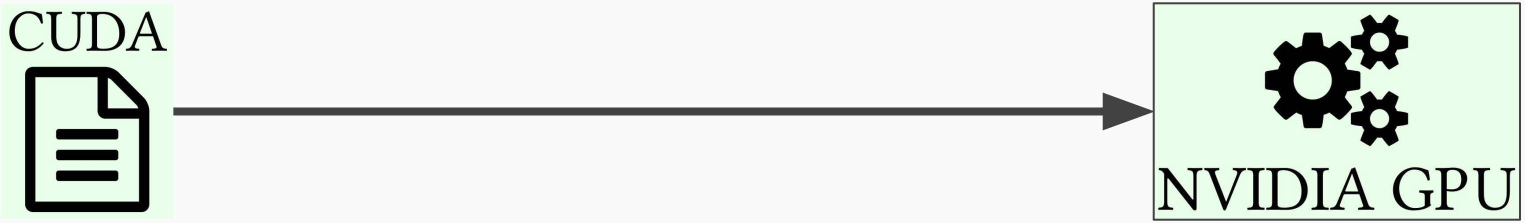
CUDA-OMP – Or: Breaking the Vendor Lock

Performance Portable Programming Through OpenMP as Target Independent Runtime Layer

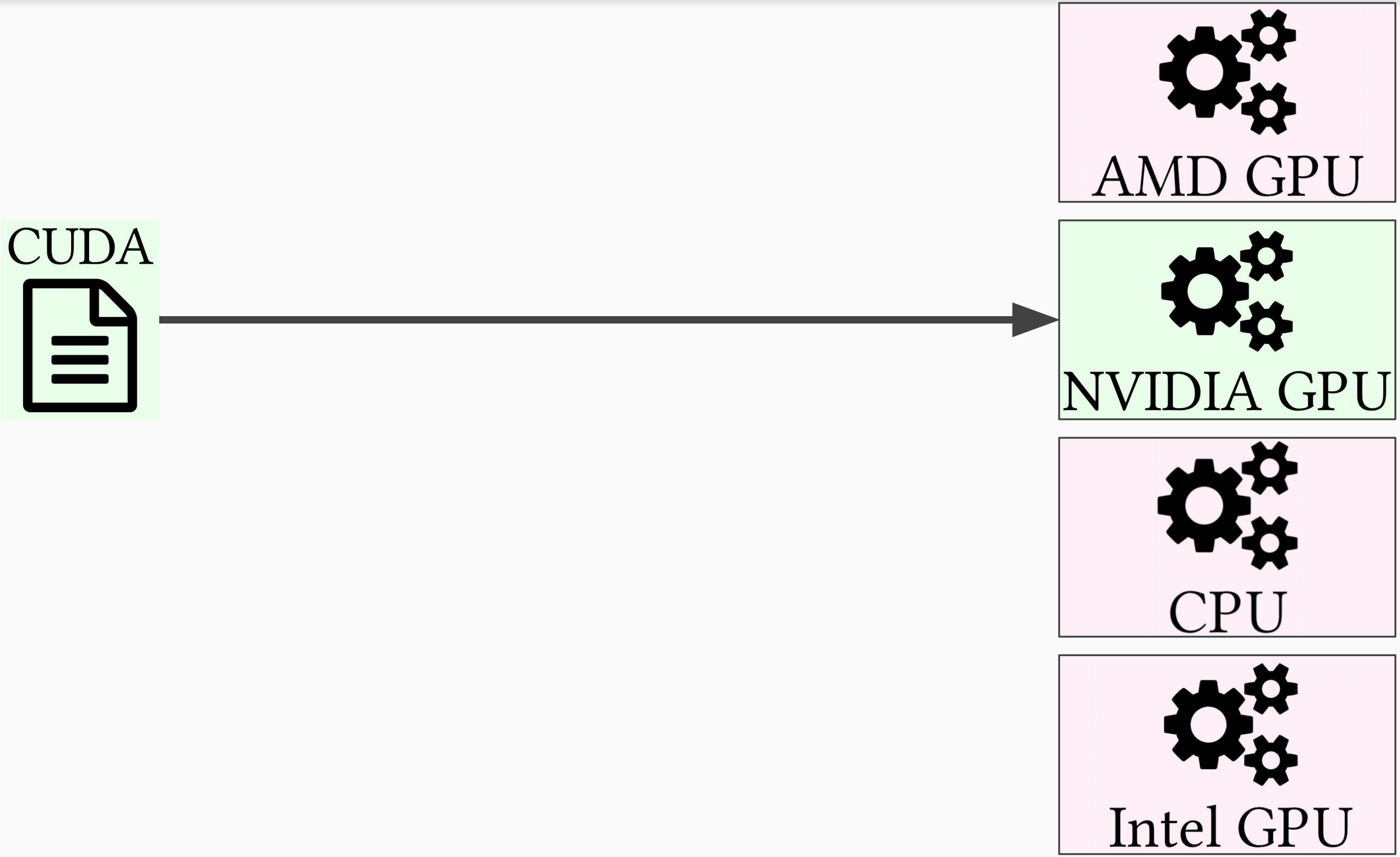
Johannes Doerfert[1,2], Mark Jasper[1], Joseph Huber[3,4], Khaled Abdelaal[5], Giorgis Georgakoudis[1], Thomas Scogland[1], Konstantinos Parasyris[1]

[1] LLNL: Lawrence Livermore National Laboratory
[2] ANL: Argonne National Laboratory (past, email is dead)
[3] ORNL: Oak Ridge National Laboratory (past, email is dead)
[4] AMD
[5] University of Oklahoma (ANL intern)

