Implementation of OpenSource Structural Engineering Application OpenSees on GPU platform

Ms Gouri Kadam¹
Ms Shweta Nayak²

¹ CDAC, Pune University Campus Pune, India. Email: gourik@cdac.in
² Department of Computer Engineering PICT, Pune, India. Email: nayakshweta19@gmail.com
INDEX

- Introduction
- Why GPU
- Methodology
- Case Study
- Results and Discussion
- Future Scope
The software framework OpenSees (Open System for Earthquake Engineering Simulation) provides a platform for structural and geotechnical engineers. OpenSees has more than 160 element types, 220 material types, 15 solution algorithms and 30 solver types.

**Static Problems:**
- Deformation analyses (1D, 2D, or 3D)
- Consolidation problems
- Soil-structure interaction problems
- Shallow foundations (e.g. bearing capacity, deformation)
- Pile foundations (e.g. vertical and lateral capacity)

**Dynamic (earthquake problems):**
- Free-field analysis
- Liquefaction-induced problems
- Soil structure interaction problems
Objective of the Study

- For earthquake simulation of large scale structural and geotechnical system parallel computation is the only option.
- Algebraic libraries developed very rapidly for solving very large number mathematical problems using efficient parallel algorithms. Accelerators such as GPUs are very efficient in handling such problems.
- Available GPU enabled version of OpenSees provided by Xinzheng Lu, Linlin Xie (Tsinghua University, China) uses CulaS4 and CulaS5, which use the Cula library. Also currently supported on window’s platform.
- GPU computing is emerging as an alternative to CPUs for throughput oriented applications because of their Number of cores embedded on the same chip.
- User friendly changes in input scripts of the application and from developers prospective great expertise in CUDA programming is not a limitation.

Source: https://goo.gl/images/Sr5sKd
Why CUSP

Libraries Considered

<table>
<thead>
<tr>
<th>Library</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CULA</td>
<td>Licensed</td>
</tr>
<tr>
<td>CUSP</td>
<td>OpenSource</td>
</tr>
<tr>
<td>cuSPARSE</td>
<td>Closed Source</td>
</tr>
<tr>
<td>PETSc</td>
<td>OpenSource</td>
</tr>
</tbody>
</table>

- **CuSP** provides a flexible, high-level interface for manipulating sparse matrices and solving sparse linear systems.
- It is the best suited library as it will give the interface for C and C++ classes.
- Most important, it is open Source and easy in new algorithm implementation compared to others.

### Hardware Used

One of the server of CAE group at CDAC (cfd-WS2) with following configuration used for testing and simulation.

- CPU : 20 x Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30 GHZ
- GPU : 2 x NVIDIA GK104GL - Quadro K4200
- RAM : 32 GB
- OS : CentOS 6.7

### Software Used

- OpenSees 2.4 and above
- gcc: 4.5.0
- make: 2.8.11
- tcl8.5
- tcl8.5-dev
- g++ :4.5.0
- gfortran
- CUDA 7.0 and higher
- CUSP library
Methodology

Basic Processing Flow

Example of Processing Flowchart

Source: https://commons.wikimedia.org/wiki/File:CUDA_processing_flow_(En).PNG
Case Study and Results

Modifications in Input Script

<table>
<thead>
<tr>
<th>Input Script (CPU)</th>
<th>GPU enabled Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>integrator LoadControl 1.0 algorithm Linear numberer RCM constraints Plain system SparseGeneral -piv analysis Static analyze 1</td>
<td>integrator LoadControl 1.0 algorithm Linear numberer RCM constraints Plain system CuSP analysis Static analyze 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
<th>Run Time (min)</th>
<th>Speed Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>SmallMP.tcl</td>
<td>6.978</td>
<td>6.441</td>
</tr>
<tr>
<td>elasticFrame.tcl</td>
<td>1.006</td>
<td>0.9</td>
</tr>
<tr>
<td>ExampleMP.tcl</td>
<td>0.854</td>
<td>0.4</td>
</tr>
<tr>
<td>PlaneShearWall.tcl</td>
<td>4.91</td>
<td>2.97</td>
</tr>
</tbody>
</table>

GPU and CPU Performance comparison
Availability in Public Domain

SUBVERSION REPOSITORIES  OPENSEES

(root)/trunk/SRC/system_of_eqn/linearSOE/sparseGEN/CuSPSolver.cpp - Rev 6505

// Written: fmk
// modified by sourik@cdac.in (CDAC, Pune)
// nayakshwetaa@gmail.com (ME student PICT,Pune)
//
#include "CuSPSolver.h"
#include <iostream>
using std::cout;

int cuSPSolver(double *C, double *B, double *host_x, int size, int nnz, int *rowStartA, int *colA, int maxIteration, double relTolerance, int preCond, int solver);

CuSPSolver::CuSPSolver(void):SparseGenRowLinSolver(SOLVER_TAGS_CUSP)
{
    single=0;
}
Future Scope

- **Multi-GPU Implementation**: All above mentioned work is done for single GPU implementation. By adding the necessary support of thrust library to CuSP or by rewriting dynamic library using multi-GPU supporting CUDA enabled libraries, implementation of other accelerators mean for heterogeneous/ hybrid architecture more speed up can be achieved. Only concern will be the increase in complexity from users perspective need to be considered.

- **Adding support for OpenCL**: The provided implementation has the limitation that it can only be used with NVIDIA GPUs. Implementing support for platform independent GPU computing APIs, such as OpenCL, would therefore make the implementation accessible to more users.
Thank you!