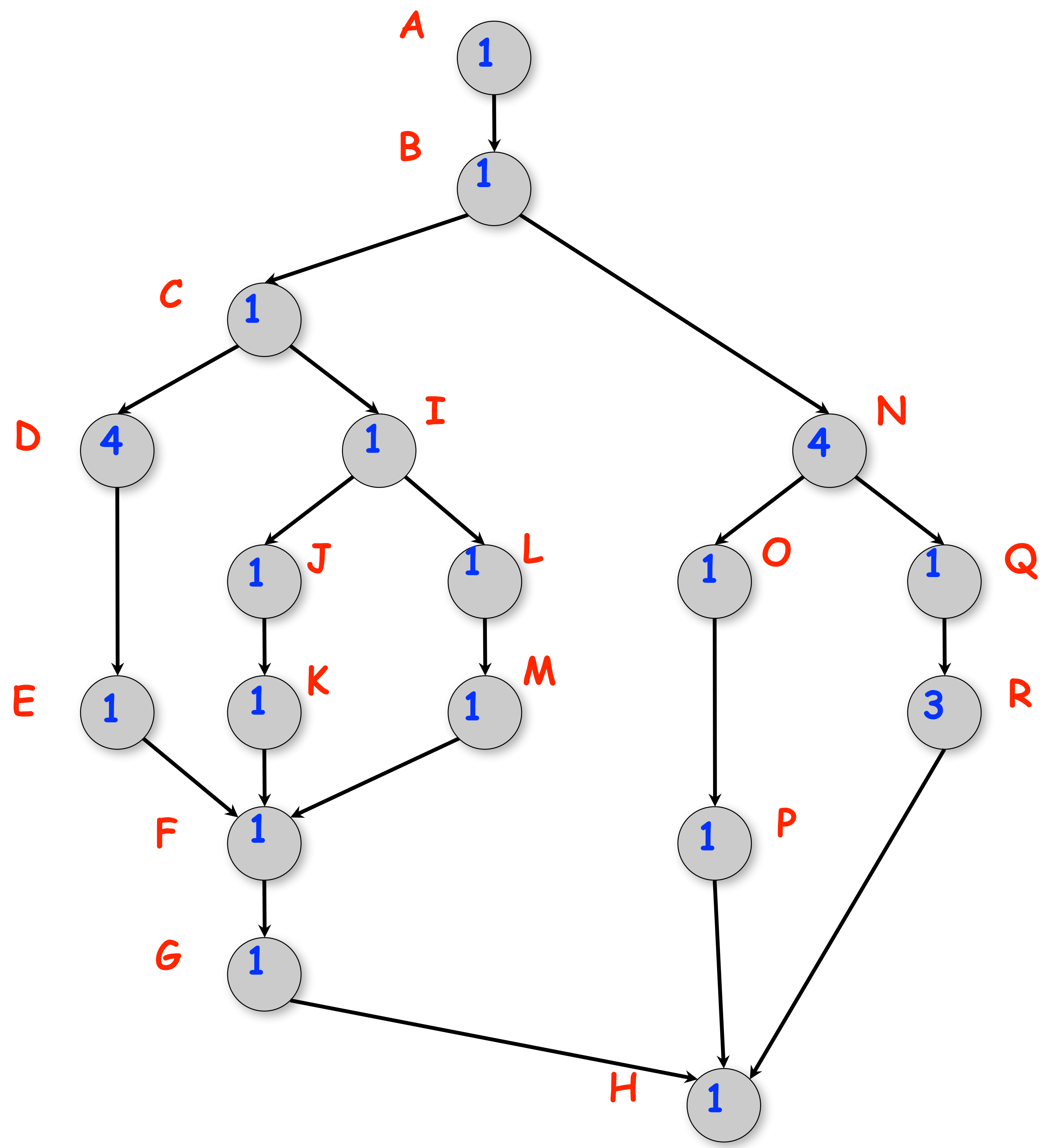
COMP 322: Fundamentals of Parallel Programming

