Parallel algorithm design

Jeremy Iverson

College of Saint Benedict & Saint John's University

Parallel algorithm design

- how do we identify concurrency in our algorithms?
- how do we assign work to processes?
- how do we distribute data to processes?

1. all of these things affect our analysis — potentially introducing parallel overhead and decreasing speedup

Problem decomposition

- the process of dividing the computation into smaller pieces of work, i.e., *tasks*
- tasks are programmer defined

1. answers questions of how do we identify concurrency in our algorithms

Example — dense matrix-multiplication

1. show this example on the board
2. just point out what we defined as tasks, and ask why? — because it was the easiest / most natural?
3. are there any other ways that we could have decomposed the problem?

Example — query processing

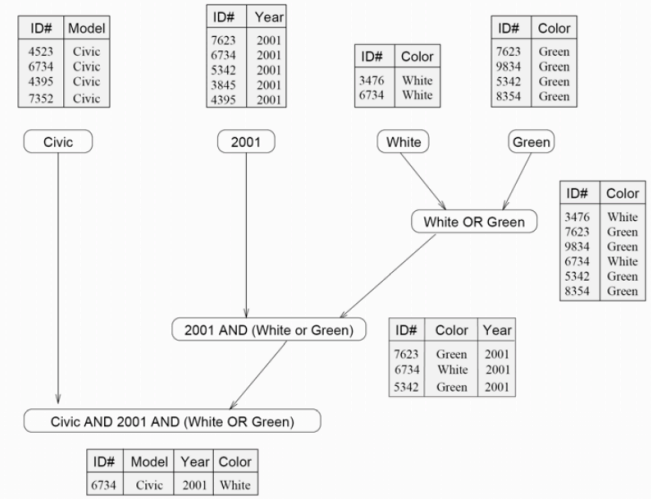
ID#	Model	Year	Color	Dealer	Price
4523	Civic	2002	Blue	MN	$18,000
3476	Corolla	1999	White	IL	$15,000
7623	Camry	2001	Green	NY	$21,000
9834	Prius	2001	Green	CA	$18,000
6734	Civic	2001	White	OR	$17,000
5342	Altima	2001	Green	FL	$19,000
3845	Maxima	2001	Blue	NY	$22,000
8354	Accord	2000	Green	VT	$18,000
4395	Civic	2001	Red	CA	$17,000
7352	Civic	2002	Red	WA	$18,000

A database storing information about used vehicles.

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

1. relation database of vehicles
2. query looks for all 2001 Civics whose color is either Green or White
3. in a relational database, this query could be processed by creating a number of intermediate tables
4. the intermediate tables are combined using set intersection / union

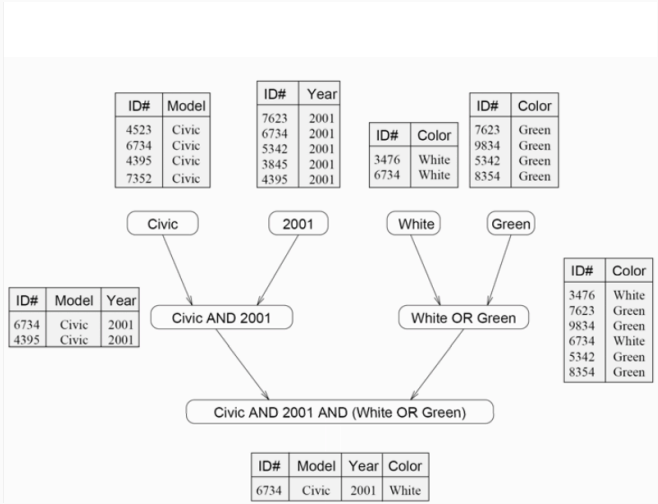
Example — query processing cont'd



The different tables and their dependencies in a query processing operation.

1. the task dependency graphs for a serialized way of organizing the computations

Example — query processing cont'd

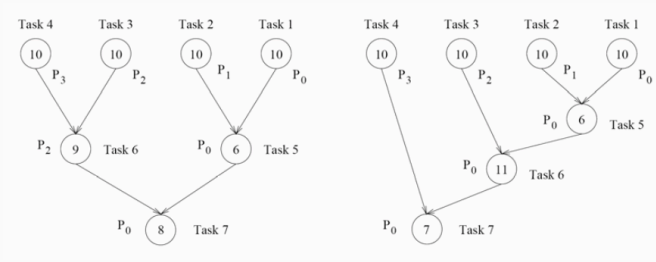


The different tables and their dependencies in a query processing operation.

1. an organization that exposes more concurrency

Task-dependency graph

- represented using a directed acyclic graph (DAG)



Task-dependency graphs and their mappings onto four processes for query processing.

useful metrics

- degree of concurrency
- critical path

1. degree of concurrency — the number of tasks that can be executed concurrently
2. critical path — sum of the weights of nodes along the longest directed path between any pair of start and finish nodes
3. the ratio of critical path to total work is the average degree of concurrency — in this example 63/27=2.33 and 64/34=1.88 respectively
4. what is the relationship between critical path and other metrics that we have studied?

Common decomposition methods

- data decomposition
- recursive decomposition
- exploratory decomposition
- speculative decomposition
- hybrid decomposition

1. up until now, our technique for identifying tasks was pretty adhoc, i.e., based on our expert experience, not a programmatic approach
2. we will focus mostly on data decomposition — it is the most common

Data decomposition

- derive the tasks by focusing on the multiplicity of the data
- consists of two steps:
  1. partition the data
  2. derive tasks from the data partitioning
- common data decompositions
  - input
  - output
  - intermediate
- owner computes rule

1. how to derive tasks from data partitioning? — owner computes rule
2. owner computes rule — tasks are all computations associated with a partition of data

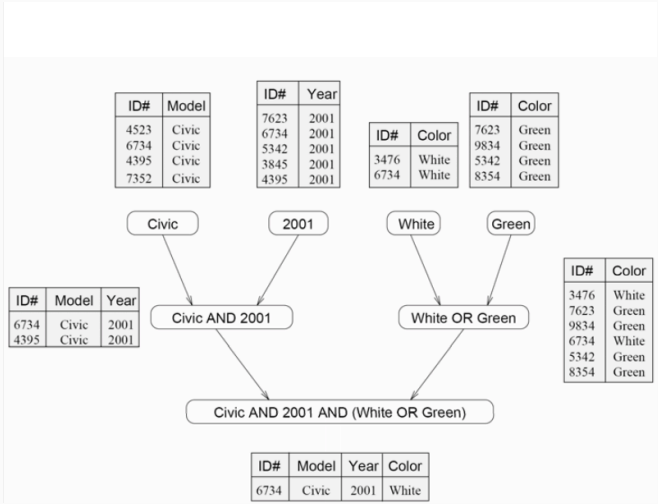
Input decomposition

1. matrix-multiplication on board

Output decomposition

1. matrix-multiplication on board

Example — query processing



The different tables and their dependencies in a query processing operation.

1. what sort of decomposition is this?

Intermediate decomposition

coming soon...

except where otherwise noted, this worked is licensed under creative commons attribution-sharealike 4.0 international license