
COMP 322: Fundamentals of Parallel Programming

Lecture 4: Parallel Speedup and Amdahl's Law

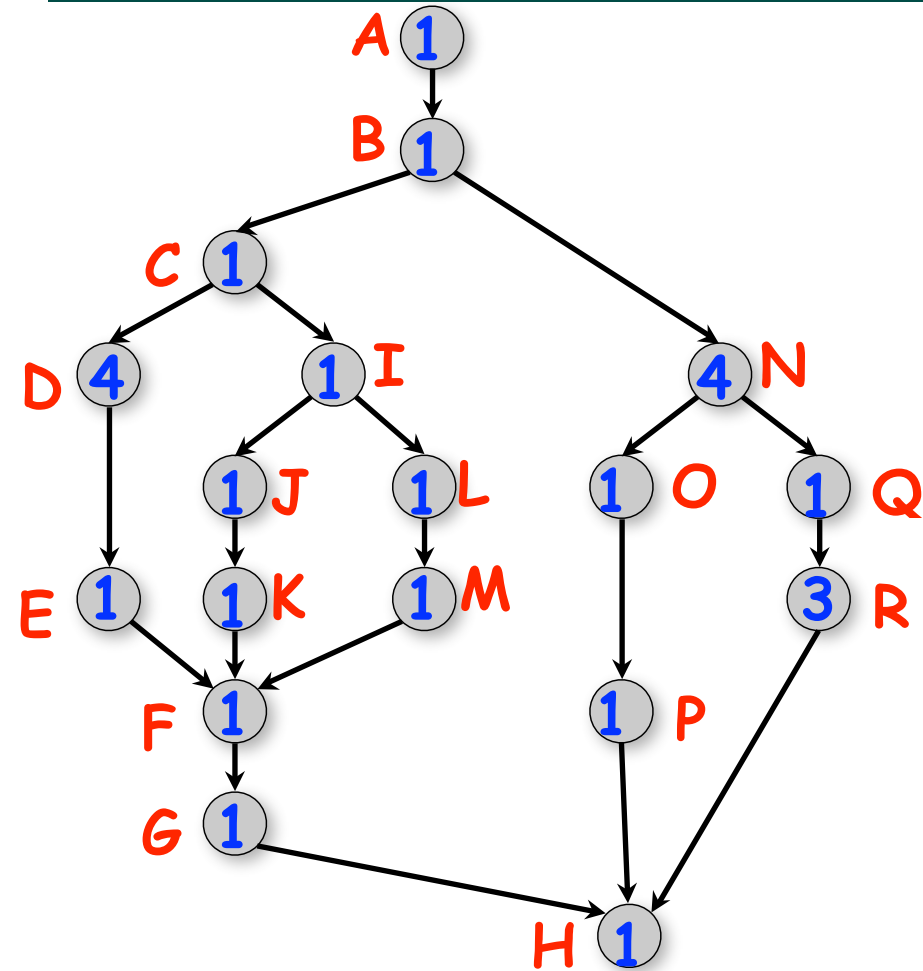
Vivek Sarkar, Eric Allen
Department of Computer Science, Rice University