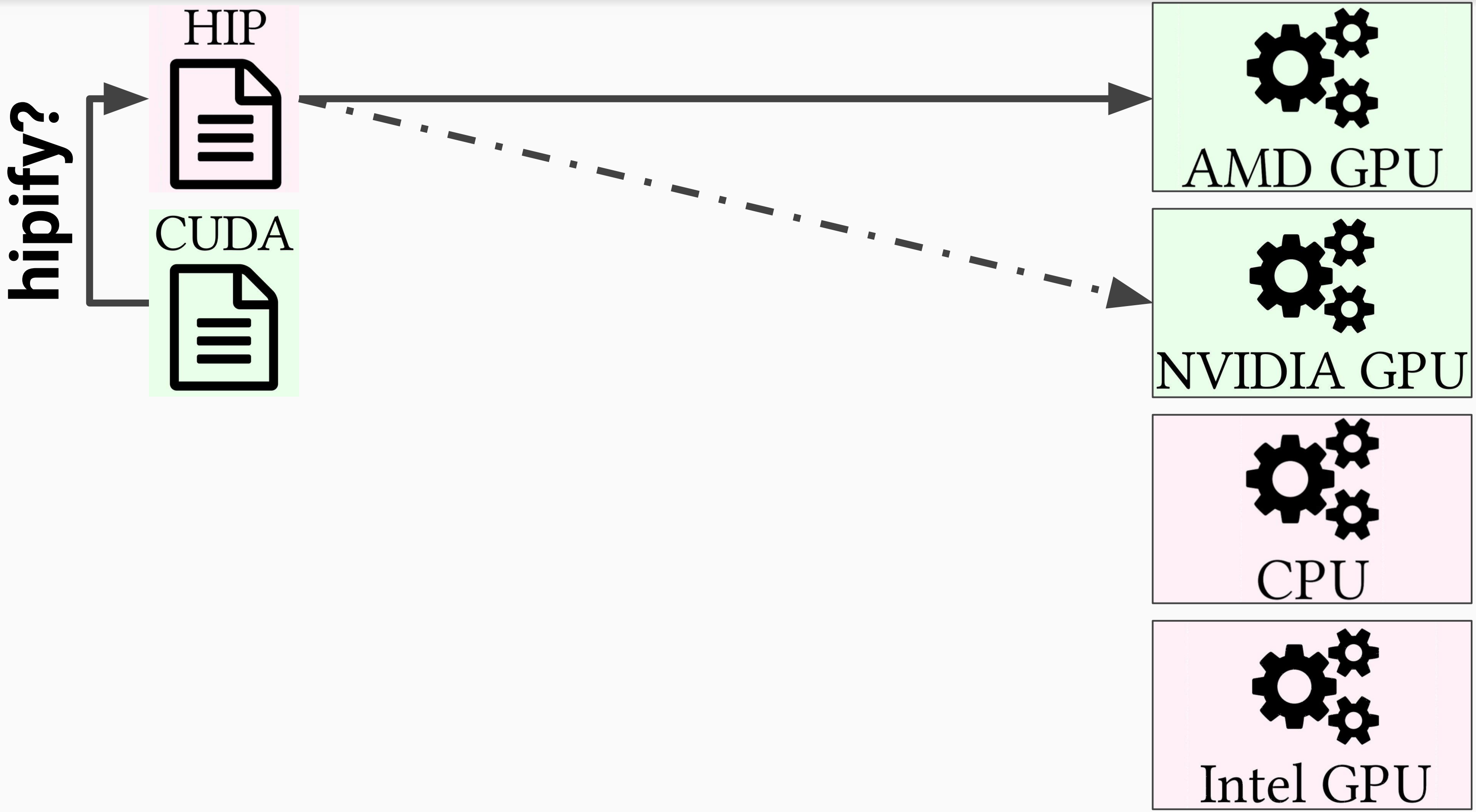
Motivation



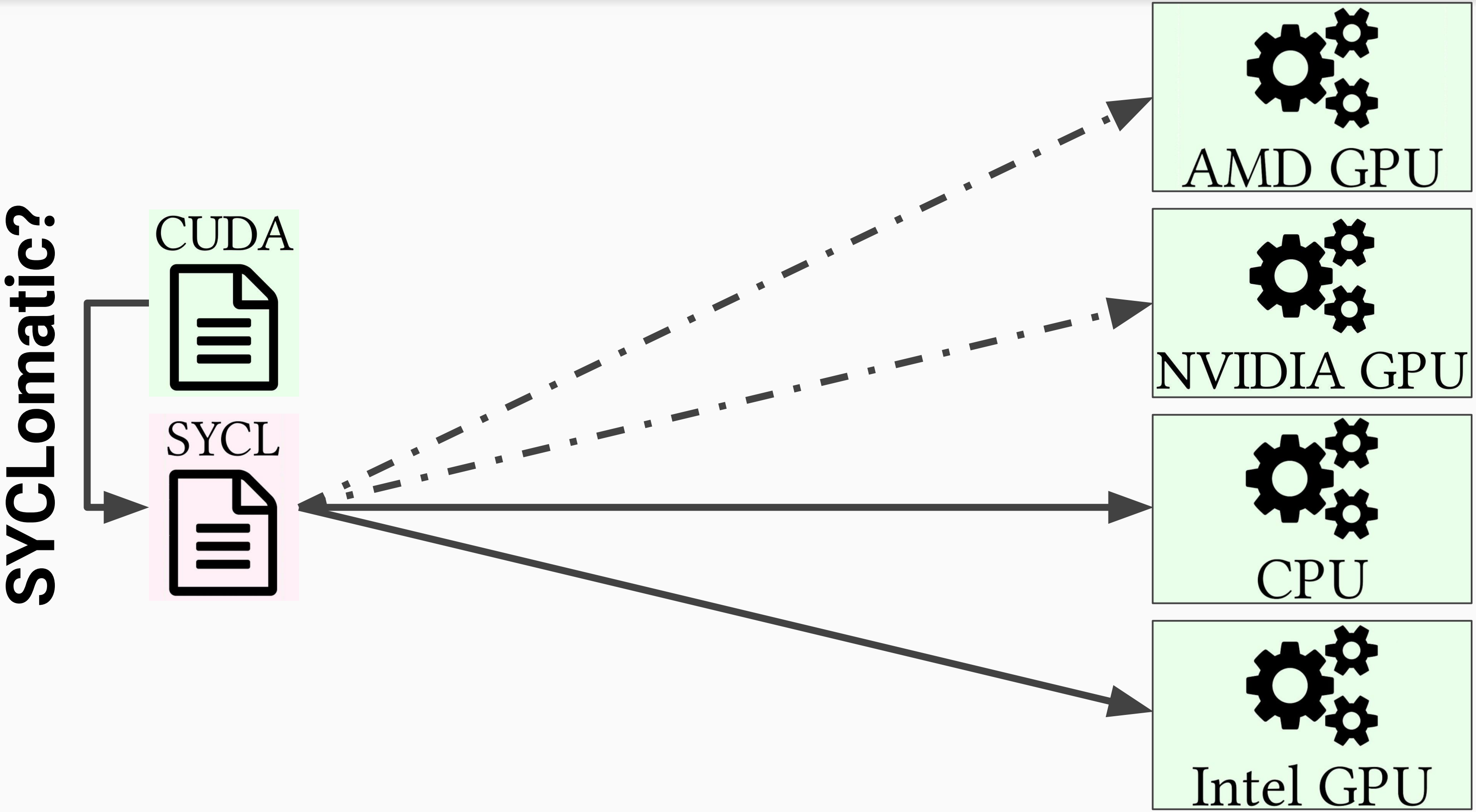
Motivation



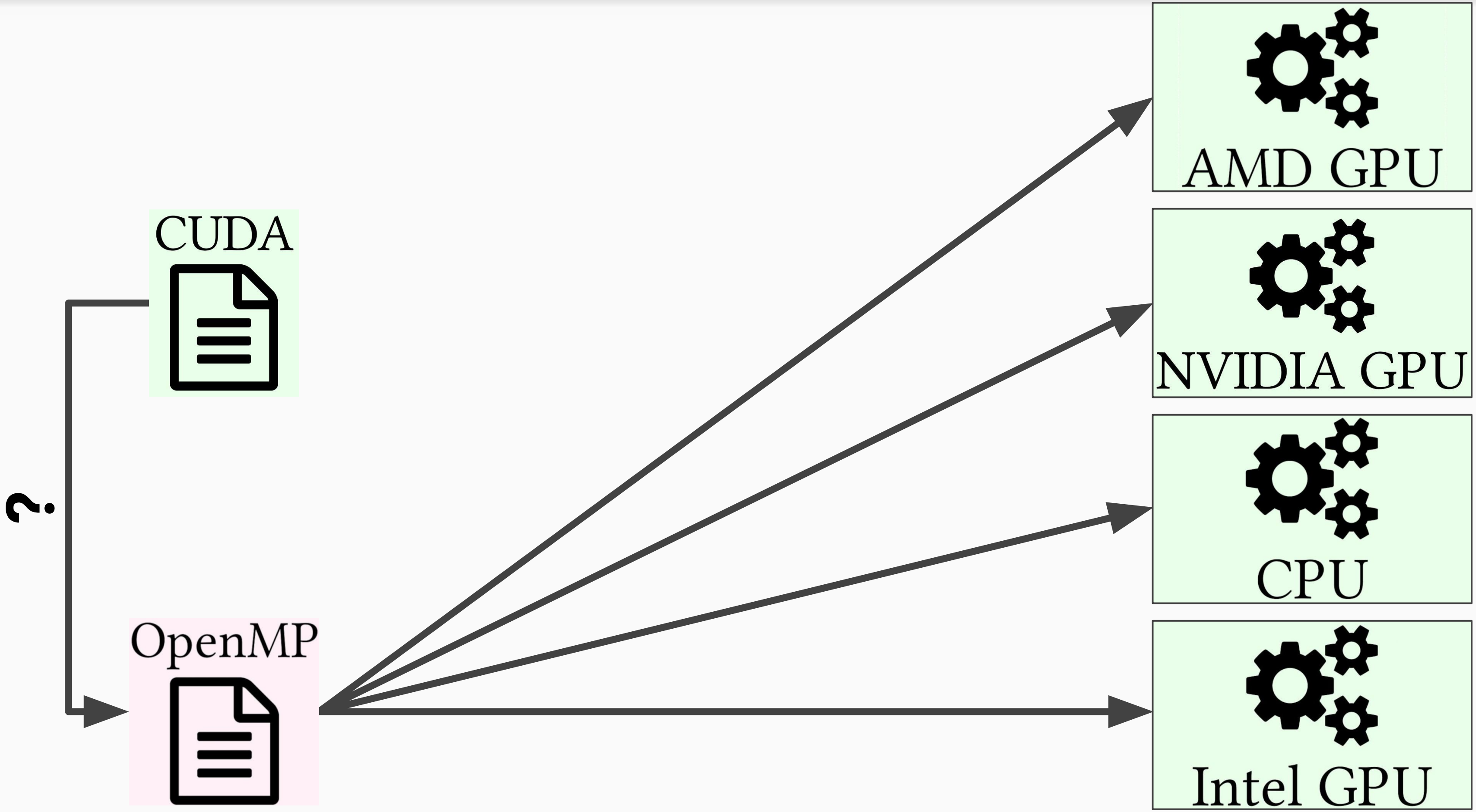
Motivation



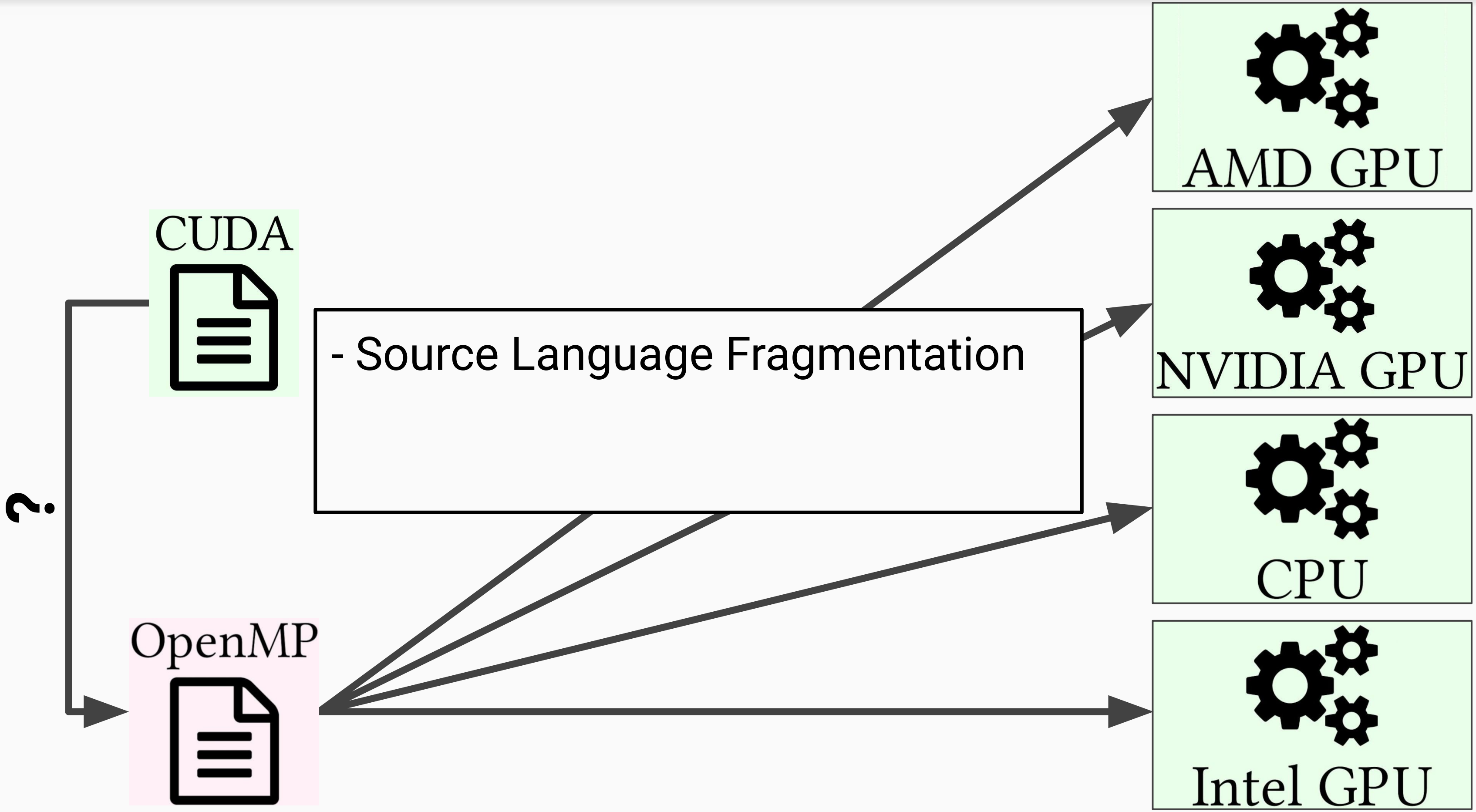
Motivation



Motivation

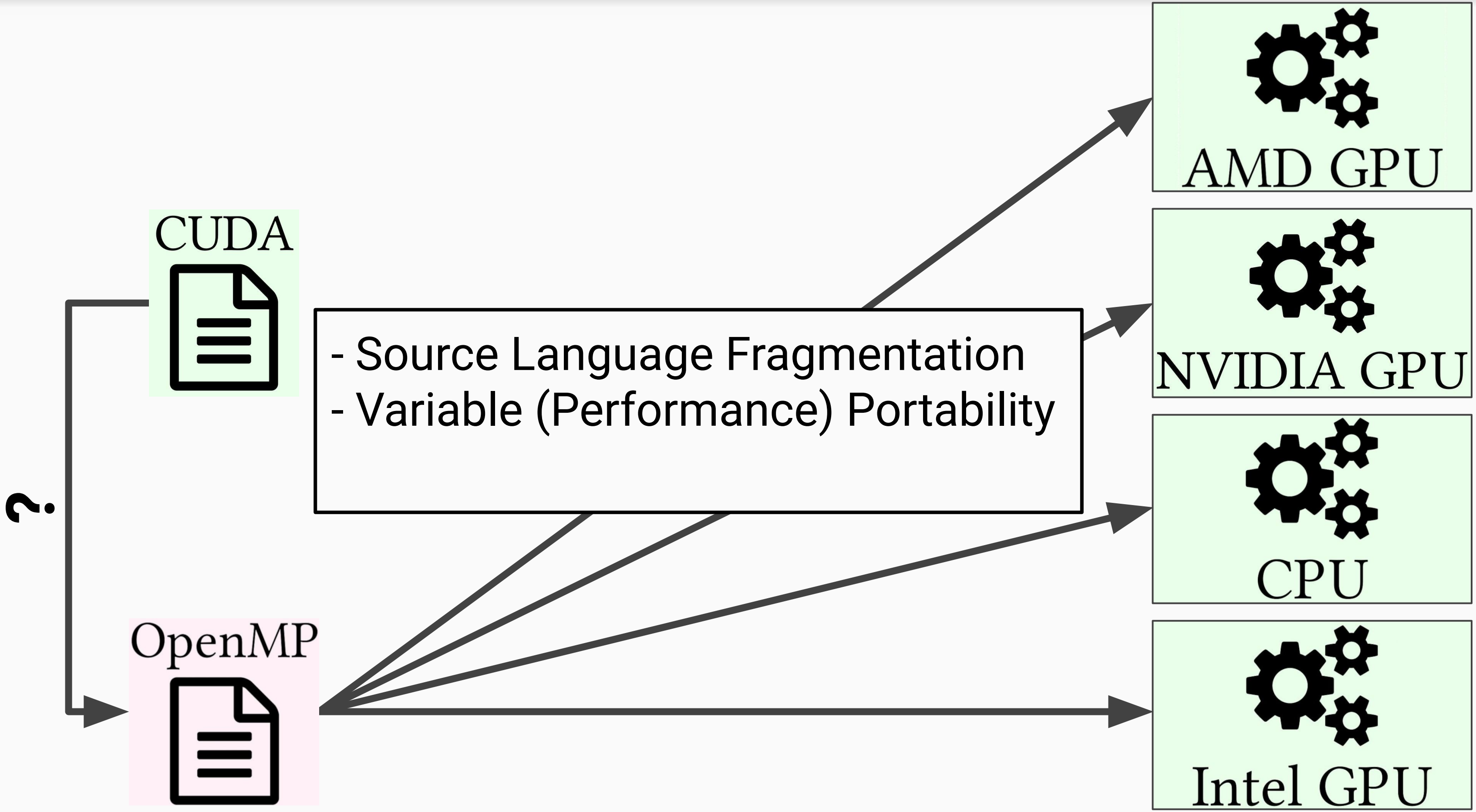


Motivation



