Enhancing Debian Packages with ROCm GPU Acceleration

Mentor: Cordell Bloor

Personal Information

Name: Utkarsh Raj

Email ID: encode.sh@protonmail.com

Location: Noida 201301, India (Current)

Time Zone: IST (GMT+5:30)

Accounts

Salsa: https://salsa.debian.org/utk4r-sh

Github: https://github.com/utk4r-sh

Skills

Prior Experience:

- Long-time Debian user (2–3 years as a daily driver)
- Proficient in Bash Scripting for automation - created an Arch Linux automated installation script
- Familiar with CMake and Make
- Comfortable working with command-line interfaces (CLIs)
- Basic experience with Git and version control

Experience Gained Recently:

- Read and practiced the basic concepts from the Debian Maintainers' Guide and Developer's Reference
- Set up the build environment and created a functional Debian source skeleton of SpFFT package from its upstream tarball using debmake
- Became familiar with Debian-specific workflows, including d/control, d/rules, and d/patches modifications
- Gained hands-on experience with:
    - ➔ autopkgtest for automated regression and functionality tests
    - ➔ Creating and applying patches - quilt, dpkg-source
    - ➔ Static and dynamic linking concepts - including RPATH, LD_LIBRARY_PATH
    - ➔ GTest filters and some familiarity with mpirun

Open-Source Contributions

Enabled ROCm support for AMD GPUs in SpFFT **[March 2025]**

https://salsa.debian.org/science-team/spfft/-/merge_requests/1

Implemented autopkgtest in SpFFT **[March 2025]**

https://salsa.debian.org/science-team/spfft/-/merge_requests/2

Special thanks to Cordell Bloor and Andrius Merkys for their valuable feedback and multiple rounds of review. Their guidance significantly broadened my understanding of Debian's packaging structure and helped improve the quality of my code.

Synopsis

This project proposes a systematic effort to enhance Debian's support for AMD GPUs by leveraging the ROCm technology. While Debian already ships many foundational ROCm components, several scientific, artificial intelligence (AI) and high-performance computing (HPC) packages with upstream ROCm support lack AMD GPU acceleration in Debian.

This project aims to:

- Enable ROCm GPU acceleration in those existing packages.
- Package new ROCm-compatible tools and libraries.
- Integrate autopkgtests to ensure continuous validation on Debian's ROCm CI.

Benefits to the Community

Powerful Open Science Ecosystem

By making GPU-accelerated computation more accessible through open-source tools, this project strengthens the foundation for collaborative, transparent, and reproducible scientific research and AI development.

Reduced Dependence on Proprietary Solutions

Enabling ROCm helps diversify the HPC ecosystem by introducing a vendor-neutral and open-source alternative to CUDA.

Better Software Compatibility for AMD Users

New tools and enhancements improve performance, stability, and feature support on AMD hardware, ensuring a smoother and more complete user experience.

Improved Testing & Reliability

By expanding autopkgtest coverage in Debian CI, this project improves long-term package resilience, reduces regressions, and ensures the timely delivery of well-tested, production-grade software.

Technical Approach

This project will follow a structured plan.

Analysis:

To structure the work effectively, I will categorize the list of potential packages as follows:

- **Category A –** Packages that support ROCm upstream but do not yet have ROCm enabled in Debian.
- **Category B –** New tools and libraries not yet packaged for Debian that would be a good addition to the ROCm ecosystem.
- **Category C –** Existing Debian packages featuring ROCm that currently lack autopkgtest support.

This categorization will help ensure consistent progress across all areas.

Strategy:

To ensure steady progress and adaptability:

- I will discuss with my mentor and prioritize packages based on feasibility, expected impact, and alignment with project goals.
- I plan to work on multiple tasks in parallel to make up for the response time and review cycles.
- I'll maintain a consistent and efficient workflow while strictly following Debian's packaging standards and best practices.

Package Tracker	Status (Example)	Contribution(s)
SpFFT	Merged	rocm, autopkgtest
Kokkos	In progress	-
RPP	In progress	-

Throughout this project, I aim to steadily grow my skills while contributing meaningfully to the Debian project.

Milestones

This project will be completed in three milestones.

1. **Enabling ROCm support for Category A:**
    - ➔ Study the CMake Flags that enable ROCm support in that particular package.
    - ➔ Modify the debian/rules file of each package to enable ROCm features during build, ensuring support for all compatible GPU architectures.
    - ➔ Write specific patches for ROCm if necessary.
    - ➔ Add the required ROCm dependencies in debian/control that are part of Debian's official repositories.

