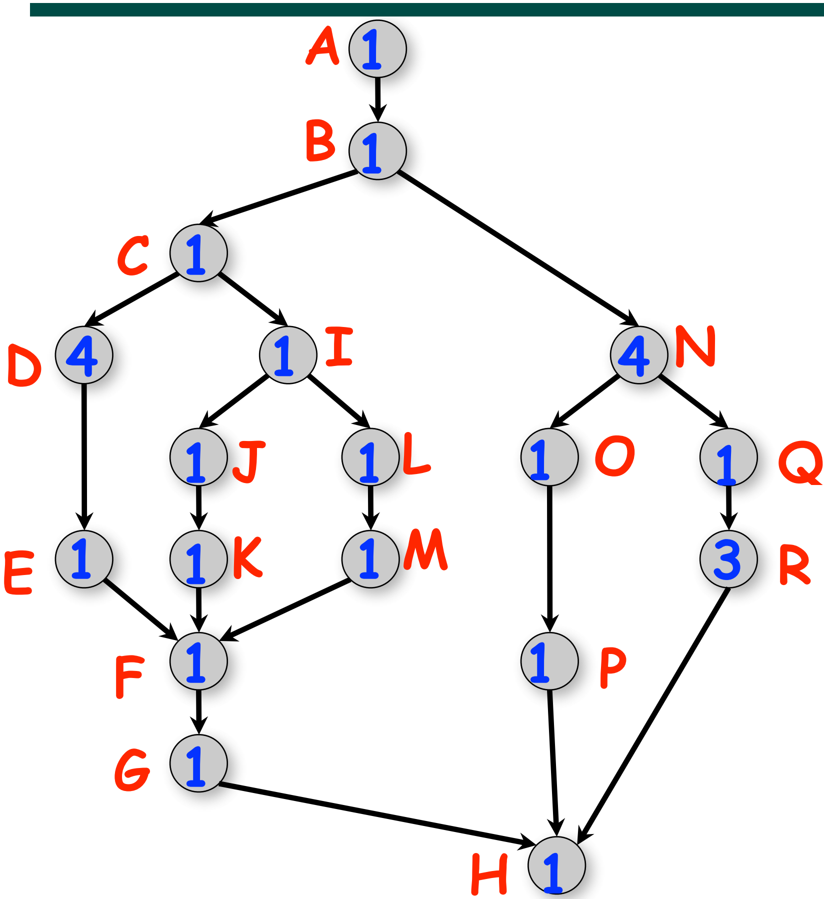
# COMP 322: Fundamentals of Parallel Programming

# Lecture 4: Parallel Speedup and Amdahl's Law

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

http://comp322.rice.edu

# One Possible Solution to Worksheet 3 (Multiprocessor Scheduling)




There are 4 idle slots in this schedule — can we do better than $T_2$ = 15 ?

| 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

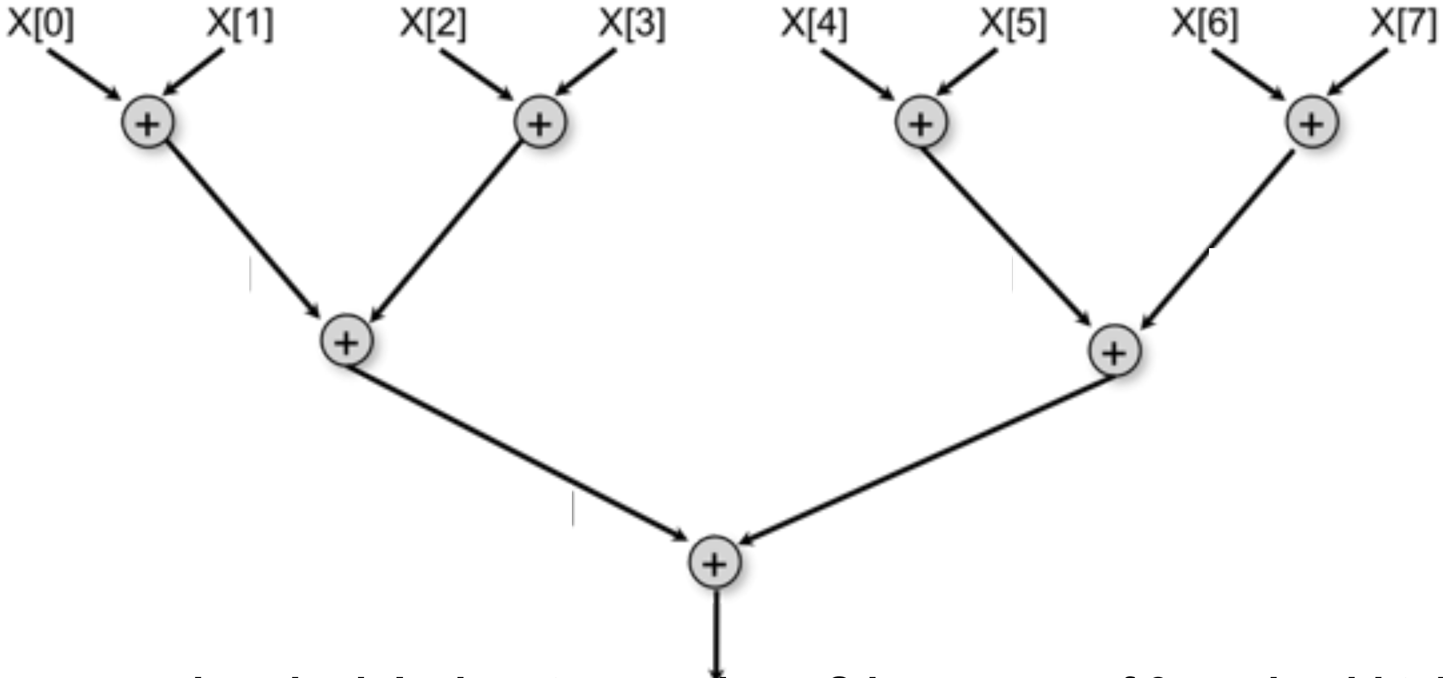
  $\max(CPL, WORK/2) <= T_2 < CPL + WORK/2$

