Analytical Modeling of Parallel Programs

S. Oliveira

Sequential X Parallel

- Evaluation of Sequential Algorithm depends on 'execution time' which depends on the 'input size'

- Evaluation of Parallel Algorithm depends on 'execution time' which depends on (1) input size, and (2) Number of processing elements.

Total Parallel Overhead

- Parallel Program – Spends extra time in:
  - Interprocess Communication
  - Idling
  - Excess Computation
- What is Idling?

  Elements in parallel system become idle from:
    - Load Imbalance
    - Synchronization
    - Presence of Serial Components

- If all processing elements are not ready for synchronization at the same time, the ones that are ready sooner will be “IDLE” until all the rest are ready.
- What is Excess Computation?

When we compare the difference between the computation performed by the parallel program and the best serial program – it’s the excess computation overhead incurred by parallel program.

Performance Metrics

To determine the best algorithm, we have to examine the benefits of parallelism. A number of metrics have been used based on desired outcome. These metrics are:

- Execution Time
- Total Parallel Overhead
- Speedup
- Efficiency
- Cost

Execution Time

- Execution Time:
  - Serial Runtime: Time elapsed between the beginning and the end of its execution on a sequential computer.
  - Parallel Runtime: Time that elapses from the moment a parallel computation starts to the moment the last processing element finishes execution.
- Serial Runtime = T_s
- Parallel Runtime = T_p

Total Parallel Overhead

- Overhead Function/Total Overhead of a parallel system is the total time collectively spent by all the processing elements over and above that required by the fastest known sequential algorithm.

- pT_p is the total time spent in solving a problem summed over all processing elements.
  - $T_o = pT_p - T_s$

Speedup (S)

- Speedup is a measure that captures the relative benefit of solving a problem in parallel.

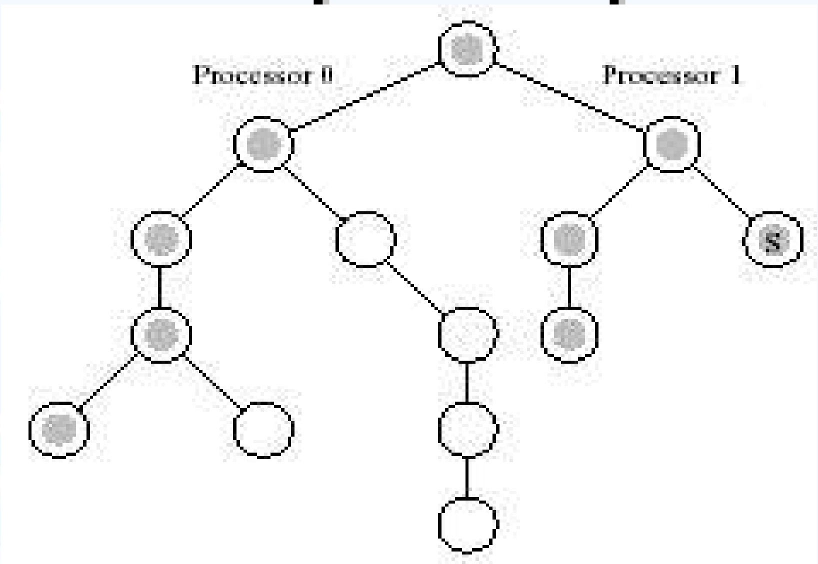
- Speedup = ratio of the serial runtime of the best sequential algorithm for solving a problem to the time taken by the parallel algorithm to solve the same problem on p processing elements.

Note that

- Speedup should not exceed the number of processing elements P.
- Speedup = P iff none of the processing elements spends more than T_s/P
- Speedup > P ➔ processing element spends less than time T_s/P solving the problem. This is called superlinear Speed up.

- Superlinear Speedup occurs if:
  - Work performed by a serial algorithm is greater than its parallel formulation or
  - Hardware features put the serial implementation at a disadvantage.
- Things that affect superlinearity:
  - Increased cache hit ratio resulting from lower problem size per processor -note superlinear speedup
  - Exploratory Decomposition: The Work performed by parallel and serial algorithms is different.

Superlinear Speedup Example



(It takes one t_c to visit one node)

Serial Formulation expands the entire tree => 14 t_c

Total Parallel work = 9 t_c (9 node expansions)

Parallel Time = 5 tc

Speedup (P =2) = 14 t_c / 5 t_c = 2.8 > 2 !

Efficiency

- Definition: Ratio of Speedup to the number of processing elements P. E=S/P.
- Efficiency is a measure of the fraction of time for which a processing element is usefully employed. (100% devoted to computation of the algorithm)

Cost

- Cost is the product of parallel runtime and the number of processing elements used. It reflects the sum of time that each processing element spends solving the problem.
- Cost of solving a problem on a single processing element is the execution time of the fastest known sequential algorithm.
- Cost optimal parallel system has efficiency $\Theta(1)$
- Cost = work = processor-time-work

Effect of Granularity on Performance

Assume P = processing elements and n=input data.

- To Increase Granularity ➔ we assign larger pieces of n to each P.
- Using fewer than the max # of P to execute a parallel algorithm ➔ scaling down parallel system in terms of P.
- As the number of P decreases, the computational time at each processor increases ➔ total parallel runtime increases but p x (parallel time) does not increase.
- Therefore if a parallel system with n processing elements is cost optimal, using p processing elements (p<n) to simulate n processing elements preserves cost-optimality.