A Graph Model for Large Scale Simulation on High Performance Computing Cluster using Task Parallel Approach
Computer Cluster

Computer Cluster: A set of connected computers to execute a task in parallel, thus they can be viewed as a single system.

• For 50 years individual computers were being designed to have higher computational power on a single chip (CPUs)

Freebies are over for software developers as limits to transistor density approach

• Problems are getting bigger and bigger (we need more computational power)
• Software developers are already switching to parallel computing via multiple-core CPU transparently
• We must define models to solve our largest problems in parallel
Graph Modeling Applications

Graph: A system of similar entities (vertices) and their relations (edges).

A long list of systems, problems and processes can be modeled by a Graph, but some of them are too large to be simulated with a single ordinary computer like:

- **Biology**: Human genome, Human brain, disease propagation
- **Network & Communication**: Cellular phone network, network topology
- **Social Networks**: Human societies, virtual societies
- **Physics**: Quantum mechanics, astronomy science
Graph Partitioning

Graphs can be partitioned using similarities, features and connectivity between vertices that depend on the type of the system.

A model is needed, to simulate or execute each partition on one node of a computer cluster.

This example shows a graph with 4 partitions and their boundary relations.
Partitioning

Assume a graph with

11 vertices
Dividing to The Two Partitions

- It can be divided into Two subgraphs
- Each cut edge creates one port
- An entity is needed on each port to emulate the neighbor to that port
Adding Boundary Vertex

- By Adding Boundary vertex on each port each subgraph will be completed.
- Now there are two graphs one with 7 vertices and the other 8 vertices.
Simple Speed-up by partitioning

For a graph with n nodes:

if simply assume a linear relation between operations and nodes.

Operation complexity : $O(n)$

With $p$ partitions and adding $b$ boundary vertices : $O((n+b)/p)$

Usually $b \ll n \Rightarrow O((n+b)/p) \approx O(n/p)$

Doing all partitions in parallel $\Rightarrow$ Speed-up $\approx P$
Real Speed-up by partitioning

- In reality speed-up in a parallel execution depend on:
  - Solving method for states
  - Communication delay between partitions
- Usually because states solving methods has $O(n^2)$ or $O(n^3)$ operations, partitioning should give a $P^2$ or $P^3$ speed-up gain
- In practice because the graph is divided into the partitions each subgraph needs to reconcile on its boundaries with neighbors
- Boundary reconciliation needs communication and extra operations that can reduce speed-up dramatically
Speed-up in Power Network Solution

Equations to solve in each step:

\[ I = Y \times V \quad \text{\textit{LU Decomposition}} \quad Y = L \times U \quad \rightarrow \quad I = L \times U \times V \]

\[ X = U \times V \rightarrow I = L \times X \]

Equations will be solved for \( I \) with backward and forward technique

For a \( n \) buses power grid in each step:

Computational Complexity: \( O\left(\frac{2}{3}n^3\right) + O(n^2) \)

Divided by \( p \) for each partition: \( O\left(\frac{2}{3} \frac{n^3}{p^3}\right) + O\left(\frac{n^2}{p^2}\right) \)

Speed-up: \( O\left(p^3 \times \frac{2n+3}{2n+3p}\right) \) for \( n \gg p \rightarrow Speed-up \approx p^3 \)
Numbers in Practice

O(p^3) speed-up is impossible in practice.

Main reason for lower performance is:

- Delay in communication between partitions (MPI communication)

Also by adding more partitions, we should have more speed-up in theory; however, in practice more partitioning requires more data communications and more operations for boundary states reconciliation.
Simulation Process in a Boundary Vertex

- In each simulation step, every boundary vertex has to communicate with the neighbor’s boundary vertex and try to reconcile its states.
Coordinating Simulation

- Each subgraph has two states whether it reached an agreement with its neighbors or not.
- The whole system goes to the next simulation step when every subgraph reaches an agreement with its neighbors.
- Simulation coordinator collects information from all subgraphs and sends suitable messages to them to go to next simulation round or next reconciliation loop.
By Using OOD/P and MPI as communication tools between processes
We created a model to execute a very large simulation on HPC^2.
The model consists of 5 main objects:

1. Vertex
2. Boundary Vertex
3. Edge
4. Boundary Edge
5. Super Vertex
Use cases for OOD/P (Object Oriented Design/Programming)

- Creating a graph dynamically from graph data
- Handling events
- Solving for states
- Forwarding to next simulation state
- Reporting the states
- Reconciling boundary states
- Facilitating communications between processes
Power Network Simulation

With our software model we simulate a power network.

- Analogy between Power Network and the Graph Simulator

![Diagram showing the analogy between Power Network and Graph Simulator](Diagram.png)
Computer Clusters at MS State

We work with shadow II in MSSTATE which is the most powerful supercomputer at Mississippi State University.

Shadow II has 110 nodes
2 Xeon processors 10 cores each
2 Xeon Phi co-processor 60 core each
512 GB RAM
Power Network Simulation Results

Reconciliation Vs Partitions

Speed-up Vs Partitions

(number of nodes 28674)
Speed-up Results

- Seed-up has been calculated for 4 different power grid sizes.
- There is always a peak in speed-up (gain) after which the MPI delay produces a net reduction in speed-up.
Conclusions

➢ Graph simulator model solves artificial power grid up to 142,000,000 buses on Shadow II. 180 was the maximum speed-up with more than 330 partitions.

➢ Faster reconciliation is the key to getting better performance on this model.

➢ Partitioning has major effect on reconciliation and performance.
Thank You

Questions?