
COMP 322: Fundamentals of Parallel Programming

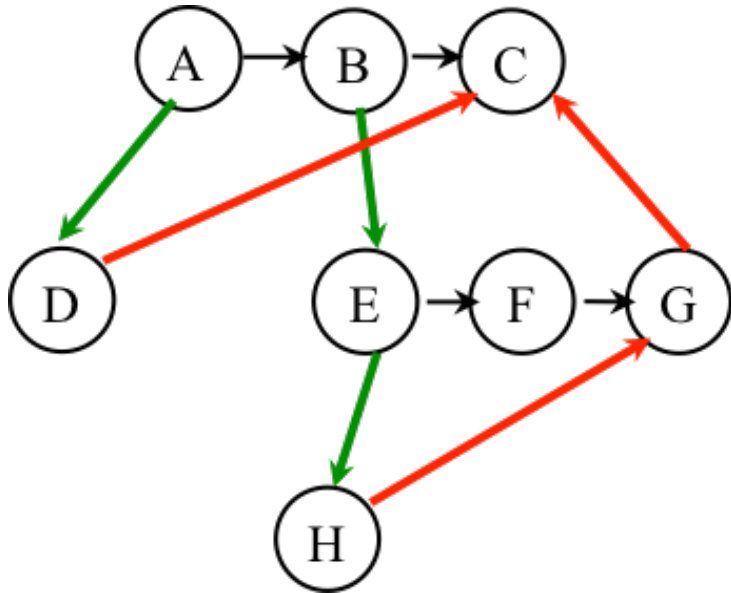
Lecture 3: Multiprocessor Scheduling

Mack Joyner and Zoran Budimlić
{mjoyner, zoran}@rice.edu

<http://comp322.rice.edu>



One Possible Solution to Worksheet 2 (Reverse Engineering a Computation Graph)



```
1. A ();
2. finish { // F1
3.   async D ();
4.   B ();
5.   async {
6.     E ();
7.     finish { // F2
8.       async H ();
9.       F ();
10.    } // F2
11.   G ();
12. }
13. } // F1
14. C ();
```

Observations:

- Any node with out-degree > 1 must be an async (must have an outgoing **spawn edge**)
- Any node with in-degree > 1 must be an end-finish (must have an incoming **join edge**)
- Adding or removing transitive edges does not impact ordering constraints



Ordering Constraints and Transitive Edges in a Computation Graph

- The primary purpose of a computation graph is to determine if an ordering constraint exists between two steps (nodes)
 - Observation: Node A must be performed before node B if there is a path of directed edges from A and B
- An edge, $X \rightarrow Y$, in a computation graph is said to be *transitive* if there exists a path of directed edges from X to Y that does not include the $X \rightarrow Y$ edge
 - Observation: Adding or removing a transitive edge does not change the ordering constraints in a computation graph



What is the critical path length of this parallel computation?

```
1. finish { // F1
2.   async A; // Boil water & pasta (10)
3.   finish { // F2
4.     async B1; // Chop veggies (5)
5.     async B2; // Brown meat (10)
6.   } // F2
7.   B3; // Make pasta sauce (5)
8. } // F1
```

