Contents

- Preliminaries of parallel algorithm design
  - Decomposition
  - Task dependency
    - Task dependency graph
  - Granularity
  - Concurrency
  - Task interaction
Parallel algorithm design

- Identifying tasks which can be done concurrently
- Mapping these tasks onto multiple processes
- Distributing data: input, intermediate and output
- Managing shared data
- Synchronizing

Preliminaries

- Decomposition: dividing work into smaller tasks
  - Task: units of computation
  - Tasks can be related or unrelated
- Task dependency
  - Some tasks may require completion of other tasks
  - Independent tasks are candidates for parallelism
- Task dependency graph
  - It has an arrow from task x to task y if x must be completed before y
Matrix-vector multiplication

• Each task is to compute dot product of 1 row of A times the vector b
• There is no order requirement for computation
• Task dependency graph has no edges

Evaluating an expression

• Compute $a(x) \times b(x) \times (c(x) + d(x))$
• Assume $a$, $b$, $c$ and $d$ are functions
• In general there can be many task dependency graphs
Database Query Processing

- A relational database of vehicles
- A query looks for all 2001 Civics whose color is either Green or White

MODEL="Civic" AND YEAR="2001" AND (COLOR="Green" OR COLOR="White")

<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>

Table 3.1 A database storing information about used vehicles.

Database Query 1

Figure 3.2 The different tables and their dependencies in a query processing operation.
Granularity, Concurrency and Task Interaction

- **Granularity**
  - task size
- **Concurrency**
  - number of simultaneous tasks
- **Interaction**
  - sharing data, synchronizing
**Granularity**

- The size of tasks after decomposition is granularity
- Fine-grained: large number of small tasks
  - matrix-vector multiply as decomposed earlier
- Coarse-grained: small number of large tasks
  - blocked matrix-vector multiply

---

**Matrix-Vector Multiply**

Coarse-grained

\[
\begin{array}{cccc}
0 & 1 & \ldots & n \\
\hline
\text{Task 1} & \text{Task 2} & \text{Task 3} & \text{Task 4} \\
\end{array}
\]

*Figure 3.4*  Decomposition of dense matrix-vector multiplication into four tasks. The portions of the matrix and the input and output vectors accessed by Task 1 are highlighted.
Concurrentcy

- Maximum degree of concurrency
  - Maximum number of tasks that can be executed in a program at any given time
  - May be misleading
- Concurrency increases as granularity decreases
  - There is a bound on how fine-grained a decomposition permits
- Concurrency depends on the task dependency graph
  - The same granularity does not guarantee the same degree of concurrency

Critical Path

- Critical path
  - The longest path between any pair of start and finish nodes
  - Start node: node with no incoming edges
  - Finish node: node with no outgoing edges
- Critical path length (with weights)
  - The sum of the weights of the nodes along the critical path
  - Weight of a node: the size of the amount of work of the node
- Average degree of concurrency
  - The ratio of the total amount of work to the critical path length
    - It takes into account the weight of a node
Critical Path Example (with weight)

Total of all task times = 41
Critical path time is 6+7+8 = 21
Average concurrency = 41/21 = 1.95

Task Graphs for Query Example

The average degree of concurrency of the two task-dependency graphs is 2.33 and 1.88
Interaction

- Interaction: how data is shared between tasks
  - Which tasks communicate with which other tasks
- Reflected in a task interaction graph
- Sparse matrix-vector multiplication
  - Decomposed by partitioning output vector $y$ into one task per entry
  - Task $i$ owns row $i$ of $A$ and $b[i]$
  - Tasks must get some $b$ data from other tasks

**Sparse Matrix-Vector Multiplication: $y=Ab$**

![Sparse Matrix-Vector Multiplication Diagram](image)

**Figure 3.5** A decomposition for sparse matrix-vector multiplication and the corresponding task-interaction graph. In the decomposition Task $i$ computes $\sum_{0 \leq j \leq 11, A[i,j] \neq 0} A[i,j]b[j]$. 
Processes and Mapping

- **Process**
  - Refer to a processing or computing agent that performs tasks
    - An abstract entity that uses code and data to produce the output of the task

- **Processes vs. Processors**
  - Processes are logical computing agents that perform tasks
  - Processors are hardware units that physically perform computations

- **Mapping**
  - The mechanism by which tasks are assigned to processes for execution
  - Task-dependency and task-interaction graphs play an important role in mapping

Mapping Tasks to Processes

Figure 3.7 Mappings of the task graphs of Figure 3.5 onto four processes.
Summary

- Preliminaries of parallel algorithm design
  - Decomposition: dividing work into smaller tasks
  - Task dependency graph
  - Granularity
  - Concurrency
  - Task interaction
  - Task mapping