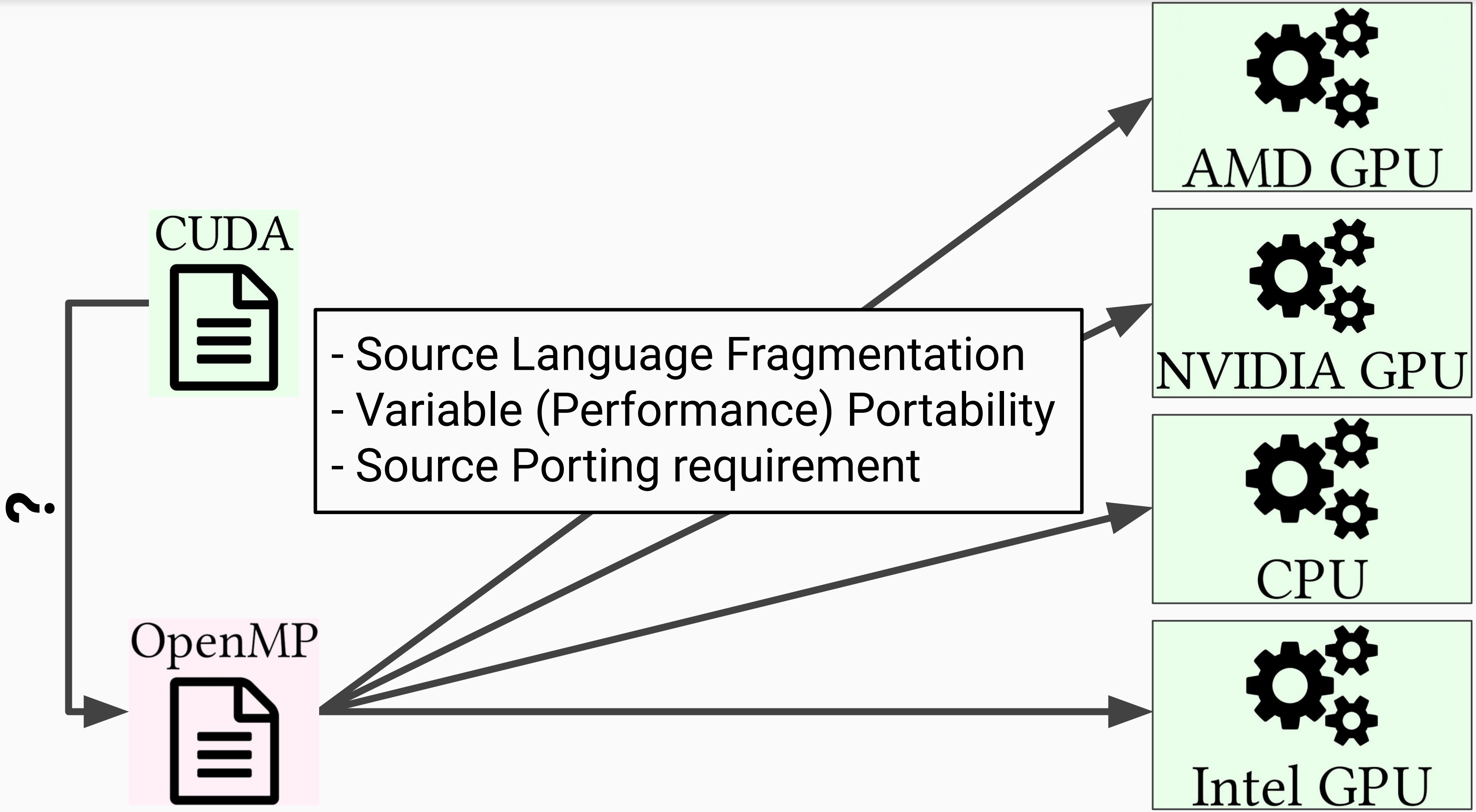
- Source Language Fragmentation

Motivation



- Source Language Fragmentation
- Variable (Performance) Portability

Motivation



CUDA

?

OpenMP

- Source Language Fragmentation
- Variable (Performance) Portability
- Source Porting requirement

