Streamlining Fedora Linux Distributions for RISC-V: A Scalable and Automated Approach

Surendra Billa, Arif Badar, Rushikesh Jadhav,
Yogeshwar Sonawane, Sanjay Wandhekar
C-DAC, Pune, India

CONNECTING THE DOTS
ISC High Performance

International Workshop on RISC-V for HPC at ISC

June 13, 2025

Outline

- Introduction
- Motivation
- Technical Background
- Proposed Architecture
- Development Approach
- Results & Performance Analysis
- Conclusion
- Future Work

Introduction

- RISC-V ISA: A Game Changer
  - **Open-source** and **modular** by design, RISC-V enables customizable, **cost-effective** processor development.
  - Suited for a wide range of systems: from **embedded** devices to **HPC servers**.

- The Challenge: Software Ecosystem Gaps
  - Major Linux distributions (e.g., Debian, Fedora, Ubuntu) target **general-purpose** RV64GC systems.
  - **Custom hardware configurations**, especially those using minimal RISC-V extensions like RV64G, lack **upstream** support.

Motivation

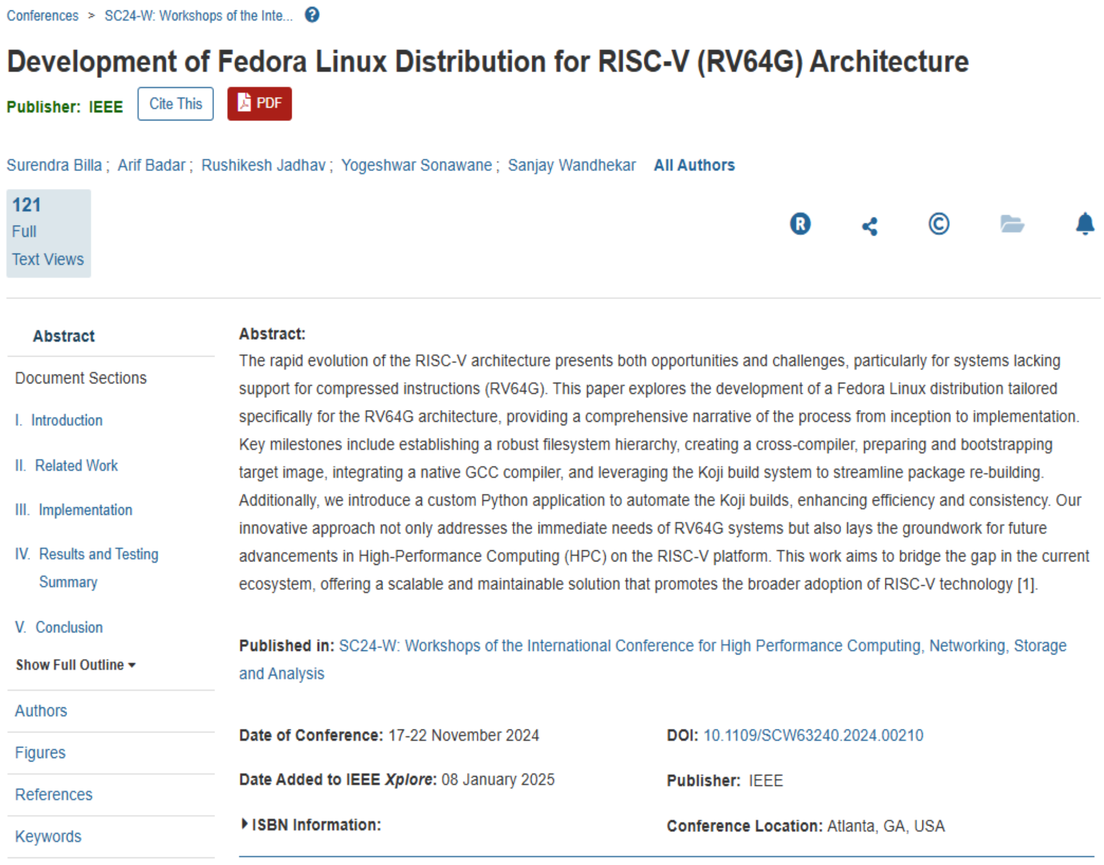
- Enable Broader RISC-V Adoption
  - Lightweight RISC-V designs help meet **cost** and **complexity** constraints ideal for academia, startups, and research.
  - However, absence of tailored Linux distributions limits **usability** and **experimentation**.

- Limitations of Existing Build Systems
  - **Buildroot and Yocto** are for custom embedded systems: They create minimal, hardware-specific Linux images. Buildroot produces **static images** and Yocto has limited support for **package managers**.
  - Standard distributions like Debian and Fedora : They offer full-featured package management(like dnf or apt) with access to large official repositories.

Technical Background

Our prior work on RV64G Fedora Port:

- Presented and published our work titled:
  "Development of Fedora Linux Distribution for RISC-V (RV64G) Architecture“



Conferences > SC24-W: Workshops of the Inte...

Development of Fedora Linux Distribution for RISC-V (RV64G) Architecture

Surendra Billa ; Arif Badar ; Rushikesh Jadhav ; Yogeshwar Sonawane ; Sanjay Wandhekar All Authors

121
Full
Text Views

Abstract:
The rapid evolution of the RISC-V architecture presents both opportunities and challenges, particularly for systems lacking support for compressed instructions (RV64G). This paper explores the development of a Fedora Linux distribution tailored specifically for the RV64G architecture, providing a comprehensive narrative of the process from inception to implementation. Key milestones include establishing a robust filesystem hierarchy, creating a cross-compiler, preparing and bootstrapping target image, integrating a native GCC compiler, and leveraging the Koji build system to streamline package re-building. Additionally, we introduce a custom Python application to automate the Koji builds, enhancing efficiency and consistency. Our innovative approach not only addresses the immediate needs of RV64G systems but also lays the groundwork for future advancements in High-Performance Computing (HPC) on the RISC-V platform. This work aims to bridge the gap in the current ecosystem, offering a scalable and maintainable solution that promotes the broader adoption of RISC-V technology [1].

