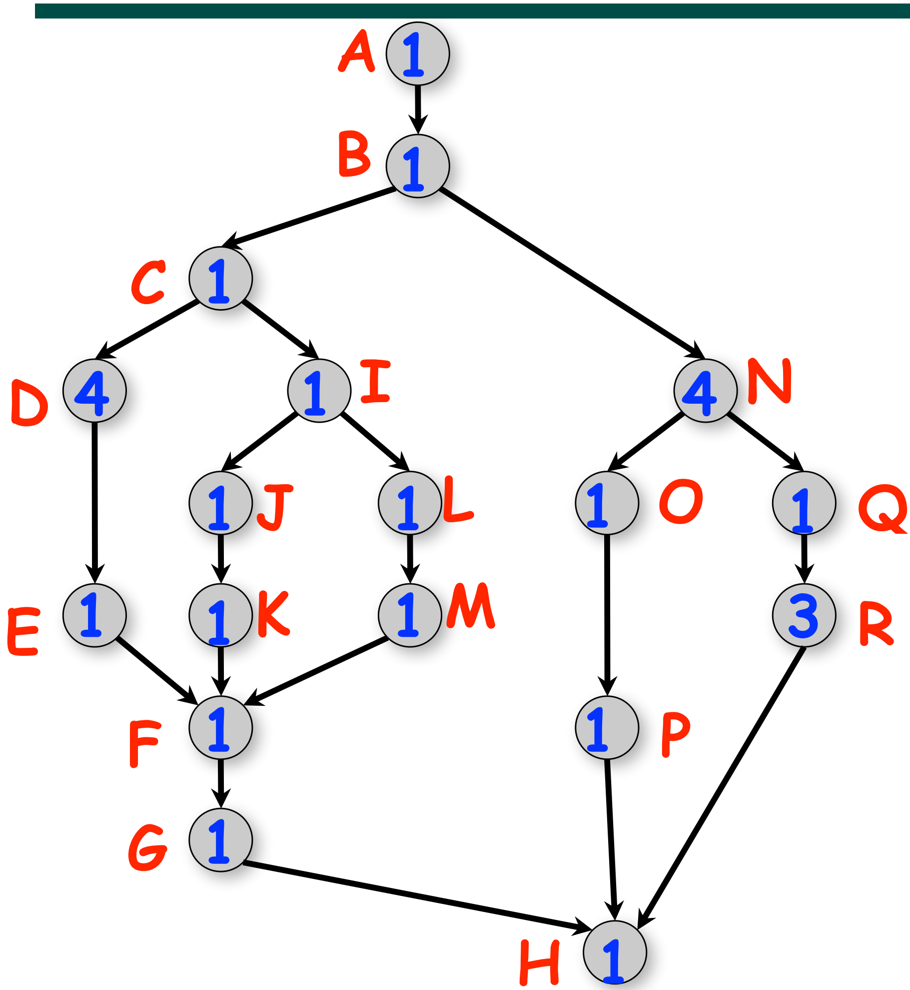
COMP 322: Fundamentals of Parallel Programming

Lecture 4: Parallel Speedup and Amdahl's Law

Instructors: Vivek Sarkar, Shams Iman
Department of Computer Science, Rice University
{vsarkar, shams}@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