AMD GPU

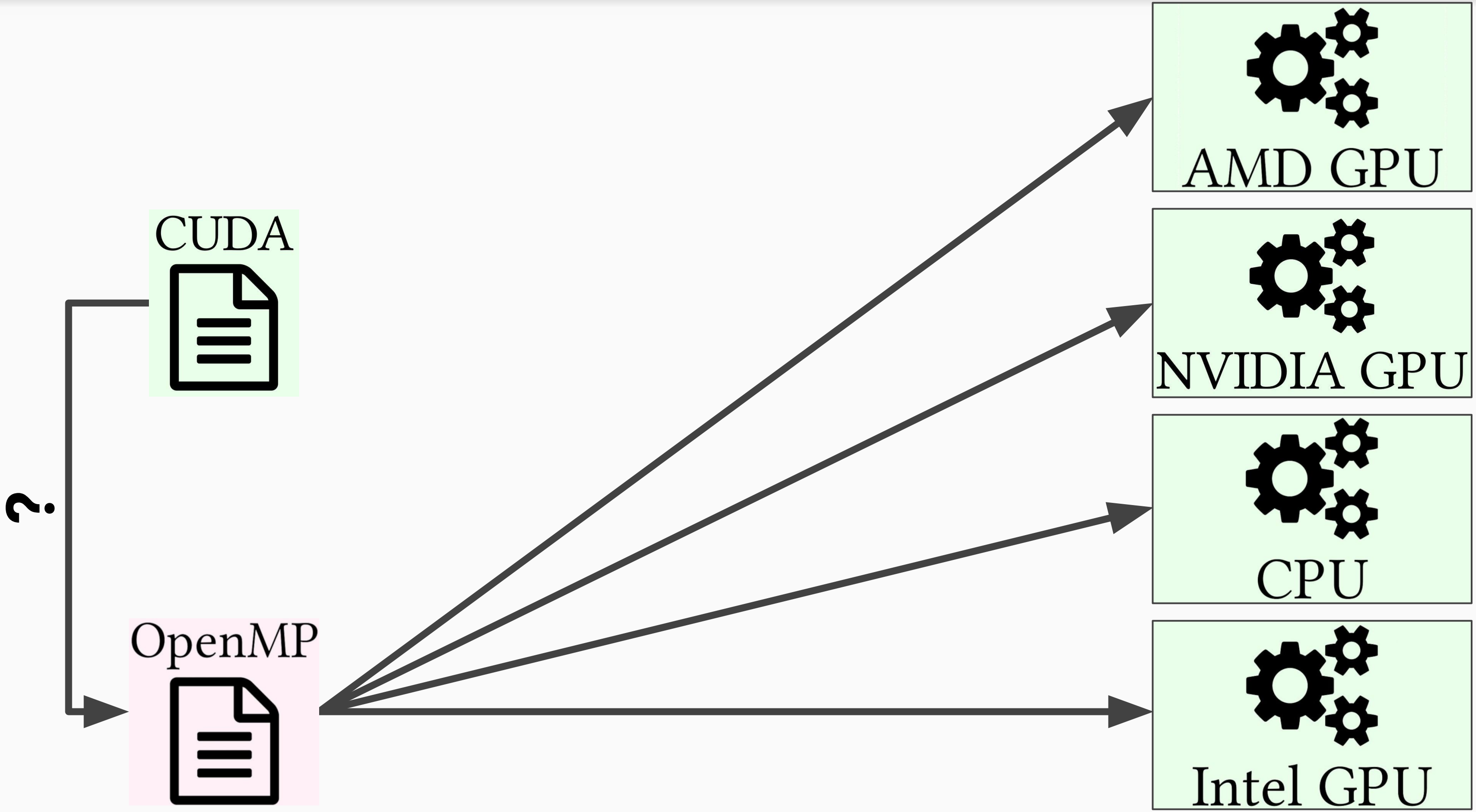
NVIDIA GPU

CPU

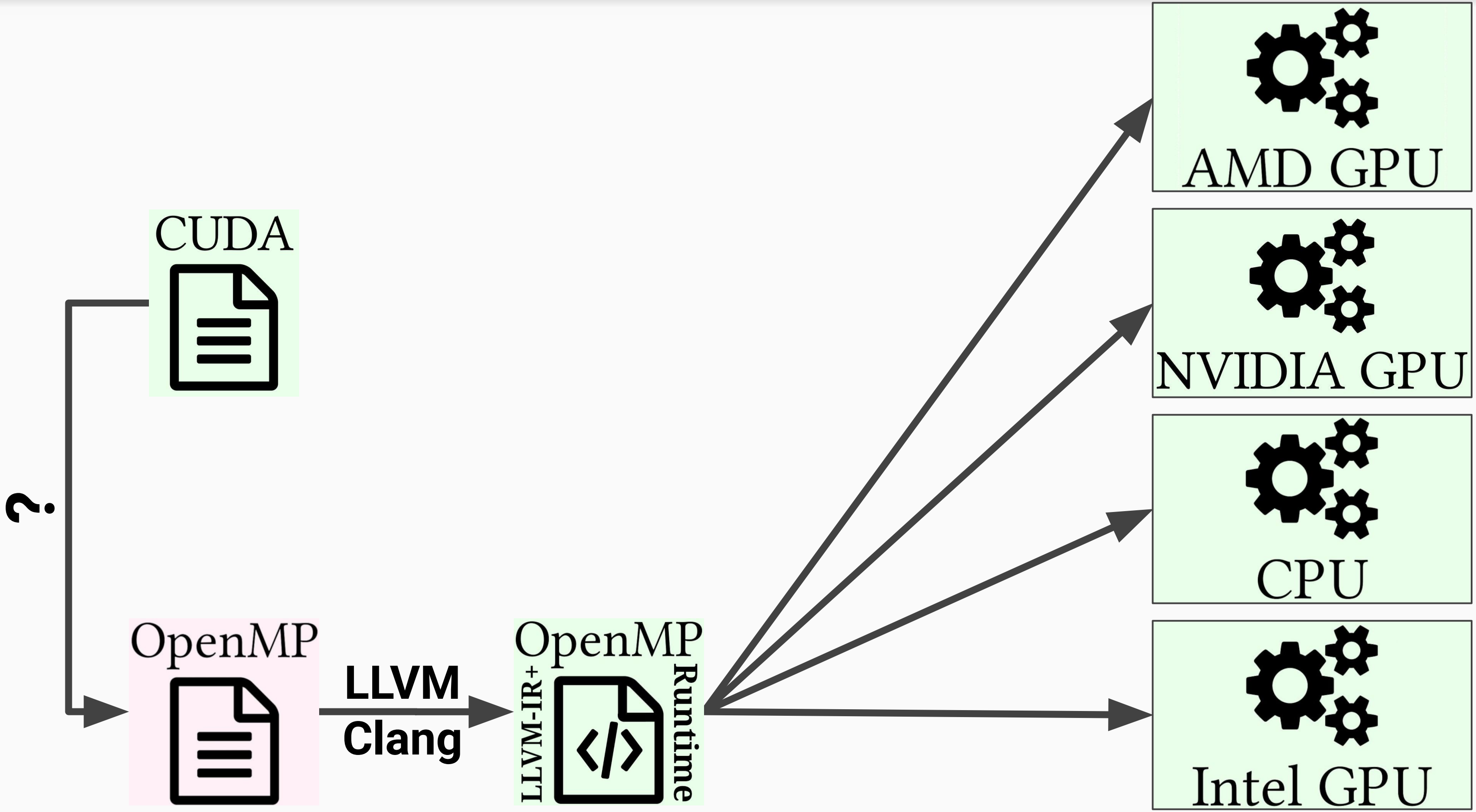
Intel GPU

Approach

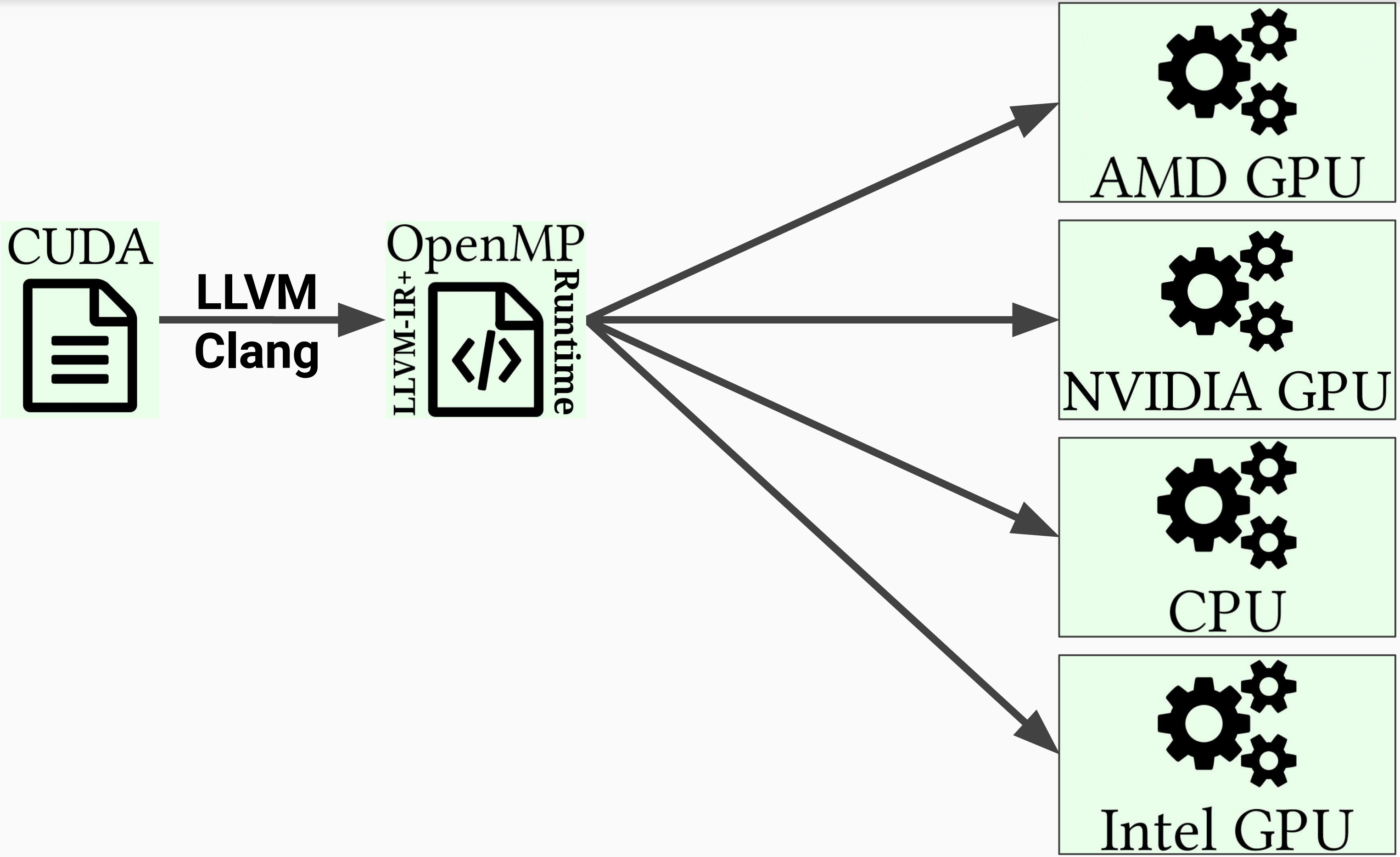
Approach



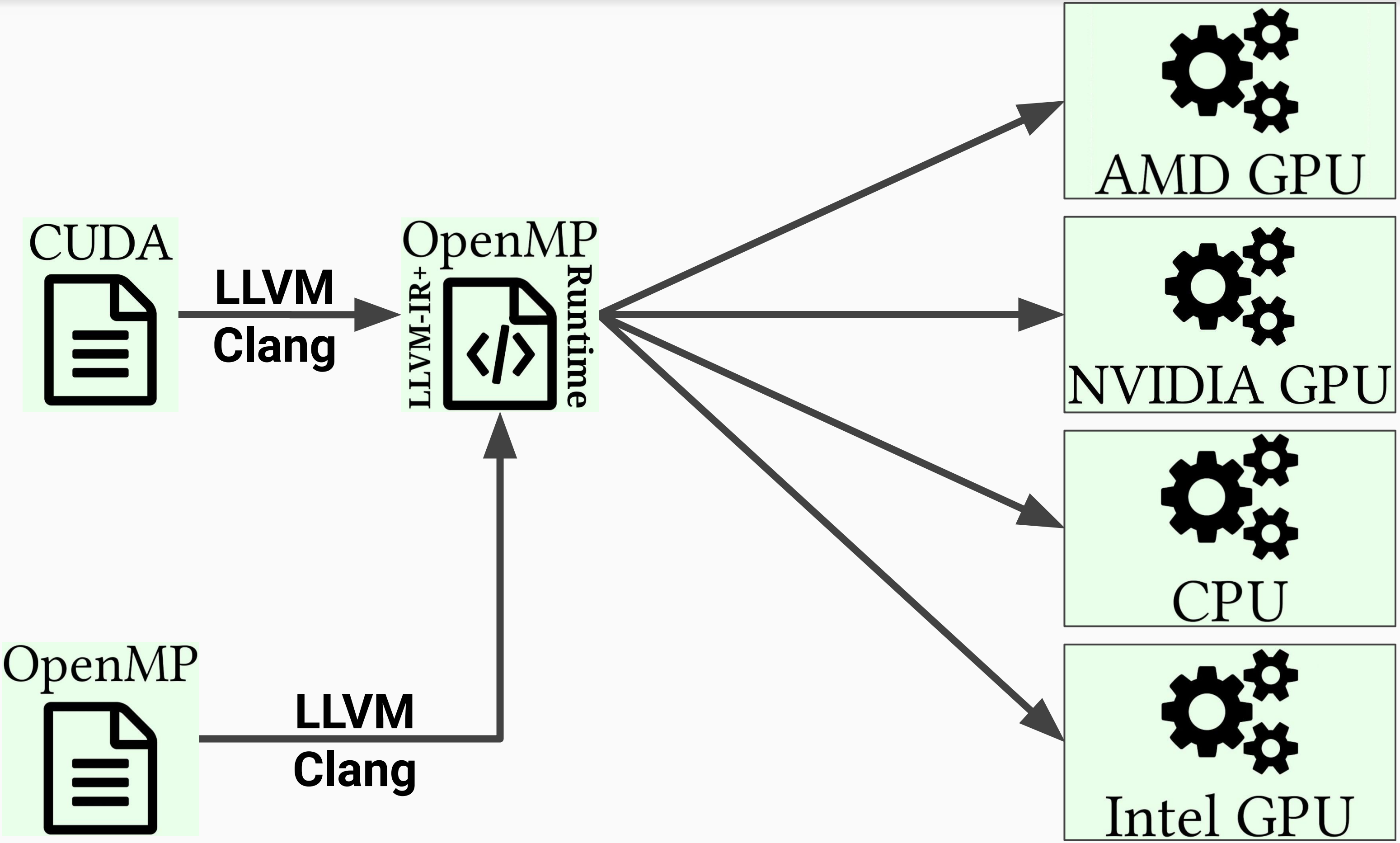
Approach


AMD GPU
NVIDIA GPU
CPU
Intel GPU
CUDA
?
OpenMP
LLVM
Clang
OpenMP
LLVM-IR+
Runtime

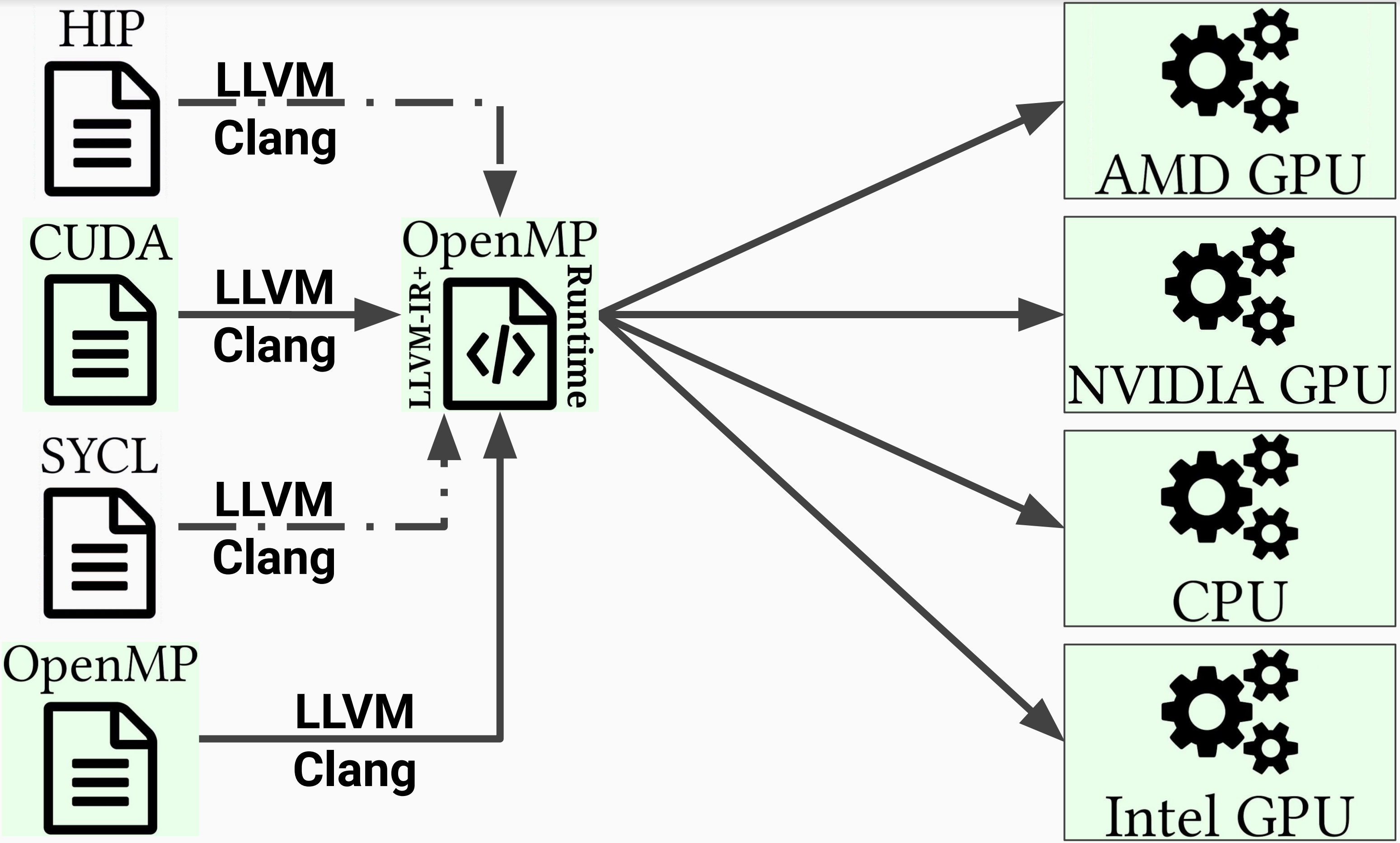
Approach


CUDA
LLVM
Clang
OpenMP
LLVM-IR+
Runtime
AMD GPU
NVIDIA GPU
CPU
Intel GPU

Approach

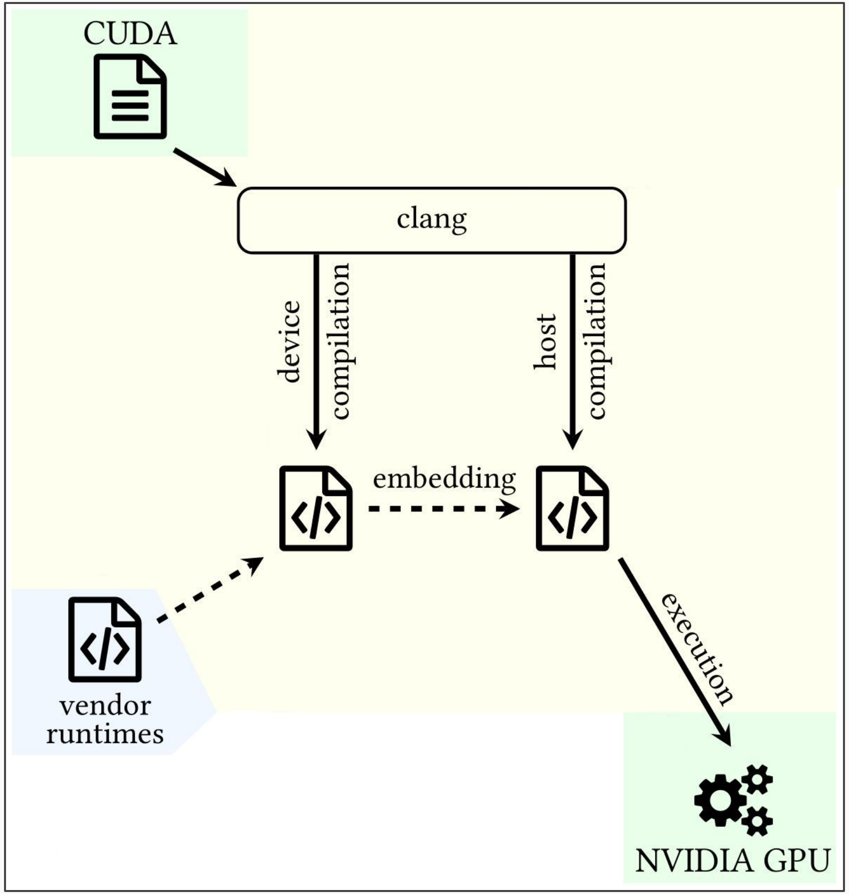

CUDA
LLVM
Clang
OpenMP
LLVM-IR+
Runtime
OpenMP
LLVM
Clang
AMD GPU
NVIDIA GPU
CPU
Intel GPU

