user-defined weights $w_1, \ldots, w_6$
Ensure: PA, CQ, MP, FI, ODA, HCS

// The procedure consists of 4 main steps.
// Step 1: Lines 3-11 are used to calculate PA, CQ and MP
Count the duration time of test cases for BS on both architectures as $T_{k,\text{RISC-V}}$ and $T_{k,\text{ARM}}$.
Count the instruction count of test cases for BS on both architectures as $N_{k,\text{RISC-V}}$ and $N_{k,\text{ARM}}$.
Count the L1 dcache load misses of test cases for BS on both architectures as $M_{k,\text{RISC-V}}$ and $M_{k,\text{ARM}}$.
$S_T \leftarrow$ geometric mean of $\left(T_{k,\text{ARM}}/T_{k,\text{RISC-V}}\right)$
$S_N \leftarrow$ geometric mean of $\left(N_{k,\text{ARM}}/N_{k,\text{RISC-V}}\right)$
$S_M \leftarrow$ geometric mean of $\left(M_{k,\text{ARM}}/M_{k,\text{RISC-V}}\right)$
$PA \leftarrow \min(w_1,\ w_1 \cdot S_T)$
$CQ \leftarrow \min(w_2,\ w_2 \cdot S_N)$
$MP \leftarrow \min(w_3,\ w_3 \cdot S_M)$
// Step 2: Lines 13-21 are used to calculate FI
Count the passed test cases for BS on both architectures as $Q_{\text{RISC-V}}$ and Q_{ARM}.
Collect the total number of test cases as Q_{total}.
$R_{P,\text{RISC-V}} \leftarrow Q_{\text{RISC-V}}/Q_{\text{total}}$ *(pass rate of RISC-V)*
$R_{P,\text{ARM}} \leftarrow Q_{\text{ARM}}/Q_{\text{total}}$ *(pass rate of ARM)*
if $R_{P,\text{ARM}} == 0$ **then**
  $FI \leftarrow w_4$
else
  $FI \leftarrow \min(w_4,\ w_4 \cdot R_{P,\text{RISC-V}}/R_{P,\text{ARM}})$
end if
// Step 3: Lines 23-36 are used to calculate ODA
Collect the total number of commits for both architectures in the official project of BS as $C_{\text{RISC-V}}$ and C_{ARM}.
Count the past year's commits for both architectures in the official BS project as $RecentC_{\text{RISC-V}}$ and $RecentC_{\text{ARM}}$.
if $C_{\text{ARM}} == 0$ **then**
  $ODA_1 \leftarrow 0.3 \cdot w_5$
else
  $ODA_1 \leftarrow \min\left(0.3 \cdot w_5,\ 0.3 \cdot w_5 \cdot C_{\text{RISC-V}}/C_{\text{ARM}}\right)$
end if
if $RecentC_{\text{ARM}} == 0$ **then**
  $ODA_2 \leftarrow 0.7 \cdot w_5$
else
  $Ratio \leftarrow RecentC_{\text{RISC-V}}/RecentC_{\text{ARM}}$
  $ODA_2 \leftarrow \min\left(0.7 \cdot w_5,\ 0.7 \cdot w_5 \cdot Ratio\right)$
end if
$ODA \leftarrow ODA_1 + ODA_2$
// Step 4: Lines 38-46 are used to calculate HCS
Count the heterogeneous supports for BS on both architectures as $HCSN_{\text{RISC-V}}$ and $HCSN_{\text{ARM}}$.
Collect the total number of heterogeneous features in BS as HCS_{total}.
$HCSR_{\text{RISC-V}} \leftarrow HCSN_{\text{RISC-V}}/HCS_{\text{total}}$
$HCSR_{\text{ARM}} \leftarrow HCSN_{\text{ARM}}/HCS_{\text{total}}$
if $HCSR_{\text{ARM}} == 0$ **then**
  $HCS \leftarrow w_6$
else
  $HCS \leftarrow \min\left(w_6,\ w_6 \cdot HCSR_{\text{RISC-V}}/HCSR_{\text{ARM}}\right)$
end if