Published in: SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis

Date of Conference: 17-22 November 2024

DOI: 10.1109/SCW63240.2024.00210

Date Added to IEEE *Xplore*: 08 January 2025

Publisher: IEEE

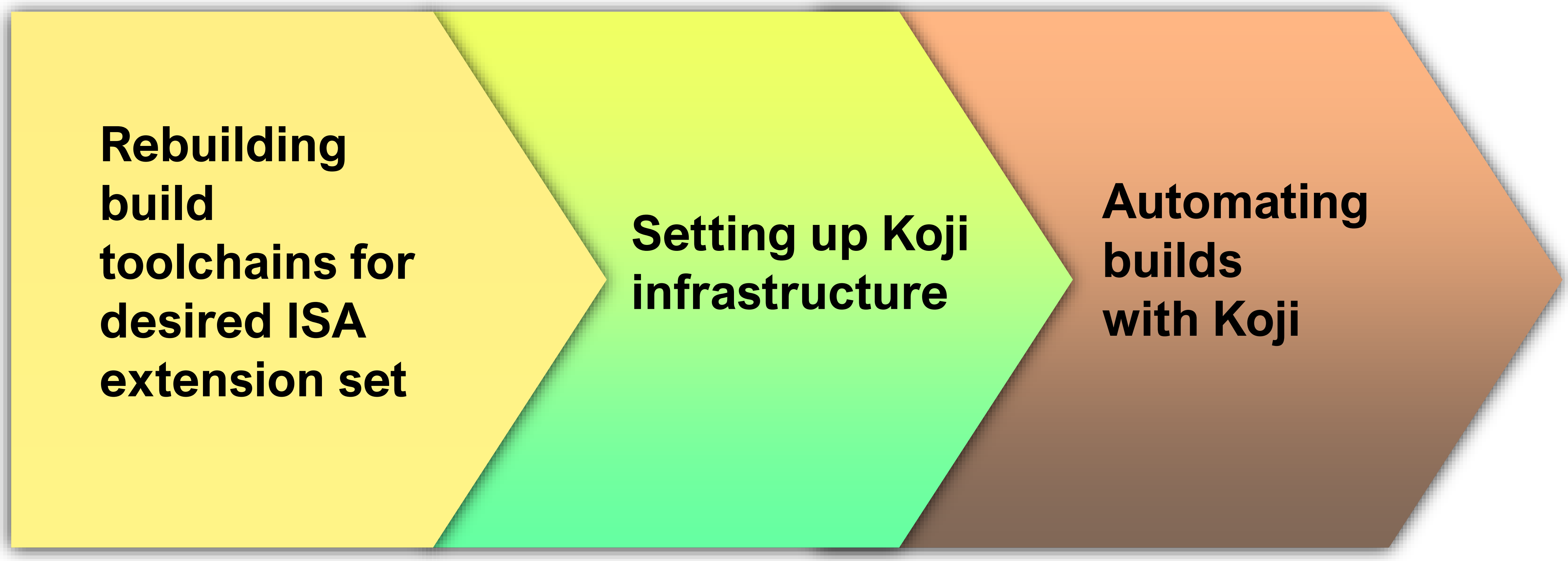
ISBN Information:

Conference Location: Atlanta, GA, USA

Technical Background

- Prior Approach (Based on Linux From Scratch - LFS)
  - Built a **minimal** Fedora Linux system for **RV64G** from source.
  - Manually compiled essential packages and dependencies.
- Challenges Faced
  - Involving considerable **human effort** and **time-consuming**.
  - Required setup of **Mock** and **Koji** to enable package builds.
  - Even with Koji, sequential package builds **limited scalability**.

