3D Acoustic Wave Simulation with Multi-GPU NCCL Communication and MPI Group Parallelism

Copyright statements

This project uses `OpenMPI` for distributed-memory parallelism via MPI. OpenMPI is an open-source implementation of the Message Passing Interface, licensed under the New BSD License.

This project uses the `NVIDIA CUDA Toolkit` for GPU programming. CUDA is provided by NVIDIA under the NVIDIA Software License Agreement.

This project uses `NCCL` for fast multi-GPU communication. NCCL is distributed by NVIDIA under a permissive BSD-like license.

This project uses `libxml2` for XML configuration parsing. libxml2 is developed by the GNOME project and released under the MIT License.

This project uses `segy.h` from the open-source Seismic Unix (SU) package developed by the Colorado School of Mines, which is released under a BSD-style license.

Project Structure

This project implements a 3D acoustic wave finite-difference simulation using CUDA and MPI. It leverages **NCCL** for communication between multiple GPUs in a domain decomposition scheme, and employs **MPI** for shot-domain parallelism.

- `main.cpp`: Main framework
- `kernel.cu`: CUDA kernel implementations
- `fd.h`: Custom header file
- `segy.h`: Segy header file
- `Makefile`: Compilation instructions
- `parameter.xml`: XML parameter card
- `run.sh`: Sample launch script

Build Instructions

Before compiling, make sure you have the following:

- **CUDA Toolkit**
- **MPI**
- **NCCL**
- **libxml2**

Compile

```
make
```

This produces the executable `GPU3D`.

To clean up:

```
make clean
```

Parameters (XML)

All simulation parameters are read from an XML file named `parameter.xml`.

```xml
<parameter name="fm">10</parameter>          <!-- Source central frequency -->
<parameter name="dy">10</parameter>          <!-- Grid spacing in y -->
<parameter name="dx">10</parameter>          <!-- Grid spacing in x -->
<parameter name="dz">10</parameter>          <!-- Grid spacing in z -->
<parameter name="pml">50</parameter>         <!-- PML thickness -->
<parameter name="ny">676</parameter>         <!-- Grid points in y -->
<parameter name="nx">676</parameter>         <!-- Grid points in x -->
<parameter name="nz">210</parameter>         <!-- Grid points in z -->
<parameter name="disx_shot_grid">10</parameter>    <!-- Shot grid interval in x -->
<parameter name="disy_shot_grid">10</parameter>    <!-- Shot grid interval in y -->
<parameter name="sourcex_min_grid">100</parameter> <!-- Minimum grid of shot in x -->
<parameter name="sourcex_max_grid">100</parameter> <!-- Maximum grid of shot in x -->
<parameter name="sourcey_min_grid">100</parameter> <!-- Minimum grid of shot in y -->
<parameter name="sourcey_max_grid">100</parameter> <!-- Maximum grid of shot in y -->
<parameter name="sz">0</parameter>           <!-- Shot depth -->
<parameter name="gz">0</parameter>           <!-- Receiver depth -->
<parameter name="scale_y">4</parameter>      <!-- Receiver grid interval in y -->
<parameter name="scale_x">4</parameter>      <!-- Receiver grid interval in x -->
<parameter name="gpu_num">8</parameter>      <!-- Number of GPUs per node -->
<parameter name="id_of_group">4</parameter>  <!-- Number of processes/GPUs per process group -->
<parameter name="dt">5e-4</parameter>        <!-- Sampling interval -->
<parameter name="t">4.0</parameter>          <!-- Total sampling time -->
<parameter name="fvel">/path/to/vel.bin</parameter>       <!-- Velocity model (binary file) -->
```

```
<parameter name="outfile">/path/to/out.segy</parameter>   <!-- Output SEG-Y filename -->
```

Ensure paths `fvel` and `outfile` are absolute or valid relative paths.

Run Instructions

To run the simulation:

```
bash run.sh
```

Or manually:

```
mpirun -np 9 --hostfile nodefile ./GPU3D parameter.xml
```

The number of processes is equal to the total number of GPU cards actually used plus one (as master process).

Output

The output is a **SEG-Y** file containing seismic records, written to the path defined by the `outfile` parameter.
