Lecture 4: Principles of Parallel Algorithm Design (part 1)
Constructing a Parallel Algorithm

• identify portions of work that can be performed concurrently
• map concurrent portions of work onto multiple processes running in parallel
• distribute a program’s input, output, and intermediate data
• manage accesses to shared data: avoid conflicts
• synchronize the processes at stages of the parallel program execution
Task Decomposition and Dependency Graphs

**Decomposition:** divide a computation into smaller parts, which can be executed concurrently

**Task:** programmer-defined units of computation.

**Task-dependency graph:**
- **Node** represents a task.
- **Edge** represents control dependence.
Example 1: Dense Matrix-Vector Multiplication

- Computing $y[i]$ only use $i$th row of $A$ and $b$ – treat computing $y[i]$ as a task.

- Remark:
  - Task size is uniform
  - No dependence between tasks
  - All tasks need $b$
Example 2: Database Query Processing

- Executing the query:
  Model = “civic” AND Year = “2001” AND (Color = “green” OR Color = “white”)

on the following database:

<table>
<thead>
<tr>
<th>ID#</th>
<th>Model</th>
<th>Year</th>
<th>Color</th>
<th>Dealer</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>4523</td>
<td>Civic</td>
<td>2002</td>
<td>Blue</td>
<td>MN</td>
<td>$18,000</td>
</tr>
<tr>
<td>3476</td>
<td>Corolla</td>
<td>1999</td>
<td>White</td>
<td>IL</td>
<td>$15,000</td>
</tr>
<tr>
<td>7623</td>
<td>Camry</td>
<td>2001</td>
<td>Green</td>
<td>NY</td>
<td>$21,000</td>
</tr>
<tr>
<td>9834</td>
<td>Prius</td>
<td>2001</td>
<td>Green</td>
<td>CA</td>
<td>$18,000</td>
</tr>
<tr>
<td>6734</td>
<td>Civic</td>
<td>2001</td>
<td>White</td>
<td>OR</td>
<td>$17,000</td>
</tr>
<tr>
<td>5342</td>
<td>Altima</td>
<td>2001</td>
<td>Green</td>
<td>FL</td>
<td>$19,000</td>
</tr>
<tr>
<td>3845</td>
<td>Maxima</td>
<td>2001</td>
<td>Blue</td>
<td>NY</td>
<td>$22,000</td>
</tr>
<tr>
<td>8354</td>
<td>Accord</td>
<td>2000</td>
<td>Green</td>
<td>VT</td>
<td>$18,000</td>
</tr>
<tr>
<td>4395</td>
<td>Civic</td>
<td>2001</td>
<td>Red</td>
<td>CA</td>
<td>$17,000</td>
</tr>
<tr>
<td>7352</td>
<td>Civic</td>
<td>2002</td>
<td>Red</td>
<td>WA</td>
<td>$18,000</td>
</tr>
</tbody>
</table>
• **Task:** create sets of elements that satisfy a (or several) criteria.

• **Edge:** output of one task serves as input to the next
• An alternate task-dependency graph for query

• Different task decomposition leads to different parallelism
Granularity of Task Decomposition

- **Fine-grained** decomposition: large number of small tasks
- **Coarse-grained** decomposition: small number of large tasks

Matrix-vector multiplication example

-- **coarse-grain**: each task computes 3 elements of $y[]$

![Diagram of matrix-vector multiplication example](image)
Degree of Concurrency

• **Degree of Concurrency**: # of tasks that can execute in parallel
  -- *maximum degree of concurrency*: largest # of concurrent tasks at any point of the execution
  -- *average degree of concurrency*: average # of tasks that can be executed concurrently

• Degree of Concurrency vs. Task Granularity
  – Inverse relation
Critical Path of Task Graph

- **Critical path**: The longest directed path between any pair of *start node* (node with no incoming edge) and *finish node* (node with outgoing edges).

- **Critical path length**: The sum of weights of nodes along critical path.

- **Average degree of concurrency** = total amount of work / critical path length
Example: Critical Path Length

Task-dependency graphs of query processing operation

Left graph:
- Critical path length = 27
- Average degree of concurrency = $63/27 = 2.33$

Right graph:
- Critical path length = 34
- Average degree of concurrency = $64/34 = 1.88$
Limits on Parallelization

• Facts bounds on parallel execution
  – Maximum task granularity is finite
    • Matrix-vector multiplication $O(n^2)$
  – Communication between tasks
• **Speedup** = sequential execution time/parallel execution time
• **Parallel efficiency** = sequential execution time/(parallel execution time $\times$ processors used)