Parallel Algorithm Design

K Hari Babu
Department of Computer Science & Information Systems

BITS Pilani
Pilani Campus

Designing Parallel Programs

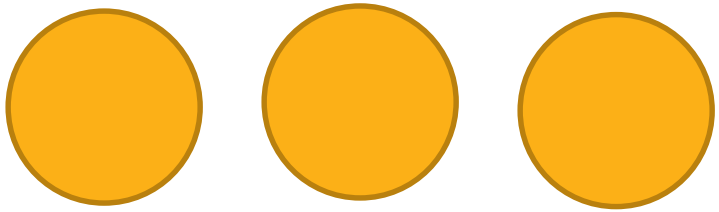
- Steps for parallel program design and implementation
- Identifying Parallelism
  - Computational task partitioning. Aggregate tasks when needed.
  - Dependence analysis to derive a task graph
- Mapping & Scheduling of parallelism
  - Map tasks =⇒ processors (cores)
  - Order execution
- Parallel Programming
  - Coding
  - Debugging
- Performance Evaluation

Task Decomposition

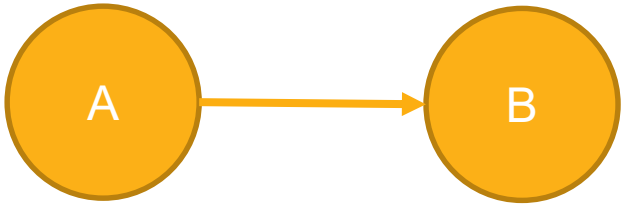
- The first step in developing a parallel algorithm is to decompose the problem into tasks that can be executed concurrently
- A given problem may be decomposed into tasks in many different ways.
- Tasks may be of same, or different sizes.
- A decomposition can be illustrated in the form of a directed graph with nodes corresponding to tasks and edges indicating that the result of one task is required for processing the next. Such a graph is called a *task dependency graph* or *data dependency graph*.

Task Graphs

- A set of Tasks



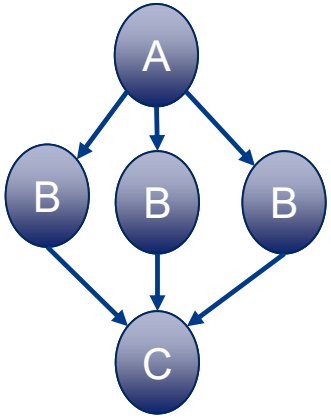
- Data dependence among tasks



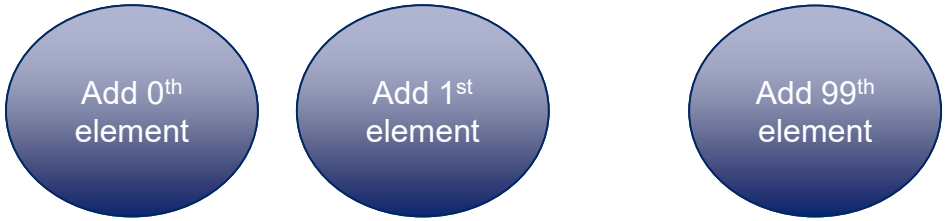
Seeking Concurrency

BITS Pilani

- Parallel computers are more available than ever, but in order to take advantage of multiple processors, programmers and/or compilers must be able to identify operations that may be performed in parallel (i.e., concurrently)
- Data parallelism
  - A data dependence graph exhibits data parallelism when there are independent tasks applying the same operation to different elements of a dataset.



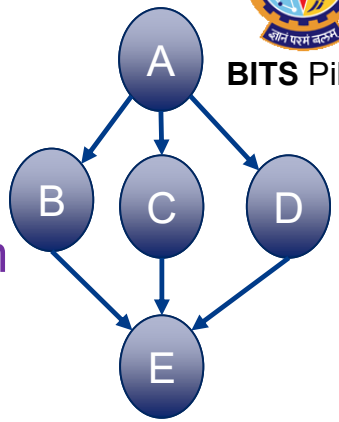

```
2    for(int i=0;i<100;i++)
3        a[i]=b[i]+c[i];
```



