High Performance Optimizations and Dynamic Load Balancing for Computational Aerodynamics Solvers

Blessy Boby TVE15CS020
Gmon K TVE15CS024
Shilpa S TVE15CS055
Robin R LTVE15CS068

Under the guidance of Prof. Dr. Ajeesh Ramanujan
College of Engineering, Trivandrum

Project under the guidance of
Harichand M. V, Scientist Engineer
Vikram Sarabhai Space Centre

TABLE OF CONTENTS

Introduction

Introduction

Load Balancing

1. It is a challenging and interesting problem that appear in all High Performance Computing(HPC) environments.
2. Most of the applications running in HPC environments use MPI based parallel computing.
3. This approach needs load assignment, before starting the computations, which is difficult in many problems where computational load changes dynamically.

In VSSC Aeronautics entity,

1 Parallel computing is used to solve Computational Fluid Dynamics problems.
2 Memory bandwidth utilisation of CFD problems are very low, making them computationally inefficient.

Motivation

The motivation for doing this project was primarily an interest in undertaking a project at VSSC,ISRO. The opportunity to learn about a new area of computing not covered in lectures was appealing.

Problem Statement

Problem Statement

Develop a distributed MPI framework for solving computational problems efficiently. Challenge here is to design data layouts which can effectively utilise memory bandwidth. Multiple re-ordering schemes, along with data duplication can be attempted to improve the memory performance.

While solving computational problems in a distributed MPI environment, it is important to ensure load balancing. But the static load balancing algorithms can fail if the computational loads vary over time. A distributed work queue may be attempted to solve this problem

Objectives

Objectives

Two problems can be attempted as part of this Project

- Improve the dynamic load balancing capability in an MPI environment for solving CFD problems
- Improve data locality of CFD problems

Scope of Project

Scope of project

Develop a frame work for solving Computational Fluid Dynamics Problems keeping in mind

1. Explore graph re-ordering, data layout modification for increased compute efficiency.
2. Dynamic load balancing.
3. Hybrid CPU-GPU computing.
4. Compare the results with default MPI method, Charm++ and Petsc.

Requirements

Requirements

1. METIS software for partitioning graph into sub-graph.
2. OPENMPI for implementing MPI i.e, passing messages between sub-graphs.
3. Operating system used are both ubuntu / Linux.
4. MPI programs are written in c++ language.
5. Languages used for implementing parallel programs are charm++ and petsc.

Charm++

1. Parallel programming with migratable objects.
2. Programming model based on graphs where edges represent data dependancy.

Petsc

1. Its a library for large scale scientific computing with traditional programming model.
2. Its a much lower level library compared to charm++.

FORMAT OF INPUT (MESH)

- The SU2 mesh format carries an extension of .su2, and the files are in a readable ASCII format.
- As an unstructured code, SU2 requires information about both the node locations as well as their connectivity.
- The connectivity description provides information about the types of elements (triangle, rectangle, tetrahedron, hexahedral, etc.) that make up the volumes in the mesh and also which nodes make up each of those elements.

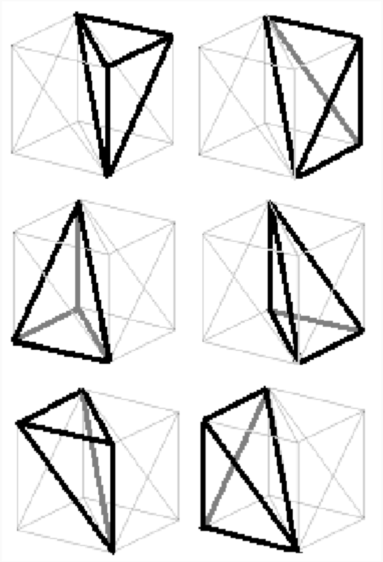
- The first line of the .su2 mesh declares the dimensionality of the problem (NDIME).
- The next part of the file describes the interior element connectivity, which begins with the NELEM(Number of Elements in the mesh) keyword
- Remaining part contains the information about Element in the format

```
Connectivity node1 node2 etc...
```