If enabling ROCm breaks existing functionality, I will create a separate binary package for ROCm based on maintainers' preferences.

I plan to begin with Kokkos, a widely used performance portability model

https://github.com/kokkos/kokkos

I'll follow a similar approach I had for SpFFT.

https://salsa.debian.org/science-team/spfft/-/merge_requests/1/diffs#8756c63497c8dc39f7773438edf53b220c773f67

I believe the first step would be to follow through the documentation and identify the CMake flags and required dependencies that are relevant to ROCm.

For instance, to build for GFX940 architecture with HIP support, these CMake flags seem relevant.

```
-DKokkos_ENABLE_HIP=ON
-DKokkos_ARCH_AMD_GFX940=ON
```

I took reference from Kokkos documentation listing supported device backends and specific flags for HIP enablement.

https://kokkos.org/kokkos-core-wiki/get-started/configuration-guide.html

I will then begin enabling ROCm features across all compatible GPU architectures, while limiting them to supported CPU architectures as needed.

2. Packaging New Tools and Libraries in Category B (Experimental):

- ➔ Use standard tools like debmake, debuild/dpkg-buildpackage, sbuild and lintian to create basic Debian packages for new tools and libraries.
- ➔ Ensure it complies with Debian policy.

I plan to study the Debian Policy Manual thoroughly to ensure compliance with Debian's packaging standards.

https://www.debian.org/doc/debian-policy/

Once the base package is ready, I will enable ROCm features and add autopkgtests where applicable.

ROCm Performance Primitives (RPP) is a promising library for this experiment.

https://github.com/ROCm/rpp

It provides image processing primitives with HIP backend and could be a valuable addition to Debian's ROCm ecosystem.

source tree (sample):

```
rpp-<version>/
├── debian/
│   ├── control
│   ├── rules
│   └── ...
rpp_<version>.orig.tar.xz
rpp_<version>.dsc
rpp_<version>.debian.tar.xz
```

After setting up the initial structure, I will carefully configure the debian/ directory in accordance with Debian policy to ensure the package meets all required packaging standards.

3. Automated tests running on Debian CI for Category C:

- ➔ Implement autopkgtest to verify the overall functionality.
- ➔ Split the test suite into two, one that checks for GPU-accelerated functionality and the other one for normal CPU-level functionality.

I will make sure to skip GPU tests on the official Debian CI, preventing false regressions due to unavailability of an AMD GPU. These GPU tests will be routed to the ROCm CI, maintained by the AMD ROCm team, while the official Debian CI can validate mandatory CPU-level core functionality.

I will start with the packages I plan to enable ROCm support in, for instance, Kokkos, following a similar approach I had for SpFFT.

https://salsa.debian.org/science-team/spfft/-/merge_requests/2/diffs#83223a554d33ba7e1888274ff8ac43adb8147ba0

The first step will be to enable tests during build and add necessary dependencies.

```
-DKokkos_ENABLE_TESTS=ON
```

I'll create a separate binary for tests and define the appropriate location for the test executables in libkokkos-tests.install, which will depend on the overall structure and the type of tests. I'll then continue implementing autopkgtest for the same.

I did some research and noticed that Kokkos has ctests.

debian/tests/control (sample):

```
Test-Command: rocm-test-launcher debian/tests/run-testdir
Features: test-name=libkokkos-tests
Depends: pkg-rocm-tools, <ctest dependencies>
Restrictions: allow-stderr
```

I'll try to split it into GPU and CPU tests as required. Since Kokkos has ctests, I'll refer to its documentation for correct syntax of filters to skip all those lines that involve GPU-inclusive operations, and define it as a 1st line argument of libkokkos-cpu-tests.

I'll make similar modifications for libkokkos-gpu-tests as necessary, adding a skippable tag so it is gracefully skipped on the official Debian CI infrastructure.

As GPU functionality is hardware-dependent, I expect autopkgtest to run GPU tests exclusively on the ROCm CI infrastructure, where AMD GPUs are available.

Deliverables

Enhanced GPU Support in Existing Packages

Enable ROCm support in at least 6 existing packages that already have upstream ROCm support. These enhancements will be contributed through merge requests and integrated into the Debian archive.

New Tools and Libraries for the ROCm Ecosystem

Package at least 2 new tools or libraries that will be useful for AMD GPU users and are currently missing from Debian. These will be created using debmake and maintained according to Debian packaging standards.

Improved autopkgtest Coverage on Debian CI and ROCm CI

Implement autopkgtest for both enhanced and newly created packages to ensure stable functionality across all supported architectures. These tests will be integrated into the official Debian CI and AMD's ROCm CI infrastructures.