Process Overview



Rebuilding build toolchains for desired ISA extension set

Setting up Koji infrastructure

Automating builds with Koji

Development Approach

Patching gcc and binutils rpm package for rv64g extension set

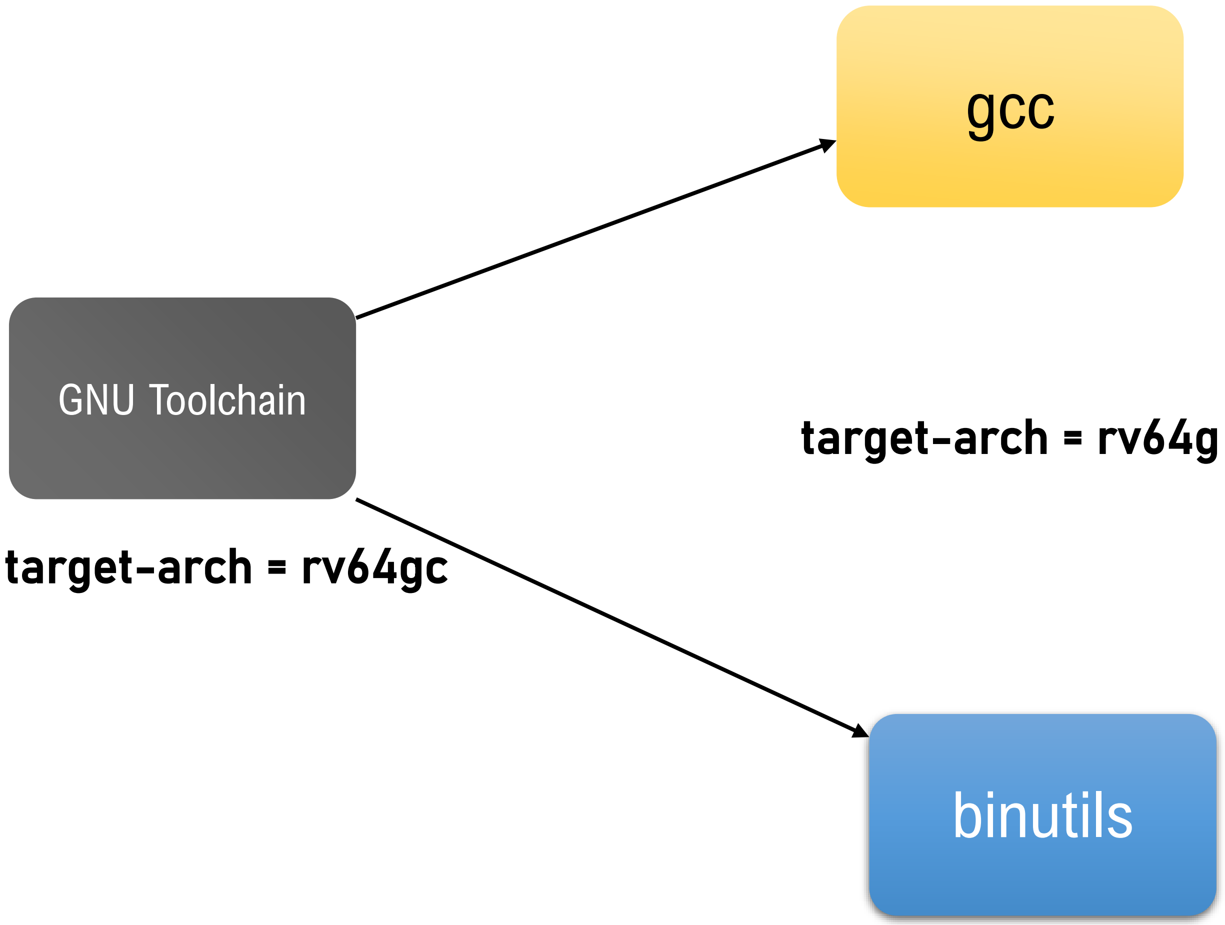
- Updated **GCC** specification file

```
@ -1163,7 +1163,7 @@ CONFIGURE_OPTS="\

%ifarch riscv64
    - —with-arch=rv64gc --with-abi=lp64d --with-multilib-list=lp64d \
    + —with-arch=rv64g --with-abi=lp64d —-with-multilib-list=lp64d \
%endif
```

