AOMP: OpenMP Target Offloading for AMD GPUs

Jan-Patrick Lehr
Member of Technical Staff
Software Development Engineer

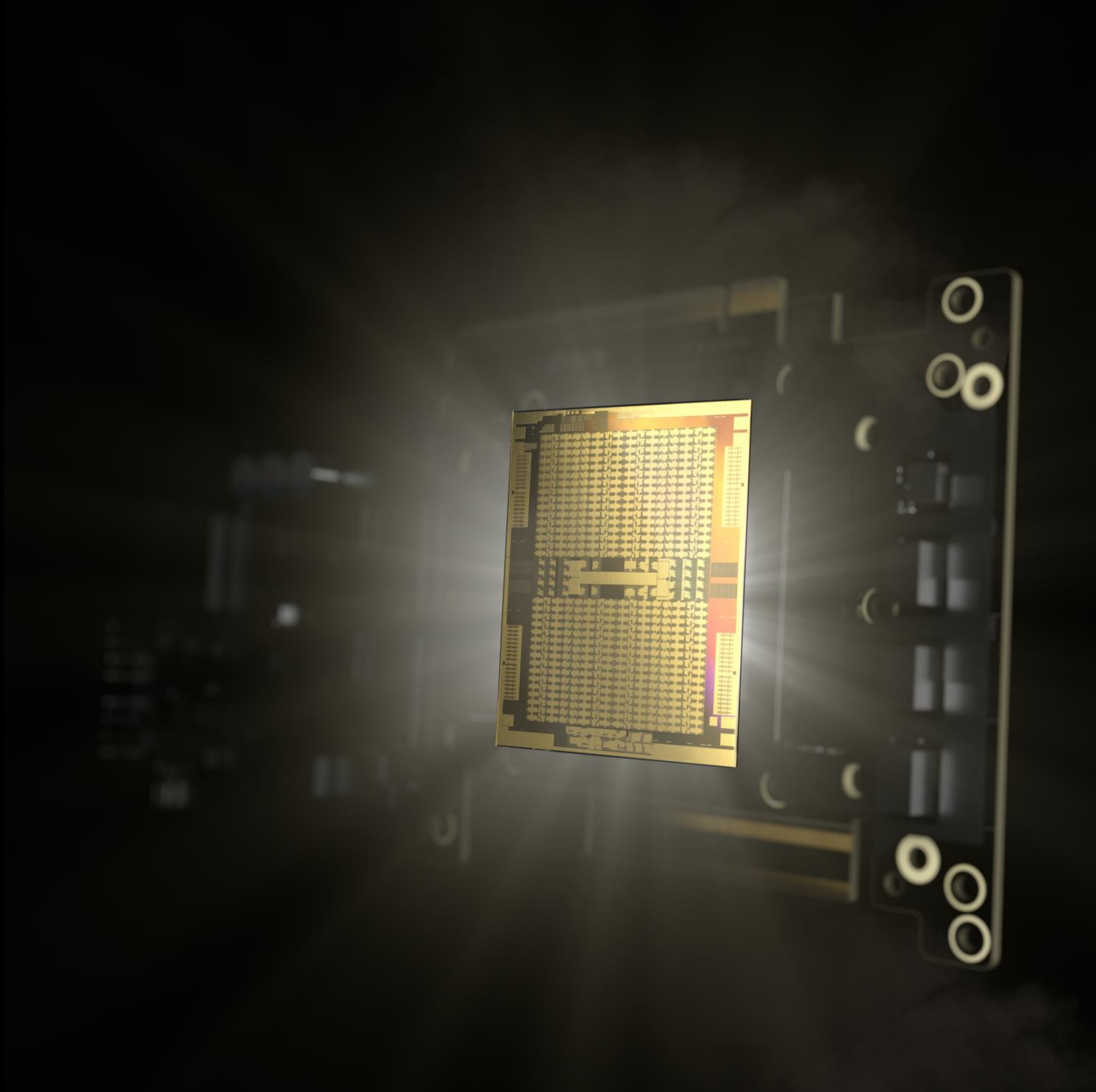
AMD
together we advance_

Disclaimer

Agenda