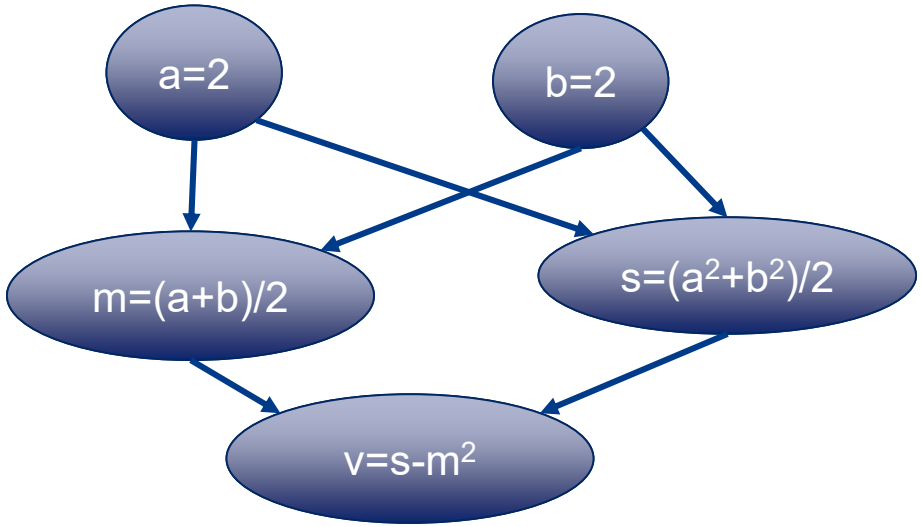
While we could draw one vertex for each step in the algorithm, it is better to draw a vertex for each step of the algorithm for each data element because it exposes more opportunities for parallelism.

Seeking Concurrency

- Functional parallelism
  - A data dependence graph exhibits functional parallelism when there are independent tasks applying different operations to different data elements




```
1    a=2;
2    b=3;
3    m = (a+b)/2;
4    s=(a^2 + b^2)/2;
5    v= s-m^2;
```



While we could draw one vertex for each step in the algorithm, it is better to draw a vertex for each step of the algorithm for each data element because it exposes more opportunities for parallelism.

Seeking Concurrency

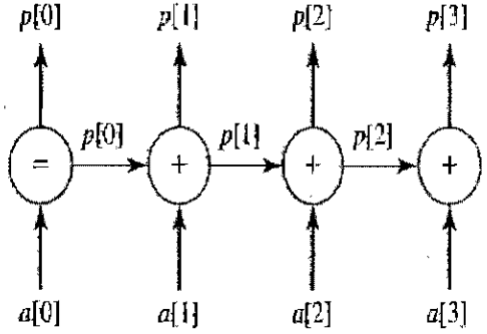
BITS Pilani

- Pipelining
  - A data dependence graph forming a simple path or chain admits no parallelism if only a single problem instance must be processed.
  - If problem instances need to be processed, each problem instance an be divided into stages and concurrency equal to the number of stages can be achieved.

```
 8    p[0]=a[0]
 9   for(int i=1;i<=3;i++){
10       p[i]=p[i-1]+a[i]
11   }
```

```
13   p[0]=a[0]
14   p[1]=p[0]+a[1]
15   p[2]=p[1]+a[2]
16   p[3]=p[2]+a[3]
```

Suppose 2 sets of nos:
{1,2,3,4}
{5,6,7,8}



If multiple instances of a problem are there, operation C may be performed on instance i, while operation B is performed on instance i+1 and operation A is performed on instance i+2.

Data Clustering

- Data clustering is the process of organizing a dataset into groups, or clusters of "similar" items
- Algorithm
  1. Input N documents
  2. For each of the N documents generate a D-dimensional vector indicating how well it covers the D diffent topics
  3. Choose the K initial cluster centers using a random sample
  4. Repeat the following steps for *I* iterations or until the performance function converges whichever comes first