Step B1



Step B2



Step B3

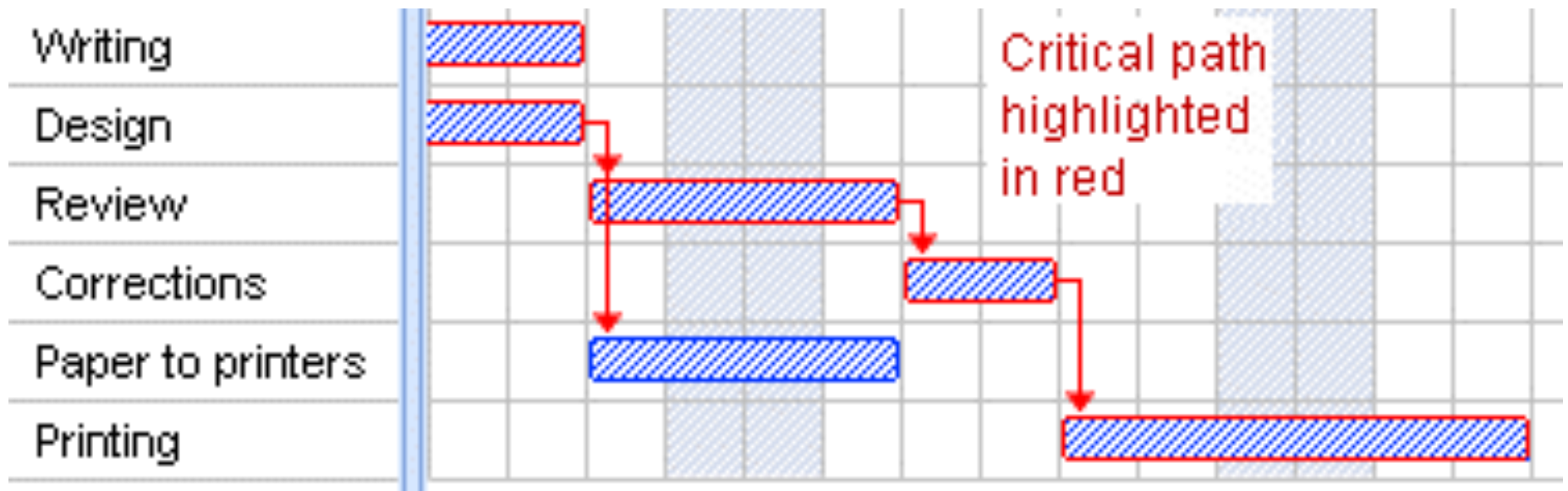


Step A



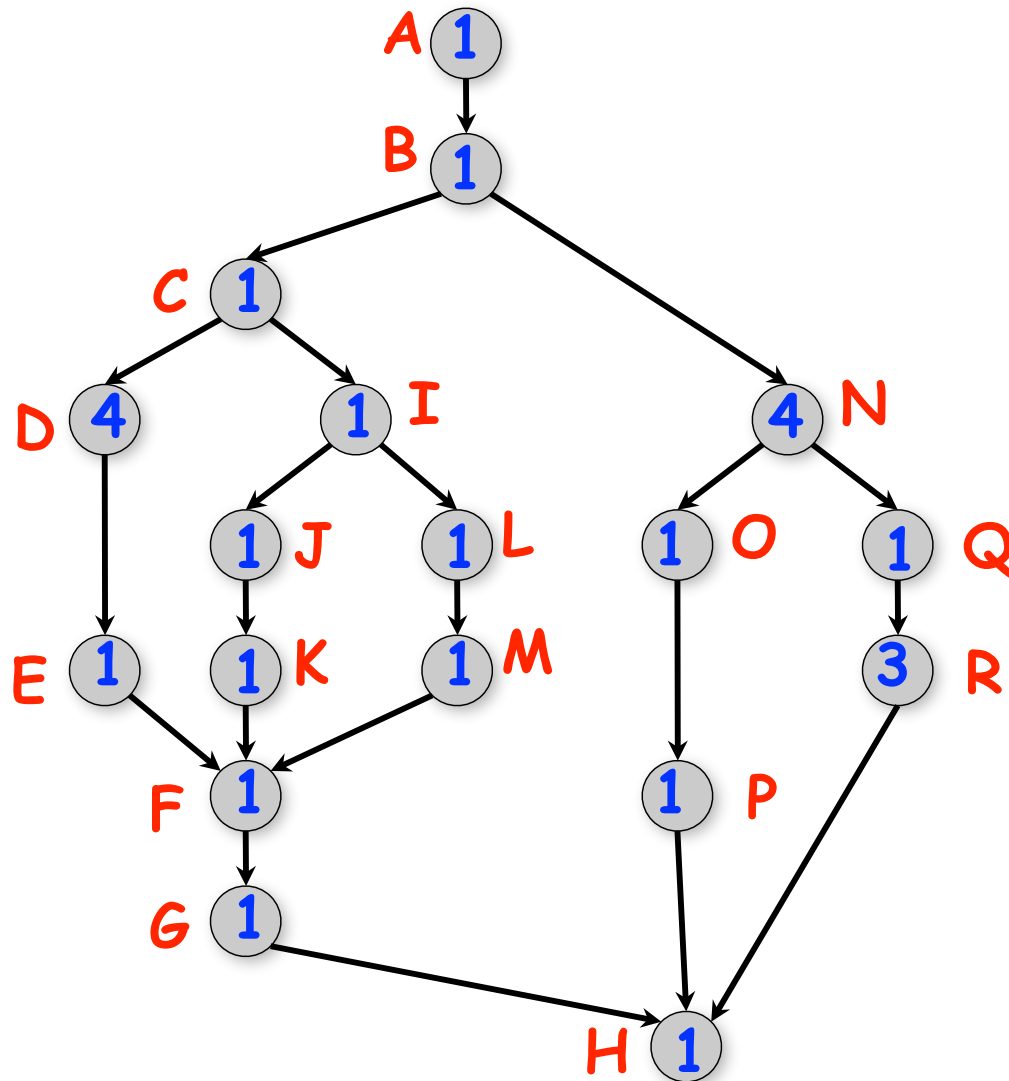
Computation Graphs are used in Project Scheduling as well

- Computation graphs are referred to as “Gantt charts” in project management
- Sample project for preparing a printed document
 - Source: <http://www.gantt.com/creating-gantt-charts.htm>



Scheduling of a Computation Graph on a fixed number of processors: Example

node label = time(N), for all nodes N in the graph



Start time	Proc 1	Proc 2	Proc 3
0	A		
1	B		
2	C	N	
3	D	N	I
4	D	N	J
5	D	N	K
6	D	Q	L
7	E	R	M
8	F	R	O
9	G	R	P
10	H		
11	Completion time = 11		

NOTE: this schedule achieved a completion time of 11. Can we do better?