Lecture 4: Parallel Speedup and Amdahl's Law

Mack Joyner
mjoyner@rice.edu

http://comp322.rice.edu

One Possible Solution to Worksheet 3 (Multiprocessor Scheduling)



Start time	Proc 1	Proc 2
0	A	
1	B	
2	C	N
3	D	N
4	D	N
5	D	N
6	D	O
7	I	Q
8	J	R
9	L	R
10	K	R
11	M	E
12	F	P
13	G	
14	H	
15		

- As before, WORK = 26 and CPL = 11 for this graph
- T_2 = 15, for the 2-processor schedule on the right
- We can also see that max(CPL,WORK/2) <= T_2 < CPL + WORK/2
- There are 4 idle slots in this schedule — can we do better than T_2 = 15 ?

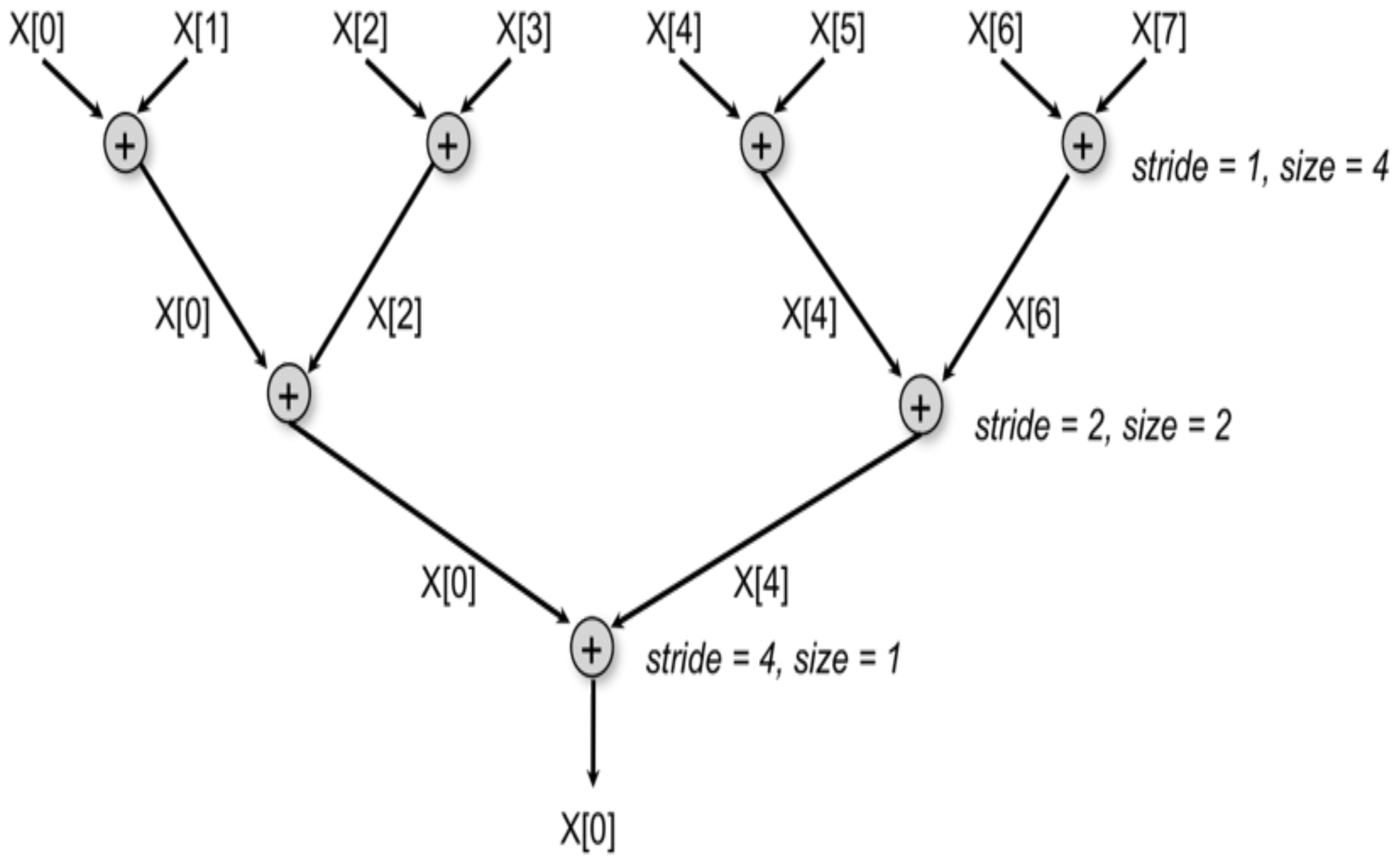
Parallel Speedup

- Define Speedup(P) = T_1 / T_P
  - —Factor by which P processors speeds up execution time relative to 1 processor, for fixed input size
  - —For ideal executions without overhead, 1 <= Speedup(P) <= P
    - —You see this with abstract metrics, but bounds may not hold when measuring real execution times with real overheads
  - —Linear speedup
    - – When Speedup(P) = k*P, for some constant k, 0 < k < 1

- Ideal Parallelism = WORK / CPL = T_1 / T_∞

  = Parallel Speedup on an unbounded (infinite) number of processors

Computation Graph for Recursive Tree approach to computing Array Sum in parallel



Assume greedy schedule, input array size S is a power of 2, each add takes 1 time unit

- WORK(G) = S-1, and CPL(G) = log2(S)
- Define T(S,P) = parallel execution time for Array Sum with size S on P processors
- Use upper bound T(S,P) <= WORK(G)/P + CPL(G) as a worst-case estimate

T(S,P) = WORK(G)/P + CPL(G) = (S-1)/P + log2(S) ⇒ Speedup(S,P) = T(S,1)/T(S,P) = (S-1)/((S-1)/P + log2(S))

How many processors should we use?

