Installation cheat sheet for Kokkos :: page 1

Requirements

Compiler

Compiler	Minimum version	Notes
ARM Clang	20.1	
Clang	10.0.0	For CUDA
Clang	8.0.0	For CPU
GCC	8.2.0	
Intel LLVM	2023.0.0	For SYCL
Intel LLVM	2021.1.1	For CPU
MSVC	19.29	
NVCC	11.0	
NVHPC	22.3	
ROCM	5.2.0	

Build system

Build system	Minimum version	Notes
CMake	3.25.2	For Intel LLVM full support
CMake	3.21.1	For NVHPC support
CMake	3.18	For better Fortran linking
CMake	3.16	

How to integrate Kokkos

Note the difference in the version number between `x.y.z` and `x.y.zz`.

As an external dependency

Configure, build and install Kokkos

```
git clone -b x.y.zz https://github.com/kokkos/kokkos.git
cd kokkos
cmake -B build \
    -DCMAKE_CXX_COMPILER=<your C++ compiler> \
    -DCMAKE_INSTALL_PREFIX=path/to/kokkos/install \
    <Kokkos compile options>
cmake --build build
cmake --install build
```

Setup, and configure your code

```
find_package(Kokkos x.y.z REQUIRED)
target_link_libraries(
    my-app
    Kokkos::kokkos
)
```

```
cd path/to/your/code
cmake -B build \
    -DCMAKE_CXX_COMPILER=<your C++ compiler> \
    -DKokkos_ROOT=path/to/kokkos/install
```

As an internal dependency

Setup with a Git submodule

```
git submodule add -b x.y.zz https://github.com/kokkos/kokkos.git
↪  tpls/kokkos
```

```
add_subdirectory(path/to/kokkos)
target_link_libraries(
    my-app
    Kokkos::kokkos
)
```

Setup with FetchContent

```
include(FetchContent)
FetchContent_Declare(
    kokkos
    URL https://github.com/kokkos/kokkos/releases/download/x.y.zz
↪   /kokkos-x.y.zz.tar.gz
    URL_HASH SHA256=<hash for x.y.z archive>
)
FetchContent_MakeAvailable(kokkos)
target_link_libraries(
    my-app
    Kokkos::kokkos
)
```

Configure your code

```
cmake -B build \
    -DCMAKE_CXX_COMPILER=<your C++ compiler> \
    <Kokkos compile options>
```

You may combine the external/internal dependency approaches.

Kokkos compile options

Host backends

Option	Backend
`-DKokkos_ENABLE_SERIAL=ON`	Serial
`-DKokkos_ENABLE_OPENMP=ON`	OpenMP
`-DKokkos_ENABLE_THREADS=ON`	Threads

The serial backend is enabled by default if no other host backend is enabled.

Device backends

Option	Backend	Device
`-DKokkos_ENABLE_CUDA=ON`	CUDA	NVIDIA
`-DKokkos_ENABLE_HIP=ON`	HIP	AMD
`-DKokkos_ENABLE_SYCL=ON`	SYCL	Intel

You can only select the serial backend, plus another host backend and one device backend at a time.

See architecture-specific options.

Specific options

Option	Description
`-DKokkos_ENABLE_DEBUG=ON`	Activate extra debug features, may increase compile times
`-DKokkos_ENABLE_DEBUG_BOUNDS_CHECK=ON`	Use bounds checking, will increase runtime
`-DKokkos_ENABLE_EXAMPLES=ON`	Build examples
`-DKokkos_ENABLE_TUNING=ON`	Create bindings for tuning tools

Architecture-specific options

Host architectures

Host options are used for controlling optimization and are optional.

Option	Architecture
-DKokkos_ARCH_NATIVE=ON	Local host

Device architectures

Device options are mandatory. They can be deduced from the device if present at CMake configuration time.