Approach

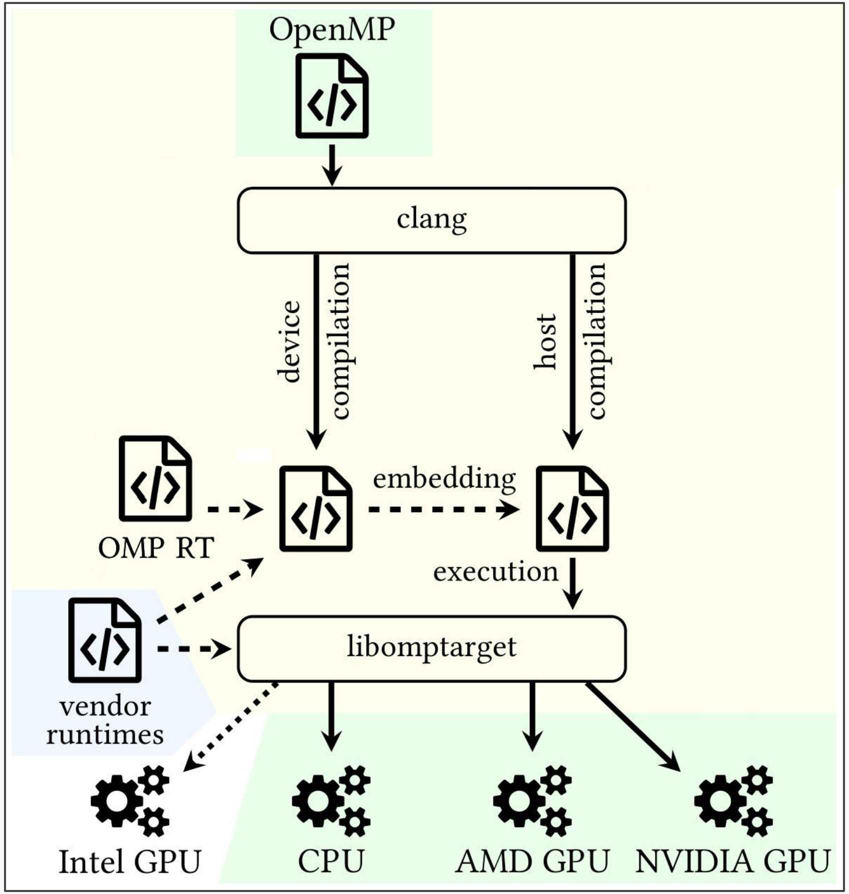

HIP
LLVM
Clang
CUDA
LLVM
Clang
SYCL
LLVM
Clang
OpenMP
LLVM
Clang
OpenMP
LLVM-IR+
Runtime
AMD GPU
NVIDIA GPU
CPU
Intel GPU

Details

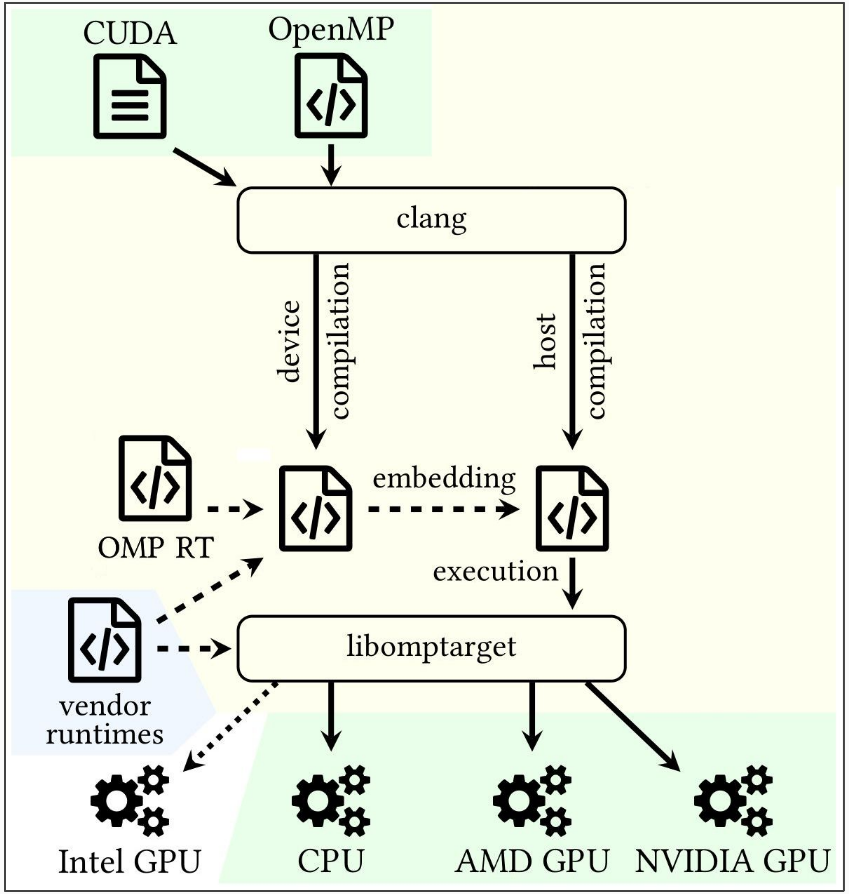
Overview – CUDA via OpenMP Offload Compilation



Overview – CUDA via OpenMP Offload Compilation

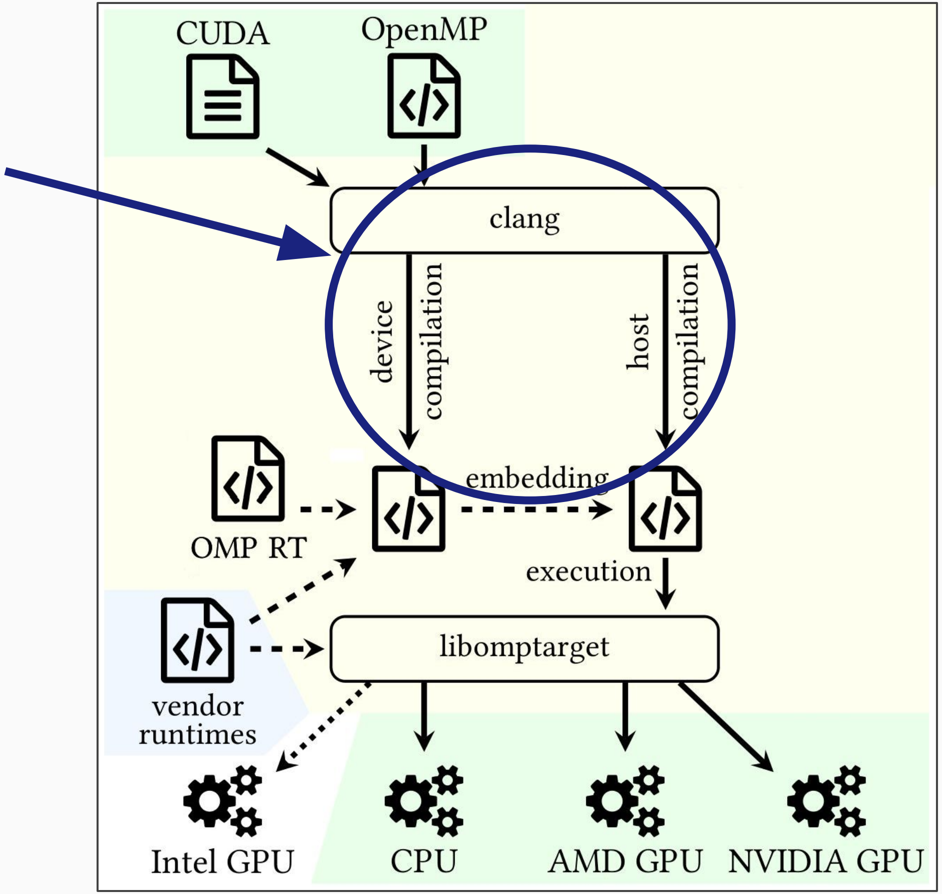


Overview – CUDA via OpenMP Offload Compilation



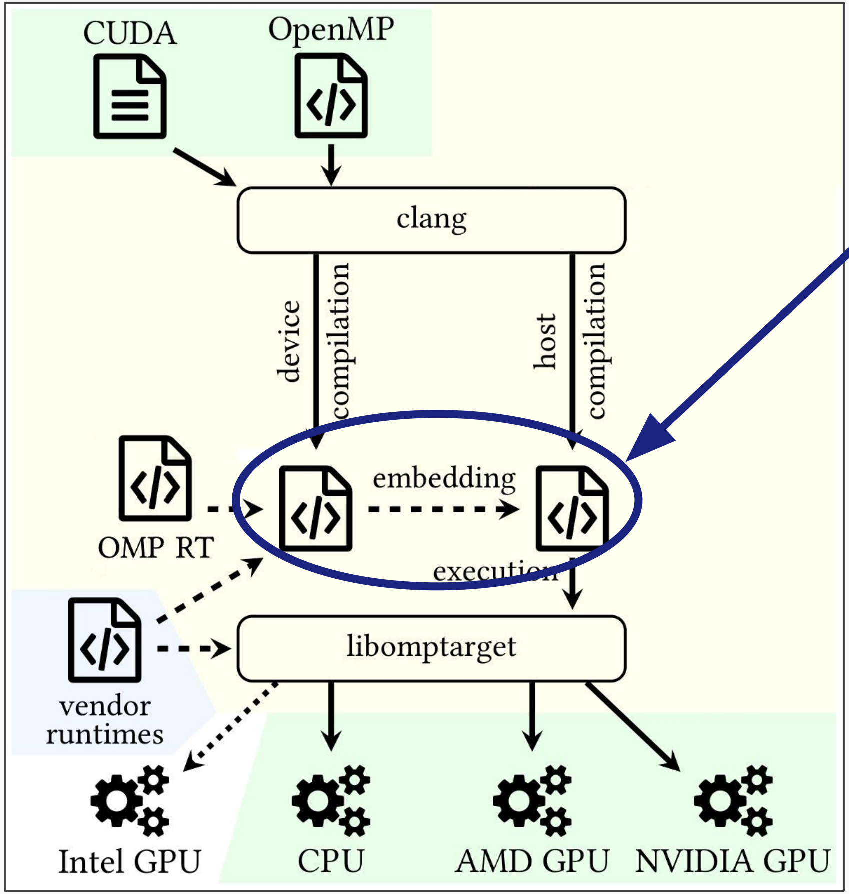
Overview – CUDA via OpenMP Offload Compilation

New Offload Driver
- Language agnostic
- “Classic” design
- Static library support
- LTO-capable



Overview – CUDA via OpenMP Offload Compilation

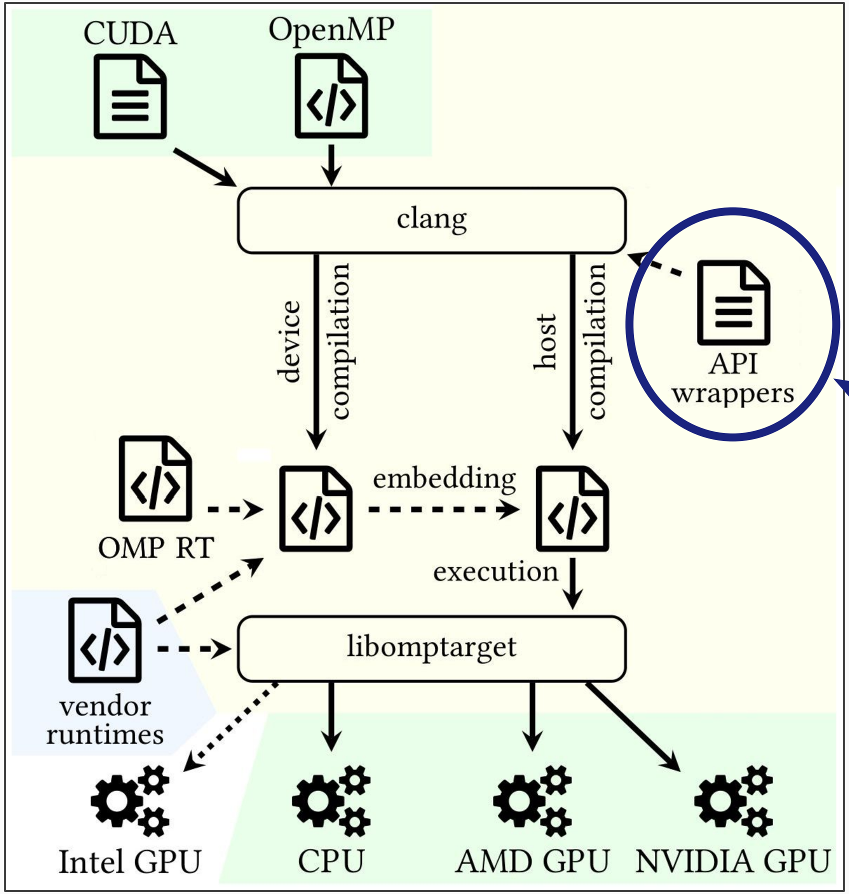
New Offload Driver
- Language agnostic
- “Classic” design
- Static library support
- LTO-capable



Novel Embedding
- Language agnostic
- Metadata enriched
- Multi-device support
- ELF-tooling available