Contact email: vsarkar@rice.edu

<https://wiki.rice.edu/confluence/display/PARPROG/COMP322>



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

$$\max(\text{CPL}, \text{WORK}/2) \leq T_2 < \text{CPL} + \text{WORK}/2$$

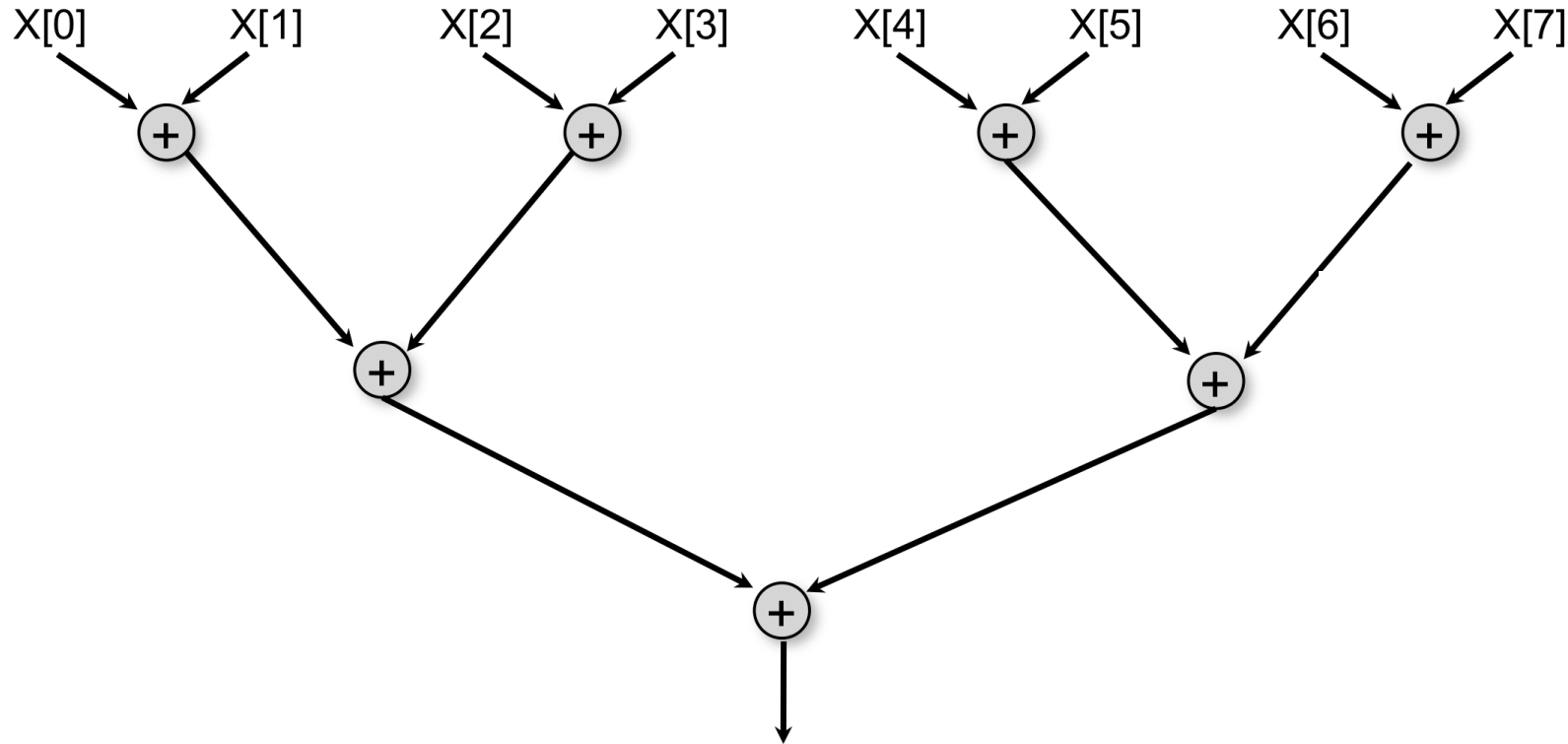


Parallel Speedup

- Define $\text{Speedup}(P) = T_1 / T_P$
 - 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, $1 \leq \text{Speedup}(P) \leq P$
 - Linear speedup
 - When $\text{Speedup}(P) = k \cdot P$, for some constant k , $0 < k < 1$
- Ideal Parallelism = 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:

- $WORK(G) = S-1$
- $CPL(G) = \log_2(S)$
- Estimate $T_p = WORK(G)/P + CPL(G) = (S-1)/P + \log_2(S)$
- Within a factor of 2 of any schedule's execution time



How many processors should we use?