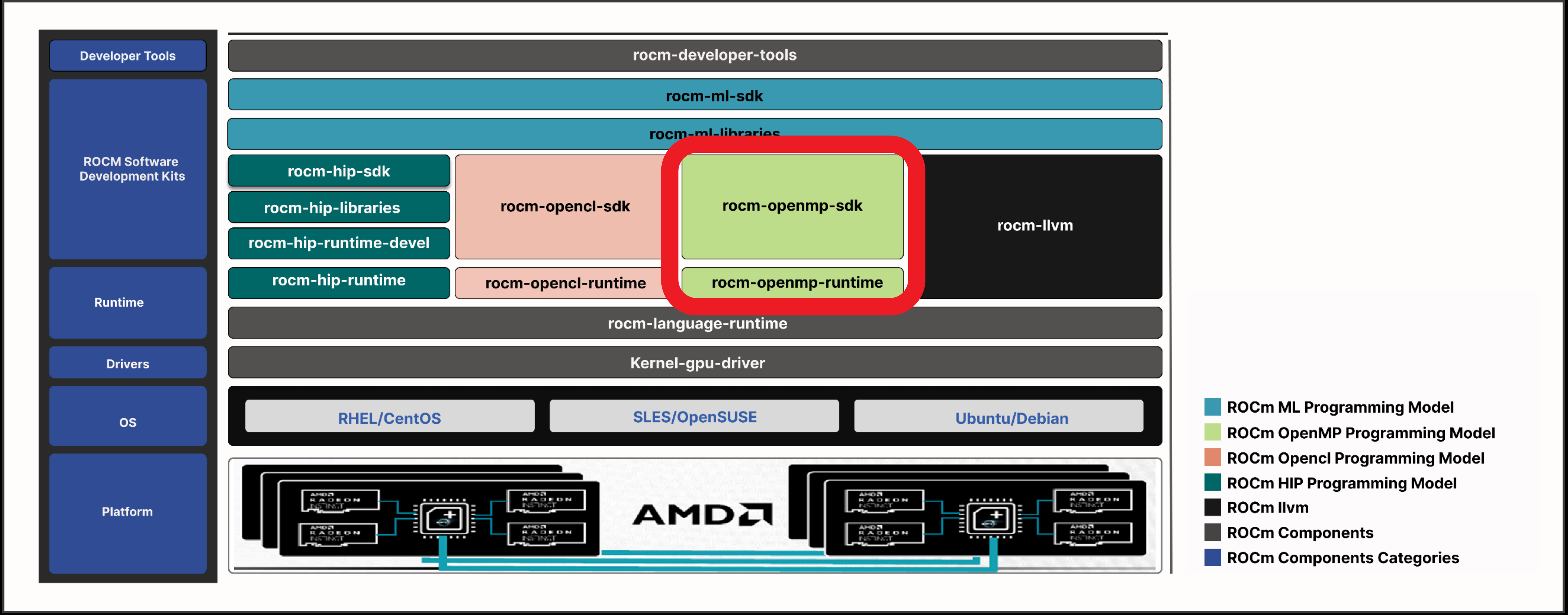
1. A high-level overview of the ROCm™ software stack
2. AOMP and its (software) dependencies
3. AOMP compilation/linking process
4. AOMP architecture
5. Example OpenMP target offload

