

GSOHC: Global Synchronization Optimization for Heterogeneous Computing (Artifact)

Authors:

Soumik Kumar Basu, Jyothi Vedurada

Department of Computer Science and Engineering, IIT Hyderabad, India



Overview

GSOHC is a compiler optimization framework designed to improve CPU-GPU collaborative computing by relocating global synchronization barriers and synchronous memory transfers for better overlap and performance. It achieves up to **1.9× speedup** across a variety of GPUs.

This repository provides the complete **artifact** to reproduce the results from our ECOOP 2025 paper.



Artifact Contents

- **Docker Image:** `gsohc_artifact_image.tar`
 - **LLVM Project:**
Source code under `llvm-project/llvm/lib/Transforms/gsohc/`
 - **Benchmarks:**
 - `GSOHC_Benchmarks/HeCBench/`
 - `GSOHC_Benchmarks/PolyBench/`
 - **Scripts:**
 - `unzip.sh`
 - `compile_all.py`
 - `test_all.py`
 - `plots.py`
 - `checker.sh`
 - `min_eval.py`
 - `build_llvm.sh`
 - `stat_run.py`
-



Hardware Requirements

- **NVIDIA GPU** (CUDA support required)
 - **CPU:** x86 architecture
 - **Disk Space:** ≥ 60 GiB
 - **RAM:** ≥ 4 GiB (16 GiB recommended)
-



Software Requirements

These must be installed on your **host machine** before running the docker image:

- **NVIDIA CUDA Toolkit** (Follow [official guide](#))
- **Docker** ([Install Docker](#))
- **NVIDIA Container Toolkit** ([nvidia-docker installation guide](#))

After running Docker, the container comes pre-installed with:

- CUDA 11.0+
- CMake \geq 3.20.0
- Python \geq 3.8
- zlib \geq 1.2.3.4
- GNU Make (3.79 or 4.3)
- PyYAML \geq 5.1
- LLVM-14.x
- Ubuntu 22.04



Quick Start Guide

1. Load the Docker Image

```
sudo docker load -i gsohc_artifact_image.tar
```

2. Run the Docker Container

```
sudo docker run --gpus all --name gsohc -it gsohc_artifact
```

3. Unzip and Build LLVM Project

```
python3 unzip.py  
./build_llvm.sh
```

You will be asked for the number of threads to parallelize the LLVM build. Provide **half the number of your CPU cores** for optimal performance.

4. Verify Installation

```
chmod +x checker.sh  
./checker.sh
```

You should see confirmation that all dependencies are correctly installed.



Selective Benchmark Run (Minimal Evaluation)

Compile and run a random **n** benchmarks:

```
python3 run_benchmarks.py --num [n]
```

Compile and run a specific benchmark:

```
python3 run_benchmarks.py --one [benchmark_folder_name]
```

Example: To compile and run only **adam-cuda** benchmark program use the following command:

```
python3 run_benchmarks.py --one adam-cuda
```

Running Benchmarks

Compile all Benchmarks

```
python3 compile_all.py
```

✓ Output: **[SUCCESS]** All benchmarks compiled successfully!

Run all Benchmarks

```
python3 test_all.py
```

✓ Output: **[SUCCESS]** All benchmarks completed successfully!
Results will be stored in **benchmark_results.csv** inside each benchmark's folder.



Plotting Results

Consolidate results and generate a speedup bar chart:

```
python3 plots.py
```

This will generate **benchmark_speedups.png**.



Static Analysis Report

Generate the static analysis statistics:

```
python3 stat_run.py
```

Results:

- `analysis_report.csv`
- `analysis_report.png`

Exporting Results

To export plots/tables from Docker to your host:

```
sudo docker cp gsohc:/workspace/benchmark_speedups.png [path/to/host]
sudo docker cp gsohc:/workspace/analysis_report.png [path/to/host]
```



License

The artifact is distributed under a **MIT License**.



Notes

- **Artifact Size:** ~10 GiB
- **Tested Platforms:**
 - Ubuntu 22.04 LTS + RTX A4000
 - Debian 11 + NVIDIA P100
 - Ubuntu 22.04 LTS + NVIDIA A100

For any queries or issues, feel free to contact:

- Soumik Kumar Basu: cs21resch11004@iith.ac.in
-