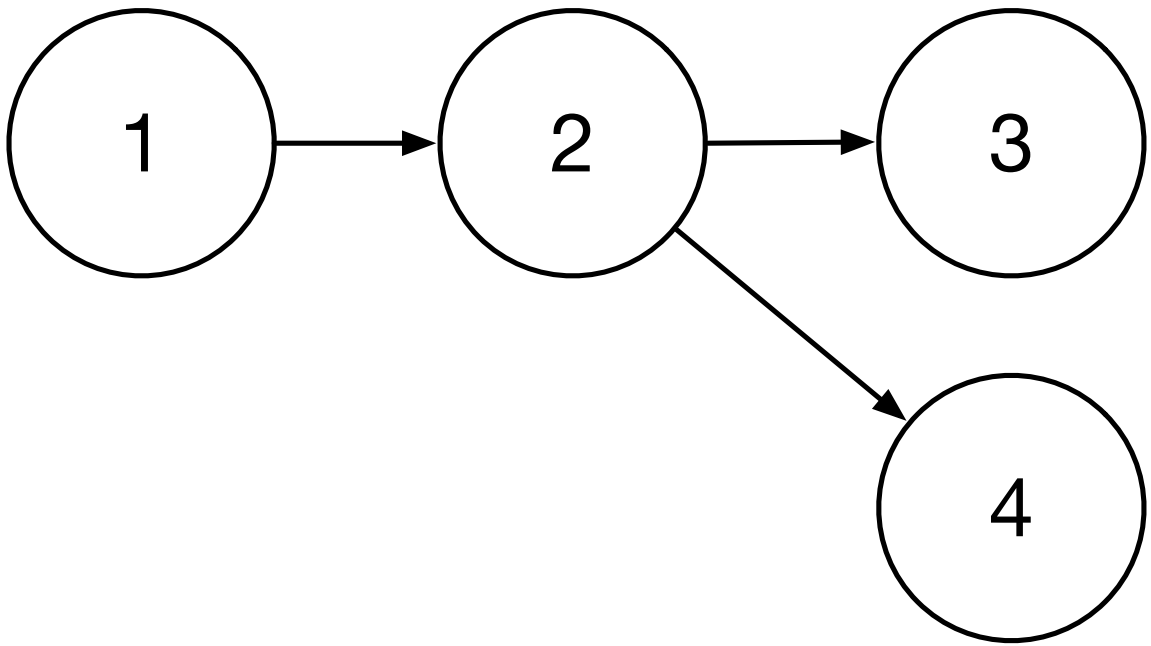
Chapter 2: Execution Model

Elements of Parallel Computing

Eric Aubanel

Real Data Dependencies

1 $a \leftarrow 5$
2 $b \leftarrow 3 + a$
3 $c \leftarrow 2 * b - 1$
4 $d \leftarrow b + 1$



Anti-Dependence

1 $a \leftarrow 8$
2 $b \leftarrow 7 * a$
3 $c \leftarrow 3 * b + 1$
4 $b \leftarrow 2 * a$
5 $c \leftarrow (a + b)/2$

A false dependence

Output Dependence

1 $a \leftarrow 3$
2 $b \leftarrow 5 + a$
3 $a \leftarrow 42$
4 $c \leftarrow 2 * a + 1$

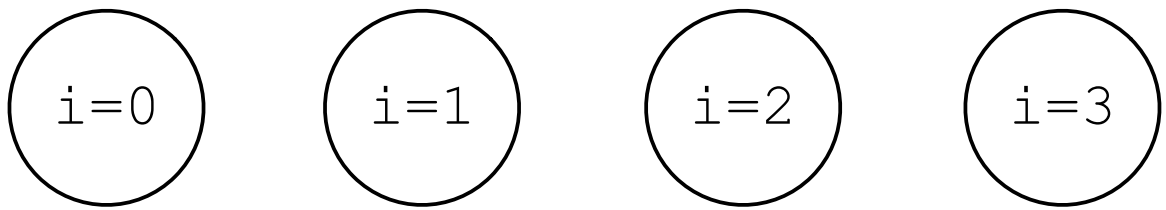
Also a false dependence

Task Graph Model

- Directed Acyclic Graph (DAG)
  - vertex: execution of a task
  - edge: real data dependence
- $u \rightarrow v$: task u must execute before task v
  - Implies communication
- Graph can have non-negative weights: computation (vertex) and communication (edge) cost
- Control dependencies encapsulated in tasks

