Performance Metrics

Performance Metrics

- Analytical models for parallel computing are used to analyze and predict the performance of parallel algorithms and systems.
- This models provide insights into the scalability, efficiency, and resource utilization of parallel programs running on parallel computing architectures.
- Following are few analytical model commonly used:
    - Execution time models
    - Speecup models
    - Work span models
    - Queuing models
    - Communication models

Performance Metrics

- **Execution time:**

  A sequential algorithm is usually evaluated in terms of its execution time, expressed as a function of the size of its input.
  The execution of a parallel program depends not only on the input size but also on the number of processing elements used, and their relative computation and inter-process communication speeds.

Performance Metrics

- **Execution time:**

  A sequential algorithm is usually evaluated in terms of its execution time, expressed as a function of the size of its input.
  The execution of a parallel program depends not only on the input size but also on the number of processing elements used, and their relative computation and inter-process communication speeds.

  Sources of overhead:
  Inter-process interaction
  Idling
  Excess computation

Performance Metrics

- **Execution time:**
  The serial runtime of a program is the time elapsed between the beginning and the ending of its execution on a serial computer.

Performance Metrics

- **Execution time:**
  The serial runtime of a program is the time elapsed between the beginning and the ending of its execution on a serial computer.

$$T_s$$

Performance Metrics

- **Execution time:**

  The serial runtime of a program is the time elapsed between the beginning and the ending of its execution on a serial computer.

  $$T_S$$

  The parallel runtime is the time that elapses from the moment a parallel computation starts to the moment the last processing element finishes execution.

Performance Metrics

- **Execution time:**
  The serial runtime of a program is the time elapsed between the beginning and the ending of its execution on a serial computer.

$$T_s$$

The parallel runtime is the time that elapses from the moment a parallel computation starts to the moment the last processing element finishes execution.

$$T_p$$

Performance Metrics

- **Total parallel overhead:**
  The overhead incurred by a parallel program are encapsulated in a single expression referred to as the overhead function.
  It is defined as the total time collectively spent by all the processing elements over and above that required by the fastest known sequential algorithm for solving the same problem on a single processing element

Performance Metrics

- **Total parallel overhead:**

  The overhead incurred by a parallel program are encapsulated in a single expression referred to as the overhead function.

  It is defined as the total time collectively spent by all the processing elements over and above that required by the fastest known sequential algorithm for solving the same problem on a single processing element

$$T_o$$

Performance Metrics

- **Total parallel overhead:**

  The overhead incurred by a parallel program are encapsulated in a single expression referred to as the overhead function.
  It is defined as the total time collectively spent by all the processing elements over and above that required by the fastest known sequential algorithm for solving the same problem on a single processing element

$$T_o$$

$$T_0 = pT_p - T_s$$

Performance Metrics

- **Speedup:**
  Speed is the measure that captures the relative benefit of solving a problem in parallel.
  It is defined as the ratio of the time taken to solve a problem on a single processing element to the time required to solve the same problem on a parallel computer with ‘p’ identical processing elements.

Performance Metrics

- **Speedup:**
  Speed is the measure that captures the relative benefit of solving a problem in parallel.
  It is defined as the ratio of the time taken to solve a problem on a single processing element to the time required to solve the same problem on a parallel computer with ‘p’ identical processing elements.

$$S$$

Performance Metrics

- **Speedup:**
  Speed is the measure that captures the relative benefit of solving a problem in parallel.
  It is defined as the ratio of the time taken to solve a problem on a single processing element to the time required to solve the same problem on a parallel computer with ‘p’ identical processing elements.

$$S$$

$$S = \frac{T_s}{T_p}$$

Performance Metrics

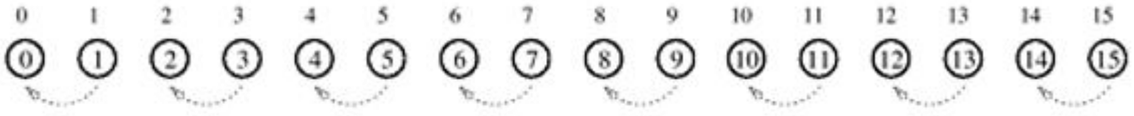
- **Speedup Example:**

  Adding ‘n’ number using
  ‘n’ processing elements.