  $\max(CPL, WORK/2) \le T_2 < CPL + WORK/2$

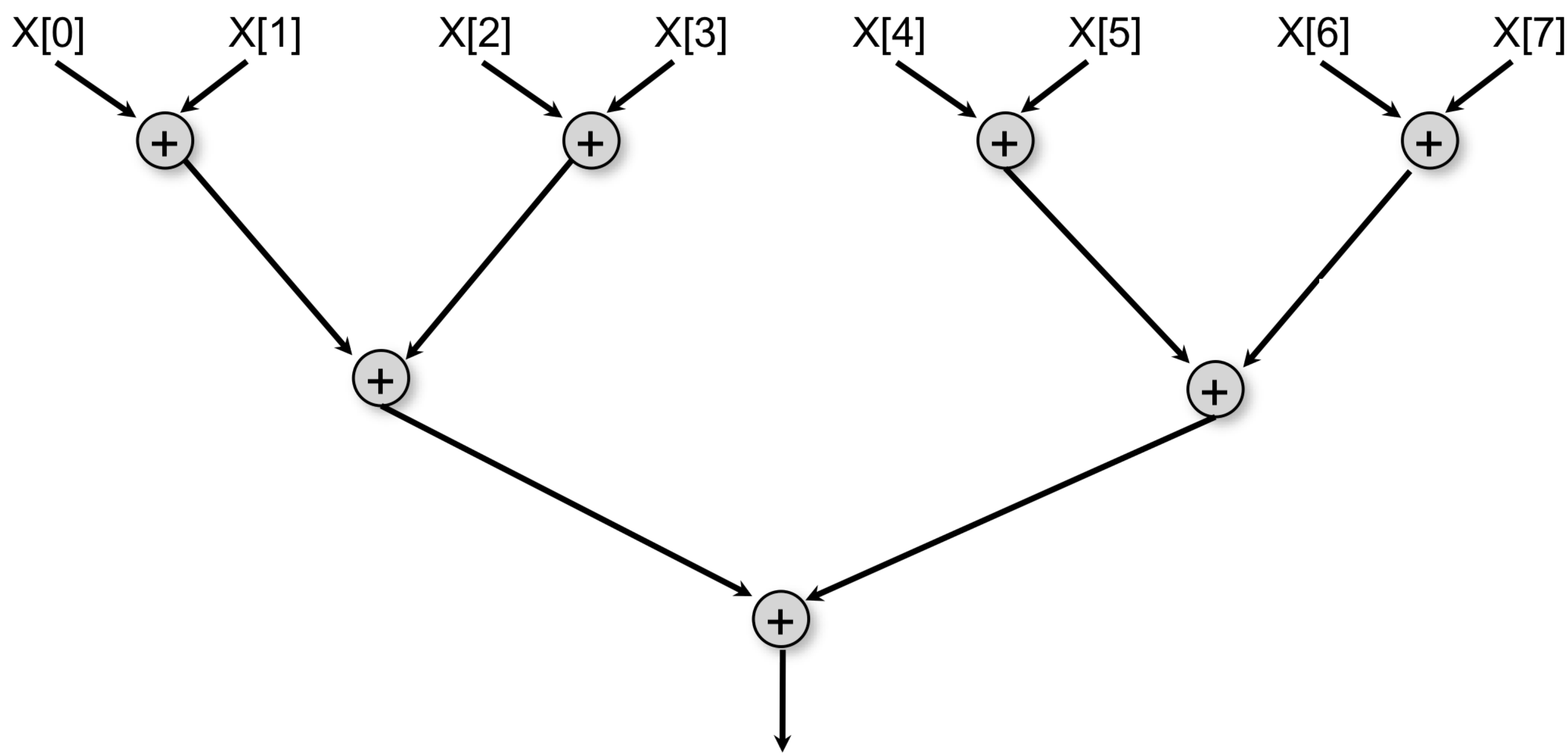
Parallel Speedup

- Define Speedup(P) = T_1 / T_P
  - —Factor by which the use of P processors speeds up execution time relative to 1 processor, for a fixed input size
  - —For ideal executions without overhead, 1 <= Speedup(P) <= P
  - —Linear speedup
    - When Speedup(P) = k*P, for some constant k, 0 < k < 1
- Ideal Parallelism

= WORK / CPL

= Parallel Speedup on an unbounded number of processors

Reduction Tree Schema for computing Array Sum in parallel



Assume input array size = S, and each add takes 1 unit of time:

- WORK(G) = S-1
- CPL(G) = log2(S)
- Use upper bound to estimate T_P = WORK(G)/P + CPL(G)

  = (S-1)/P + log2(S)