Table 5: Ecosystem Maturity of Media and AI Computing Software

Software	PA	CQ	MP	FI	ODA	HCS	Total Score
PyTorch	3.69	6.73	8.74	19.31	0.05	7.50	46.02
MXNet	6.78	6.64	9.84	20.00	7.00	0.00	50.26
Caffe	6.37	0.45	2.63	20.00	7.00	10.00	46.45
OpenCV	6.67	16.50	19.43	20.00	4.22	10.00	76.82
OpenBLAS	1.11	0.08	0.63	20.00	7.70	0.00	29.52
MNN	5.15	11.63	13.72	19.94	0.00	7.50	57.94
FFmpeg	7.00	16.30	20.00	20.00	0.67	10.00	73.97
scikit-learn	10.19	16.90	12.14	19.62	0.00	10.00	68.85
MLlib	4.34	13.37	15.12	19.67	0.00	10.00	62.50
CNTK	3.14	13.38	12.32	18.26	7.00	10.00	64.10
Mahout	0.19	1.65	3.82	20.00	7.00	10.00	42.66
ONNX	4.79	7.15	7.54	19.81	0.00	7.50	46.79
TensorFlow	4.64	6.85	6.30	19.29	0.09	10.00	47.17
x264	5.34	12.36	15.56	20.00	0.03	10.00	63.29
x265	5.12	10.70	16.42	20.00	0.00	10.00	62.24
Avg.	4.97	9.38	10.95	19.73	2.72	8.17	55.91

Analysis on PA reveals persistent gaps in computational efficiency, particularly evidenced by Storm's 1.40 and openGauss's 2.19 scores. These findings collectively indicate that while RISC-V achieves foundational maturity in big data processing, bridging gaps with comprehensive maturity necessitates refinements targeting PA, CQ, MP, ODA and HCS.

Figure 3(c) shows the ecological scoring radar chart for scenario S_3. In the domain of big data processing, RISC-V achieves perfect scores in FI (1.00/1.00), indicating strong capabilities in these areas. However, it shows weaknesses in ODA (0.23/1.00) and CQ (0.45/1.00). Moderate scores on PA (0.32/1.00) and MP (0.46/1.00) suggest that there is room for improvement in these dimensions.

Insight 3: In the scenario of big data processing, performance can be significantly enhanced through a combination of software and hardware optimizations. At the software level, improvements are achieved by optimizing source code and employing instruction set extensions—such as those for sorting, hashing, and compression—to accelerate key operations. MP scores can be further improved by implementing cache partitioning strategies for shuffle data and utilizing prefetching techniques.

5.4 Virtualization and Containerization Scenario

S_4 refers to the virtualization and containerization scenario. Table 7 presents the detailed ecological maturity scores of the software components in this scenario.

The overall score for this scenario is 52.28. Our comparative analysis between iSulad and Docker in the RISC-V ecosystem reveals distinct performance characteristics: iSulad achieves a higher composite score of 65.66 compared to Docker, suggesting better architectural alignment with RISC-V specifications. However, both exhibit notable shortcomings in computational efficiency optimization, primarily attributed to insufficient low-level performance tuning, indicating substantial potential for improvement in this dimension.

Table 6: Ecosystem Maturity of Big Data Processing Software

Software	PA	CQ	MP	FI	ODA	HCS	Total Score
Hadoop	8.19	11.69	11.24	20.00	0.00	10.00	61.12
Kafka	6.41	10.45	13.32	20.00	0.00	10.00	60.18
MySQL	6.17	20.00	20.00	20.00	0.00	10.00	76.17
openGauss	2.19	5.68	4.24	20.00	7.00	10.00	49.11
Redis	3.34	3.40	3.02	20.00	0.04	10.00	39.80
MariaDB	16.73	8.15	10.05	20.00	2.02	10.00	66.95
Storm	1.40	3.35	2.92	20.00	7.00	10.00	44.67
Avg.	6.35	8.96	9.26	20.00	2.29	10.00	56.86

Table 7: Ecosystem Maturity of Virtualization and Containerization Software

Software	PA	CQ	MP	FI	ODA	HCS	Total Score
iSulad	1.76	12.99	11.32	19.59	10.00	10.00	65.66
Docker	1.31	2.57	3.98	19.85	1.18	10.00	38.89
Avg.	1.54	7.78	7.65	19.72	5.59	10.00	52.28

Although composite score suggests an intermediate maturity level of RISC-V, critical gaps persist in PA, CQ and other dimensions despite FI, where HCS's full score still attributes to the symmetrical lack of heterogeneous support on both ecosystems. Through further optimization of the toolchain and enhanced contributions from the open-source community, this ecosystem could be significantly improved.