Overview – CUDA via OpenMP Offload Compilation

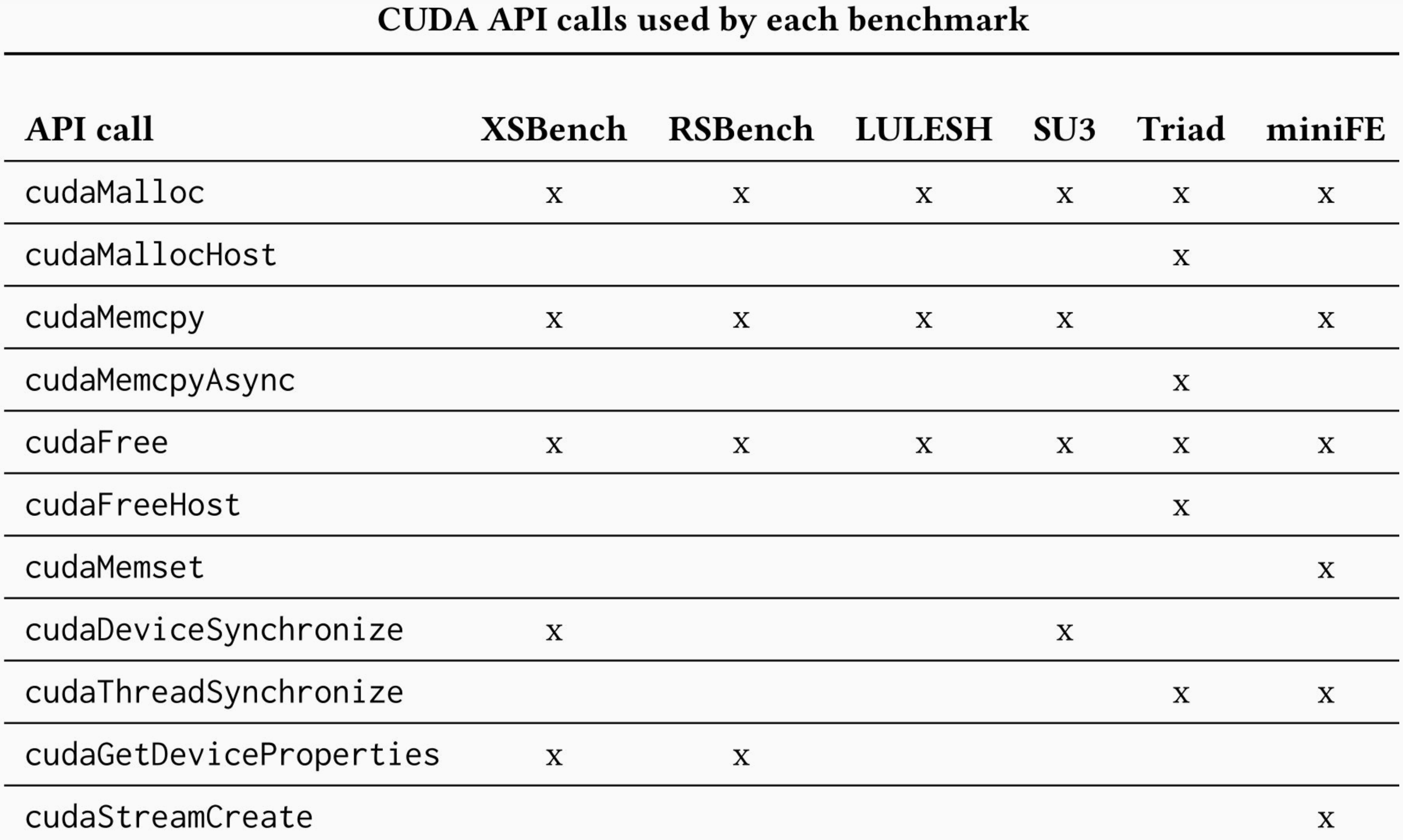
New Offload Driver
- Language agnostic
- “Classic” design
- Static library support
- LTO-capable



Novel Embedding
- Language agnostic
- Metadata enriched
- Multi-device support
- ELF-tooling available

CUDA API Wrappers
- Map to OpenMP RT
- Use existing & new APIs
- Incl. compiler used APIs

Wrapped (User) CUDA APIs

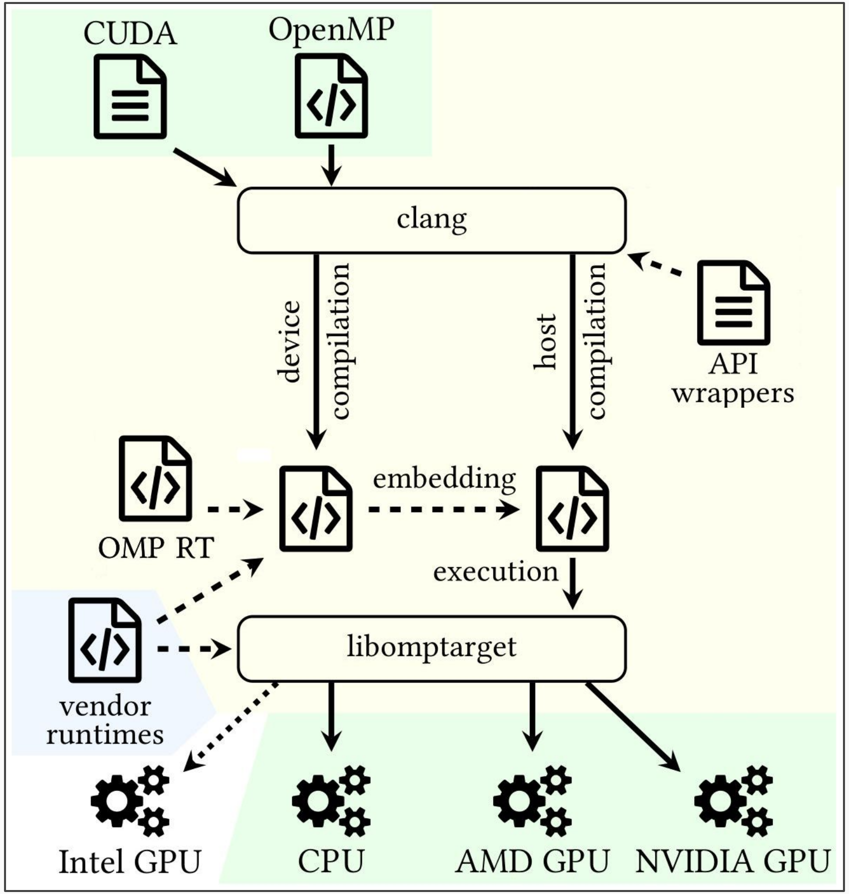


CUDA API calls used by each benchmark

API call	XSBench	RSBench	LULESH	SU3	Triad	miniFE
`cudaMalloc`	x	x	x	x	x	x
`cudaMallocHost`					x	
`cudaMemcpy`	x	x	x	x		x
`cudaMemcpyAsync`					x	
`cudaFree`	x	x	x	x	x	x
`cudaFreeHost`					x	
`cudaMemset`						x
`cudaDeviceSynchronize`	x			x		
`cudaThreadSynchronize`					x	x
`cudaGetDeviceProperties`	x	x				
`cudaStreamCreate`						x

Overview – CUDA via OpenMP Offload Compilation

New Offload Driver
- Language agnostic
- “Classic” design
- Static library support
- LTO-capable



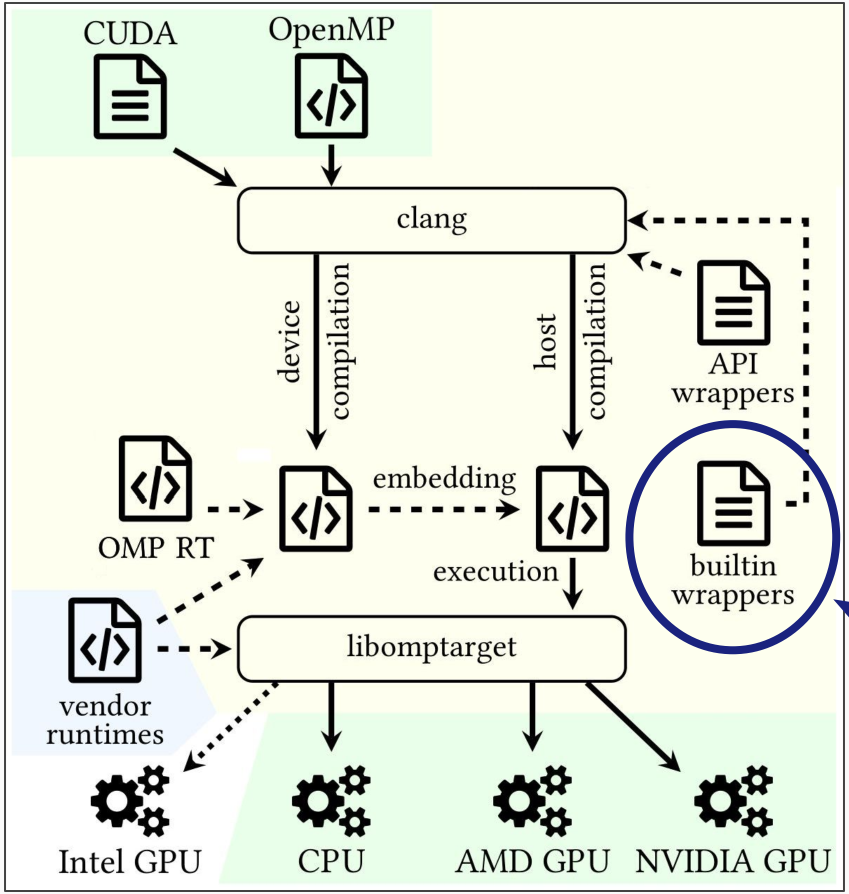
Novel Embedding
- Language agnostic
- Metadata enriched
- Multi-device support
- ELF-tooling available

CUDA API Wrappers
- Map to OpenMP RT
- Use existing & new APIs
- Incl. compiler used APIs

Overview – CUDA via OpenMP Offload Compilation

New Offload Driver

- Language agnostic
- “Classic” design
- Static library support
- LTO-capable



Novel Embedding

- Language agnostic
- Metadata enriched
- Multi-device support
- ELF-tooling available

CUDA API Wrappers

- Map to OpenMP RT
- Use existing & new APIs
- Incl. compiler used APIs

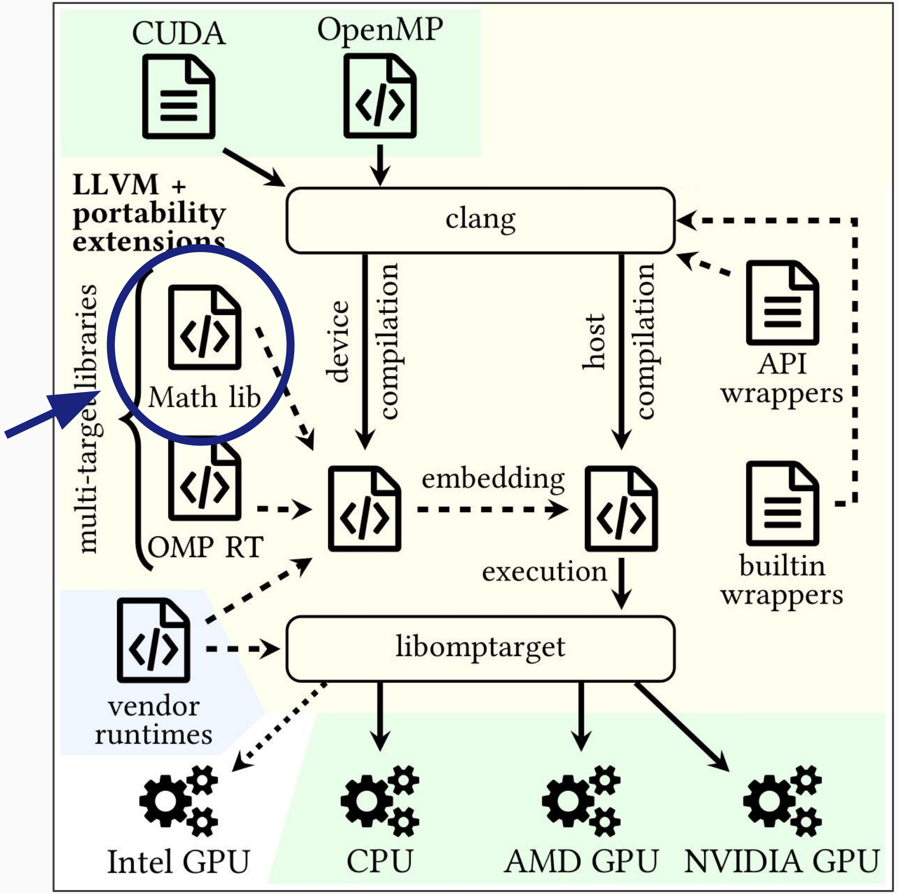
CUDA Builtin Wrappers

- Map to OpenMP RT
- Use existing & new APIs
- Incl. compiler used APIs

Overview – CUDA via OpenMP Offload Compilation

New Offload Driver
- Language agnostic
- “Classic” design
- Static library support
- LTO-capable

Target libm.a
- Map to target math impl.
- Multi-(sub-)target support
- Enable math optimization



Novel Embedding
- Language agnostic
- Metadata enriched
- Multi-device support
- ELF-tooling available