# 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
    - —This is what you will see with abstract metrics, but these bounds may not hold when we start 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 = $T_1 / T_\infty$

  = 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

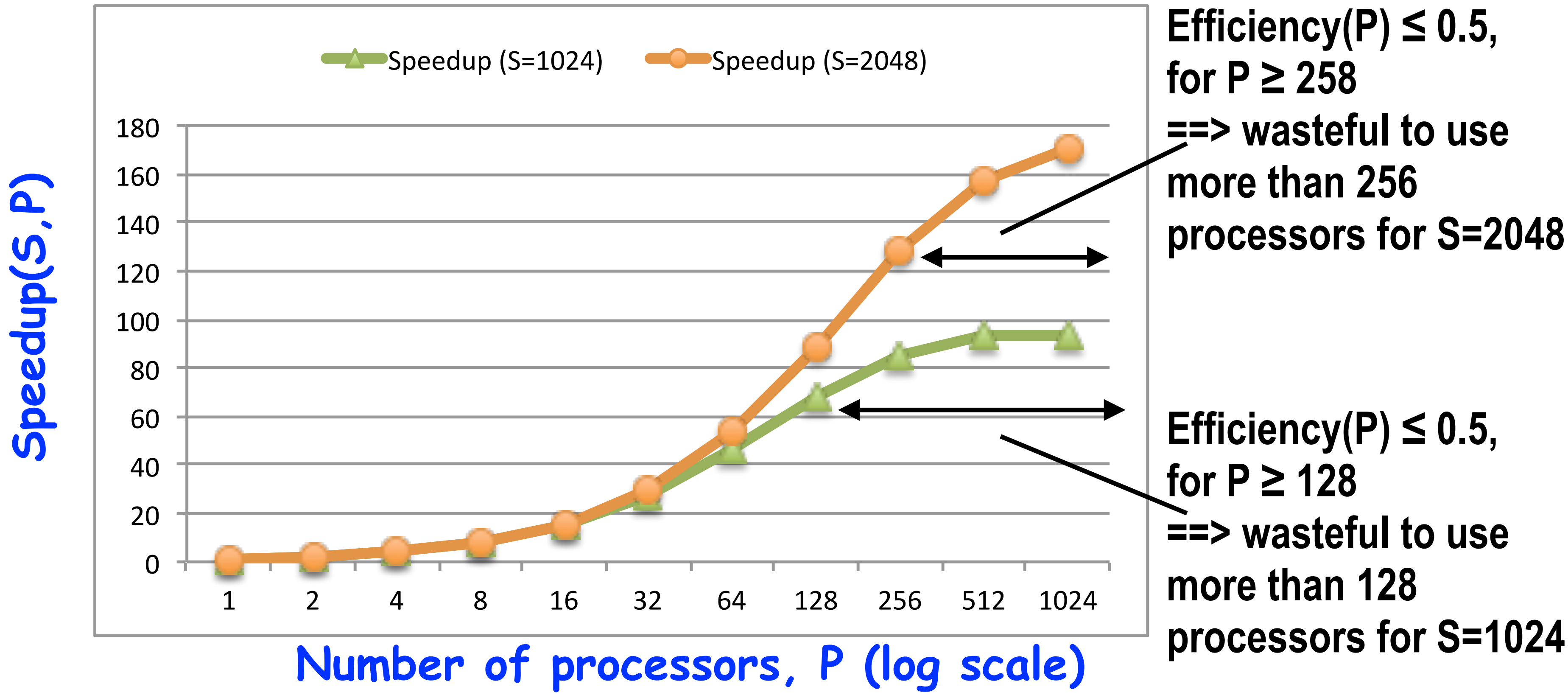
- WORK(G) = S-1, and CPL(G) = log2(S)
- Define T(S,P) = parallel execution time for Array Sum with size S on P processors
- Use upper bound T(S,P) <= WORK(G)/P + CPL(G) as a worst-case estimate
  - T(S,P) = WORK(G)/P + CPL(G) = (S-1)/P + log2(S)
- ⇒ Speedup(S,P) = T(S,1)/T(S,P) = (S-1)/((S-1)/P + log2(S))