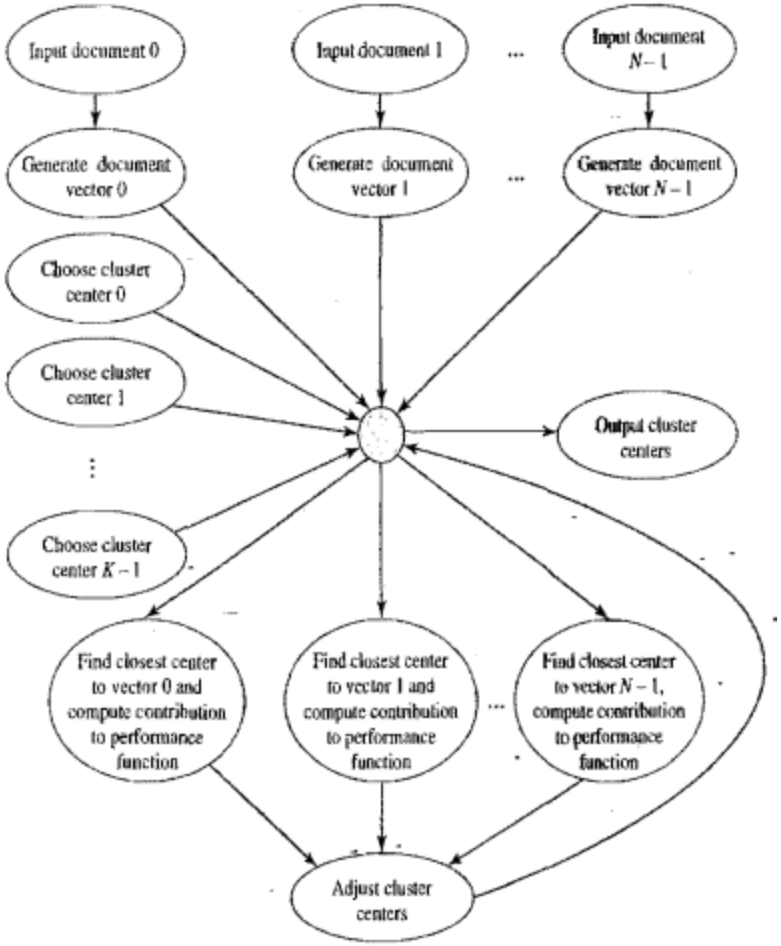
  (a)for each of the N documents, find the closest center and compute its contribution to the performance function

  (b) Adjust the K cluster centers to try to improve the value of the performance function
  5. OUTPUT K centers

Data Clustering

BITS Pilani

- Our first step in the analysis is to draw a data dependence graph.



A good data dependence graph makes data and functional parallelism easy to find.

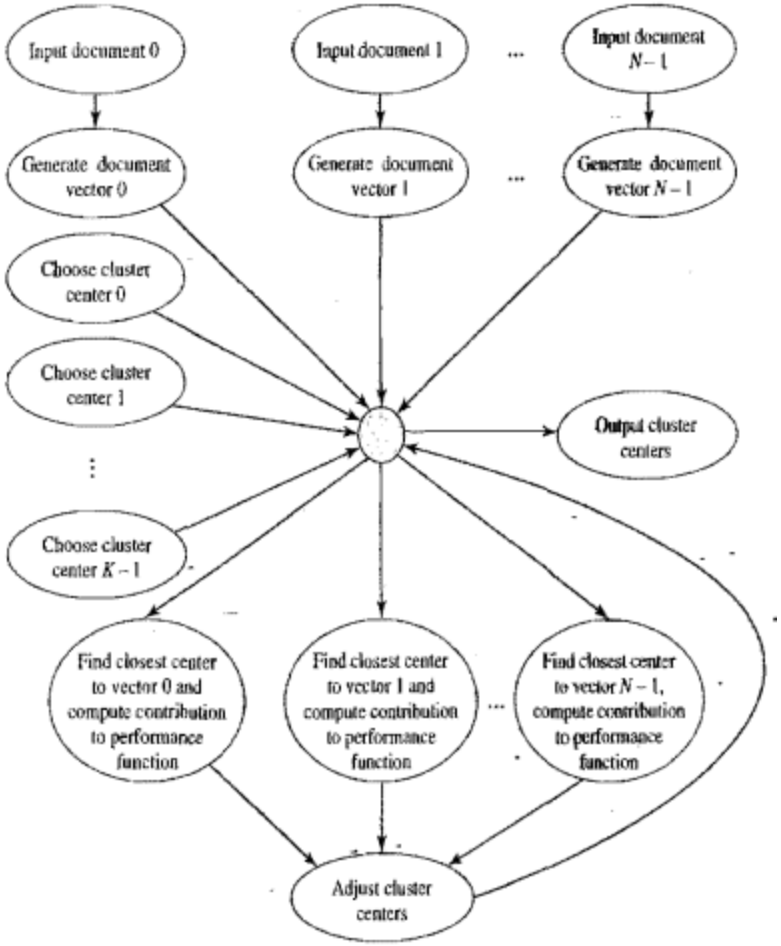
<u>Opportunities for data parallelism:</u>

- Each document may be input in parallel.
- Each document vector may be generated in parallel.
- The original cluster centers may be generated in parallel.
- The closest cluster center to each document vector and that vector's contribution to the overall performance function may be computed in parallel.