Figure 3(d) presents the ecological scoring radar chart for this scenario. RISC-V demonstrates exceptional performance in HCS with a perfect score of 1.00/1.00 and nearly perfect FI at 0.99/1.00, highlighting its robust capabilities in these areas. However, it exhibits significant weaknesses in PA, scoring only 0.08/1.00, and CQ, with a score of 0.39/1.00. The ODA and MP metrics are moderate, achieving scores of 0.56/1.00 and 0.38/1.00, respectively.

Insight 4: In the scenario of virtualization and containerization, performance bottlenecks in RISC-V systems are primarily attributed to the lack of hardware-assisted virtualization support (i.e., the H-extension). This limitation fundamentally restricts hypervisor efficiency and impedes the scalability of virtualized workloads.

5.5 Cloud Computing and Cloud-native Scenario

S_5 refers to the cloud computing and cloud-native scenario. Table 8 presents the detailed ecological maturity scores of the software components in this scenario.

The composite score for this scenario is 48.92. The ecological maturity in the cloud computing and cloud-native scenario is relatively low within the RISC-V ecosystem, and there is significant room for improvement in the listed software. While the scenario performs well in terms of FI, there is still a need for substantial improvements in performance optimization, compilation quality, open-source community engagement and heterogeneous computing support which gets confusing full scores due to the same reason.

Table 8: Ecosystem Maturity of Cloud Computing and Cloud-native Software

Software	PA	CQ	MP	FI	ODA	HCS	Total Score
Grafana	1.57	3.82	3.49	20.00	0.00	10.00	38.88
Kubernetes	4.68	6.82	13.40	20.00	0.22	10.00	55.12
Helm	0.97	3.04	4.95	20.00	2.55	10.00	41.51
Prometheus	4.51	8.26	17.53	20.00	0.04	10.00	60.16
Avg.	2.93	5.49	9.84	20.00	0.70	10.00	48.92

Only with more resources invested into these dimensions will it be possible to further advance the maturity and widespread adoption of the cloud computing and cloud-native scenario within the RISC-V ecosystem. Figure 3(e) shows the ecological scoring radar chart for this scenario. In the scenario of cloud computing and cloud-native, RISC-V achieves perfect scores in FI, indicating strong capabilities in these areas. However, it shows significant weaknesses in ODA (0.07/1.00) and CQ (0.27/1.00). Additionally, moderate Memory Performance (0.49/1.00) and very low PA at 0.15/1.00 highlight critical areas for improvement.

Insight 5: In the scenario of cloud computing and the cloud-native, suboptimal performance and limited open-source contributions jointly hinder the maturity of the RISC-V ecosystem. These limitations are evident in two key areas: (1) inadequate language runtime support, requiring further research on ISA extensions with runtime detection, compiler backend optimizations, and standard library enhancements for key languages such as Golang and OpenJDK; and (2) limited architecture-specific adaptation in cloud-native software, where compatibility is primarily achieved through patch-based modifications. In contrast, full native support for architectures like ARM eliminates the need for runtime emulation, thereby reducing performance overhead.

5.6 Summarizing the Five Scenarios

As illustrated in Figure 3(f), this study aggregates the average scores across six evaluation dimensions for 30 software applications spanning five scenarios (S_1–S_5). The composite results are visualized using a normalized radar chart. RISC-V attains near-perfect scores in FI (0.99/1.00), demonstrating strong capabilities in this area. However, it exhibits weaknesses in ODA (0.26/1.00) and PA (0.24/1.00). Meanwhile, its moderate scores in CQ (0.46/1.00) and MP (0.52/1.00) indicate room for further improvement in these aspects.

Insight 6: RISC-V's overall maturity is constrained by notable deficiencies in PA and ODA, as well as suboptimal performance in CQ and MP. Future efforts to enhance RISC-V maturity should focus on addressing these low-performing areas to narrow the maturity gap while capitalizing on its strengths in specialized scenarios. By leveraging RISC-V's instruction set extensibility and hardware-software co-design capabilities, we propose the following enhancement strategies across critical dimensions: (1) source code optimization, (2) ISA extension co-optimization, (3) memory hierarchy innovation, (4) parameter tuning framework.

6 Insights from the Perspective of Six Dimensions

In this section, from the perspective of six dimensions, we present our analysis and the main findings.

6.1 Performance Advancement

With 57% of the 30 software packages scoring PA below 5, developing high-performance processor cores and improving system-on-chip design are crucial steps toward enhancing the maturity of the RISC-V ecosystem in the future.