ROCm™ Software Stack

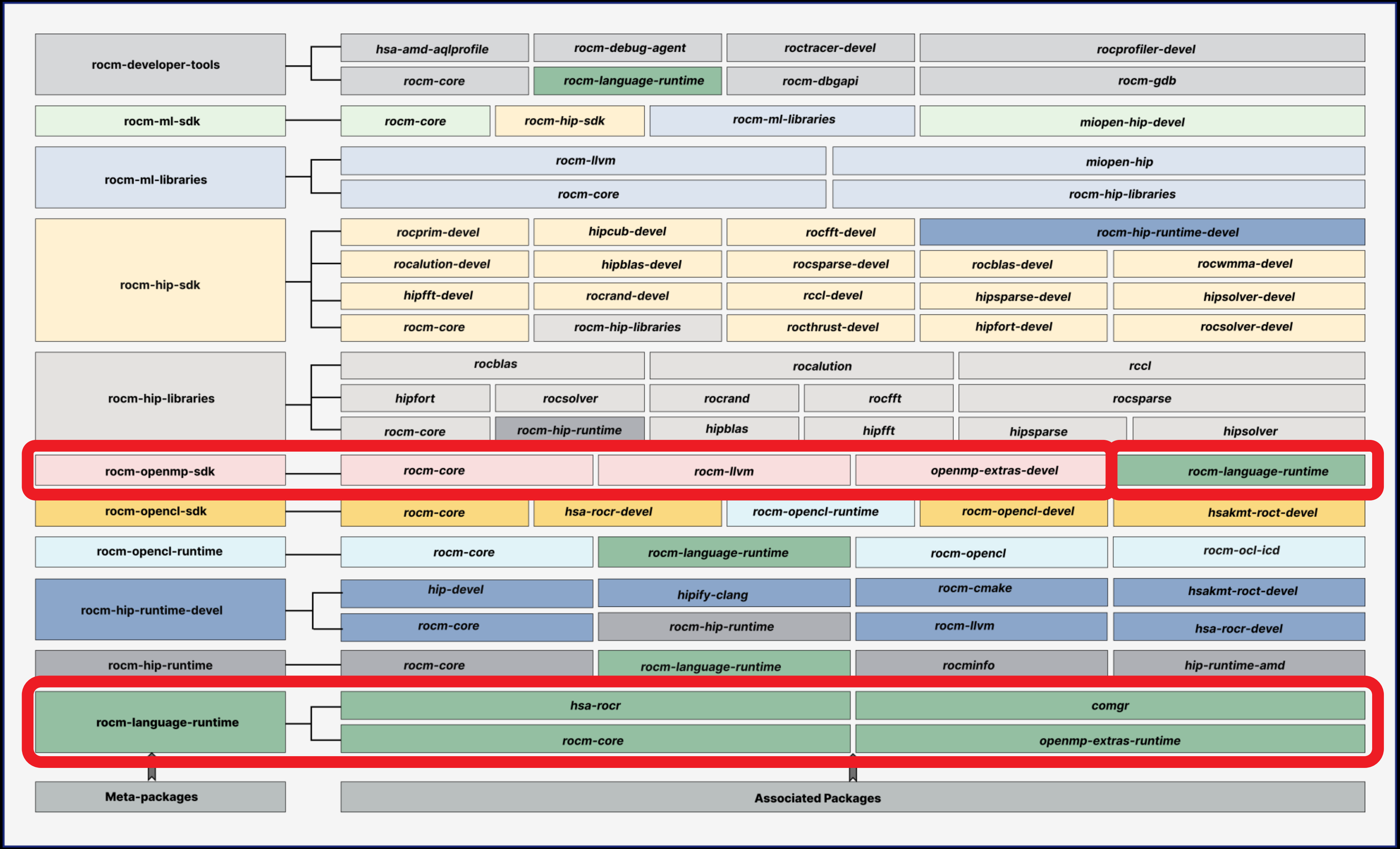


With AMD ROCm™ open software platform built for flexibility and performance, the HPC and AI communities can gain access to open compute languages, compilers, libraries and tools designed to accelerate code development and solve the toughest challenges in the world today.

ROCm™ Software Stack (Meta Packages)



ROCm™ Associated Packages



Meta-packages	Associated Packages
rocm-developer-tools	hsa-amd-aqlprofile, rocm-debug-agent, roctracer-devel, rocprofiler-devel, rocm-core, rocm-language-runtime, rocm-dbgapi, rocm-gdb
rocm-ml-sdk	rocm-core, rocm-hip-sdk, rocm-ml-libraries, miopen-hip-devel
rocm-ml-libraries	rocm-llvm, miopen-hip, rocm-core, rocm-hip-libraries
rocm-hip-sdk	rocprim-devel, hipcub-devel, rocfft-devel, rocm-hip-runtime-devel, rocalution-devel, hipblas-devel, rocsparse-devel, rocblas-devel, rocwmma-devel, hipfft-devel, rocrand-devel, rccl-devel, hipsparse-devel, hipsolver-devel, rocm-core, rocm-hip-libraries, rocthrust-devel, hipfort-devel, rocsolver-devel
rocm-hip-libraries	rocblas, rocalution, rccl, hipfort, rocsolver, rocrand, rocfft, rocsparse, rocm-core, rocm-hip-runtime, hipblas, hipfft, hipsparse, hipsolver
rocm-openmp-sdk	rocm-core, rocm-llvm, openmp-extras-devel, rocm-language-runtime
rocm-opencl-sdk	rocm-core, hsa-rocr-devel, rocm-opencl-runtime, rocm-opencl-devel, hsakmt-roct-devel
rocm-opencl-runtime	rocm-core, rocm-language-runtime, rocm-opencl, rocm-ocl-icd
rocm-hip-runtime-devel	hip-devel, hipify-clang, rocm-cmake, hsakmt-roct-devel, rocm-core, rocm-hip-runtime, rocm-llvm, hsa-rocr-devel
rocm-hip-runtime	rocm-core, rocm-language-runtime, rocminfo, hip-runtime-amd
rocm-language-runtime	hsa-rocr, comgr, rocm-core, openmp-extras-runtime