AMD GPU architectures (HIP)

Option	Arch.	Associated cards
-DKokkos_ARCH_AMD_GFX942_APU=ON	GFX942 APU	MI300A
-DKokkos_ARCH_AMD_GFX942=ON	GFX942	MI300X
-DKokkos_ARCH_AMD_GFX90A=ON	GFX90A	MI210, MI250, MI250X
-DKokkos_ARCH_AMD_GFX908=ON	GFX908	MI100
-DKokkos_ARCH_AMD_GFX906=ON	GFX906	MI50, MI60
-DKokkos_ARCH_AMD_GFX1103=ON	GFX1103	Ryzen 8000G, Radeon 740M, 760M, 780M, 880M, 980M
-DKokkos_ARCH_AMD_GFX1100=ON	GFX1100	7900xt
-DKokkos_ARCH_AMD_GFX1030=ON	GFX1030	V620, W6800

Intel GPU architectures (SYCL)

Option	Architecture
-DKokkos_ARCH_INTEL_GEN=ON	Generic JIT
-DKokkos_ARCH_INTEL_XEHP=ON	Xe-HP
-DKokkos_ARCH_INTEL_PVC=ON	GPU Max (Ponte Vecchio)
-DKokkos_ARCH_INTEL_DG1=ON	Iris XeMAX
-DKokkos_ARCH_INTEL_GEN12=ON	Gen12
-DKokkos_ARCH_INTEL_GEN11=ON	Gen11

NVIDIA GPU architectures (CUDA)

Option	Arch.	CC	Associated cards
-DKokkos_ARCH_HOPPER90=ON	Hopper	9.0	H200, H100
-DKokkos_ARCH_ADA89=ON	Ada	8.9	GeForce RTX 40 series, RTX 6000/5000 series, L4, L40
-DKokkos_ARCH_AMPERE86=ON	Ampere	8.6	GeForce RTX 30 series, RTX A series, A40, A10, A16, A2
-DKokkos_ARCH_AMPERE80=ON	Ampere	8.0	A100, A30
-DKokkos_ARCH_TURING75=ON	Turing	7.5	T4
-DKokkos_ARCH_VOLTA72=ON	Volta	7.2	
-DKokkos_ARCH_VOLTA70=ON	Volta	7.0	V100
-DKokkos_ARCH_PASCAL61=ON	Pascal	6.1	P6, P40, P4
-DKokkos_ARCH_PASCAL60=ON	Pascal	6.0	P100
-DKokkos_ARCH_MAXWELL53=ON	Maxwell	5.3	
-DKokkos_ARCH_MAXWELL52=ON	Maxwell	5.2	M6, M60, M4, M40
-DKokkos_ARCH_MAXWELL50=ON	Maxwell	5.0	M10

Examples for the most common architectures

Current CPU with OpenMP

```
cmake \
    -B build \
    -DCMAKE_BUILD_TYPE=Release \
    -DKokkos_ENABLE_OPENMP=ON \
    -DKokkos_ARCH_NATIVE=ON
```

AMD MI300A APU with HIP

```
export HSA_XNACK=1
cmake \
    -B build \
    -DCMAKE_CXX_COMPILER=hipcc \
    -DCMAKE_BUILD_TYPE=Release \
    -DKokkos_ENABLE_HIP=ON \
    -DKokkos_ARCH_AMD_GFX942_APU=ON
```

The environment variable is required to access host allocations from the device.

AMD MI250 GPU with HIP

```
cmake \
    -B build \
    -DCMAKE_CXX_COMPILER=hipcc \
    -DCMAKE_BUILD_TYPE=Release \
    -DKokkos_ENABLE_HIP=ON \
    -DKokkos_ARCH_AMD_GFX90A=ON
```

Intel GPU Max 1550 (Ponte Vecchio) with SYCL

```
cmake \
    -B build \
    -DCMAKE_CXX_COMPILER=icpx \
    -DCMAKE_BUILD_TYPE=Release \
    -DKokkos_ENABLE_SYCL=ON \
    -DKokkos_ARCH_INTEL_PVC=ON \
    -DCMAKE_CXX_FLAGS="-fp-model=precise"
```

The last option is required for math operators precision.

NVIDIA H100 GPU with CUDA

```
cmake \
    -B build \
    -DCMAKE_BUILD_TYPE=Release \
    -DKokkos_ENABLE_CUDA=ON \
    -DKokkos_ARCH_HOPPER90=ON
```

NVIDIA A100 GPU with CUDA

```
cmake \
    -B build \
    -DCMAKE_BUILD_TYPE=Release \
    -DKokkos_ENABLE_CUDA=ON \
    -DKokkos_ARCH_AMPERE80=ON
```