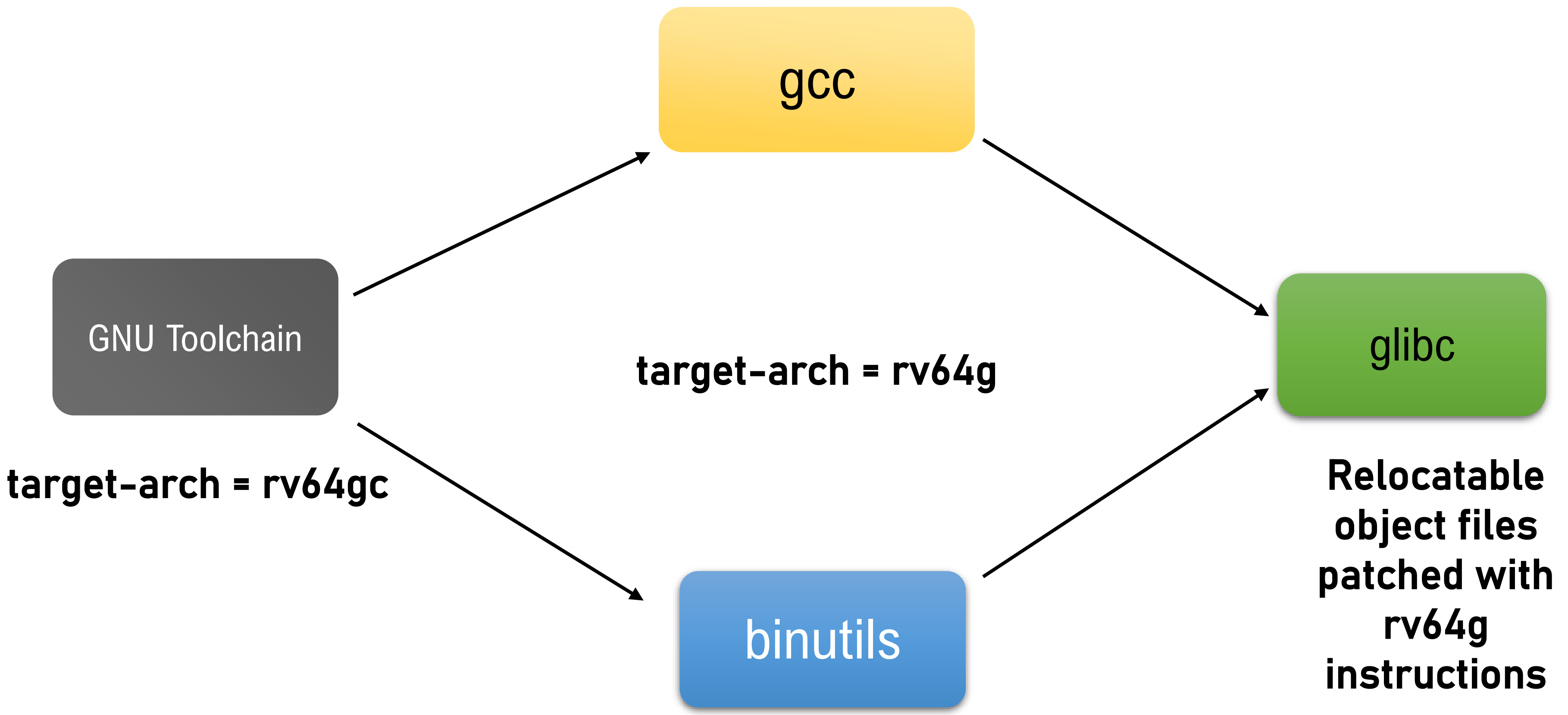
Development Approach

Rebuilding gcc and binutils with existing toolchain



Development Approach

Rebuilding glibc with patched compiler and assembler



GNU Toolchain

target-arch = rv64gc

gcc

binutils

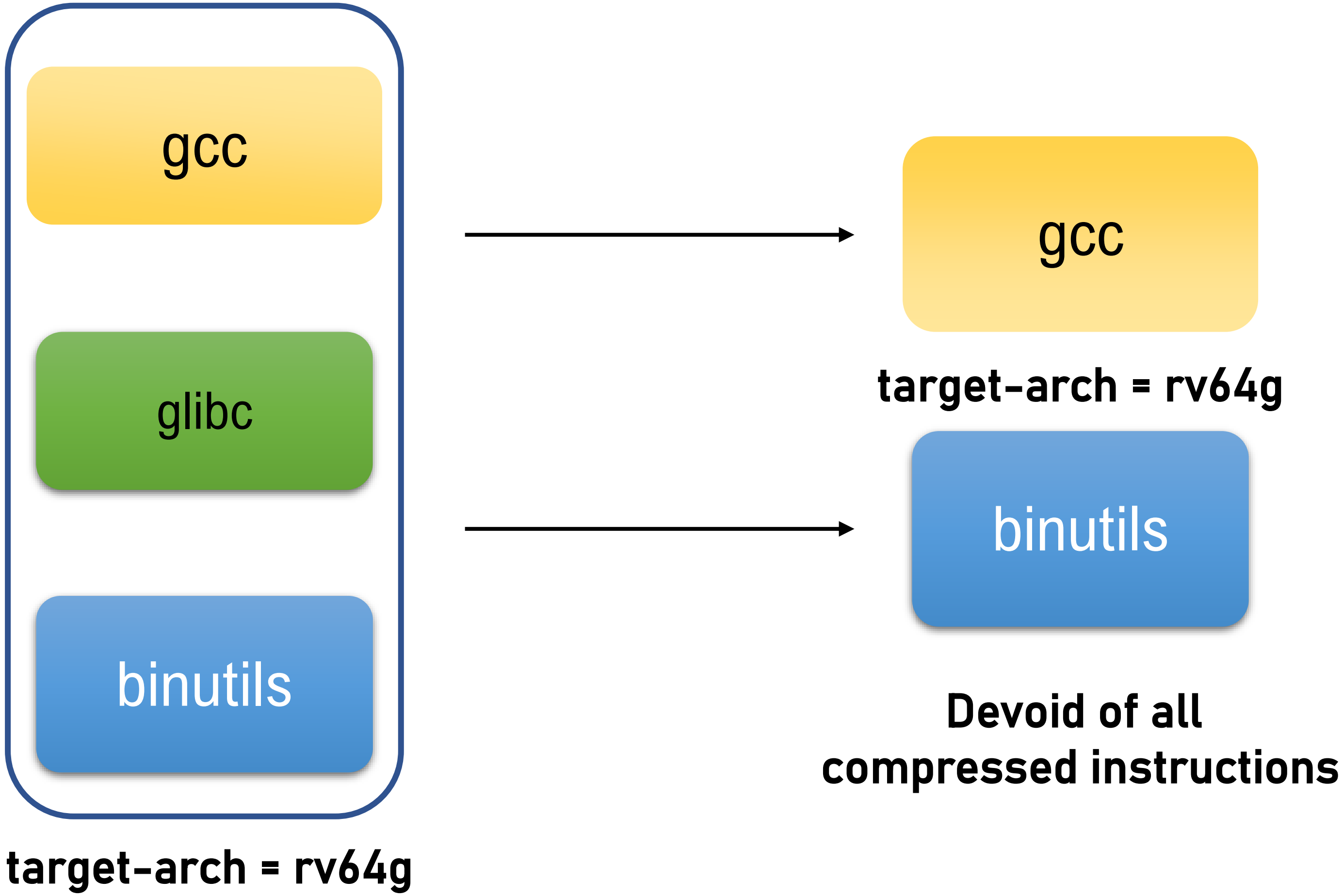
target-arch = rv64g