Insight 7: RISC-V attains 0.24 of ARM's baseline performance in PA dimension. This indicates that, in terms of performance enhancement or optimization, RISC-V currently lags significantly behind ARM, suggesting a need for improvement in this critical area to enhance its competitiveness. The PA currently represents the weakest dimension in the evaluation framework, with multiple software systems demonstrating suboptimal scores in this critical dimension. This pattern highlights persistent deficiencies in performance optimization for the RISC-V ecosystem.

6.2 Compilation Quality

With 27% of the 30 software packages attaining a CQ score below 5, and 53% having a CQ score below 10, our results reveal a statistically significant gap in compilation efficiency between RISC-V and ARM. Improving the adaptive ISA extension and compiler optimization framework is paramount for improving compilation quality [7].

Insight 8: RISC-V achieves 46% of ARM's baseline performance in the CQ dimension. Improving compiler effectiveness requires several coordinated efforts: integrating RISC-V–specific optimization passes into mainstream toolchains (e.g., enhanced register allocation, RVV-aware vectorization); enabling mature support for advanced techniques such as Profile-Guided Optimization (PGO) and Link-Time Optimization (LTO); strengthening JIT compiler infrastructure for language runtimes like *OpenJDK* and *Python*; and tuning critical runtime libraries (e.g., `libc`, `libm`) with low-level assembly optimizations. Additionally, systematically deploying auto-vectorization can reduce dynamic instruction counts by converting scalar operations into SIMD-parallel execution, thereby improving compilation quality for RISC-V's variable-length vector extension (RVV). In parallel, broader upstream engagement in GCC/LLVM and the adoption of ML-assisted tuning approaches (e.g., via MLIR) will be key to enhancing long-term compiler robustness and performance.

6.3 Memory Performance

Detailed analysis finds that 27% of the 30 software packages have an MP score below 5 and 43% have an MP score below 10. The distribution exhibits considerable divergence between the two ecosystems. To bridge the gap, memory hierarchy optimization, prefetch mechanisms, and other optimization methods are of importance.

Insight 9: RISC-V scores 0.52 on the MP dimension relative to ARM, indicating a moderate level of efficiency in handling memory operations. To improve this, efforts beyond core development are needed. The open-source community should prioritize the design of advanced memory hierarchies with competitive PPA characteristics—namely, high performance, low power consumption, and efficient area utilization. Key components include optimized cache replacement policies, prefetchers and memory controllers, which are critical for enhancing system-wide memory efficiency.

6.4 Functional Integrity

It is seen that 70% of the 30 software packages have a full FI score, and 96% have an FI score greater than 19, which shows that RISC-V can serve as a sustainable alternative solution for ARM in diverse scenarios.

Insight 10: RISC-V scores 0.99 on FI relative to ARM. This nearly perfect score suggests that RISC-V excels in ensuring the correctness and reliability of its operations, almost matching ARM's performance in functional integrity.

6.5 Open-source Development Activity

In terms of ODA, RISC-V attains a relatively lower score than most influence factors. Detailed analysis reveals some remarkable insights: a) The score combines both historical total submissions and contributions from the past year, and since ARM was introduced earlier than RISC-V, ARM has significantly higher historical submission volumes, which brings it advantage in the historical activity. b) As shown in Table 2, in the past year, contributions to OpenJDK and OpenBLAS for RISC-V are inferior to ARM, but OpenCV's contribution was about double of ARM's, suggesting that RISC-V's ODA is closing the gap with ARM's. c) For most software, RISC-V has a much lower submission count compared to ARM, indicating a need for more collaboration among researchers across various scenarios of the ecosystem. Open-source communities can further engage developers by hosting open contests and events.

Insight 11: RISC-V scores 0.26 on the ODA dimension relative to ARM, indicating significantly lower community engagement and contribution. To address this weakness, we recommend establishing a structured, goal-driven incentive mechanism for open-source development, whereby specific tasks are clearly defined and responsibility is assigned to designated contributors or organizations. In addition, implementing quantitative evaluation mechanisms for project quality and contributor influence can help foster a more active and impactful open-source ecosystem.