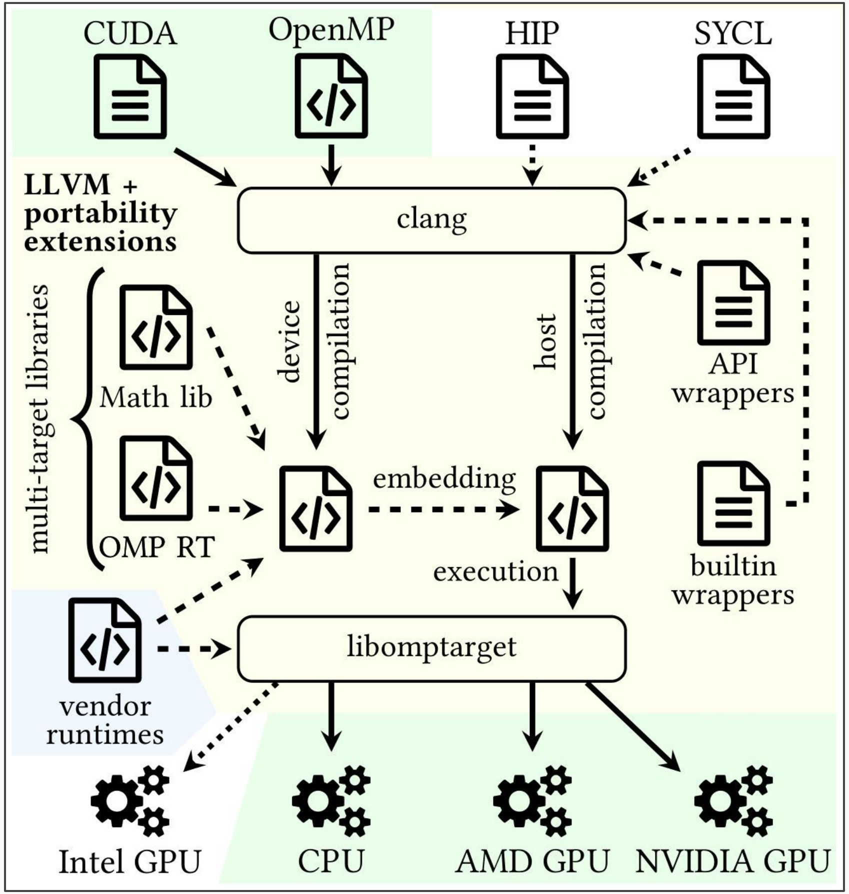
CUDA API Wrappers
- Map to OpenMP RT
- Use existing & new APIs
- Incl. compiler used APIs

CUDA Builtin Wrappers
- Map to OpenMP RT
- Use existing & new APIs
- Incl. compiler used APIs

Overview – CUDA via OpenMP Offload Compilation

New Offload Driver
- Language agnostic
- “Classic” design
- Static library support
- LTO-capable

Target libm.a
- Map to target math impl.
- Multi-(sub-)target support
- Enable math optimization



Novel Embedding
- Language agnostic
- Metadata enriched
- Multi-device support
- ELF-tooling available

CUDA API Wrappers
- Map to OpenMP RT
- Use existing & new APIs
- Incl. compiler used APIs

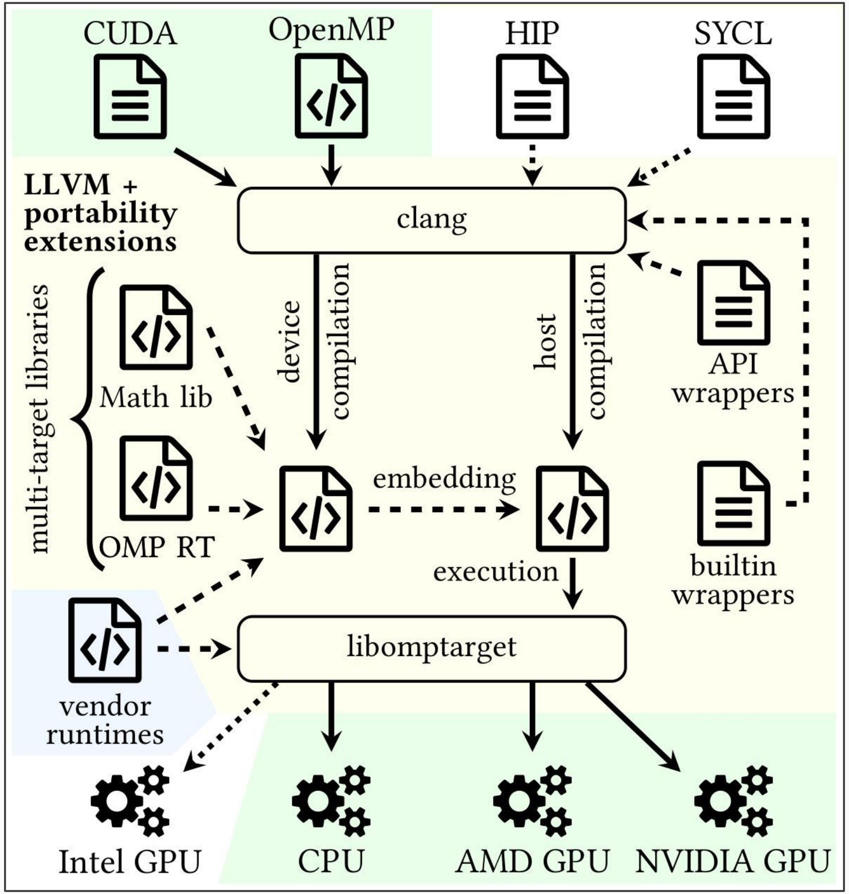
CUDA Builtin Wrappers
- Map to OpenMP RT
- Use existing & new APIs
- Incl. compiler used APIs

Overview – CUDA via OpenMP Offload Compilation

New Offload Driver
- Language agnostic
- “Classic” design
- Static library support
- LTO-capable

Target libm.a
- Map to target math impl.
- Multi-(sub-)target support
- Enable math optimization

Results:
- Portable CUDA
- Interoperable
  CUDA (+ HIP) + OpenMP
- Access to OpenMP features



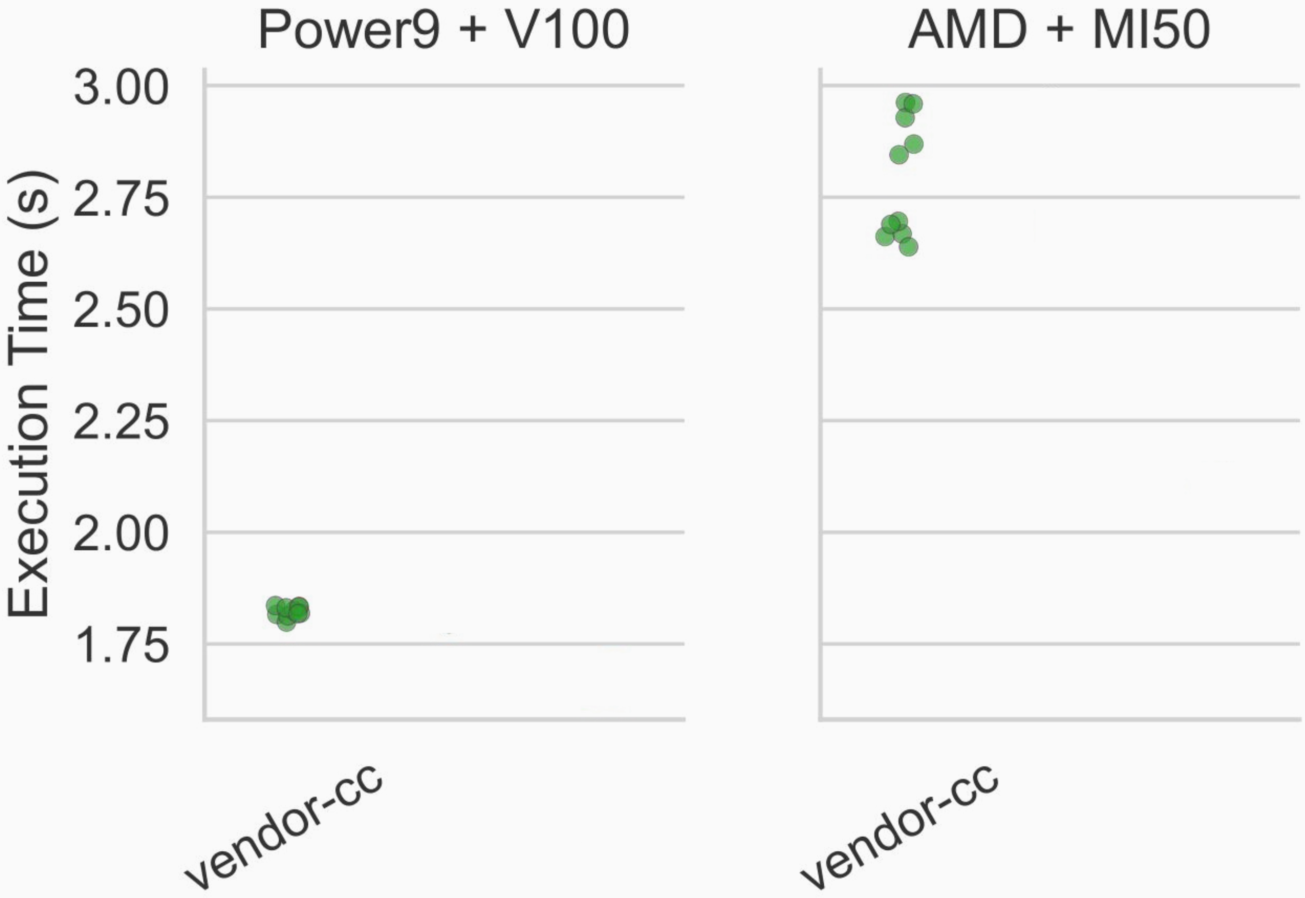
Novel Embedding
- Language agnostic
- Metadata enriched
- Multi-device support
- ELF-tooling available

CUDA API Wrappers
- Map to OpenMP RT
- Use existing & new APIs
- Incl. compiler used APIs

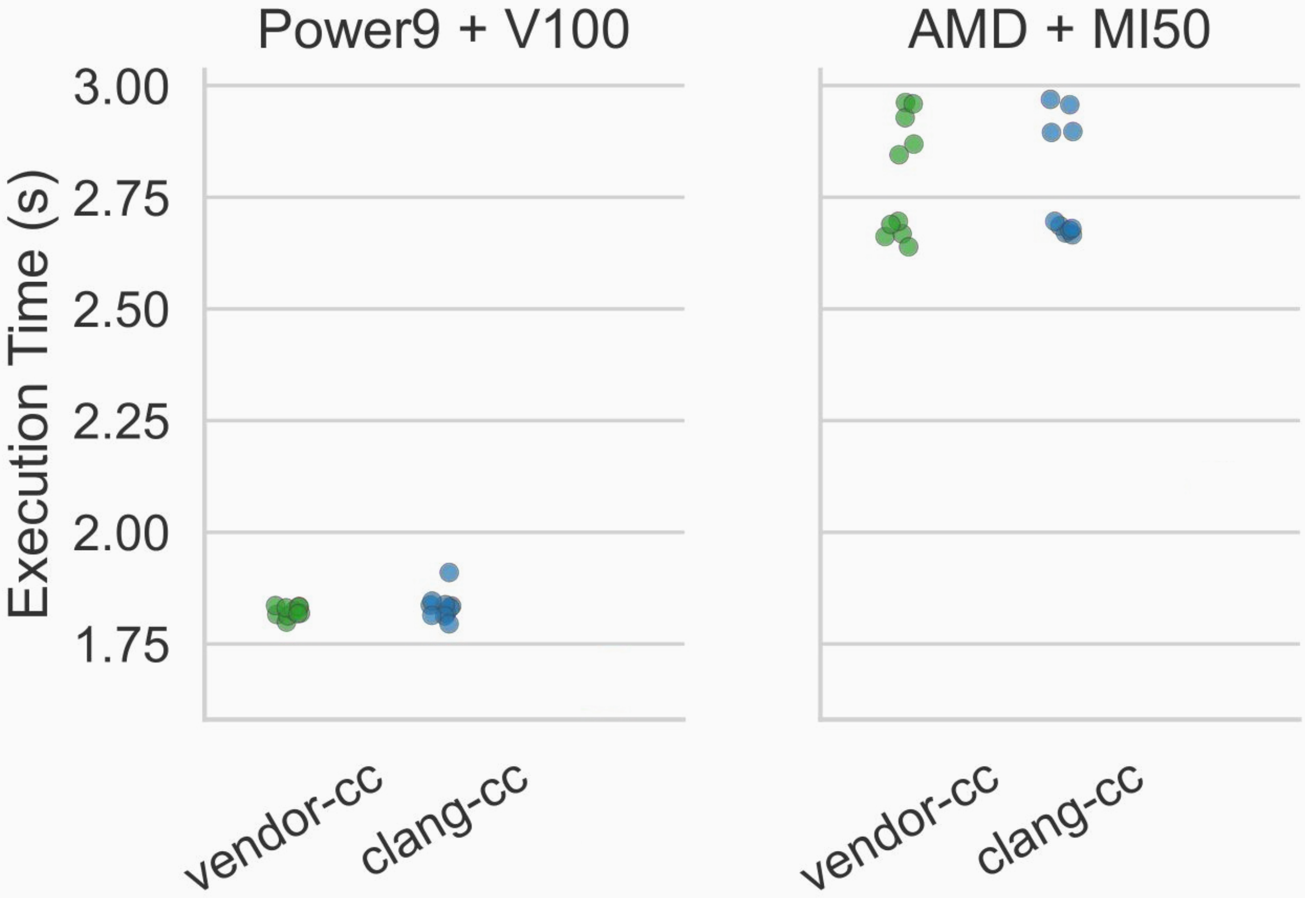
CUDA Builtin Wrappers
- Map to OpenMP RT
- Use existing & new APIs
- Incl. compiler used APIs