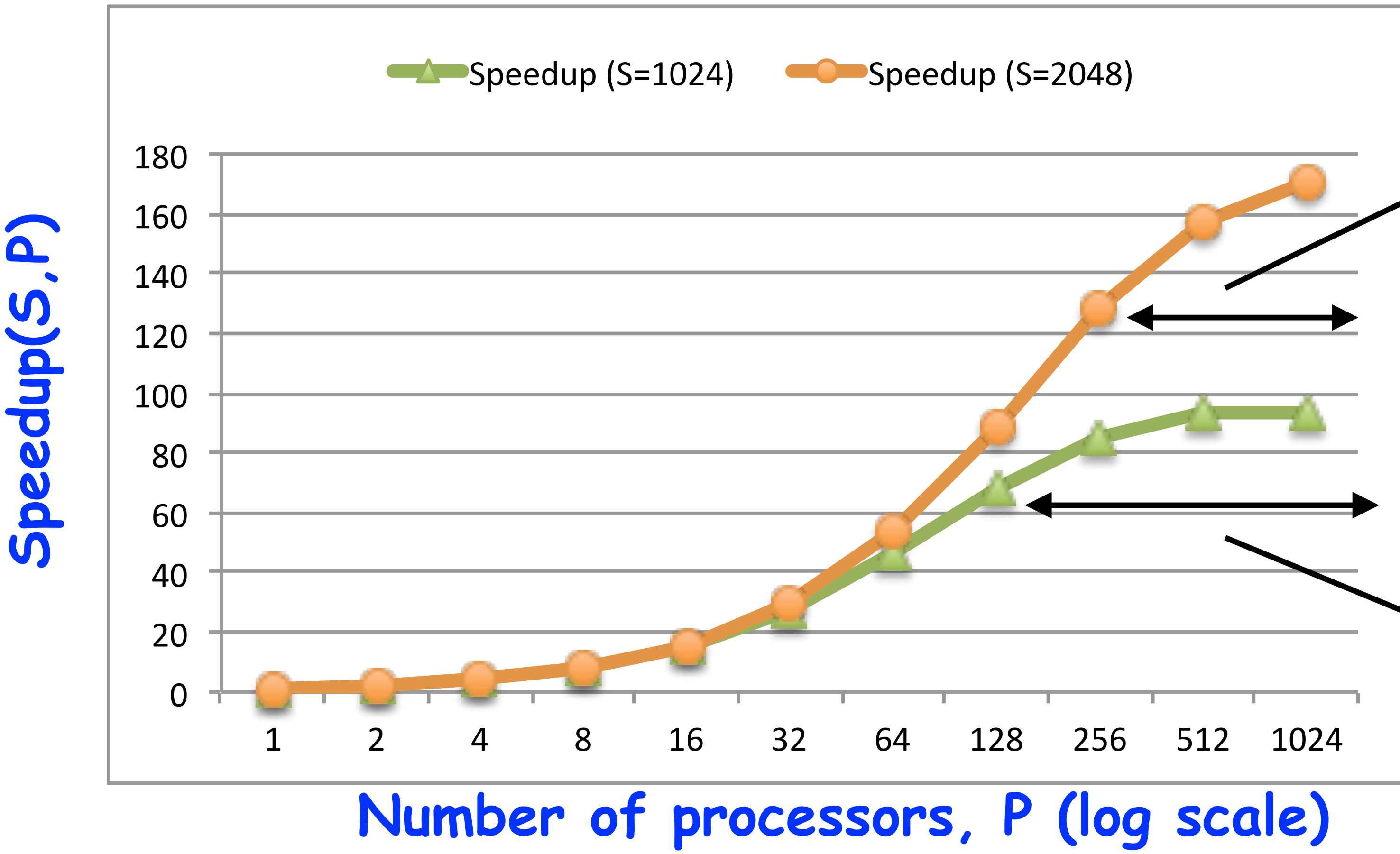
  - Within a factor of 2 of any greedy schedule's execution time

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
- Half-performance metric
  - $S_{1/2}$ = input size that achieves Efficiency(P) = 0.5 for a given P
  - Figure of merit that indicates how large an input size is needed to obtain efficient parallelism
  - A larger value of $S_{1/2}$ indicates that the problem is harder to parallelize efficiently
- How many processors to use?
  - Common goal: choose number of processors, P for a given input size, S, so that efficiency is at least 0.5

ArraySum: Speedup as function of array size, S, and number of processors, P

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



Efficiency(P) ≤ 0.5, for P ≥ 258
==> wasteful to use more than 256 processors for S=2048

Efficiency(P) ≤ 0.5, for P ≥ 128
==> wasteful to use more than 128 processors for S=1024

Amdahl's Law [1967]

