# Faiss GPU CUDA 12.8 Build Guide for Ubuntu 24.04.3 LTS

This guide shows how to build Faiss GPU with CUDA 12.8 from source on Ubuntu 24.04.3 LTS, completely avoiding conda dependencies.

## Prerequisites Verification

First, verify your system has a compatible NVIDIA GPU:

```bash
# Check GPU compatibility
nvidia-smi
lspci | grep -i nvidia

# Verify CUDA 12.8 installation
nvcc --version
nvidia-smi  # Should show CUDA Version: 12.8
```

## RECOMMENDED APPROACH: Faiss GPU without cuVS

This approach provides excellent GPU performance with significantly simpler build process.

### Step 1: Install System Dependencies

```bash
```

```
# Update package lists
sudo apt update && sudo apt upgrade -y

# Install build essentials
sudo apt install -y \
  build-essential \
  cmake \
  git \
  ninja-build \
  pkg-config \
  curl \
  wget

# Install BLAS library (OpenBLAS)
sudo apt install -y \
  libopenblas-dev \
  liblapack-dev

# Install OpenMP
sudo apt install -y libomp-dev

# Install Python development headers
sudo apt install -y \
  python3-dev \
  python3-pip \
  python3-numpy \
  python3-setuptools \
  python3-wheel

# Install SWIG for Python bindings
sudo apt install -y swig

# Verify versions
cmake --version  # Should be >= 3.24
python3 --version
swig -version
```

## Step 2: Install CUDA 12.8 Toolkit (if not already installed)

```
bash
```

```
# Add NVIDIA repository
wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2404/x86_64/cuda-keyring_1.1-1_all.deb
sudo dpkg -i cuda-keyring_1.1-1_all.deb

# Update package lists
sudo apt update

# Install CUDA toolkit
sudo apt install -y cuda-toolkit-12-8

# Set environment variables
echo 'export CUDA_HOME=/usr/local/cuda-12.8' >> ~/.bashrc
echo 'export PATH=$CUDA_HOME/bin:$PATH' >> ~/.bashrc
echo 'export LD_LIBRARY_PATH=$CUDA_HOME/lib64:$LD_LIBRARY_PATH' >> ~/.bashrc
echo 'export CUDACXX=$CUDA_HOME/bin/nvcc' >> ~/.bashrc

# Reload environment
source ~/.bashrc

# Verify CUDA installation
nvcc --version
```

## Step 3: Clone and Configure Faiss

bash

```
# Clone Faiss repository
cd ~
git clone https://github.com/facebookresearch/faiss.git
cd faiss

# Create build directory and configure
cmake -B build \
  -DCMAKE_BUILD_TYPE=Release \
  -DBUILD_SHARED_LIBS=ON \
  -DFAISS_ENABLE_GPU=ON \
  -DFAISS_ENABLE_CUVS=OFF \
  -DFAISS_ENABLE_PYTHON=ON \
  -DFAISS_ENABLE_C_API=OFF \
  -DFAISS_OPT_LEVEL=generic \
  -DBUILD_TESTING=OFF \
  -DCMAKE_CUDA_COMPILER=/usr/local/cuda-12.8/bin/nvcc \
  -DCMAKE_CUDA_ARCHITECTURES="60;61;70;75;80;86;89;90" \
  -DPython_EXECUTABLE=$(which python3) \
  -DCMAKE_INSTALL_PREFIX=/usr/local \
  .

# Check configuration output for any errors
```

## Step 4: Build Faiss

```
bash

# Build the library
make -C build -j$(nproc) faiss

# Build Python bindings
make -C build -j$(nproc) swigfaiss

# Verify build
ls -la build/faiss/libfaiss.so
ls -la build/faiss/python/
```

## Step 5: Install Faiss

```
bash
```

```
# Install C++ library system-wide (optional)
sudo make -C build install

# Install Python bindings
cd build/faiss/python
python3 setup.py install --user

# Alternative: Install in virtual environment
# python3 -m venv ~/faiss-env
# source ~/faiss-env/bin/activate
# python3 setup.py install
```

## Step 6: Test Installation

bash

```
# Test basic import
python3 -c "import faiss; print(f'Faiss version: {faiss.__version__}')"

# Test GPU functionality
python3 -c "
import faiss
import numpy as np

# Check GPU availability
print(f'Number of GPUs: {faiss.get_num_gpus()}')

# Test basic GPU index
d = 64
nb = 1000
xb = np.random.random((nb, d)).astype('float32')

# CPU index
index_cpu = faiss.IndexFlatL2(d)
index_cpu.add(xb)

# GPU index
if faiss.get_num_gpus() > 0:
    res = faiss.StandardGpuResources()
    index_gpu = faiss.index_cpu_to_gpu(res, 0, index_cpu)
    print('GPU index created successfully!')

    # Test search
    xq = np.random.random((5, d)).astype('float32')
    D, I = index_gpu.search(xq, 5)
    print(f'Search completed. Shape: {D.shape}')
else:
    print('No GPU available')
"
```

## ADVANCED APPROACH: Faiss GPU with cuVS

**Warning**: This approach is significantly more complex and requires building multiple dependencies from source.

### Additional Prerequisites for cuVS

```
bash
```

```
# Install additional dependencies for cuVS
sudo apt install -y \
  libnvidia-ml-dev \
  libcublas-dev-12-8 \
  libcurand-dev-12-8 \
  libcusparse-dev-12-8 \
  libcusolver-dev-12-8 \
  libcufft-dev-12-8

# Install required system libraries for RAPIDS ecosystem
sudo apt install -y \
  libssl-dev \
  libffi-dev \
  libbz2-dev \
  libsqlite3-dev \
  libncurses5-dev \
  libgdbm-dev \
  liblzma-dev \
  uuid-dev \
  libreadline-dev
```