Define Efficiency(P) = Speedup(P)/ P = $T_1/(P * T_P)$

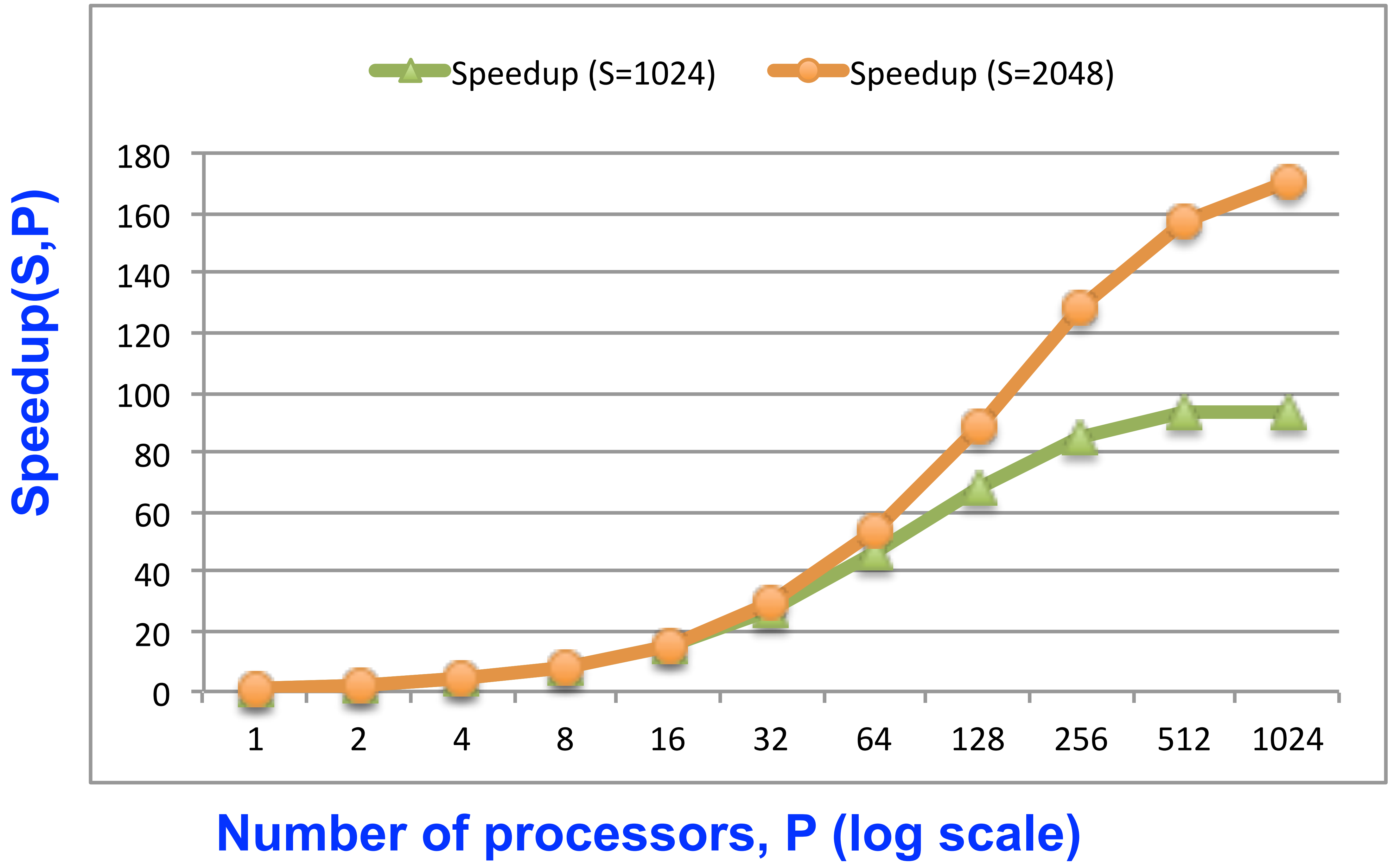
- —Processor efficiency --- figure of merit that indicates how well a parallel program uses available processors
- —For ideal executions without overhead, 1/P <= Efficiency(P) <= 1
- —Efficiency(P) = 1 (100%) is the best we can hope for

How many processors should we use?

What should be the minimum efficiency to determine how many processors we should use?

Array Sum: Speedup as a function of array size S and number of processors P

- Speedup(S,P) = T(S,1)/T(S,P) = (S-1)/((S-1)/P + $\log_2$(S))
- Asymptotically, Speedup(S,P) → (S-1)/$\log_2$S, as P → infinity