glibc

Relocatable
object files
patched with
rv64g
instructions

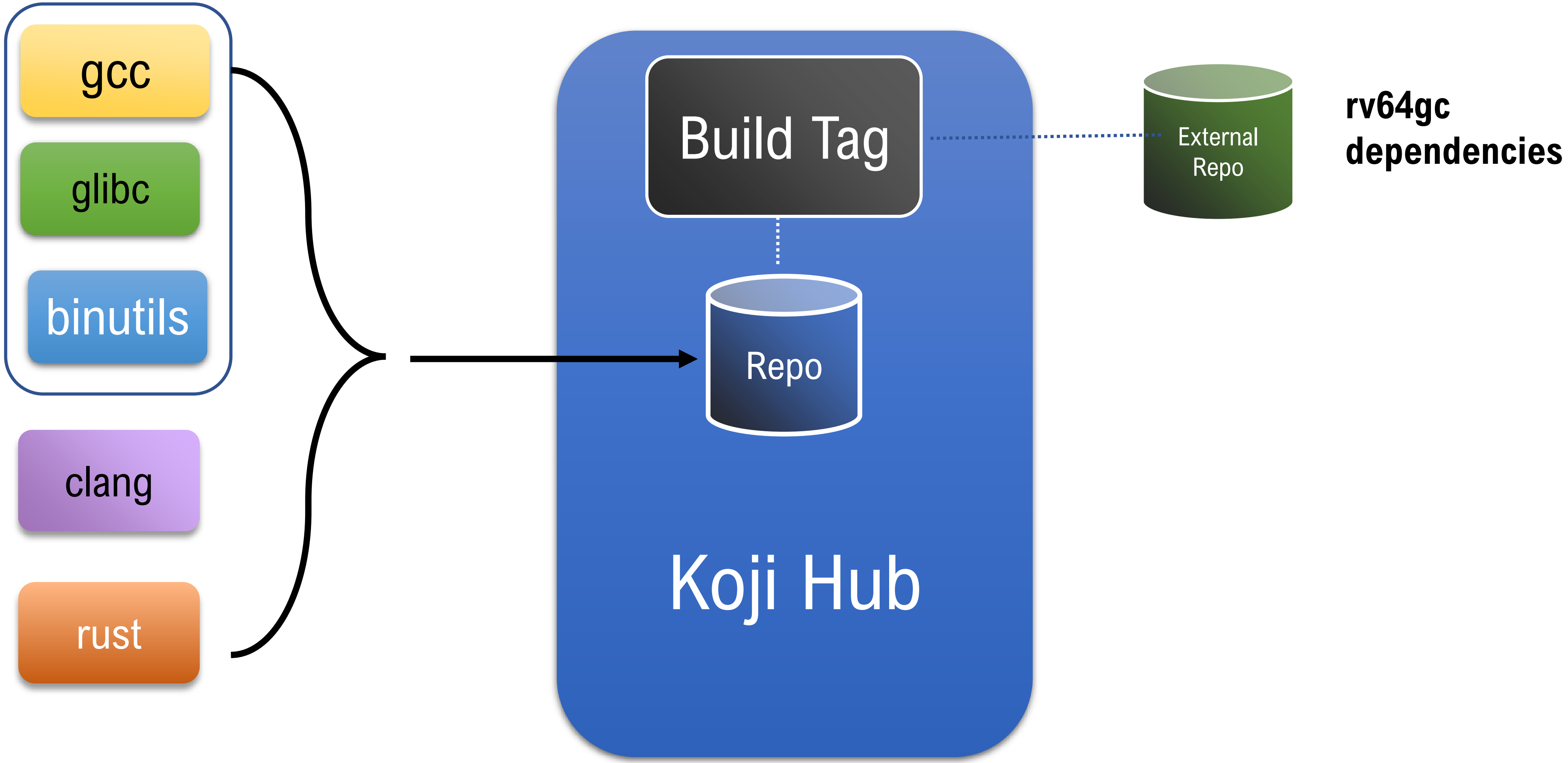
Development Approach

Rebuilding gcc and binutils again with the new toolchain



Development Approach

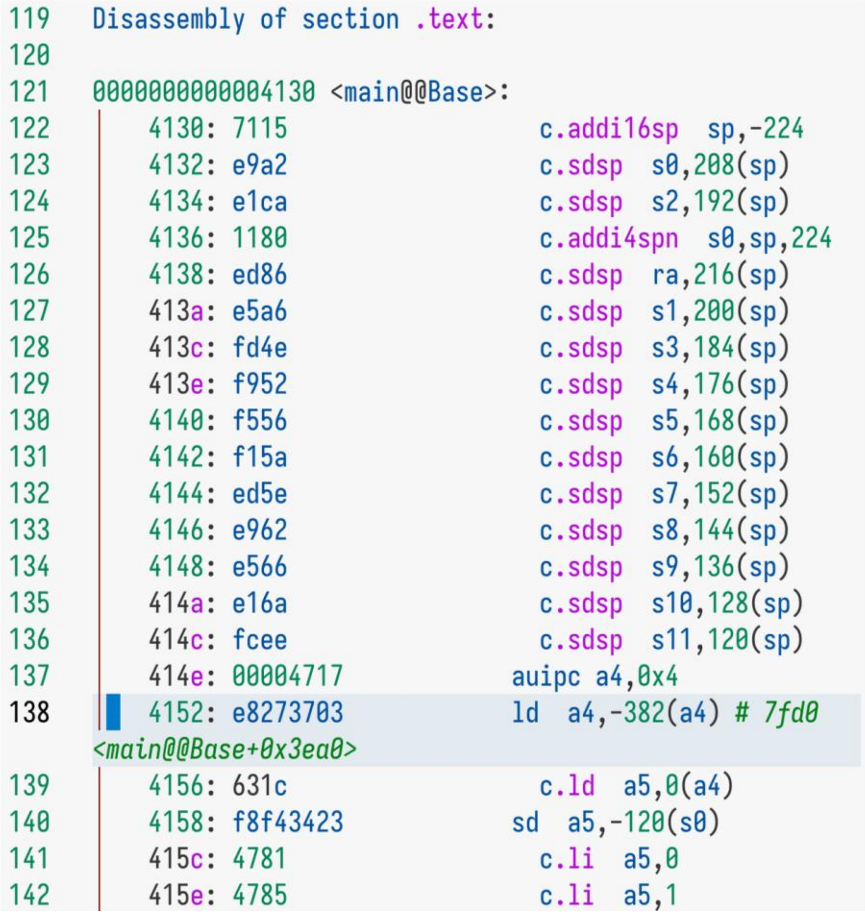
Integrating rv64g toolchain in Koji repository