Hardware Limitations

Currently, I do not have local access to a ROCm-compatible GPU, which has limited my ability to run the GPU tests.

I have discussed this with the mentor, Cordell Bloor. I believe he can grant me remote access to a GPU server either at the University of Calgary or the University of Oregon.

I plan to connect via ssh, as I have some experience using it for remote access.

```
ssh -p <port_number> <username>@<domain>
```

Academics

University:	JSS Academy of Technical Education Noida
Degree:	Bachelor of Technology
Course:	Electronics and Communication Engineering
Semester:	4th (As of April 2025)

Background

Although I'm new to open-source development, I'm very excited to learn and contribute! My academic background in Electronics has fueled my interest in Computational Physics, Linux Systems, and High-Performance Computing (HPC), especially GPU acceleration - areas I intend to explore further from both software and hardware perspectives.

I've always been inspired by large collaborative projects like the Linux Kernel, APT, and Wayland - where small consistent contributions come together to create something powerful, stable, and widely impactful. Debian, which unifies these into a stable system has always been the backbone of my development environment. I personally wish to contribute to this effort and evolve as a Debian Developer through hands-on practice and impactful work.

During the last few weeks, I've contributed to the Debian ecosystem by submitting merge requests for the SpFFT package, under the mentorship of Cordell Bloor. My tasks were to enable ROCm in this package and implement autopkgtest, which gave me valuable insight into Debian packaging, testing workflows, and the structured process of maintaining high-quality code.

This project strongly aligns with my interests and long-term goals in open-source development and HPC. I see it as a great starting point to improve my technical skills, make meaningful contributions to open-source, and learn from experienced developers on real-world systems that matter.

It's the only GSoC project I'll be applying for this year.

Timeline

Phase	Milestones
Community Bonding Period (May 8 – June 2)	• Get to know the mentor and the Debian community, and establish a fast, effective communication channel with the mentor via Matrix, IRC, Google Meet or any other preferred platform. • Remotely access the available ROCm compatible hardware and set up a test environment. • Read the Debian's Policy Manual, relevant ROCm documentation and check out any past packaging efforts by other developers. • Gain more familiarity with autopkgtest and CI workflows.
Week I – VI (June 2 – July 14)	**First 3 Weeks:** • Enable ROCm in at least 3 upstream packages that have ROCm support, starting with Kokkos. • Start implementing autopkgtest for those packages. • Submit patches for review and address mentor feedback. **Next 3 Weeks:** • All pending issues must have been resolved. • Select 1 new useful tool/library, ROCm Performance Primitives (RPP) library. • Package it using Debian standard tools and set up the debian directory ensuring best practices. • Once basic packaging is done, enable ROCm and begin implementing tests, if applicable. **Milestone I and III (50% Complete):** At least 3 packages should have ROCm enabled with autopkgtest running and be in a mergeable state. **Milestone II (50% Complete):** At least 1 new addition to the Debian ROCm ecosystem.

Week VII – XII (July 14 – August 25)	All MRs from earlier should have been approved. ● Re-evaluate whether to target more packages or less based on past experience. ● Revise the timeline accordingly and reprioritize remaining tasks based on actual progress, complexity of packages, and mentor feedback. **Milestone I and III (100% Complete):** If there is no change in pace, at least 3 more packages should have ROCm enabled with autopkgtest running and be in a mergeable state. **Milestone II (100% Complete):** If there is no change in pace, at least 1 more new addition to the Debian ROCm ecosystem.
Final Week	During the final phase of this project: ● All pending issues must have been resolved. ● All MRs from earlier should have been approved. ● Final review. ● Write the final GSoC report. **Overall:** At least 6 packages should have ROCm enabled with autopkgtest running and be in a mergeable state. At least 2 new additions to the Debian ROCm ecosystem.

Stretch Goals

This project allows researchers and developers to leverage the full potential of their AMD GPUs and access optimized packages more easily, streamlining scientific, high-performance computing and AI workflows on Debian-based systems.

If time permits, I would personally like to write a technical article aimed at showcasing how ROCm is transforming the open-source HPC ecosystem - particularly in scientific simulations and AI workloads. This article will include benchmark results from packages enhanced during this GSoC project (e.g., SpFFT, Kokkos), comparing performance with and without ROCm GPU acceleration on supported hardware.

It will highlight key performance gains, the current state of ROCm in Debian, and its potential to advance reproducible and scalable scientific research - aiming to promote further adoption and contributions.