Scheduling of a Computation Graph on a fixed number of processors, P

- Assume that node N takes $\text{TIME}(N)$ regardless of which processor it executes on, and that there is no overhead for creating parallel tasks
- A schedule specifies the following for each node
 - $\text{START}(N)$ = start time
 - $\text{PROC}(N)$ = index of processor in range $1 \dots P$

such that

- $\text{START}(i) + \text{TIME}(i) \leq \text{START}(j)$, for all CG edges from i to j
(Precedence constraint)
- A node occupies consecutive time slots in a processor (Non-preemption constraint)
- All nodes assigned to the same processor occupy distinct time slots (Resource constraint)



Greedy Schedule

- A greedy schedule is one that never forces a processor to be idle when one or more nodes are ready for execution
- A node is **ready** for execution if all its predecessors have been **executed**
- **Observations**
 - $T_1 = \text{WORK}(G)$, for all greedy schedules
 - $T_\infty = \text{CPL}(G)$, for all greedy schedules
- where $T_p(S)$ = execution time of schedule S for computation graph G on P processors



Lower Bounds on Execution Time of Schedules

- Let T_p = execution time of a schedule for computation graph G on P processors
 - T_p can be different for different schedules, for same values of G and P
- Lower bounds for all greedy schedules
 - Capacity bound: $T_p \geq \text{WORK}(G)/P$
 - Critical path bound: $T_p \geq \text{CPL}(G)$
- Putting them together
 - $T_p \geq \max(\text{WORK}(G)/P, \text{CPL}(G))$



Upper Bound on Execution Time of Greedy Schedules

Theorem [Graham '66]. Any greedy scheduler achieves

$$T_p \leq \text{WORK}(G)/P + \text{CPL}(G)$$

Proof sketch:

Define a time step to be **complete** if P processors are scheduled at that time, or **incomplete** otherwise

complete time steps $\leq \text{WORK}(G)/P$

incomplete time steps $\leq \text{CPL}(G)$

Start time	Proc 1	Proc 2	Proc 3
0	A		
1	B		
2	C	N	
3	D	N	I
4	D	N	J
5	D	N	K
6	D	Q	L
7	E	R	M
8	F	R	O
9	G	R	P
10	H		
11			



Bounding the performance of Greedy Schedulers

Combine lower and upper bounds to get

$$\max(\text{WORK}(G)/P, \text{CPL}(G)) \leq T_p \leq \text{WORK}(G)/P + \text{CPL}(G)$$

Corollary 1: Any greedy scheduler achieves execution time T_p that is within a factor of 2 of the optimal time (since $\max(a,b)$ and $(a+b)$ are within a factor of 2 of each other, for any $a \geq 0, b \geq 0$).

Corollary 2: Lower and upper bounds approach the same value whenever

- There's lots of parallelism, $\text{WORK}(G)/\text{CPL}(G) \gg P$
- Or there's little parallelism, $\text{WORK}(G)/\text{CPL}(G) \ll P$



Abstract Performance Metrics (Lab 1)

- Basic Idea
 - Count operations of interest, as in big-O analysis, to evaluate parallel algorithms
 - Abstraction ignores many overheads that occur on real systems
- Calls to `doWork()`
 - Programmer inserts calls of the form, `doWork(N)`, within a step to indicate abstraction execution of N application-specific abstract operation
 - e.g., in the Homework 1 programming assignment (Parallel Sort), we will include one call to `doWork(1)` in each call to `compareTo()`, and ignore the cost of everything else
- Abstract metrics are enabled by calling `HjSystemProperty.abstractMetrics.set(true)` at start of program execution
- If an HJ program is executed with this option, abstract metrics can be printed at end of program execution with calls to `abstractMetrics().totalWork()`, `abstractMetrics().criticalPathLength()`, and `abstractMetrics().idealParallelism()`



Announcements & Reminders

- **IMPORTANT:**
 - Watch video & read handout for topic 1.5 for next lecture on Wednesday, Jan 17th**
- **HW1 was posted on the course web site (<http://comp322.rice.edu>) on Jan 10th, and is due on Jan 24th**
- **Quiz for Unit 1 (topics 1.1 - 1.5) is due by Jan 26th on Canvas**
- **See course web site for all work assignments and due dates**
- **Use Piazza (public or private posts, as appropriate) for all communications re. COMP 322**
- **See Office Hours link on course web site for latest office hours schedule.**