# 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.
- 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 (50%)

# ArraySum: Speedup as 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 [1967]

- If $q \le 1$ 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) ≤ 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

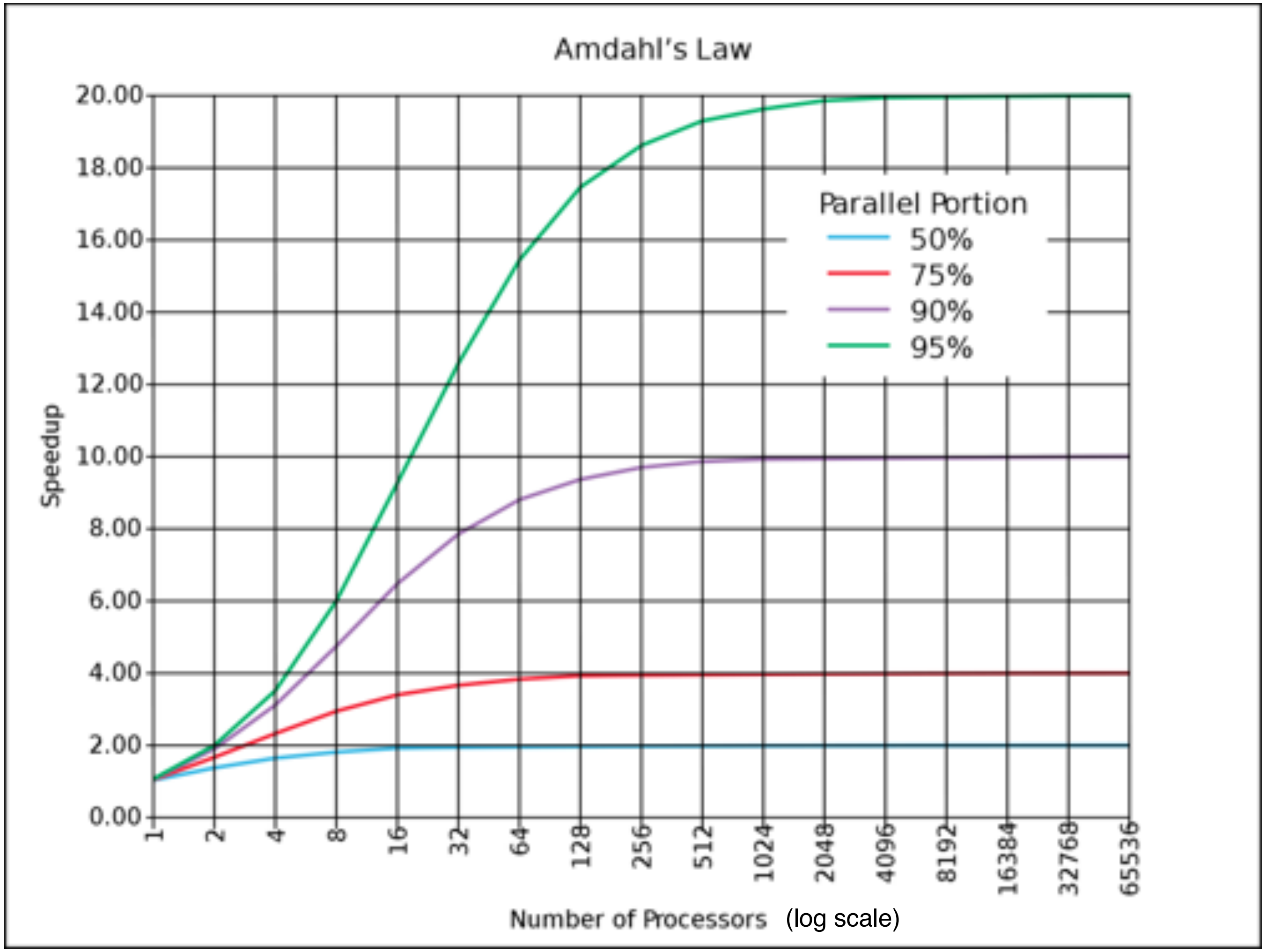



Figure source: http://en.wikipedia.org/wiki/Amdahl's law

# Announcements & Reminders

- **IMPORTANT:**
  - —Watch video & read handout for topic 2.1 for next lecture on Monday, Jan 22nd
- 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
- Midterm exam will be on Thursday, Feb 22nd. Time is TBD.
- 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.