# Building AMD ROCm from Source on a Supercomputer

Cristian Di Pietrantonio
cdipietrantonio@pawsey.org.au

2023 Cray User Group

# Pawsey Supercomputing Research Centre

Headquarted in Perth, Western Australia, Pawsey has a 20-year long history. Offers critical support to radioastronomy research around the Square Kilometre Array (SKA). The centre underwent a 70m capital refresh financed by the Australian government. Currently employs 60+ staff.

# The Setonix supercomputer


- Australia's most powerful research supercomputer.
- HPE Cray EX system with 200'000 AMD Zen3 CPU cores and 750+ MI250X GPUs.
- 50 PFLOPS, 90% coming from AMD GPUs.
- 15PB `/scratch` storage.
- $15^{\text{th}}$ in TOP500, $4^{\text{th}}$ in Green500.
- Artwork by aboriginal artist Margaret Whitehurst.

# Outline

Pawsey presents ROCm-from-source, a source build system for ROCm.

- Background and motivation
- ROCm: an overview
- Related work
- Introduction to ROCm-from-source
- Deployment on Setonix (HPE Cray Ex system)
- Conclusion and future work

# Background and motivation

AMD GPUs are adopted by major supercomputers in the world. The corresponding software development platform and runtime is ROCm.

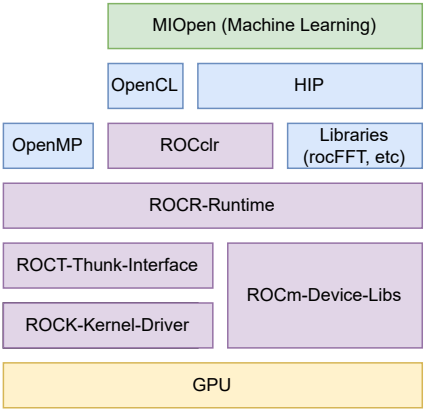
- Still in rapid development, with features added at each release.
- Critical to provide researchers with the latest version.
- Provided installation methods not suitable for fast deployment on supercomputers.

We want:

- a non-root installation method;
- custom installation path;
- explicit dependency versions; and,
- a method easy to execute and to update.

# ROCm: an overview

- Kernel driver & low-level API.
- HSA Runtime & device libraries.
- LLVM as the compiler infrastructure.
- Several programming models and libraries.
- Tens of projects overall.



# Comparison with Spack

## Related work

ROCm support in Spack has come a long way since this work started. Despite that, there are a couple of reasons to prefer ROCm-from-source to Spack now that ROCm is rapidly evolving.

- A Spack deployment might not be updated as often (hence outdated recipes).
- Updating ROCm-from-source it is easier than updating many recipes.
- ROCm-from-source installs all ROCm projects in one go.
- ROCm-from-source tries to retain the "official" installation tree structure (might be important for the Cray environment).
- Spack may be the way to go when ROCm is mature?

# Other scripted builds

## Related work

```
# Maintainer: Torsten Keßler <t dot kessler at posteo dot de>

pkgname=rocwmma
pkgver=5.4.1
pkgrel=1
pkgdesc='Library for accelerating mixed precision matrix multiplication'
arch=('x86_64')
url='https://rocwmma.readthedocs.io/en/latest/index.html'
license=('MIT')
depends=('hip' 'rocblas' 'openmp')
makedepends=('rocm-cmake' 'doxygen')
_git='https://github.com/ROCmSoftwarePlatform/rocWMMA'
source=("$pkgname-$pkgver.tar.gz::$_git/archive/rocm-$pkgver.tar.gz")
sha256sums=('641d2730db737edcade8da6b3f77ce85d4ad460e0902c2b688df2d51fb13f9f0')
_dirname="$(basename "$_git")-$(basename "${source[0]}" .tar.gz)"

build() {
    cmake \
      -Wno-dev \
      -B build \
      -S "$_dirname" \
      -DCMAKE_INSTALL_PREFIX=/opt/rocm \
      -DCMAKE_BUILD_TYPE=None \
      -DROCWMMA_BUILD_TESTS=OFF \
      -DROCWMMA_BUILD_SAMPLES=OFF
  cmake --build build
}

package() {
    DESTDIR="$pkgdir" cmake --install build

    install -Dm644 "$_dirname/LICENSE.md" "$pkgdir/usr/share/licenses/$pkgname/LICENSE"
}
```

- There exist other scripted builds, motivated by machine learning applications.
- They rely on packet managers for dependencies.
- Not ready to be executed as is (interpreted recipes, extensive modification required).
- Not comprehensive of all ROCm projects.
- They provided a good source of information about CMake options.
- AMD provides the AOMP project, a scripted build of AMD's fork of LLVM.

# ROCm-from-source

ROCm-from-source is a source build system for ROCm written entirely in Shell.

- Requires only a minimal set of external dependencies (AMD kernel drivers, common Linux commands).
- No root permissions required, dependencies also built from source.
- Convenient set of Shell functions make the build process easy to understand and work on.
- Can enable projects that are still experimental (e.g. rocWMMA).
- Can be used to build containers (e.g. only install necessary components).
- Available at `https://github.com/PawseySC/rocm-from-source`.

# ROCm-from-source

Installing ROCm from source is as easy as running the following commands:

```
git clone --branch rocm5.4.3rev0 \
    https://github.com/PawseySC/rocm-from-source.git
export ROOT_INSTALL_DIR=/software/setonix/2022.11/rocm
./rocm-from-source/install_rocm.sh
```

# The driver script

The `install_rocm.sh` script orchestrates the execution of various other helper scripts with the goal of installing ROCm.