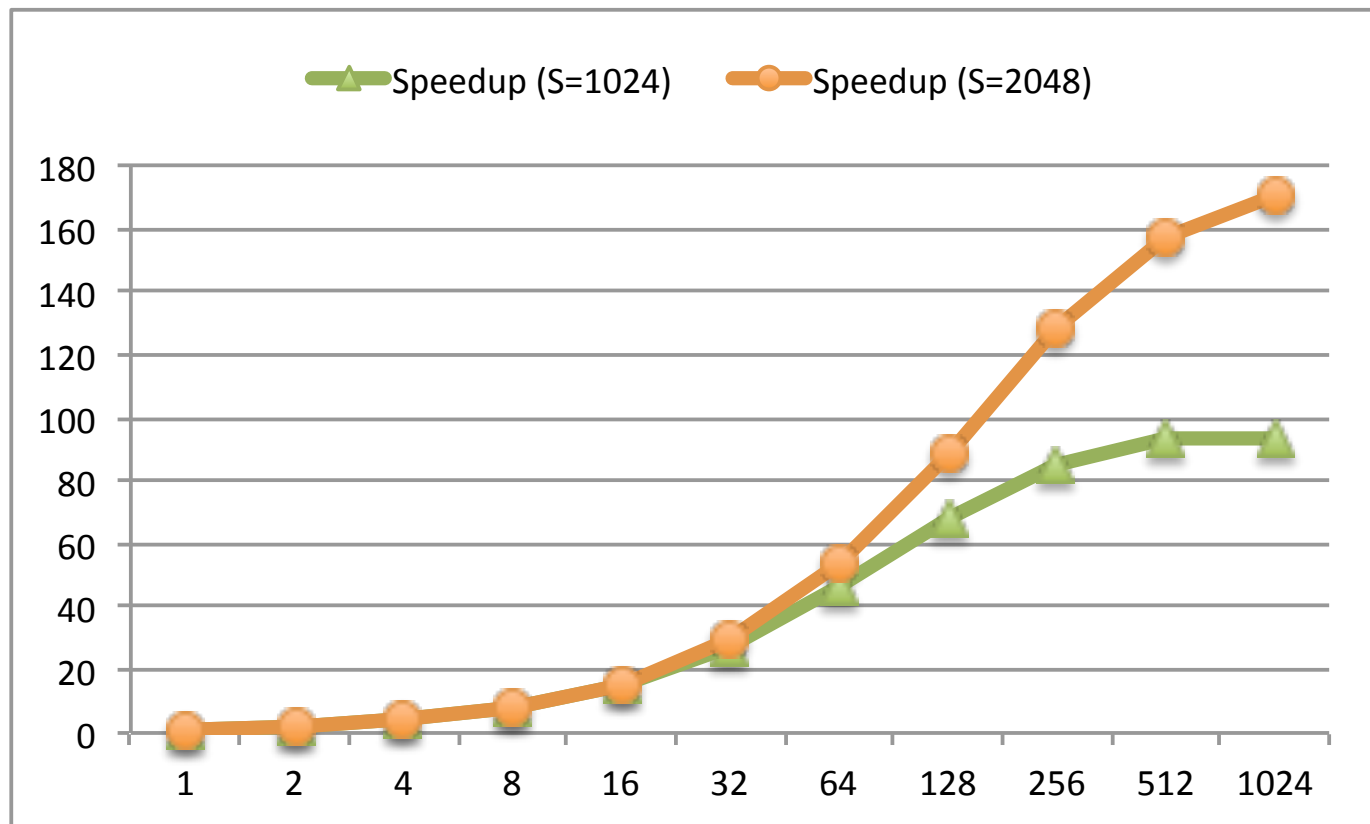
- **Define Efficiency(P) = Speedup(P)/ P = $T_1/(P * T_P)$**
 - Processor efficiency --- figure of merit that indicates how well a parallel program uses available processors
 - For ideal executions without overhead, $1/P \leq \text{Efficiency}(P) \leq 1$
- **Half-performance metric**
 - $S_{1/2}$ = input size that achieves $\text{Efficiency}(P) = 0.5$ 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

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

Speedup(S,P)



Number of processors, P (log scale)



Amdahl's Law [1967]

- 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 $\text{Speedup}(S,P) \leq 1/q$.
- Observation follows directly from critical path length lower bound on parallel execution time
 - $\text{CPL} \geq q * T(S,1)$
 - $T(S,P) \geq q * T(S,1)$
 - $\text{Speedup}(S,P) = T(S,1)/T(S,P) \leq 1/q$
- 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