6.6 Heterogeneous Computing Support

HCS is crucial in data centers and high-performance computing, as it can provide higher computational power and accelerate processing speeds. Although RISC-V demonstrates performance comparable to that of ARM in specific scenarios of its HCS, this does not equate to RISC-V's HCS having reached a mature stage. As illustrated in Table 3, even ARM exhibits limited support coverage for test cases in our evaluation. Consequently, RISC-V's HCS currently

approximates ARM's baseline capabilities but still faces a considerable gap in achieving a robust and mature software ecosystem. RISC-V needs to enhance its heterogeneous computing support, particularly with GPU, TPU, and similar technologies.

Insight 12: RISC-V scores 0.88 on the HCS dimension relative to ARM, indicating that both architectures exhibit comparable capabilities in supporting heterogeneous computing workloads. ARM's advantage stems from decades of commercial deployment and sustained investment in optimizing hardware-software co-design for diverse workloads across CPUs, GPUs, and specialized accelerators. In contrast, RISC-V is converging through ecosystem-driven efforts, such as the Heterogeneous Computing Working Group, which standardizes interfaces, defines open acceleration frameworks, and promotes hardware abstraction layers.

7 Insights from the Perspective of Future Prediction

In this section, from the perspective of future prediction, we investigate the following research question: How long will it take for RISC-V to achieve the same level of ecosystem maturity as ARM?

If the current maturity score is P, the full score is F and the growth rate is r, the condition to reach the full score in n years is given by the following equation Equation (18).

$$P \cdot (1+r)^n \geqslant F \tag{18}$$

Then the value of n can be calculated, with which we predict the future of RISC-V. The results are shown in Table 9. If the points for five dimensions continue to grow at a rate of 10% per year, PA and ODA are expected to reach full points in approximately 15 years, while CQ is expected to achieve a full point in 8 years. FI is already nearing the full point, and HCS is expected to reach the full score in 2 years.

Insight 13: The adoption of RISC-V processors in PCs and servers is expected to drive the development of an ecosystem with sufficient scale, tooling, and vendor support to competitively challenge the ARM architecture. If the weaknesses identified in this study are effectively addressed, RISC-V is poised to make a significant impact on the computing landscape by 2030.

Based on our findings, we present five action-oriented recommendations to accelerate the mainstream adoption of RISC-V.

- **Exploit ISA Flexibility**: Prioritize RVV/J–aware optimizations for hot kernels (e.g., GEMM/FFT/conv, hashing/compression); enable PGO/LTO by default; publish a minimal *continuous-integration (CI)* dashboard to track gains across S_1–S_5.
- **Target HPC/Server Markets**: Co-design accelerators with vector and J extensions; ship two reference pipelines (video transcode, LLM inference) that run natively on RISC-V with open drivers and reproducible builds.
- **Leverage Multidimensional Profiling**: Use C-AMAT, CPI-stack, and roofline to localize HW–SW mismatches; integrate regression guards (e.g., wall-time, cache-miss thresholds) into the scenario-level *CI pipeline*.
- **Strengthen Open-Source Collaboration**: Label RISC-V issues and mentorship tickets; adopt contributor guidelines and SLAs (e.g., median time-to-merge); expand conformance/fuzzing to prevent FI regressions.
- **Build Cross-Stack Synergy**: Define versioned cross-stack profiles aligning ISA, microarchitecture knobs, toolchains, and container images to enable predictable performance portability.

Table 9: Projections on the Time Required for the RISC-V Ecosystem to Reach a Maturity Comparable to ARM's

Dimension	Current Score	Time Required (years)		
		(r = 10%)	(r = 20%)	(r = 30%)
PA	4.86	14.85	7.76	5.40
CQ	9.13	8.23	4.30	2.99
MP	10.33	6.93	3.62	2.52
FI	19.84	0.08	0.04	0.03
ODA	2.64	13.98	7.31	5.08
HCS	8.75	1.40	0.73	0.51
Overall	55.54	6.17	3.23	2.24

8 Related Work

Over the past decade, numerous studies have investigated various aspects of the RISC-V ecosystem. Much of this research has focused on specific areas, including system-level security [15, 23, 24, 31, 33, 36], open-source development efforts [2, 6, 11, 12], and ecosystem-wide surveys [18, 26, 32]. However, to the best of our knowledge, no prior work has systematically modeled and quantified the maturity of the RISC-V ecosystem across multiple dimensions.