Evaluation

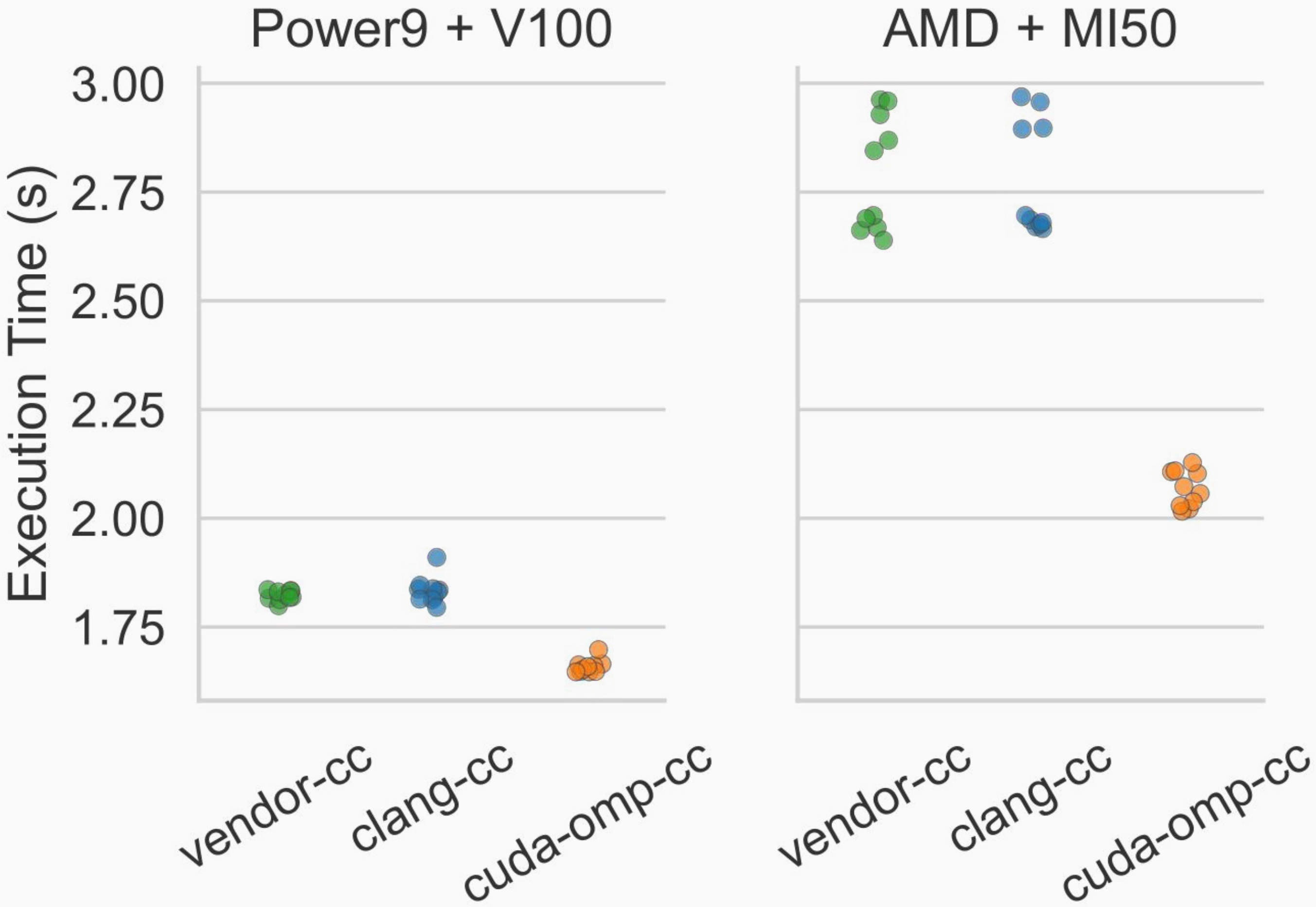
Performance Evaluation – XSBench



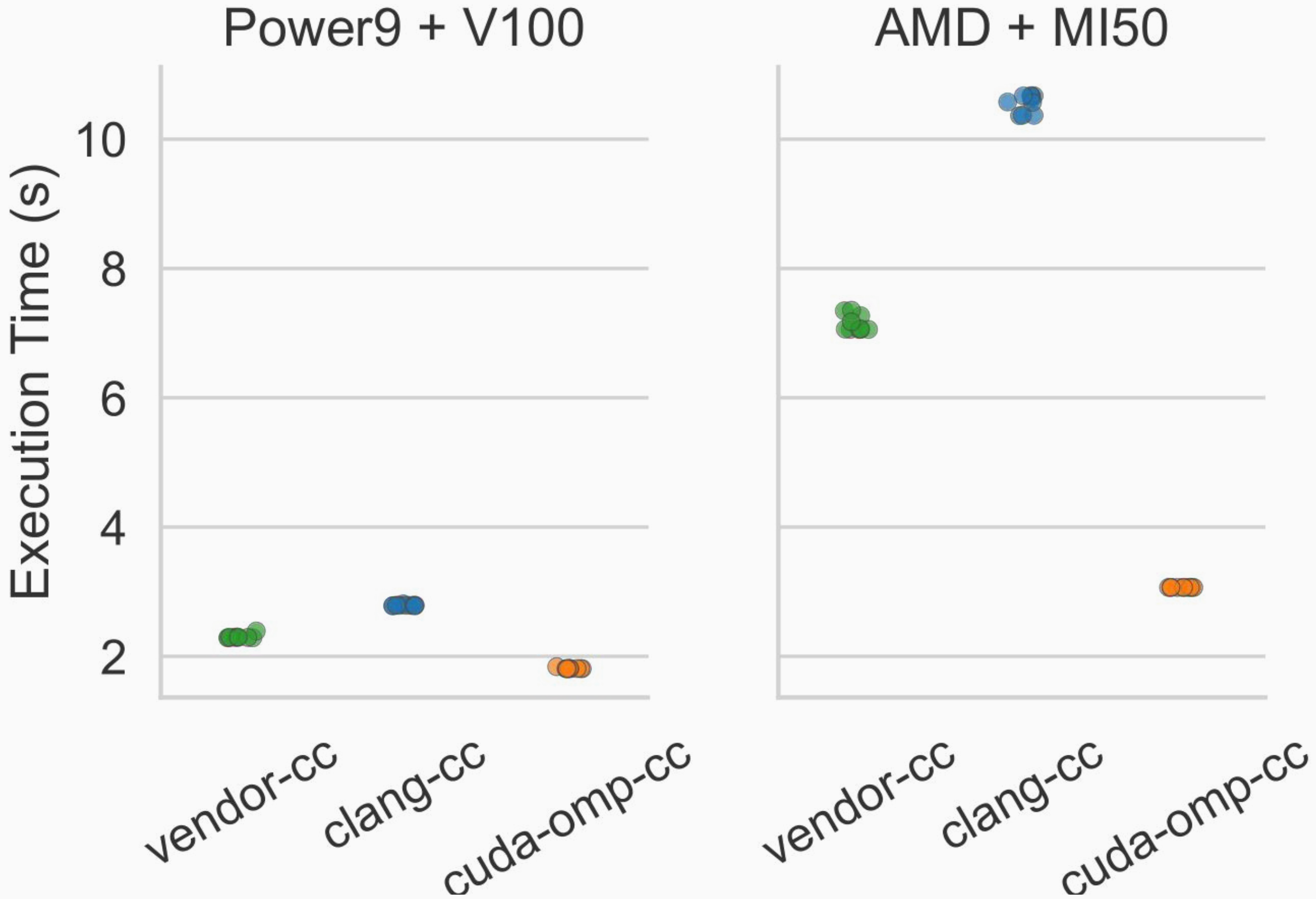
Performance Evaluation – XSBench



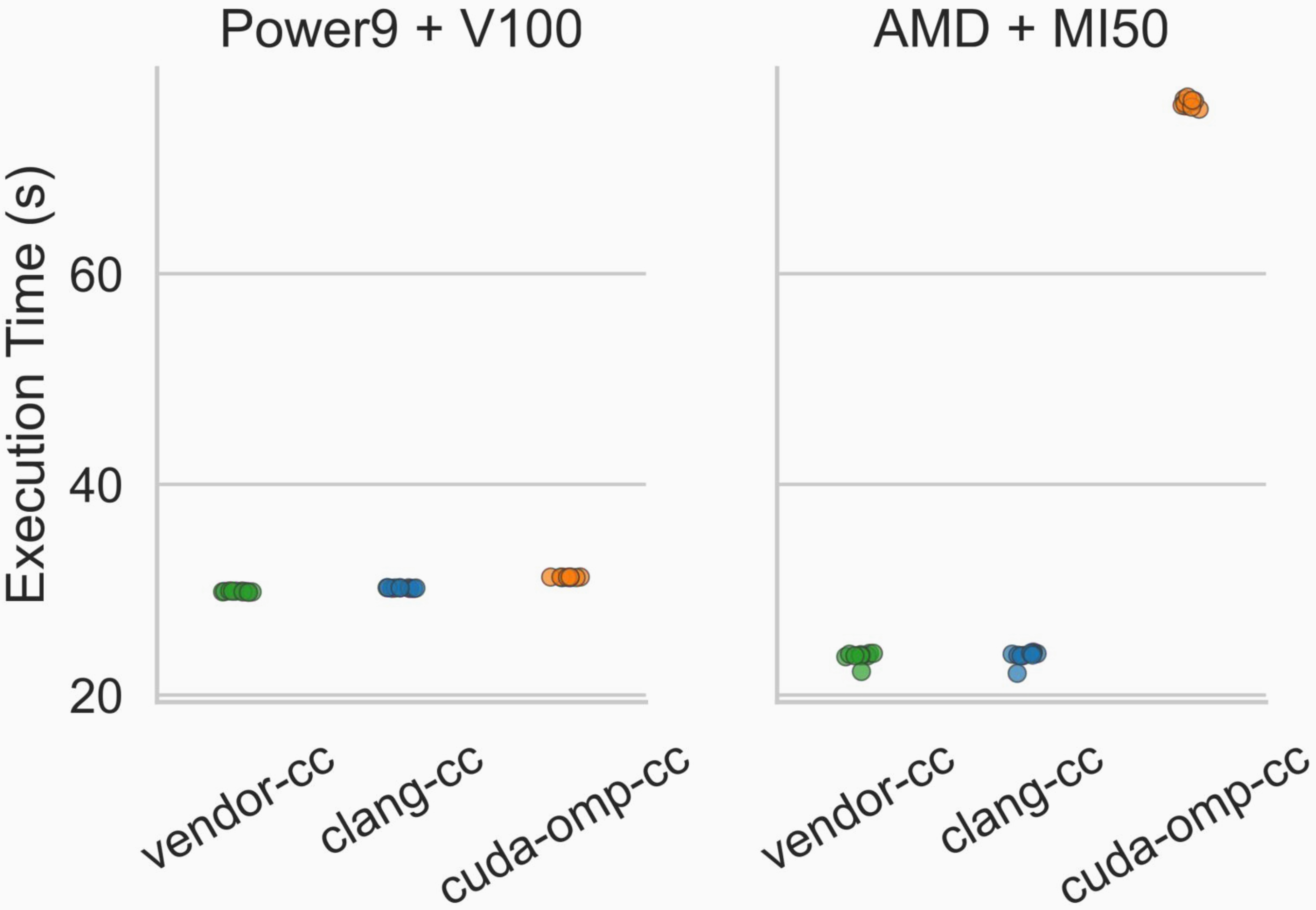
Performance Evaluation – XSBench



Performance Evaluation – XSBench



Performance Evaluation – RSBench



Performance Evaluation – Triad (Stream)