- Connectivity and number of nodes depends on the type of element

Element	Connectivity	Number of nodes
Triangle	5	3
Quadrilateral	9	4
Tetrahedral	10	4
Hexahedral	12	8
Prism	13	6
Pyramid	14	5

EXAMPLE FOR A 3D MESH



```
NDIME=3
NELEM=6
10 0 1 2 3
10 0 3 2 4
10 0 3 5 6
10 0 3 4 6
10 0 7 1 3
10 0 7 5 3
```

WHAT IS METIS?

The METIS is a software package for partitioning large irregular graphs and meshes.Its source code can be downloaded directly from `http://www.cs.umn.edu/metis`. METIS provides a set of stand-alone command-line programs for computing partitionings as well as an application programming interface (API) that can be used to invoke its various algorithms from C/C++ or Fortran programs.

APPLICATION OF METIS: PARTITIONING OF GRAPH

- METIS can partition an unstructured graph into a user-specified number k of parts using either the multilevel recursive bisection or the multilevel k-way partitioning paradigms.
- METIS' stand-alone program for partitioning a graph is gpmetis and the functionality that it provides is achieved by the METIS PartGraphRecursive and METIS PartGraphKway API routines.

APPLICATION OF METIS: PARTITIONING OF MESH

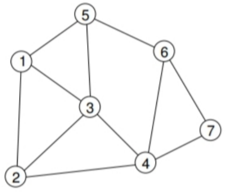
- METIS provides the **mpmetis** program for partitioning meshes arising in finite element or finite volume methods.
- This program take as input the element-node array of the mesh and compute a k-way partitioning for both its elements and its nodes.
- This program first converts the mesh into either a dual graph (i.e., each element becomes a graph vertex) or a nodal graph and then uses the graph partitioning API routines to partition this graph.
- The functionality provided by **mpmetis** is achieved by the **METIS PartMeshNodal** and **METIS PartMeshDual** API routines.

APPLICATION OF METIS: CONVERTING A MESH TO GRAPH

- METIS provides the m2gmetis program for converting a mesh into the graph format used by METIS. This program can generate either the nodal or dual graph of the mesh.
- The corresponding API routines are METIS MeshToNodal and METIS MeshToDual.

INPUT FILE FORMAT

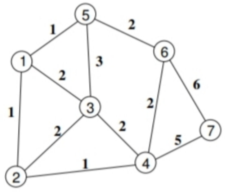
- The primary input of the mesh partitioning programs in METIS is the mesh to be partitioned.
- This mesh is stored in a file in the form of the element node array.
- A mesh with n elements is stored in a plain text file that contains n + 1 lines.
- The first line contains 2 parameters that is the number of elements n in the mesh and number of weights associated with each element(optional).



Graph File:

```
7 11
5 3 2
1 3 4
5 4 2 1
2 3 6 7
1 3 6
5 4 7
6 4
```

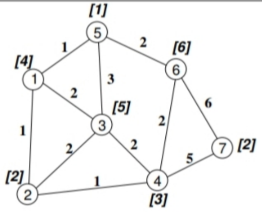
(a) Unweighted Graph



Graph File:

```
7 11 001
5 1 3 2 2 1
1 1 3 2 4 1
5 3 4 2 2 2 1 2
2 1 3 2 6 2 7 5
1 1 3 3 6 2
5 2 4 2 7 6
6 6 4 5
```

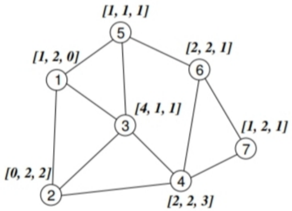
(b) Weighted Graph
Weights on edges



Graph File:

```
7 11 011
4 5 1 3 2 2 1
2 1 1 3 2 4 1
5 5 3 4 2 2 2 1 2
3 2 1 3 2 6 2 7 5
1 1 1 3 3 6 2
6 5 2 4 2 7 6
2 6 6 4 5
```