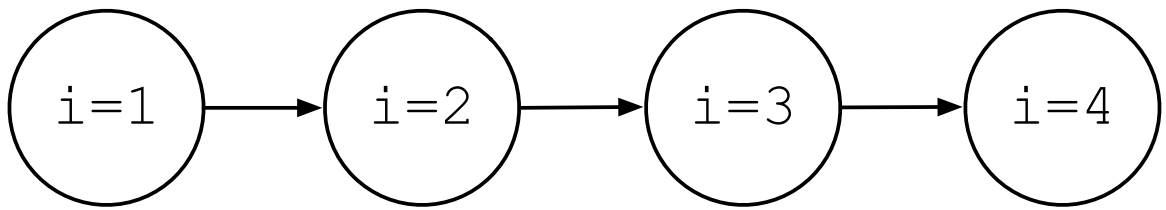
Simplest Task Graph Example

for $i \leftarrow 0$ **to** $n - 1$ **do**
    $c[i] \leftarrow a[i] + b[i]$
end



Dependence Between Iterations

for $i \leftarrow 1$ **to** $n - 1$ **do**
    $a[i] \leftarrow a[i-1] + x * i$
end



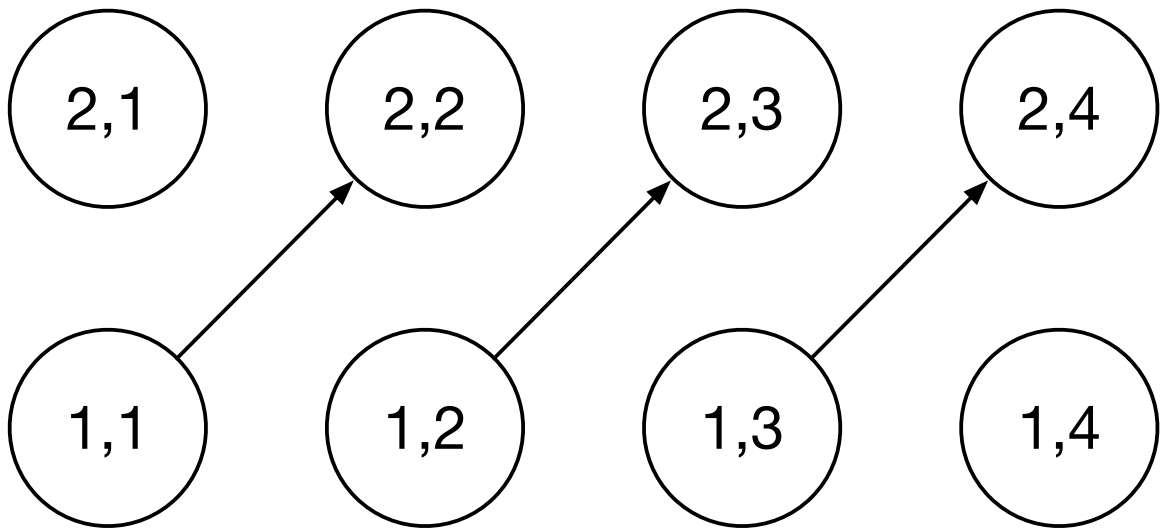
Another Task Graph

for $i \leftarrow 1$ **to** $n-1$ **do**

1 $\quad a[i] \leftarrow b[i] + x * i$

2 $\quad c[i] \leftarrow a[i-1] * b[i]$

end



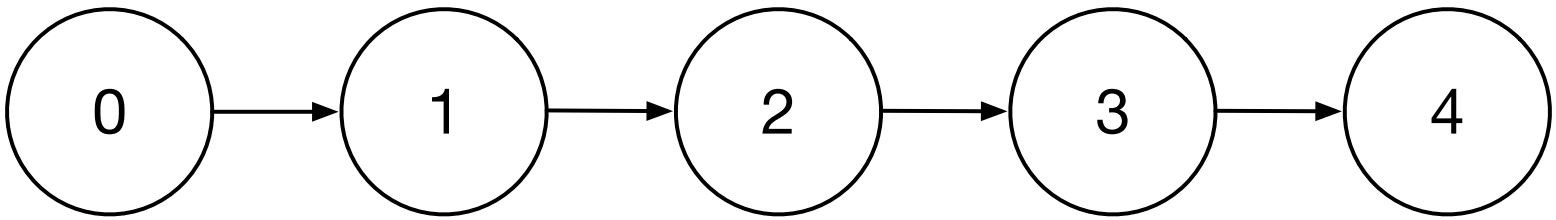
Reduction

count $\leftarrow$ 0
foreach *document in collection* **do**
    count $\leftarrow$ count + countOccurrences(word, document)
end