Software System Terminology

ROCm™

AMD ROCm™ is an open software platform for GPU compute consisting of compilers, libraries, tools, etc.

https://www.amd.com/en/graphics/servers-solutions-rocm

AOMP

AOMP is an open-source Clang/LLVM-based compiler with added support for the OpenMP® API on Radeon™ GPUs that builds on top of ROCm™

Releases are more frequent than **ROC**m™

(https://github.com/ROCm-Developer-Tools/aomp)

LLVM

Since AOMP is Clang/LLVM based, the high-level software architecture of OpenMP target offload support is identical

AOMP is both ahead and behind LLVM mainline trunk (upstream)

Jargon

- Upstream: LLVM mainline trunk (i.e., the code in their GitHub repository)

- Host: The host machine (e.g., AMD Epyc processor-based server)
- (Target) Device: An attached accelerator (e.g., AMD Instinct MI200)
- (Host) Runtime: The OpenMP runtime implementation for the host
- Libomptarget: The OpenMP target offload runtime (running on the host)
- (Target or Device) Runtime: The OpenMP runtime implementation for the target device
- Plugin: The device-specific target-runtime plugin
- Kernel: A target region executable on the device

- *Queue: An HSA (hardware) queue*
- *Signal: An HSA signal*

AOMP

Open Source OpenMP Compiler
for AMD GPUs

AOMP

- **Open Source** OpenMP compiler for AMD GPUs
  - Available from https://github.com/ROCm-Developer-Tools/aomp

- Based on Clang/LLVM
  - AOMP tracks LLVM upstream closely, typically only a few hours behind
  - Includes additional optimizations (e.g., faster reductions)
  - Includes additional features (e.g., OMPT support)
  - Fixes are commonly submitted to upstream LLVM

- Fortran support
  - Currently via classic Flang
  - Active development on LLVM Flang

Github

AOMP

Getting, building and installing AOMP (and its dependencies)

- AOMP is “standalone” → requires only the kernel module (dkms) and libdrm
- AOMP is isolated from ROCm™ installations by installing in /usr/lib/aomp and uses RPATH on runtime libs
- AOMP includes builds of related ROCm™ components

- Packages are provided on Github for
  - CentOS 7, 8, and 9
  - SLES 15 SP 4
  - Ubuntu 20.04 and 22.04