(c) Weighted Graph
Weights both on vertices and edges



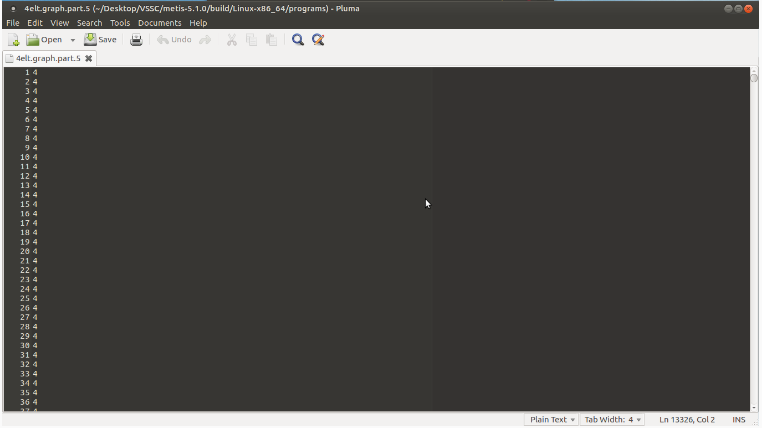
Graph File:

```
7 11 010 3
1 2 0 5 3 2
0 2 2 1 3 4
4 1 1 5 4 2 1
2 2 3 2 3 6 7
1 1 1 1 3 6
2 2 1 5 4 7
1 2 1 6 4
```

(d) Multi–Constraint Graph

OUTPUT FILE FORMAT

- The output of METIS is a partition file.
- Partition file
    - The partition file of a graph with n vertices consists of n lines with a single number per line.
    - The i'th line of the file contains the partition number that the i'th vertex belongs to.
    - Partition numbers start from 0 up to the number of partitions minus one.



4elt.graph.part.5 (~/Desktop/VSSC/metis-5.1.0/build/Linux-x86_64/programs) - Pluma

```
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
4
```

Plain Text   Tab Width: 4   Ln 13326, Col 2   INS

WHAT IS MPI?

- The **message passing interface (MPI)** is a standardized means of exchanging messages between multiple computers running a parallel program across distributed memory.
- It can be implemented using Open MPI which can be download from it’s official page:

  `\http://www.open-mpi.org/`

WHERE TO USE MPI?

- You need a portable parallel program
- You are writing a parallel library
- You have irregular or dynamic data relationships that do not fit a data parallel model

MPI SEND AND RECEIVE

Sending and receiving are the two foundational concepts of MPI. Almost every single function in MPI can be implemented with basic send and receive calls.

```
MPI_Send(
    void* data,
    int count,
    MPI_Datatype datatype,
    int destination,
    int tag,
    MPI_Comm communicator)
```

```
MPI_Recv(
void* data,
int count,
MPI_Datatype datatype,
int source,
int tag,
MPI_Comm communicator,
MPI_Status* status)
```

- The first argument is the data buffer.
- The second and third arguments describe the count and type of elements that reside in the buffer.
- MPI_Send sends the exact count of elements, and MPI_Recv will receive at most the count of elements.
- The fourth and fifth arguments specify the rank of the sending or receiving process and the tag of the message.
- The sixth argument specifies the communicator and the last argument (for MPI_Recv only) provides information about the received message.

WHY WE USE MPI?

- After the mesh file is partitioned, each node is allotted into different partitions. These partitions are distributed into different systems.
- Some times the neighbouring node of a particular node in the mesh might not belong to the same partition. So we create ghost nodes which simulate the neighbouring nodes in the partition where the particular node belongs.

WHY WE USE MPI?

- Whenever we are in need of the data from the neighbouring node, we find out the original neighbour node’s partition and retrieve the data from the node and then copy it to the ghost node in our partition.
- Here there is a need to communicate between the different partitions so that the values of ghost nodes can be updated as per the need. This is where MPI plays its role.