- If $q \leq 1$ is the fraction of WORK in a parallel program that <u>must be executed sequentially</u> for a given input size S, then the best speedup that can be obtained for that program is Speedup(S,P) $\leq$ 1/q.
- 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
- This upper bound on speedup simplistically assumes that work in program 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)
- Computation graph is more general and takes dependences into account

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

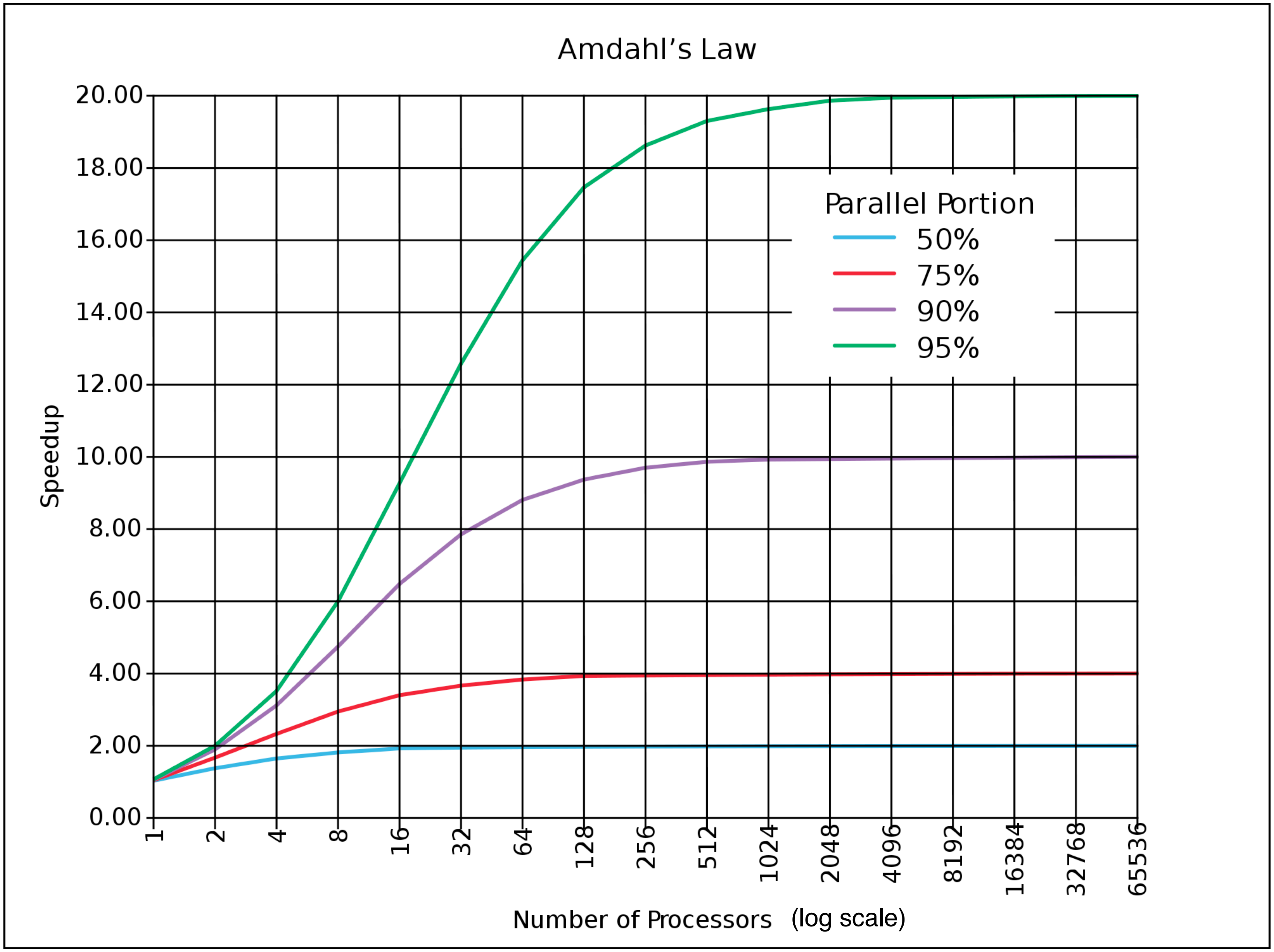


Figure source: http://en.wikipedia.org/wiki/Amdahl's law