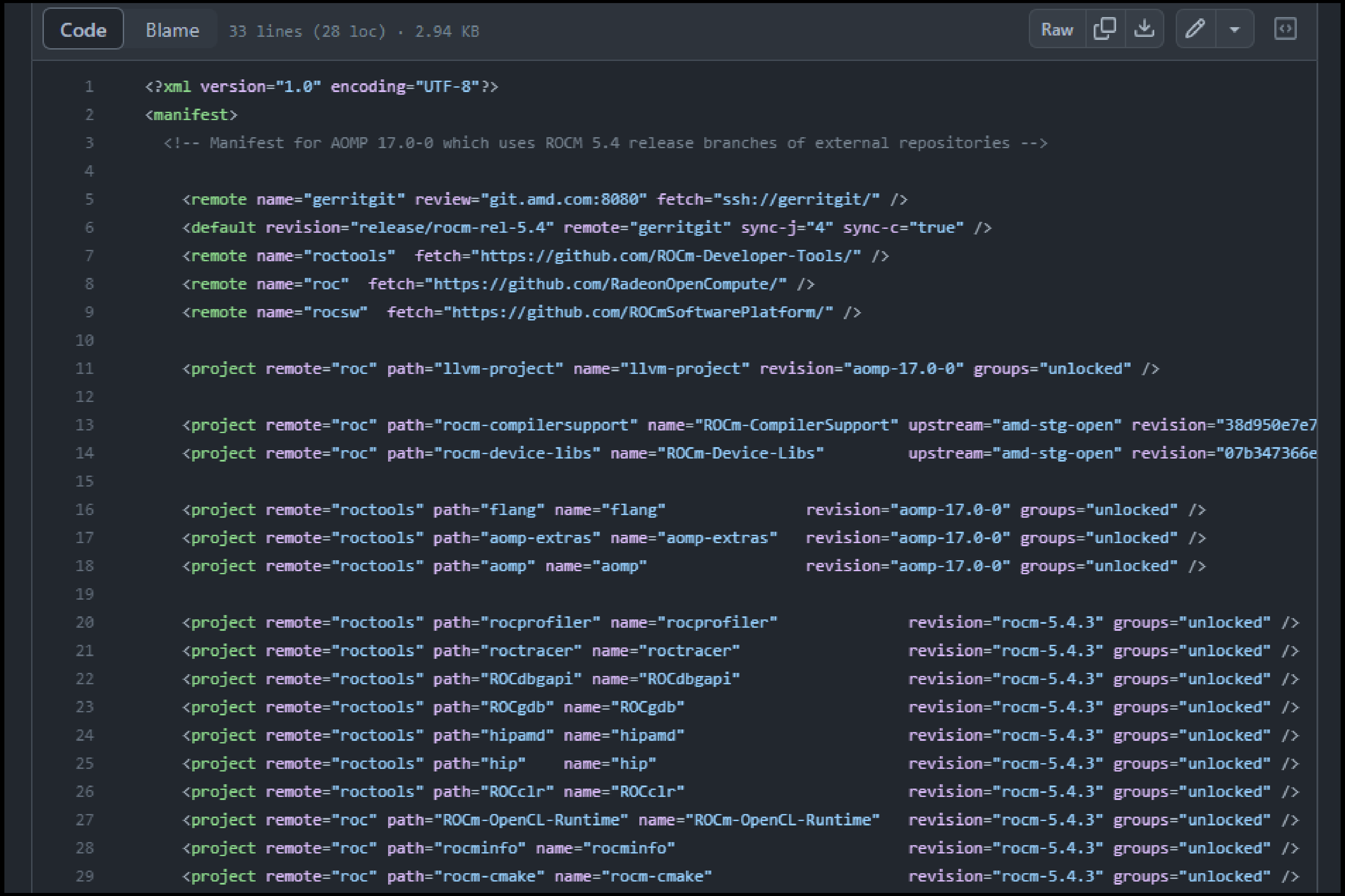
Components from ROCm™

- AOMP includes various ROCm™ components
  - The components are built from source
  - Versions and repositories are encoded in manifest file in the Github repository
  - Build scripts for individual components build the dependencies
  - All components are installed in AOMP install directory

ROCm™ Components

- rocm-compiler-support
- rocm-device-libs
- ROCprofiler
- ROCtracer
- ROCdbgapi
- ROCgdb
- Hipamd
- Hip
- ROCclr
- ROCm-OpenCL-runtime
- ROCminfo
- ROCm-cmake
- ROCR-Runtime
- ROCT-Thunk-Interface
- hipfort

AOMP Manifest File



```xml
<?xml version="1.0" encoding="UTF-8"?>
<manifest>
  <!-- Manifest for AOMP 17.0-0 which uses ROCM 5.4 release branches of external repositories -->

    <remote name="gerritgit" review="git.amd.com:8080" fetch="ssh://gerritgit/" />
    <default revision="release/rocm-rel-5.4" remote="gerritgit" sync-j="4" sync-c="true" />
    <remote name="roctools"  fetch="https://github.com/ROCm-Developer-Tools/" />
    <remote name="roc"  fetch="https://github.com/RadeonOpenCompute/" />
    <remote name="rocsw"  fetch="https://github.com/ROCmSoftwarePlatform/" />

    <project remote="roc" path="llvm-project" name="llvm-project" revision="aomp-17.0-0" groups="unlocked" />

    <project remote="roc" path="rocm-compilersupport" name="ROCm-CompilerSupport" upstream="amd-stg-open" revision="38d950e7e7
    <project remote="roc" path="rocm-device-libs" name="ROCm-Device-Libs"          upstream="amd-stg-open" revision="07b347366e

    <project remote="roctools" path="flang" name="flang"               revision="aomp-17.0-0" groups="unlocked" />
    <project remote="roctools" path="aomp-extras" name="aomp-extras"   revision="aomp-17.0-0" groups="unlocked" />
    <project remote="roctools" path="aomp" name="aomp"                 revision="aomp-17.0-0" groups="unlocked" />

    <project remote="roctools" path="rocprofiler" name="rocprofiler"              revision="rocm-5.4.3" groups="unlocked" />
    <project remote="roctools" path="roctracer" name="roctracer"                  revision="rocm-5.4.3" groups="unlocked" />
    <project remote="roctools" path="ROCdbgapi" name="ROCdbgapi"                  revision="rocm-5.4.3" groups="unlocked" />
    <project remote="roctools" path="ROCgdb" name="ROCgdb"                        revision="rocm-5.4.3" groups="unlocked" />
    <project remote="roctools" path="hipamd" name="hipamd"                        revision="rocm-5.4.3" groups="unlocked" />
    <project remote="roctools" path="hip"    name="hip"                           revision="rocm-5.4.3" groups="unlocked" />
    <project remote="roctools" path="ROCclr" name="ROCclr"                        revision="rocm-5.4.3" groups="unlocked" />
    <project remote="roc" path="ROCm-OpenCL-Runtime" name="ROCm-OpenCL-Runtime"   revision="rocm-5.4.3" groups="unlocked" />
    <project remote="roc" path="rocminfo" name="rocminfo"                         revision="rocm-5.4.3" groups="unlocked" />
    <project remote="roc" path="rocm-cmake" name="rocm-cmake"                     revision="rocm-5.4.3" groups="unlocked" />
```