Maturity models developed within the software engineering community provide useful insights but are typically tailored to specific scenarios. For example, the SEG-M^2 model by Jansen [17] focuses on software governance, while Houaich and Belaissaoui [16] proposed E-OSS to evaluate the maturity of free software. Petrinja et al. [29] introduced the OSMM framework for FLOSS development, and Cukier and Kon [8] proposed a startup ecosystem maturity model based on multiple case studies. While informative, these models rely on narrowly scoped indicators and are not easily generalizable to architecture-level ecosystems, especially those built around open ISAs such as RISC-V [4].

Recent studies comparing ARM and RISC-V reveal performance and implementation gaps. In containerized HPC deployments, ARM often outperforms RISC-V in throughput and operational simplicity, highlighting limitations of the latter [9]. Fibich and Tauner [13] benchmarked BLAS libraries on embedded platforms, showing the benefits of ARM's SIMD extensions and noting hardware constraints that impede general-purpose tuning on certain RISC-V targets. Suárez et al. [37] observe that the lower power draw of RISC-V does not reliably yield a higher performance-per-watt. Other studies [21, 22, 28, 34] report a higher emulation overhead and a lower

instruction efficiency for RISC-V relative to ARM. Nonetheless, Weaver [38] reports near-parity on scalar workloads, and Kim et al. [19, 20] achieve state-of-the-art cipher throughput on both ISAs with non-SIMD optimizations.

In contrast, our proposed framework, Sapling, introduces a scenario-agnostic methodology for evaluating ecosystem maturity, applicable to both general-purpose software environments and ISA-specific ecosystems. Unlike previous efforts that focus on single dimensions—such as the number of supported packages [26] or developer engagement [35]—Sapling employs a multi-dimensional evaluation strategy based on six rigorously selected metrics.

Moreover, whereas prior models often produce static or domain-constrained assessments, Sapling offers a scalable and extensible framework capable of capturing both current ecosystem adaptability and future evolutionary potential. In addition to maturity scores, it provides a strategic roadmap and gap analysis to support actionable decision-making for ecosystem stakeholders.

In summary, while earlier research has provided valuable insights—particularly into software coverage and developer activity—this work presents the first holistic, multi-dimensional, and architecture-aware assessment of the RISC-V software ecosystem.

9 Conclusions

In this study, we systematically quantify the maturity of the RISC-V software ecosystem through *Sapling*, an automated evaluation framework that integrates dynamic vertical analysis in six evaluation dimensions with horizontal stratification in five application scenarios. By leveraging adaptive weighting mechanisms and multi-source data fusion, Sapling constructs a comprehensive evaluation model that captures both ecosystem complexity and evolutionary trends. Our empirical analysis reveals a dualistic landscape: while RISC-V exhibits notable strengths in functional integrity verification and heterogeneous computing adaptability, critical deficiencies persist in core performance optimization, compilation toolchain compatibility, and open-source community engagement. These findings establish the first systematic, quantitative baseline for assessing RISC-V software ecosystem maturity and inform five strategic recommendations. Furthermore, the open-source implementation of Sapling provides a scalable infrastructure for continuous ecosystem monitoring, enabling stakeholders to track progress across hardware-software co-design boundaries and make data-driven decisions for ecosystem development.

Ultimately, we envision Sapling serving not merely as a static metric, but as a dynamic compass for the community, guiding resource allocation toward critical deficiencies.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. This work is supported by the National Key Research and Development Program of China under Grant No. 2023YFB4503904.

References

[1] Ilya Afanasyev and Dmitry Lichmanov. 2021. Evaluating the performance of Kunpeng 920 processors on modern HPC applications. In *International Conference on Parallel Computing Technologies*. Springer, 301–321.

[2] Krste Asanović and David A Patterson. 2014. Instruction sets should be free: The case for risc-v. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146* (2014).

[3] PLCT Lab ASE Lab and SOPHGO. 2023. RISC-V Prosperity 2036. https://rv2036.org. Accessed: Feb 1, 2025.

[4] Karthikeyan Kalyanasundaram Balasubramanian, Mirco Di Salvo, Walter Rocchia, Sergio Decherchi, and Marco Crepaldi. 2024. Designing RISC-V Instruction Set Extensions for Artificial Neural Networks: An LLVM Compiler-Driven Perspective. *IEEE Access* 12 (2024), 55925–55944. doi:10.1109/ACCESS.2024.3389673

[5] Nick Brown and Maurice Jamieson. 2024. Performance characterisation of the 64-core SG2042 RISC-V CPU for HPC. *arXiv preprint arXiv:2406.12394* (2024).