Development Approach

Comparing compiled binaries - banner

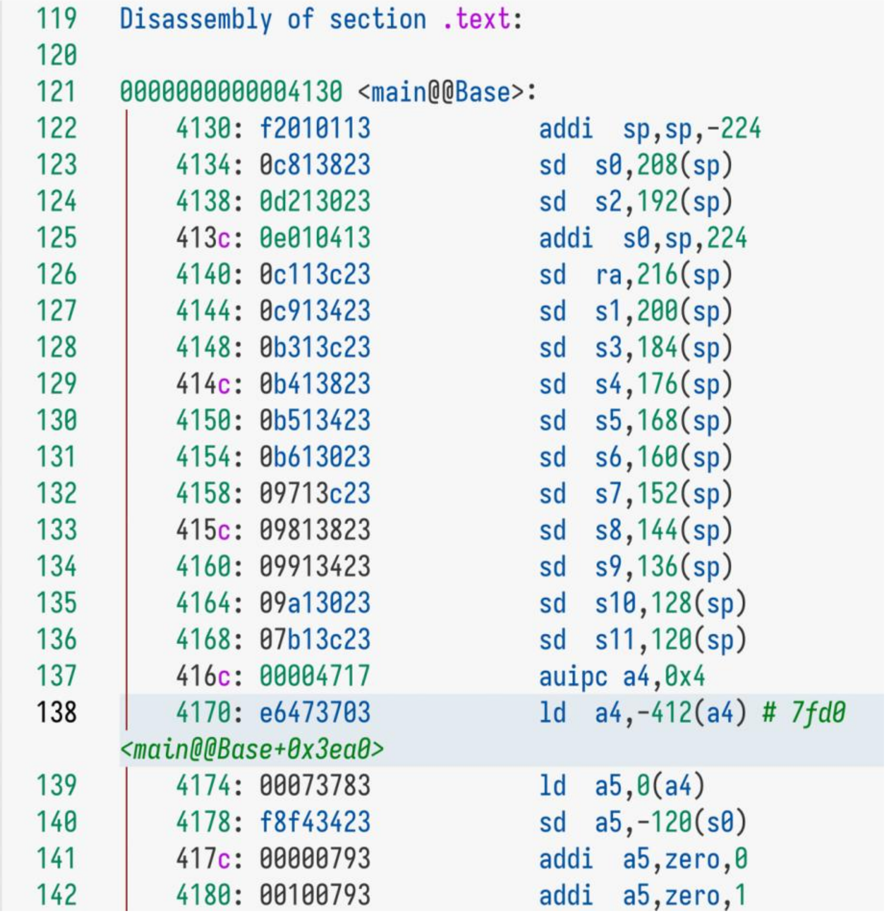
Default GNU toolchain(rv64gc)



```
Disassembly of section .text:

0000000000004130 <main@@Base>:
    4130: 7115                  c.addi16sp  sp,-224
    4132: e9a2                  c.sdsp  s0,208(sp)
    4134: e1ca                  c.sdsp  s2,192(sp)
    4136: 1180                  c.addi4spn  s0,sp,224
    4138: ed86                  c.sdsp  ra,216(sp)
    413a: e5a6                  c.sdsp  s1,200(sp)
    413c: fd4e                  c.sdsp  s3,184(sp)
    413e: f952                  c.sdsp  s4,176(sp)
    4140: f556                  c.sdsp  s5,168(sp)
    4142: f15a                  c.sdsp  s6,160(sp)
    4144: ed5e                  c.sdsp  s7,152(sp)
    4146: e962                  c.sdsp  s8,144(sp)
    4148: e566                  c.sdsp  s9,136(sp)
    414a: e16a                  c.sdsp  s10,128(sp)
    414c: fcee                  c.sdsp  s11,120(sp)
    414e: 00004717             auipc a4,0x4
    4152: e8273703             ld  a4,-382(a4) # 7fd0
<main@@Base+0x3ea0>
    4156: 631c                  c.ld  a5,0(a4)
    4158: f8f43423             sd  a5,-120(s0)
    415c: 4781                  c.li  a5,0
    415e: 4785                  c.li  a5,1
```

Patched GNU toolchain(rv64g)



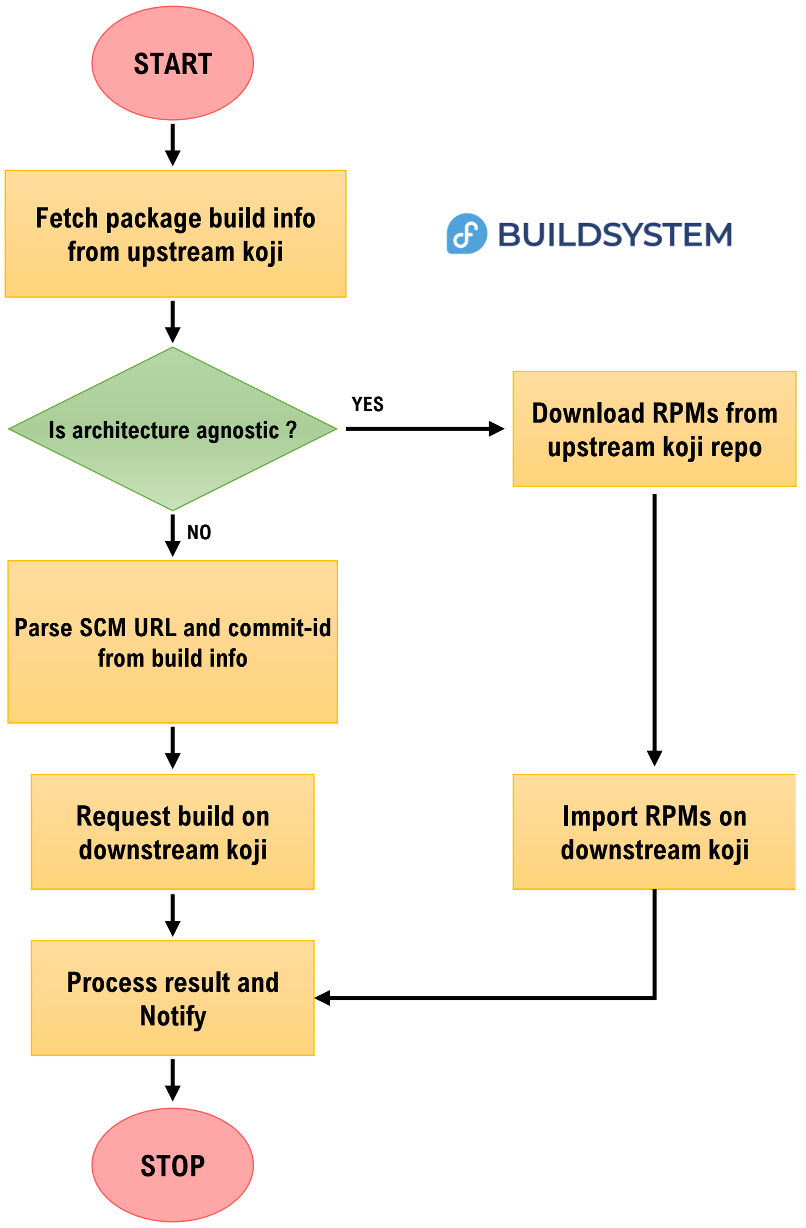
```
Disassembly of section .text:

0000000000004130 <main@@Base>:
    4130: f2010113              addi  sp,sp,-224
    4134: 0c813823              sd  s0,208(sp)
    4138: 0d213023              sd  s2,192(sp)
    413c: 0e010413              addi  s0,sp,224
    4140: 0c113c23              sd  ra,216(sp)
    4144: 0c913423              sd  s1,200(sp)
    4148: 0b313c23              sd  s3,184(sp)
    414c: 0b413823              sd  s4,176(sp)
    4150: 0b513423              sd  s5,168(sp)
    4154: 0b613023              sd  s6,160(sp)
    4158: 09713c23              sd  s7,152(sp)
    415c: 09813823              sd  s8,144(sp)
    4160: 09913423              sd  s9,136(sp)
    4164: 09a13023              sd  s10,128(sp)
    4168: 07b13c23              sd  s11,120(sp)
    416c: 00004717              auipc a4,0x4
    4170: e6473703              ld  a4,-412(a4) # 7fd0
<main@@Base+0x3ea0>
    4174: 00073783              ld  a5,0(a4)
    4178: f8f43423              sd  a5,-120(s0)
    417c: 00000793              addi  a5,zero,0
    4180: 00100793              addi  a5,zero,1
```