Used to tag dependencies for each AOMP release

- Used by AOMP team for development on most recent development branches
- Specific versions for each public release

AMD
together we advance_

AOMP

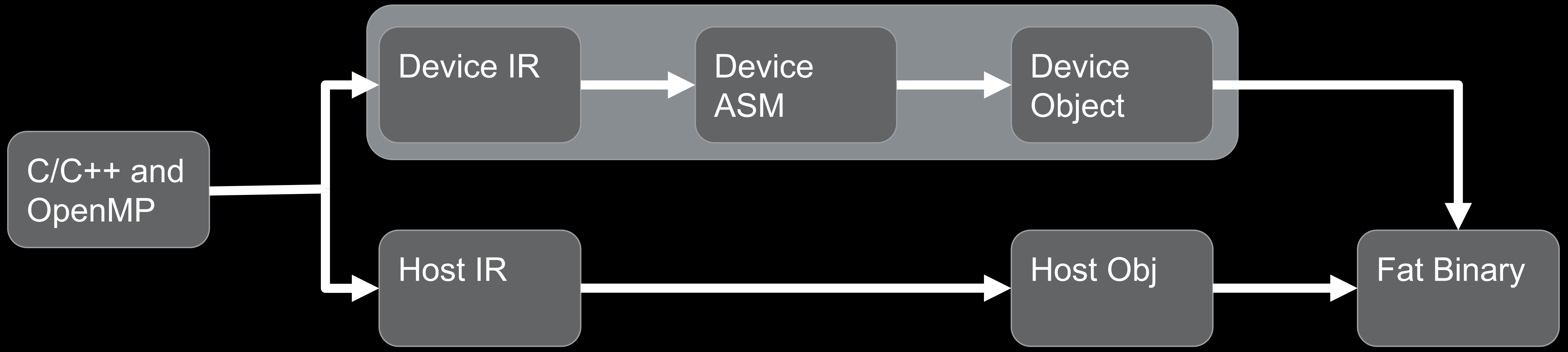
Compilation Process

Execution

Device Plugin Architecture

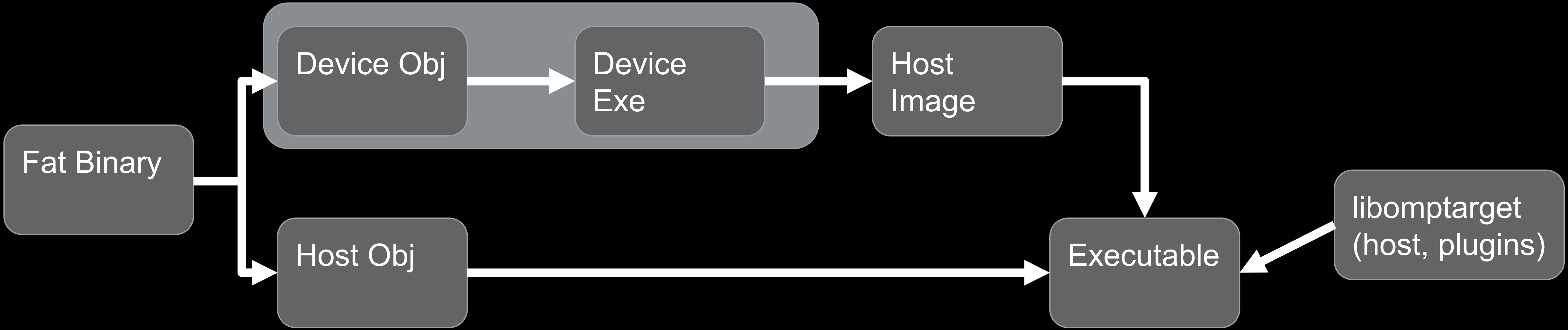
AOMP Compilation Process

- Compilation requires code generation for host and for device
  - Invoke compiler twice for the different target triples
  - Object code for both targets is output into a single fat binary