Unroll 4 iterations:

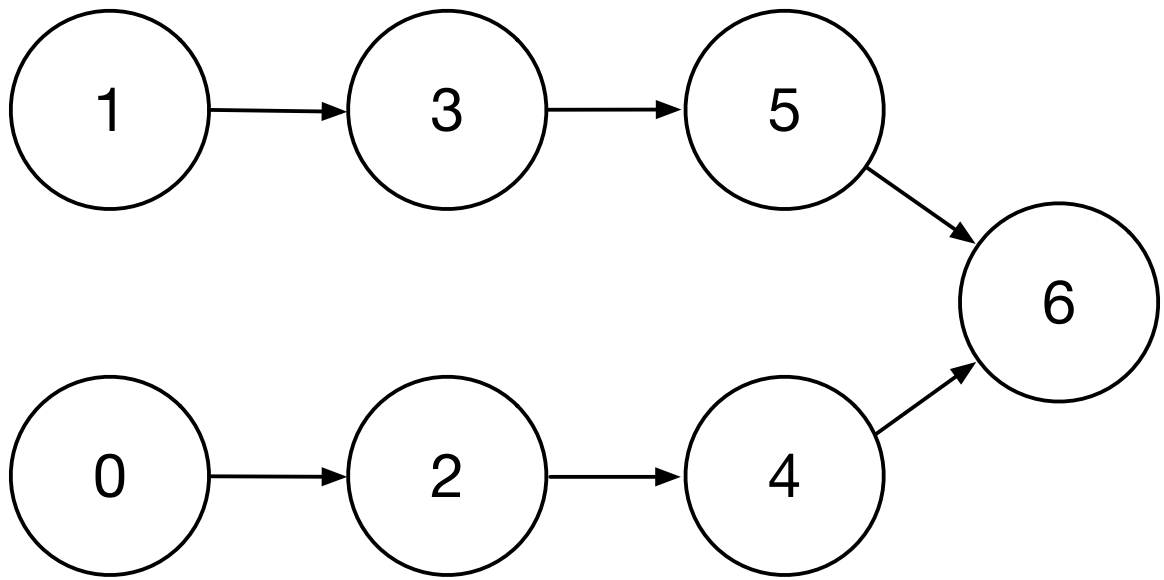
0: count $\leftarrow$ 0
1: count $\leftarrow$ count + countOccurrences(word, document1)
2: count $\leftarrow$ count + countOccurrences(word, document2)
3: count $\leftarrow$ count + countOccurrences(word, document3)
4: count $\leftarrow$ count + countOccurrences(word,