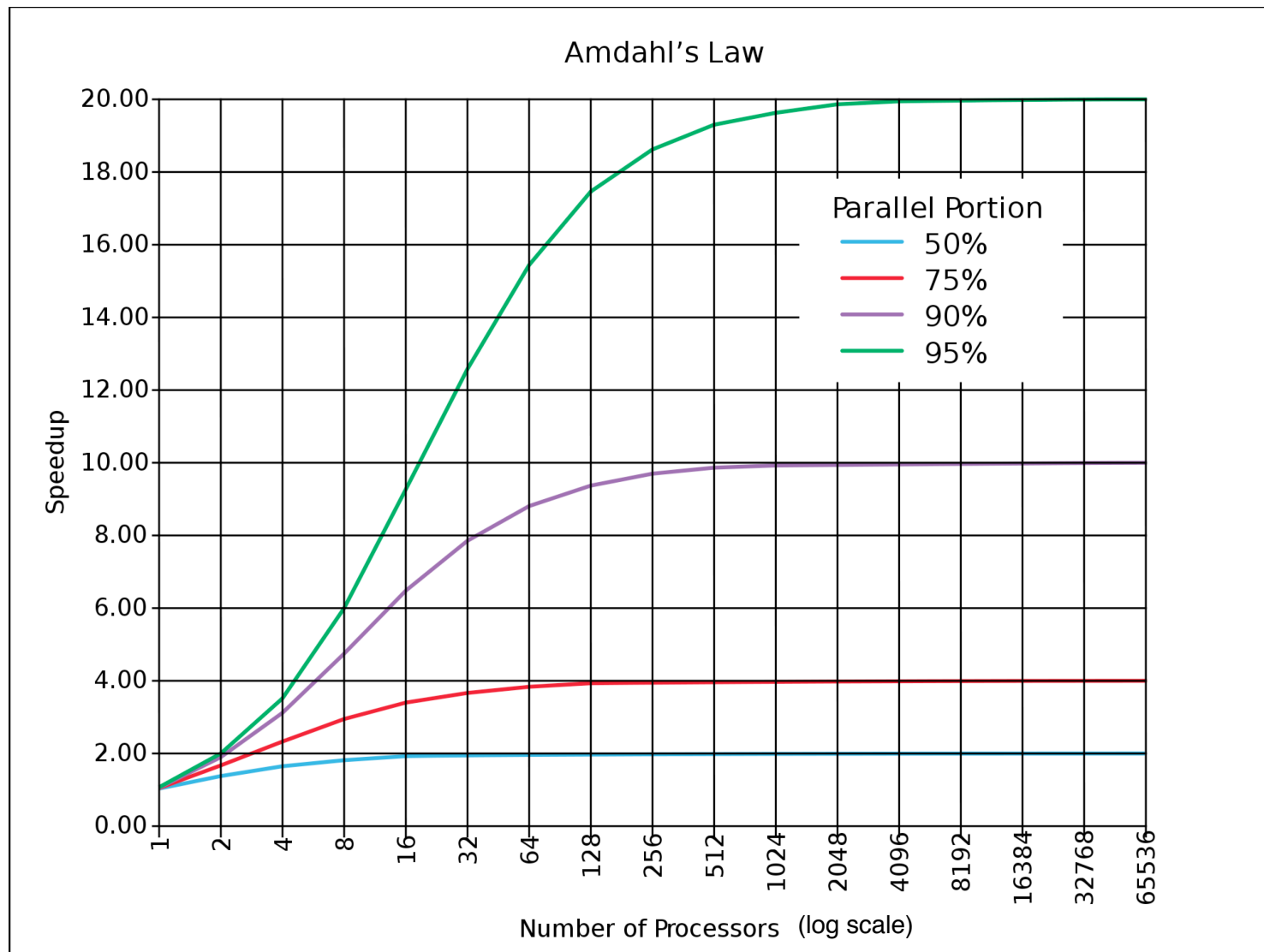


Figure source: [http://en.wikipedia.org/wiki/Amdahl's law](http://en.wikipedia.org/wiki/Amdahl's_law)