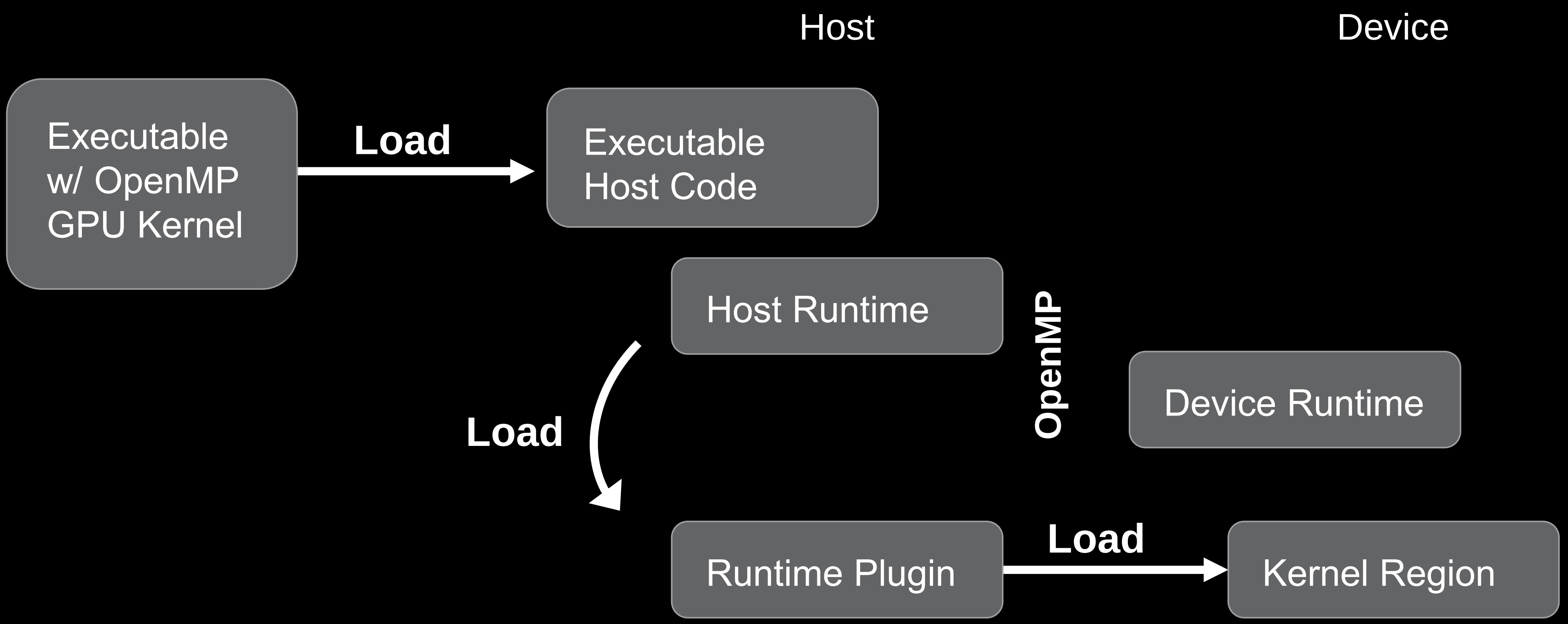


AOMP Linking Process

- Linking requires generating one host image and an embedded device executable
  - Start with fat binary and *unbundle* into device and host objects again
  - Create a correctly linked host executable with embedded device ELF objects for offload kernels
- Moving towards a simplified toolchain, closer to today's upstream LLVM



High-level Runtime Process Overview

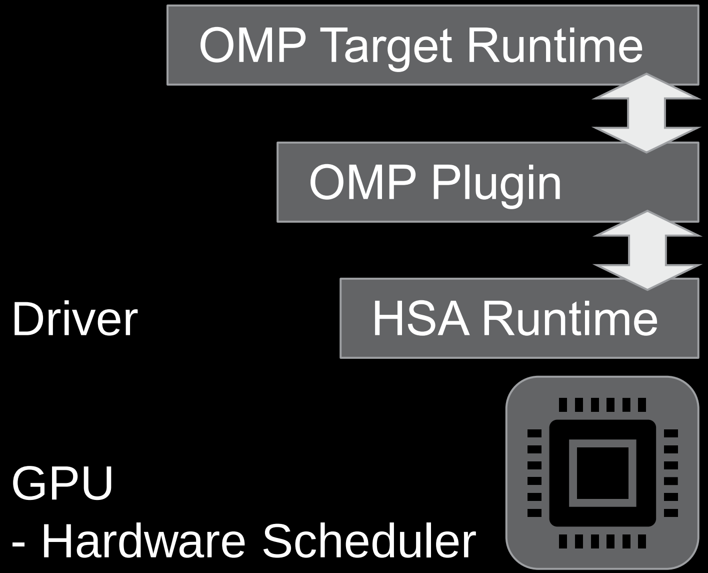


AOMP Plugin Implementation using HSA

- HSA is the Heterogenous System Architecture, managed by the HSA foundation
  - Provides standardized interface to interact with heterogenous components
  - AMD provides vendor-specific extensions to the HSA standard as part of the ROCm™ stack