Proposed System

ALGORITHM

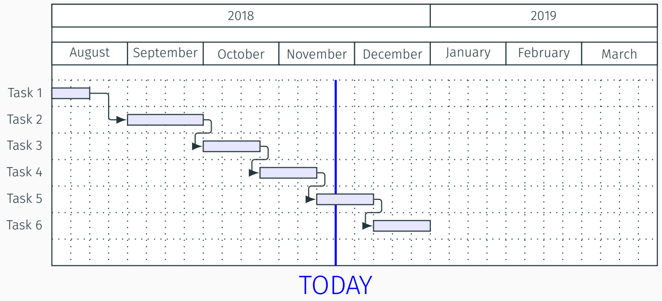
1. Read Mesh file in .su2 format
2. Check the dimension of the input
3. Store the number of vertices
4. For each element in the mesh file do the following
    - 4.1 Check the type of element
    - 4.2 Make an element to node mapping
    - 4.3 Make a node to element mapping
5. From the node to element mapping do the following.
    - 5.1 For each face for an element do the following
        - 5.1.1 Make an intersection among all the node to element mappings in the current face.
        - 5.1.2 If the cardinality of the resultant set is >= 3 then create adjacency list
6. Write the adjacency list to a file.

ALTERNATIVE ALGORITHM

1. Read Mesh file in .su2 format
2. Check the dimension of the input
3. Store the number of vertices
4. For each element in the mesh file do the following
    4.1 Check the type of element
    4.2 Make an element to node mapping
5. From the element to node mapping do the following.
    5.1 Find each face of an element
    5.2 For each face for an element do the following
        5.2.1 If this face is equal to another face of an element then make the face as edge and add this pair of elements to adjacency list.
        5.2.2 Store the face information.
6. Write the face information to file.
7. Write the adjacency list to a file.

Project Plan

Project Plan



Tasks

Task 1 : Topic selection and Research.
Task 2 : Study the given input file in mesh format.

- Analyse the structure of .su2 files.
- Discussion about developing an algorithm for conversion to adjacency list.
- Extending the idea from 2D mesh to 3D.

Task 3 : Implementation of the algorithm.

- Node to Element mapping algorithm.
- Face to Element mapping algorithm.

Task 4 :Discussion about partitioning the graph.

- Study about existing ordinary methods for partitioning.
- METIS: an effective way for partitioning graph.

Task 5 : Distribution of subgraphs among multiple processors.

- Discussion about communication methods.
- MPI: an effective interface for the communication.

Task 6 : Effective communication among processors.

- Concept of ghost vertices.
- Reducing parallel asynchronous mpi communication.

Queries from the last presentation

- Would it be feasible to use DFS/ BFS for partitioning instead of using Metis?
  DFS/BFS technique will not reduce the communication volume(represented as edgecuts in metis). The proper alternative to metis is space filling curves.

- **What is the real computation that is happening in the vertex?** Its more of a fluid mechanics computation. Mass, energy and momemtum fluxes will be computed on graph edges. Conserved fluid properties will be updated on each vertex looking at the fluxes on its edges. A close computation can be the one which appear in the paper. `https://community.dur.ac.uk/suzanne.fielding/teaching/BLT/sec1.pdf` and `https://www.comsol.com/multiphysics/fluid-flow-conservation-of-momentum-mass-and-energy`

- **What are the alternatives to MPI**
  MPI is not only the standard for communication. There are many other communication libraries like ARMCI, Portals etc. which are used for special cases. But here we work on MPI as it was the library which was suggested to us.
  https://insidehpc.com/2006/06/what-are-alternatives-to-mpi-for-communication-in-parallel-computing/

- **The final result of the project**
  Final result will be a frame work for solving CFD problems in a distributed MPI manner with dynamic load balancing taken care of. We will also attempt techniques to improve the serial code performance and data layouts for effective memory bandwidth utilization.

THANKYOU!!