Development Approach

Automating package build with Koji

- Fetch package build info from **upstream koji**
- Import packages that are machine independent (fonts, configs, keys etc)
- Get **SCM URL** and **commit ID** for regular packages
- Call build on **downstream koji** with the acquired URL and commit id
- **Monitor build**, process **result** and **notify**

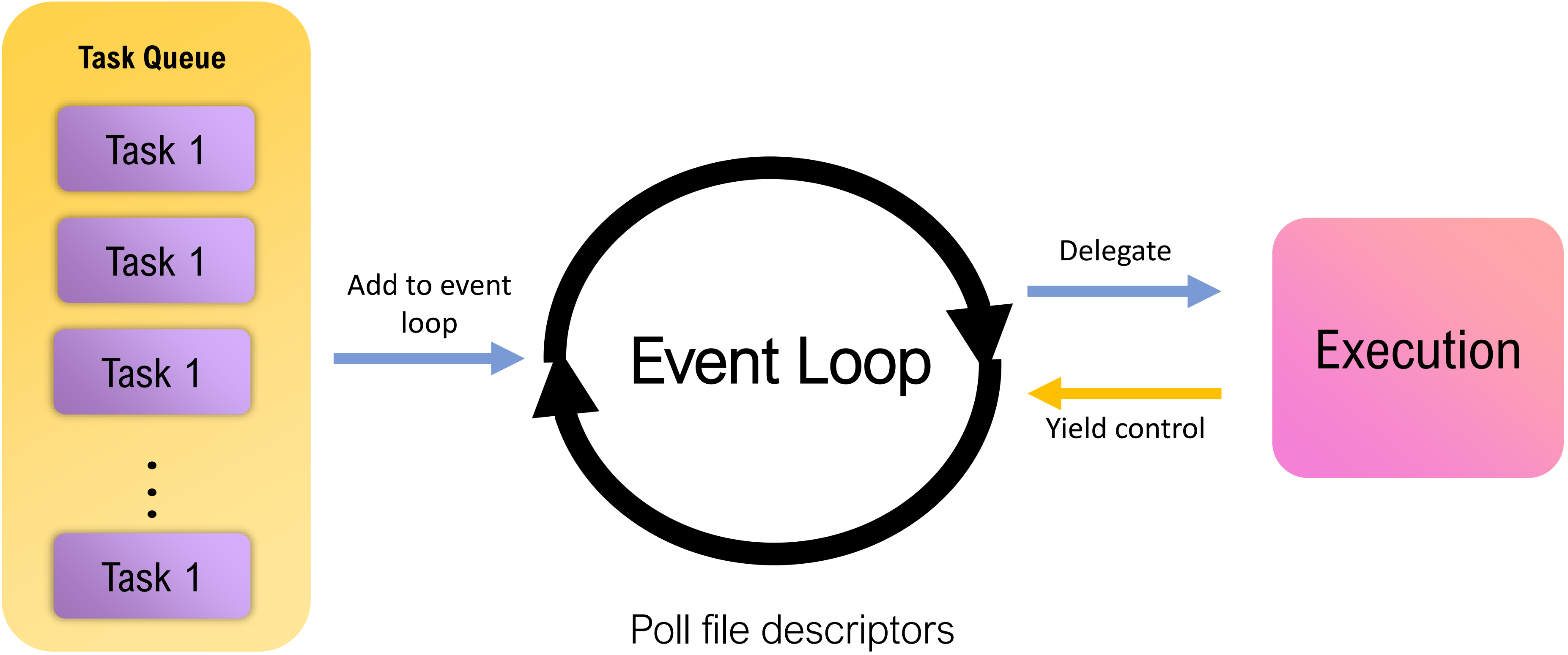

START
Fetch package build info from upstream koji
Is architecture agnostic ?
YES
Download RPMs from upstream koji repo
NO
Parse SCM URL and commit-id from build info
Request build on downstream koji
Import RPMs on downstream koji
Process result and Notify
STOP