- AMDGPU Plugin builds on top of HSA making use of
  - HSA locked or pinned memory (i.e., non-migratable)
  - HSA Signals and Queues for (asynchronous) operation dispatch
  - HSA extensions to obtain profiling information for HSA signals

- HSA is low level!
  - `hsa_signal_create( … )`
  - `hsa_signal_wait_relaxed( … )`
  - `hsa_queue_load_write_index_acquire( … )`
  - `hsa_queue_store_write_index_relaxed( … )`



Example of OpenMP Target Offload

OpenMP Target Offload Code Example

```
int main(int argc, char **argv) {
  int vals[1024] = {0};

  #pragma omp target teams distribute parallel for map(vals)
  for(int i = 0; i < 1024; ++i) {
    vals[i] = 1;
  }

  for(const auto vi : vals) {
    std::cout << vi << ‘\n’;
  }
  return 0;
}
```

OpenMP Target Offload Code Example

```
int main(int argc, char **argv) {
  int vals[1024] = {0};

  #pragma omp target teams distribute parallel for map(tofrom: vals)
  for(int i = 0; i < 1024; ++i) {
    vals[i] = 1;
  }

  for(const auto vi : vals) {
    std::cout << vi << ‘\n’;
  }
  return 0;
}
```

OpenMP Target Offload Code Example

```
int main(int argc, char **argv) {
  int vals[1024] = {0};

  #pragma omp target teams distribute parallel for map(vals)
  for(int i = 0; i < 1024; ++i) {
    vals[i] = 1;
  }

  for(const auto vi : vals) {
    std::cout << vi << ‘\n’;
  }
  return 0;
}
```

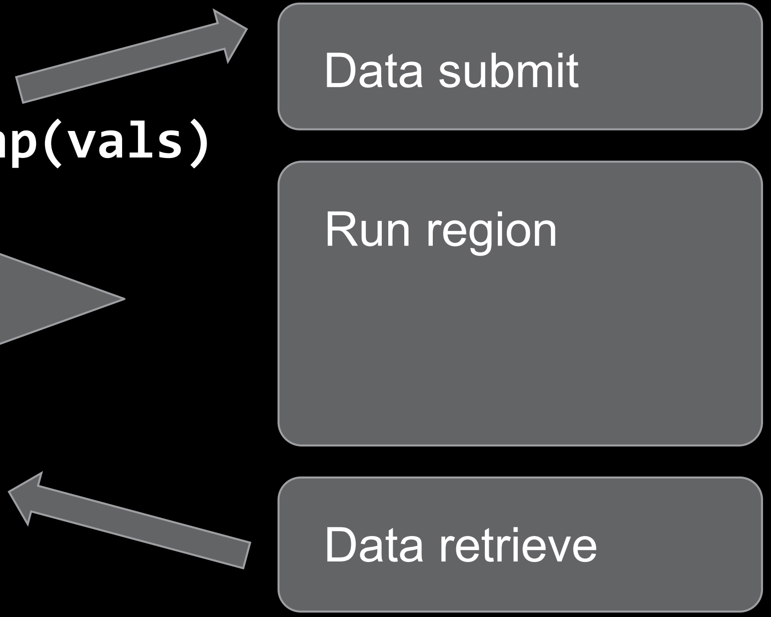
Executes on the device

Libomptarget Plugin Entry Points

```
int main(int argc, char **argv) {
  int vals[1024] = {0};

  #pragma omp target teams distribute parallel for map(vals)
  for(int i = 0; i < 1024; ++i) {
    vals[i] = 1;
  }

  for(const auto vi : vals) {
    std::cout << vi << ‘\n’;
  }
  return 0;
}
```



AOMP Summary

- Open-source compiler for OpenMP target offload based on Clang/LLVM
  - Support for C/C++/Fortran
  - Features are merged into ROCm™ releases

- Additional optimizations compared to upstream Clang/LLVM

- Provides standalone build and installation from source using provided scripts
  - Requires the kernel driver to be installed on the system
  - Pulls-in the required ROCm™ dependencies