[6] Miguel Antonio Caraveo-Cacep, Rubén Vázquez-Medina, and Antonio Hernández Zavala. 2024. A review on security implementations in soft-processors for IoT applications. *Computers & Security* 139 (2024), 103677.

[7] Enfang Cui, Tianzheng Li, and Qian Wei. 2023. RISC-V Instruction Set Architecture Extensions: A Survey. *IEEE Access* 11 (2023), 24696–24711. doi:10.1109/ACCESS.2023.3246491

[8] Daniel Cukier and Fabio Kon. 2018. A maturity model for software startup ecosystems. *Journal of Innovation and Entrepreneurship* 7 (2018). https://api.semanticscholar.org/CorpusID:256235205

[9] Vedran Dakić, Leo Mršić, Zdravko Kunić, and Goran Đambić. 2024. Evaluating arm and risc-v architectures for high-performance computing with docker and kubernetes. *Electronics* 13, 17 (2024), 3494.

[10] Joao Mario Domingos, Pedro Tomas, and Leonel Sousa. 2021. Supporting RISC-V performance counters through performance analysis tools for Linux (perf). *arXiv preprint arXiv:2112.11767* (2021).

[11] Alexander Dörflinger, Mark Albers, Benedikt Kleinbeck, Yejun Guan, Harald Michalik, Raphael Klink, Christopher Blochwitz, Anouar Nechi, and Mladen Berekovic. 2021. A comparative survey of open-source application-class RISC-V processor implementations. In *Proceedings of the 18th ACM International Conference on Computing Frontiers*. doi:10.1145/3457388.3458657

[12] Islam Elsadek and Eslam Yahya Tawfik. 2021. RISC-V Resource-Constrained Cores: A Survey and Energy Comparison. In *2021 19th IEEE International New Circuits and Systems Conference (NEWCAS)*. 1–5. doi:10.1109/newcas50681.2021.9462781

[13] Christian Fibich, Stefan Tauner, Peter Rössler, and Martin Horauer. 2020. Evaluation of Open-Source Linear Algebra Libraries targeting ARM and RISC-V Architectures. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 663–672.

[14] FPRox. 2023. Sneak peek at RISC-V RVA23(U64) profile. https://fprox.substack.com/p/risc-v-profile-rva23u64

[15] Lukas Gerlach, Daniel Weber, Ruiyi Zhang, and Michael Schwarz. 2023. A security RISC: microarchitectural attacks on hardware RISC-V CPUs. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2321–2338.

[16] Youssef Ait Houaich and Mustapha Belaissaoui. 2015. Measuring the maturity of open source software. In *2015 6th International Conference on Information Systems and Economic Intelligence (SIIE)*. 133–140. doi:10.1109/ISEI.2015.7358735

[17] Slinger Jansen. 2020. A focus area maturity model for software ecosystem governance. *Information and Software Technology* 118 (2020), 106219. doi:10.1016/j.infsof.2019.106219

[18] Stavros Kalapothas, Manolis Galetakis, Georgios Flamis, Fotis Plessas, and Paris Kitsos. 2023. A survey on risc-v-based machine learning ecosystem. *Information* 14, 2 (2023), 64.

[19] Hangi Kim, Yongjin Jeon, Giyoon Kim, Jongsung Kim, Bo-Yeon Sim, Dong-Guk Han, Hwajeong Seo, Seonggyeom Kim, Seokhie Hong, Jaechul Sung, et al. 2021. PIPO: A lightweight block cipher with efficient higher-order masking software implementations. In *Information Security and Cryptology–ICISC 2020: 23rd International Conference, Seoul, South Korea, December 2–4, 2020, Proceedings 23*. Springer, 99–122.

[20] Youngbeom Kim and Seog Chung Seo. 2022. Optimized implementation of PIPO block cipher on 32-Bit ARM and RISC-V processors. *IEEE Access* 10 (2022), 97298–97309.

[21] Ming Ling, Xin Xu, Yushen Gu, and Zhihua Pan. 2019. Does the isa really matter? a simulation based investigation. In *2019 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*. IEEE, 1–6.

[22] Yu Liu, Kejiang Ye, and Cheng-Zhong Xu. 2021. Performance evaluation of various risc processor systems: A case study on arm, mips and risc-v. In *International Conference on Cloud Computing*. Springer, 61–74.

[23] Tao Lu. 2021. A survey on risc-v security: Hardware and architecture. *arXiv preprint arXiv:2107.04175* (2021).