## Build cuVS from Source

bash

```
# Clone cuVS
cd ~
git clone --recursive https://github.com/rapidsai/cuvs.git
cd cuvs

# Configure cuVS build
cmake -S cpp -B cpp/build \
  -DCMAKE_BUILD_TYPE=Release \
  -DBUILD_SHARED_LIBS=ON \
  -DCMAKE_INSTALL_PREFIX=/usr/local \
  -DCUDA_STATIC_RUNTIME=OFF \
  -DCMAKE_CUDA_COMPILER=/usr/local/cuda-12.8/bin/nvcc \
  -DCMAKE_CUDA_ARCHITECTURES="60;61;70;75;80;86;89;90"

# Build cuVS (this may take 30+ minutes)
cmake --build cpp/build --parallel $(nproc)

# Install cuVS
sudo cmake --install cpp/build
```

## Build Faiss with cuVS Support

```
bash
```

```
# Navigate back to Faiss directory
cd ~/faiss

# Clean previous build
rm -rf build

# Configure Faiss with cuVS
cmake -B build \
  -DCMAKE_BUILD_TYPE=Release \
  -DBUILD_SHARED_LIBS=ON \
  -DFAISS_ENABLE_GPU=ON \
  -DFAISS_ENABLE_CUVS=ON \
  -DFAISS_ENABLE_PYTHON=ON \
  -DFAISS_ENABLE_C_API=OFF \
  -DFAISS_OPT_LEVEL=generic \
  -DBUILD_TESTING=OFF \
  -DCMAKE_CUDA_COMPILER=/usr/local/cuda-12.8/bin/nvcc \
  -DCMAKE_CUDA_ARCHITECTURES="60;61;70;75;80;86;89;90" \
  -DPython_EXECUTABLE=$(which python3) \
  -DCMAKE_INSTALL_PREFIX=/usr/local \
  .

# Build and install as before
make -C build -j$(nproc) faiss
make -C build -j$(nproc) swigfaiss
cd build/faiss/python
python3 setup.py install --user
```

# Troubleshooting

## Common Issues and Solutions

### CMake cannot find CUDA compiler:

bash

```
export CUDACXX=/usr/local/cuda-12.8/bin/nvcc
export CUDA_HOME=/usr/local/cuda-12.8
```

### Python bindings fail to build:

bash

```
# Make sure you have correct Python dev headers
sudo apt install python3.12-dev  # Adjust version as needed
```

**CUDA architecture errors:**

```
bash

# Check your GPU's compute capability
nvidia-smi --query-gpu=compute_cap --format=csv
# Adjust CMAKE_CUDA_ARCHITECTURES accordingly
```

**Memory issues during compilation:**

```
bash

# Reduce parallel jobs if you run out of memory
make -C build -j4  # Instead of -j$(nproc)
```

**Library linking issues:**

```
bash

# Update library cache
sudo ldconfig

# Check library paths
echo $LD_LIBRARY_PATH
ldd build/faiss/libfaiss.so
```

## Performance Verification

After successful installation, run this performance test:

```
bash
```

```
python3 -c "
import faiss
import numpy as np
import time

# Test parameters
d = 768         # dimension
nb = 100000     # database size
nq = 1000       # number of queries

print(f'Testing with {nb} vectors of dimension {d}')

# Generate random data
np.random.seed(1234)
xb = np.random.random((nb, d)).astype('float32')
xq = np.random.random((nq, d)).astype('float32')

# CPU baseline
start = time.time()
index_cpu = faiss.IndexFlatL2(d)
index_cpu.add(xb)
D_cpu, I_cpu = index_cpu.search(xq, 10)
cpu_time = time.time() - start
print(f'CPU search time: {cpu_time:.3f} seconds')

# GPU test
if faiss.get_num_gpus() > 0:
    start = time.time()
    res = faiss.StandardGpuResources()
    index_gpu = faiss.index_cpu_to_gpu(res, 0, index_cpu)
    D_gpu, I_gpu = index_gpu.search(xq, 10)
    gpu_time = time.time() - start
    print(f'GPU search time: {gpu_time:.3f} seconds')
    print(f'GPU speedup: {cpu_time/gpu_time:.2f}x')

    # Verify results are similar
    assert np.allclose(D_cpu, D_gpu, rtol=1e-5)
    print('GPU results match CPU results!')
else:
    print('No GPU available for testing')
"
```

## Notes

- **CUDA Version Compatibility**: CUDA 12.8 should work fine with this build, even though official conda packages are limited to 12.1/12.4
- **Performance**: GPU Faiss without cuVS still provides 5-10x speedup over CPU
- **cuVS Benefits**: cuVS mainly provides optimized implementations for IVF-Flat, IVF-PQ, and CAGRA indices
- **Memory Requirements**: Building from source requires ~8GB RAM, reduce parallelism if needed
- **Build Time**: Complete build takes 10-30 minutes depending on your system

## Recommendation

For most users, I recommend the **first approach (without cuVS)** as it:

- Provides excellent GPU performance
- Avoids complex dependency management
- Builds reliably on Ubuntu 24.04.3
- Fully supports CUDA 12.8
- Completely avoids conda dependencies

The cuVS approach is only worthwhile if you specifically need the CAGRA index or optimized IVF implementations.