Parallel Formulations Of Dijkstra’s All-pairs Shortest Path Algorithm

Presenter
Parthish Udayanbhai Pandya
Outline

• Introduction
• All Pairs Shortest Path algorithm & Dijkstra
• Dijkstra’s algorithm two variants
  • Source Partitioned
  • Source Parallel
• Analysis
• Results
• Conclusion
All Pairs Shortest Path

• Find the shortest path between all pairs of vertices $v_i, v_j$ such that $i \neq j$.

• An n-vertices graph, output would be an n x n matrix D where $D[i, j]$ represents the distance between two vertices $v_i, v_j$.

How do we implement Dijkstra’s Algorithm for this?
Dijkstra’s Algorithm

• Execute Single source algorithm on each process for each vertex v.

• Sequential Execution time : n times to sequential execution time of single source algorithm

\[ T_s = \Theta (n \times n^2) \]

Two Parallel Formulation :

✓ Source partitioned
✓ Source parallel

• Applicable to graph only with the non-negative weights.
Dijkstra’s Algorithm : Parallel Formulation

- **Source partitioned formulation:**
  - Partition the vertices among different processes.
  - Each process to compute single source shortest path for all vertices assigned to it.

- **Source parallel formulation:**
  - Assign each vertex to a set of processes
  - Then use parallel formulation of single source shortest path algorithm
Source Partitioned Formulation

• Use \( n \) processes.

• Each process \( P_i \) finds the shortest paths from vertex \( v_i \) to all other vertices by executing the sequential Dijkstra’s algorithm locally.

• No inter-process communication \( \Rightarrow \) Adjacency matrix is replicated at each process.

\[
T_P = \Theta(n^2) \\
S = \Theta(n^3)/\Theta(n^2) \\
E = \Theta(1)
\]

• Algorithm can use at most \( n \) processes. \( p=n \)

Iso-efficiency = \( \Theta(n^3) \)
Source Parallel Formulation

• \( p \) processes are divided into \( n \) partitions each with \( p/n \) processes \((p>n)\).
• Each partition solves one single-source shortest path problem.
• Total no of processes that can be efficiently used = \( O(n^2) \)
• **Cost Analysis**:
  
  Total Computation time: \( \Theta (n^2)/(p/n) = \Theta (n^3)/p \)
  
  Total Communication time : \( \Theta (n \log p) \)

\[ T_P = \Theta(n^3)/p + \Theta(n \log p) \]
Source Parallel Formulation (Cont’d)

• Cost optimal: If $p \log p = O(n^2)$ => $p \log n = O(n^2)$
  $$\Rightarrow p = O(n^2/\log n)$$

• Iso efficiency due to communication:
  $\Theta((p \log p)^{1.5})$

• Iso-efficiency due to concurrency:
  $\Theta(p^{1.5})$ (Because $p = O(n^2)$)

• Overall Iso efficiency: $\Theta((p \log p)^{1.5})$
Comparison of two formulations

<table>
<thead>
<tr>
<th>Type of Algorithm</th>
<th>No of Processes</th>
<th>Iso efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Partitioned</td>
<td>No more than $n$ processes</td>
<td>$\Theta(p^3)$</td>
</tr>
<tr>
<td>Source Parallel</td>
<td>Up to $n^2/\log n$ processes</td>
<td>$\Theta((p \log p)^{1.5})$</td>
</tr>
</tbody>
</table>

**Conclusion:** Source Parallel achieves more parallelism than source partitioned. It is more scalable than Source partitioned approach.
Results

Parallel Formulation for Dijkstra's All Source shortest path Algorithm

- 16*16 Parallel
- 16*16 Partitioned
Questions?
Thank You