Data Clustering

BITS Pilani

- Our first step in the analysis is to draw a data dependence graph.



A good data dependence graph makes data and functional parallelism easy to find.

<u>Opportunities for functional parallelism:</u>
The only independent sets of vertices are those representing the document input and vector generation tasks and those representing the center generation tasks. These two sets of tasks could be performed concurrently

Example: Database Query Processing

BITS Pilani

- Consider the execution of the query:

  MODEL = ``CIVIC'' AND YEAR = 2001 AND (COLOR = ``GREEN'' OR COLOR = ``WHITE)

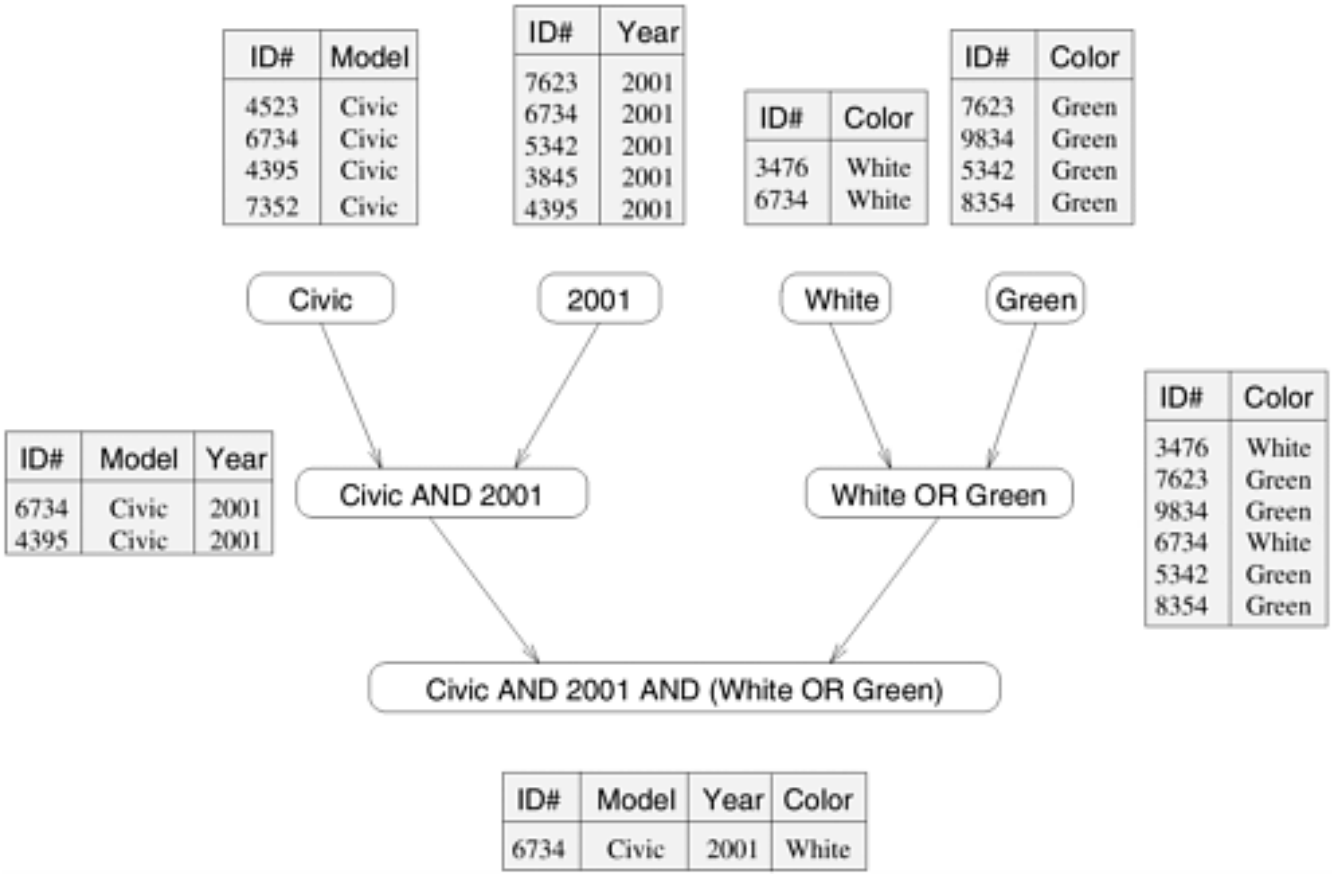
- on the following database:

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

Decomposition 1

BITS Pilani

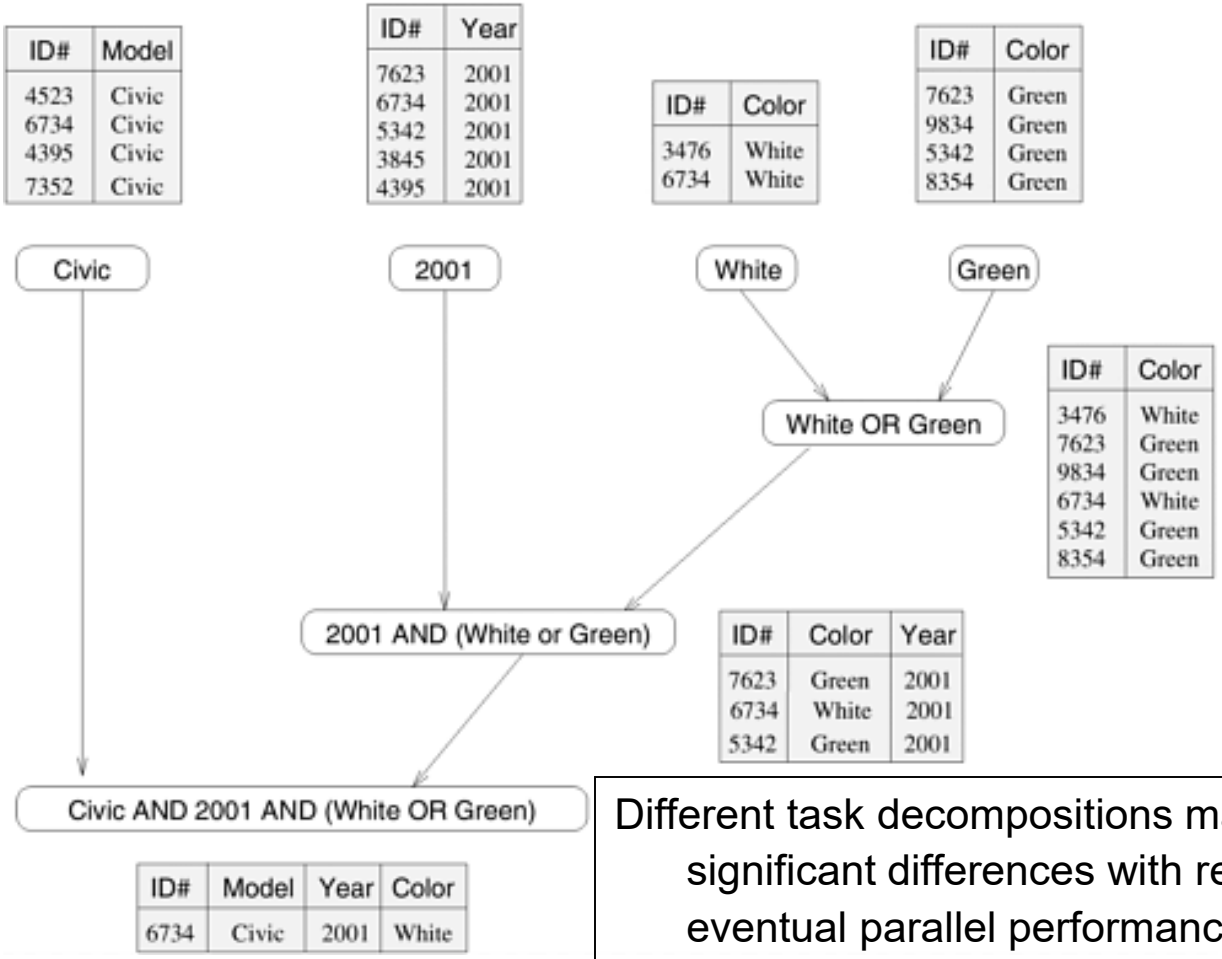
- The execution of the query can be divided into subtasks in various ways. Each task can be thought of as generating an intermediate table of entries that satisfy a particular clause.