Sequential Reduction

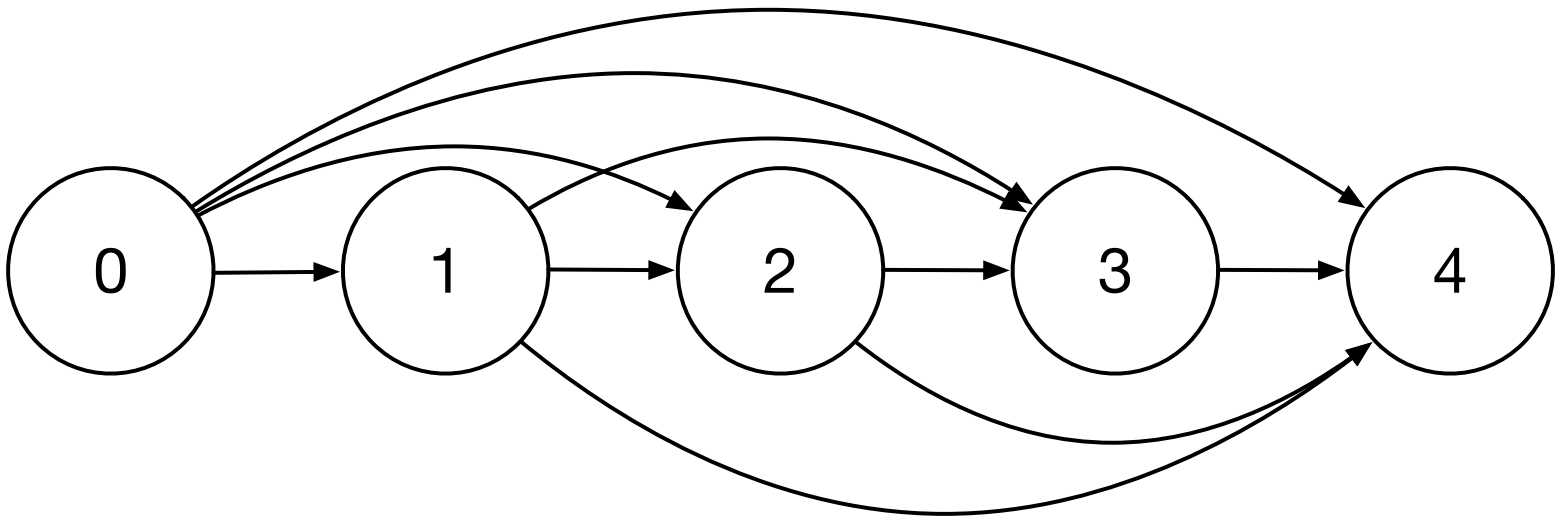


Rewrite to Create Independent Tasks

0: count1 ← 0
1: count2 ← 0
2: count1 ← count1 + countOccurrences(word, document1)
3: count2 ← count2 + countOccurrences(word, document2)
4: count1 ← count1 + countOccurrences(word, document3)
5: count2 ← count2 + countOccurrences(word, document4)
6: count ← count1 + count2

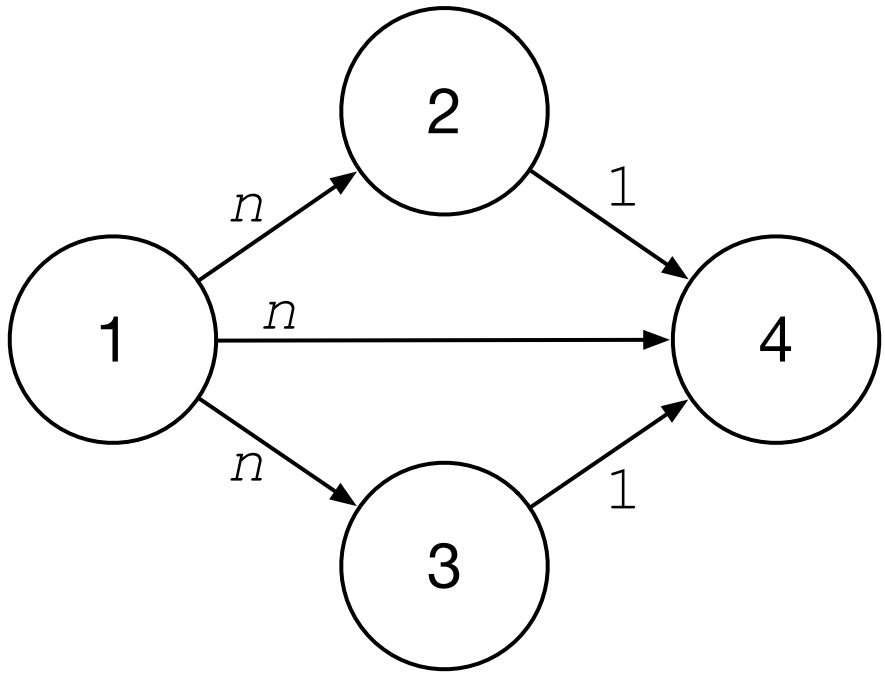


Sequential Reduction Showing Transitive Dependencies



Desirable Transitive Dependencies

1: $y \leftarrow$ `foo()`
2: $ymin \leftarrow$ `min(`y`)`
3: $ymax \leftarrow$ `max(`y`)`
4: **for** $i \leftarrow 0$ *to* $n - 1$ **do**
    $ynorm[i] \leftarrow (y[i] - ymin)/(ymax - ymin)$
end



Mapping Task Graphs