Amdahl's Law

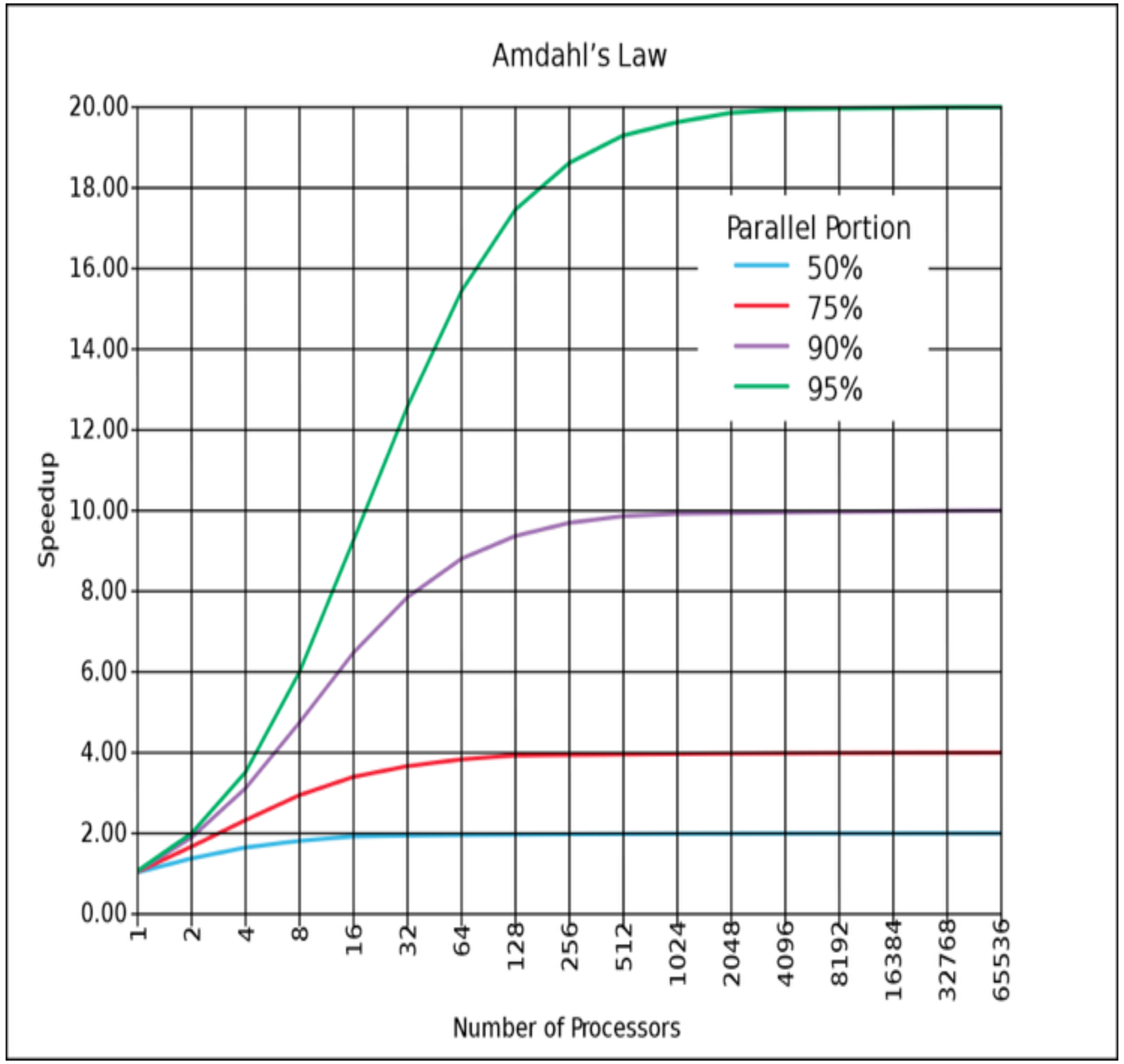
If $q \leq 1$ is the fraction of WORK in a parallel program that must be executed sequentially for a given input size S, then the best speedup that can be obtained for that program is $Speedup(S,P) \leq 1/q$.

Amdahl's Law

- Observation follows directly from critical path length lower bound on parallel execution time
  - CPL >= q * T(S,1)
  - T(S,P) >= q * T(S,1)
  - Speedup(S,P) = T(S,1)/T(S,P) <= 1/q

- Upper bound on speedup simplistically assumes that work can be divided into sequential and parallel portions
  - Sequential portion of WORK = q
    - also denoted as f_S (fraction of sequential work)
  - Parallel portion of WORK = 1-q
    - also denoted as f_p (fraction of parallel work)

Illustration of Amdahl's Law: Best Case Speedup as function of Parallel Portion



Announcements & Reminders

- No lab tomorrow
- Quiz #1 available today, due Friday, Jan. 31st at 11:59pm
- HW #1 due on Wednesday, Jan. 29th at 11:59pm
- IMPORTANT: Watch video & read handout for topic 2.1 for lecture on Friday