BUILDSYSTEM

सी-डैक CDAC

Development Approach

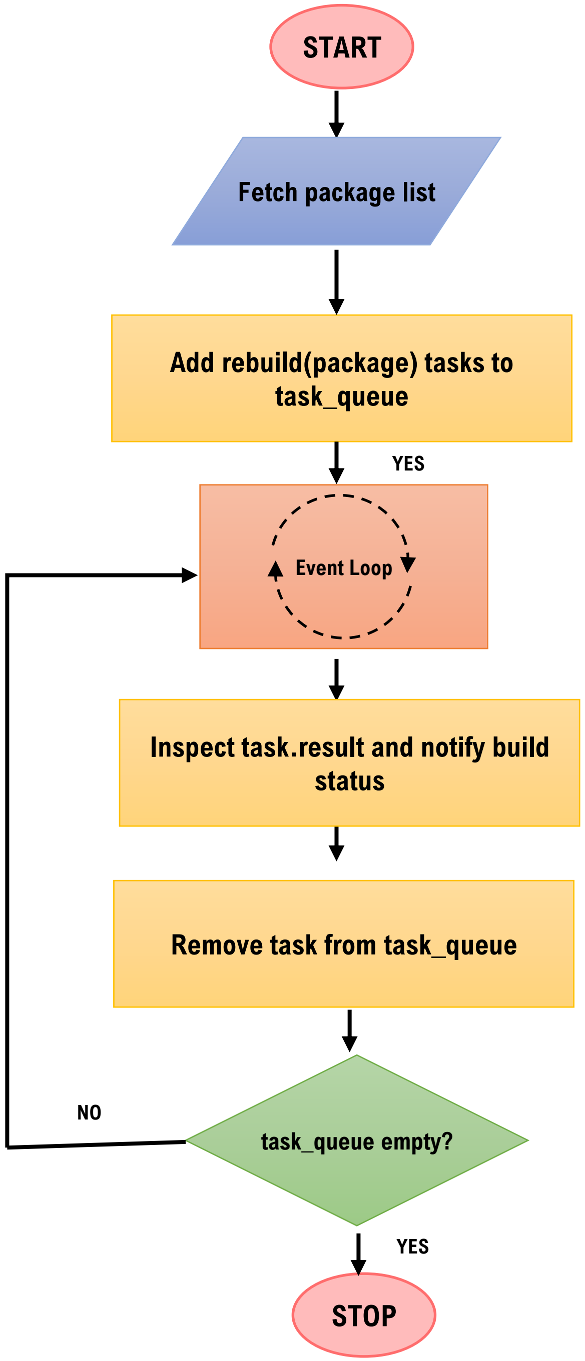
Parallelizing package builds using Asyncio event loop



Development Approach

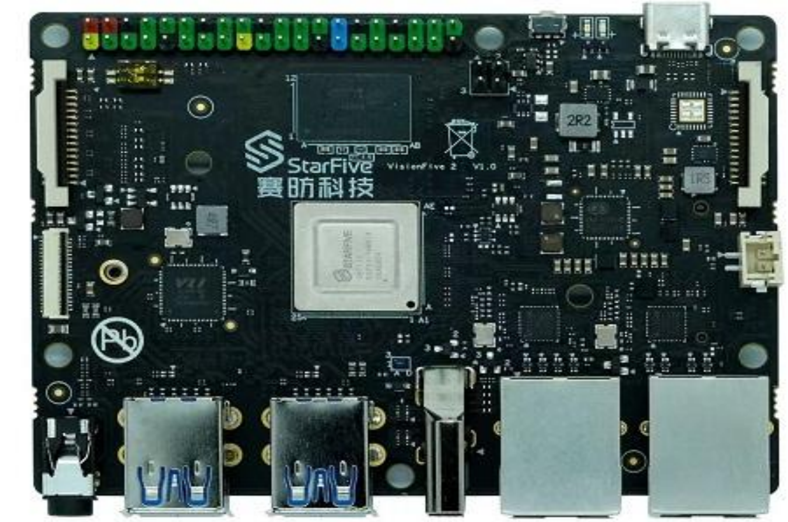
Running concurrent tasks

- Retrieve Package List
- Queue Rebuild Tasks
- Execute in Parallel
- Monitor and Notify
- Clean Up Completed Tasks
- Repeat Until Done


START
Fetch package list
Add rebuild(package) tasks to task_queue
YES
Event Loop
Inspect task.result and notify build status
Remove task from task_queue
NO
task_queue empty?
YES
STOP

Experimental Setup

- 4 Star Five Boards
- 8 GB Ram
- 4 Cores x 1.5 GHz



- 2 QEMU Instances
- 32 GB Ram
- 24 core x 2.10 GHz

Results & Performance Analysis

- Toolchain Robustness

  Packages successfully compiled with GCC (**RV64G**) are compatible with **RV64GC** binaries via LP64D ABI.

- Package Rebuild Success

  Achieved high rebuild success rates for **core components** and **libraries** using a personal Koji build system.

- Bootability on RV64G Platforms

  Verified successful boot on **StarFive, Vega**, and **QEMU** using U-Boot and GRUB across various storage.

- System Stability

  System passed **stress tests** and **HPC benchmarks** (e.g., HPL) with stable performance and no crashes.

Results & Performance Analysis

Evaluation Criteria	Manual approach	Automation Framework
Build Initiation	Requires **manual initiation** for each individual package.	**Automatically schedules** and **queues packages** for building.
Availability and Responsiveness	Build activity is **limited** on **weekends** and **holidays** due to lack of manual intervention.	**Operates continuously**, including weekends, thereby maximizing system uptime.
Efficiency in no-arch packages	Downloads are performed individually, resulting in significant **time consumption**.	Supports rapid, parallel downloads, completing the process within **seconds to a few minutes**.

Conclusion

- Modular RISC-V Support:

  Successfully developed a Fedora Linux **distribution tailored** for the **RV64G** subset, addressing key software support gaps.

- Automation & Efficiency

  Integrated the Koji build system and Python-based **automation** tools to reduce **manual effort** and enable continuous & **scalable builds**.

- Ecosystem Impact

  Presented a **reliable** and open system that helps **developers** and improves the RISC-V platform for many uses.

Future Work

- Extension-Based Customization

  Improve the system to support **RV64GC** with **additional** extensions, enabling custom Linux for various RISC-V application domains.

- Optimized Dependency Management

  Add a better system to manage **dependencies** so that rebuilding packages is faster and more reliable.

- HPC-Optimized RISC-V Server Distribution

  Develop a Fedora based RISC-V server OS optimized for **high-performance computing**.

Thank you!

yogeshwars@cdac.in

surendrab@cdac.in