Decomposition 2

BITS Pilani

- Note that the same problem can be decomposed into subtasks in other ways as well.

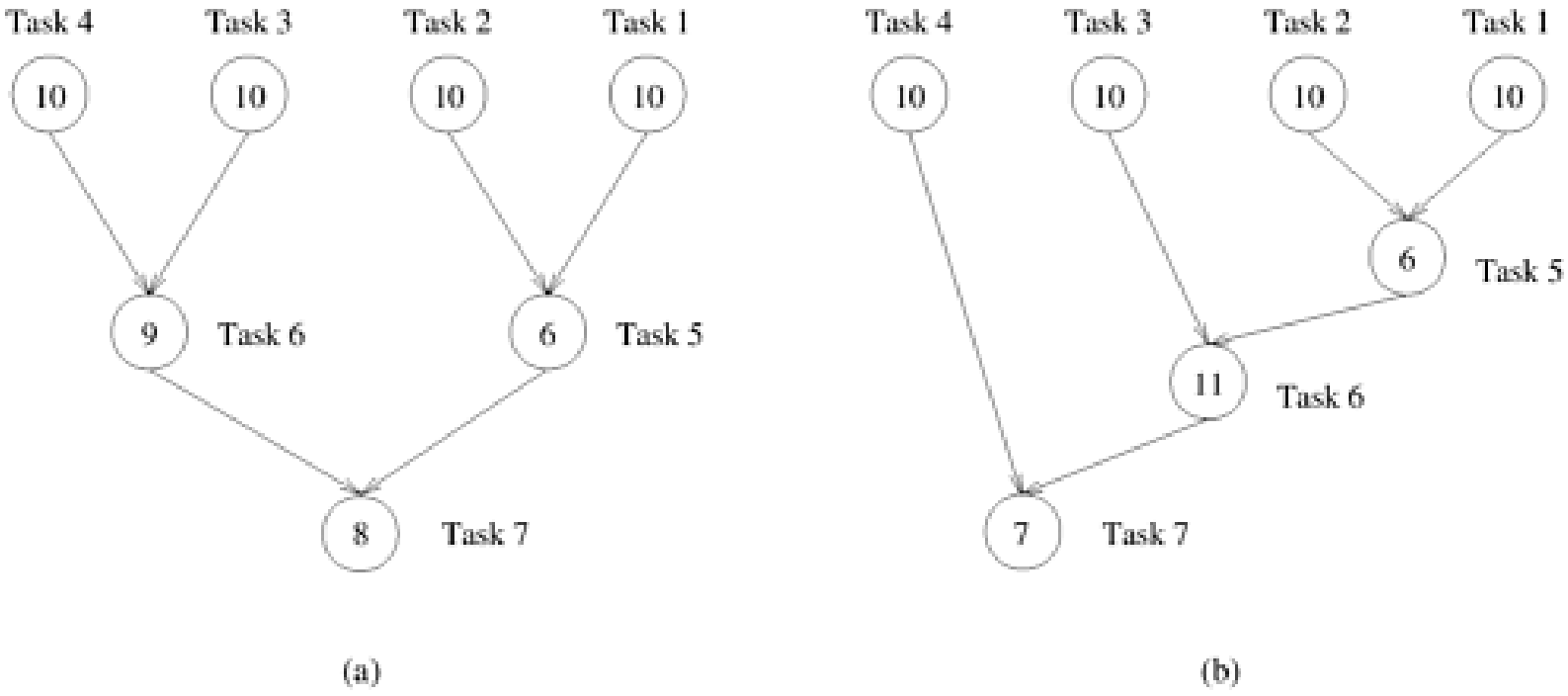


Different task decompositions may lead to significant differences with respect to their eventual parallel performance.

Comparing Decompositions

- The number of tasks that can be executed in parallel is the *degree of concurrency* of a decomposition.
- The length of the longest path in a task dependency graph is called the critical path length.
- The ratio of the total amount of work to the critical-path length is the average degree of concurrency.

Comparing Decompositions



- Maximum degree of concurrency: 4 4
- Critical path length: 27 34
- Total work: 63 64
- Average degree of concurrency: 63/27=2.34 64/34=1.88

References

- Chapter 1.6 from M.J. Quinn, *Parallel Programming in C using MPI and OpenMP*, McGraw Hill Indian Edition. 2003

innovate achieve lead

BITS Pilani
Pilani Campus

Thank You