[24] Ben Marshall, G. Richard Newell, Dan Page, Markku-Juhani O. Saarinen, and Claire Wolf. 2020. The design of scalar AES Instruction Set Extensions for RISC-V. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (Dec 2020), 109–136. doi:10.46586/tches.v2021.i1.109-136

[25] Metabase. [n. d.]. RISC-V Software Ecosystem Dashboard. https://tech.riscv.org/ecosystem/public/dashboard/9f6bf4ae-99bd-4717-b351-03465e8cf5f6

[26] Benjamin W. Mezger, Douglas A. Santos, Luigi Dilillo, Cesar A. Zeferino, and Douglas R. Melo. 2022. A Survey of the RISC-V Architecture Software Support.

IEEE Access 10 (2022), 51394–51411. doi:10.1109/ACCESS.2022.3174125
[27] openEuler. 2024. openEuler RISCV 24.03. https://www.openeuler.org/zh/blog/20240628-2403/20240628-2403.html.
[28] M Perotti, PS Davide, G Tagliavini, D Rossi, T Kurd, M Hill, L Yingying, and L Benini. [n. d.]. HW/SW approaches for RISC-V code size reduction (2020).
[29] Etiel Petrinja, Ranga Nambakam, and Alberto Sillitti. 2009. Introducing the OpenSource Maturity Model. In *2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*. 37–41. doi:10.1109/FLOSS.2009.5071358
[30] RISE Project. 2023. RISE: RISC-V Software Ecosystem. https://riseproject.dev/. Accessed: Feb 1, 2025.
[31] Majid Sabbagh and Yunsi Fei. 2021. Secure speculative execution via RISC-V open hardware design. In *Fifth Workshop on Computer Architecture Research with RISC-V (CARRV 2021)*.
[32] Jasmina Saidova. 2024. RISC-V ARCHITECTURE AND ITS ROLE IN THE NEAR FUTURE. *Journal of Advanced Scientific Research (ISSN: 0976-9595)* 5, 9 (2024).
[33] NicholasGeraldine Shirley, Yutian Gui, and Fareena Saqib. 2020. A Survey and Analysis on SoC Platform Security in ARM, Intel and RISC-V Architecture. (Jan 2020).
[34] Natalie Simson, Ares Tahigara, and Wolfgang Ecker. 2024. A Comparative Analysis of ARM and RISC-V ISAs for Deeply Embedded Systems. In *MBMV 2024; 27. Workshop*. VDE, 110–119.
[35] So Young Sohn and Min Seok Mok. 2008. A strategic analysis for successful open source software utilization based on a structural equation model. *J. Syst. Softw.* 81 (2008), 1014–1024. https://api.semanticscholar.org/CorpusID:23263414
[36] Kleber Stangherlin and Manoj Sachdev. 2022. Design and implementation of a secure RISC-V microprocessor. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 30, 11 (2022), 1705–1715.
[37] Daniel Suárez, Francisco Almeida, and Vicente Blanco. 2024. Comprehensive analysis of energy efficiency and performance of ARM and RISC-V SoCs. *The Journal of Supercomputing* 80, 9 (2024), 12771–12789.
[38] Daniel Weaver and Simon McIntosh-Smith. 2023. An empirical comparison of the RISC-V and AArch64 instruction sets. In *Proceedings of the SC'23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*. 1557–1565.
[39] XuanTie. 2023. C920: Specifications. https://xuantie.t-head.cn/product/xuantie/4082464366237126656.
[40] XuanTie. 2023. Open xuantie C906. https://xrvm.com/cpu-details?id=4056751997003636736.

A Artifact Availability

A fully functional replication package is publicly available at [https://github.com/everjcc/sapling-riscv-ecosystem].

A.1 Contents of the Repository:

- Test scripts: `run.sh`, `landscape-score.py`.
- Collected metadata: `./metadata/`.
- Evaluation results: `output.xlsx`.
- Benchmark documentation: `./benchmark-doc/`.
- Instructions and usage guide: `README.md`.

A.2 Experimental Procedure (executed via `./run.sh`):

1. Reads metadata from the `./metadata/` directory.
2. Iterates through each file with the suffix `-Landscape.xlsx`.
3. Extracts the software name (e.g., `CNTK` from `CNTK-Landscape.xlsx`).
4. Invokes `landscape-score.py` with the argument `-name ./metadata/<software_name>` to perform analysis.
5. Aggregates results into `output.xlsx`.