- Sets sensible defaults for input variables (`GFX_ARCHS`).
- Sources all other Shell script files:
  1. `utils.sh`: custom Shell functions.
  2. `install_build_deps.sh`: retrieves build dependencies (for instance, CMake and `repo`).
  3. `install_rocm_deps.sh`: installs ROCm dependencies (`libX11`, `boost`, `libdrm`, ...).
  4. `install_rocm_projects.sh`: retrieves and installs ROCm projects.
- Avoids reinstalling ROCm dependencies if not necessary.
- Create ROCm module file.

# Shell functions

Repetitive sequences of Shell commands have been wrapped in convenient high-level Shell functions.

- `wget <url-to-tar>`, `tar xf <tar-file>`, and `cd <src>` becomes `wget_untar_cd <url>`
- In the same way we define `cmake_install` and `configure_install`;
- the described approach is necessary considering the large number of projects that must be built and installed.

# Shell functions

```
wget_untar_cd () {
  url=$1
  tarfile=${url##*/}
  folder=${tarfile%.tar.gz}
  cd ${BUILD_FOLDER}
  [ -e ${tarfile} ] || \
    run_command wget "${url}"
  [ -e ${folder} ] || \
    run_command tar xf "${tarfile}"
  run_command cd "$folder"
}
```

```
configure_build () {
  run_command cd ${BUILD_FOLDER}
  wget_untar_cd $1
  if [ -e rfs_installed ] && \
    [ ${SKIP_INSTALLED} -eq 1 ]; then
  echo "Package already installed. Skipping.."
  else
  if [ $CLEAN_BUILD -eq 1 ]; then
    echo "Cleaning build directory.."
    make clean
  fi
  run_command ./configure --prefix="${INSTALL_
  run_command make -j $NCORES install
  run_command touch rfs_installed
fi
```

# Build environment, compiler and linker options

The build environment must be under tight control for the installation process to be reproducible and reliable.

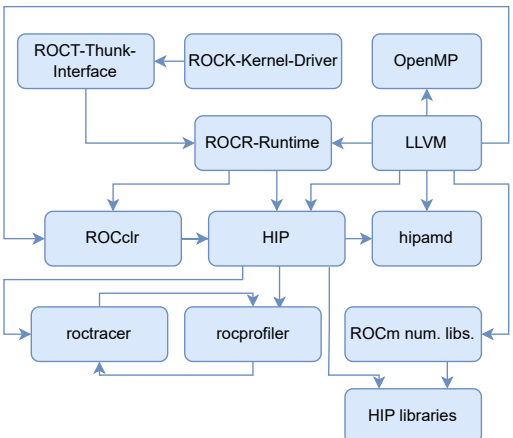
- Makes sure other installations of ROCm are not picked up.
  - Done through environment variables and patches to `CMakeLists.txt` files.
  - At runtime this is achieved using `RPATH`.
- Sets ROCm specific environment variables: `HIP_PATH`, `HSA_PATH`, `HIP_CLANG_PATH`, `ROCM_PATH`, and `HIP_RUNTIME`
- Using `gcc` to build ROCm deps and LLVM, then switch to `hipcc` (`clang`).
- Cray wrappers cause some issues at the link stage during the compilation of some packages.
- Be mindful of `libstdc++` / `libc++`

# ROCm dependencies

ROCm depends on several software libraries that may not be present on an HPE Cray system, or that are installed without all the required components.

- Examples are `libX11`, `libdrm`, `elfutils`, and `gettext`.
- Dependencies for each ROCm project were/are not documented well.
- Installed dependencies as they were discovered.
- Interestingly, the rocprofiler project requires the closed-source aqlprofiler library, developed by AMD.

# ROCm projects - dependency graph




ROCm is open source and thus all projects are available on GitHub.

- One particular repository, `https://github.com/RadeonOpenCompute/ROCm` acts as an index. The `repo` command must be used to download all the projects.
- Had to discover the dependency graph between projects, which was not well documented.
- LLVM is the most complex installation.
- LLVM Flang vs "Classic Flang".

# Patches and bug fixes

Minor bugs in build configurations and source code are routinely found in ROCm projects. Patches are generated and applied before compilation.

- Avoid the use of virtual environments.
- Wrong installation prefix for HIPIFY binaries and HIP CMake config files.
- Hardcoded `/opt/rocm` path within `CMakeLists.txt` files.
- Problematic flags in the compilation of LLVM OpenMP.
- Programming errors.

# Deployment on HPE Cray EX

Two aspects of deploying ROCm on a HPE Cray EX system.

- **Build and installation process**. Minimal external dependencies, so not harder than installing any other package from source.
- **Integration with other applications and tools**
  - Currently installed as part of the `PrgEnv-gnu` software stack, like all other software.
  - ROCm libraries are integrated into the Spack software manager as external packages.
  - More work needs to be done as our installation did not work with AMD tools such as Omniperf and Omnitrace.
- Currently, several projects are using HIP and HIP libraries from our custom installation.

# Deployment on HPE Cray EX - continued

Pawsey staff have started exploring the way Cray Programming Environments leverage ROCm to offload computations to AMD GPUs. We used a matrix multiplication example using OpenACC offloading. When using our ROCm build together with `crayftn`, the following error message suggests more work is to be done on our side.

```
Warning: Cannot find all neccessary
path for loaded rocm version!!!
lld: error: undefined symbol:
__ockl_get_num_groups
>>> referenced by [...] cce-openmp__llc.amdgpu
```

Seems like OpenACC directives are translated to OpenMP ones. Further investigations running the compiler in verbose mode confirms it.

# Conclusion and future work

This paper introduced a source build process for AMD ROCm that targets installations on a supercomputer.

- HIP and numerical libraries have been taken care of.
- More work needed for OpenMP offloading to work. Especially with the Cray compilers.
- Testing with other tools such as Omnitrace and Omniperf.
- Compiling machine learning libraries and containers.