Performance Metrics

- **Speedup Example:**

  Adding ‘n’ number using ‘n’ processing elements.

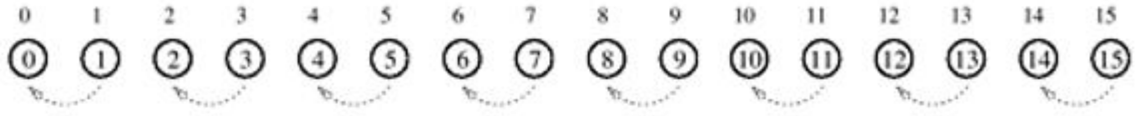


(a) Initial data distribution and the first communication step

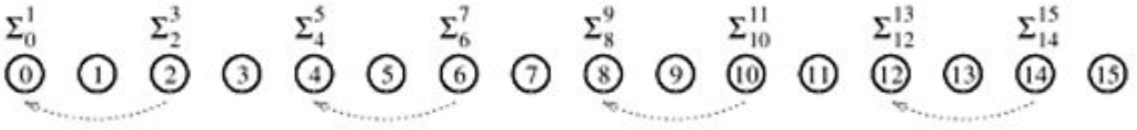
Performance Metrics

- **Speedup Example:**

  Adding ‘n’ number using ‘n’ processing elements.



(a) Initial data distribution and the first communication step

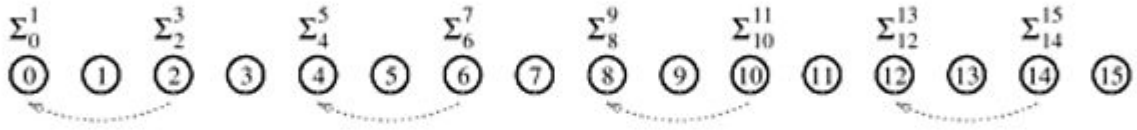


(b) Second communication step

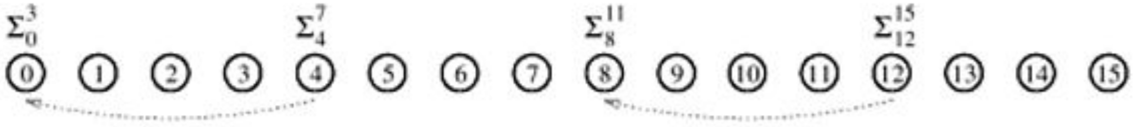
Performance Metrics

- **Speedup Example:**

  Adding 'n' number using 'n' processing elements.



(b) Second communication step

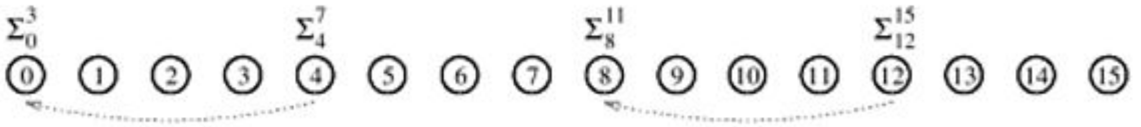


(c) Third communication step

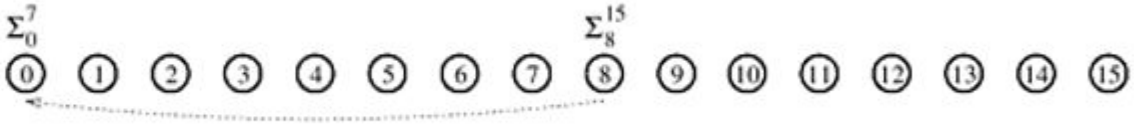
Performance Metrics

- **Speedup Example:**

  Adding ‘n’ number using ‘n’ processing elements.



(c) Third communication step

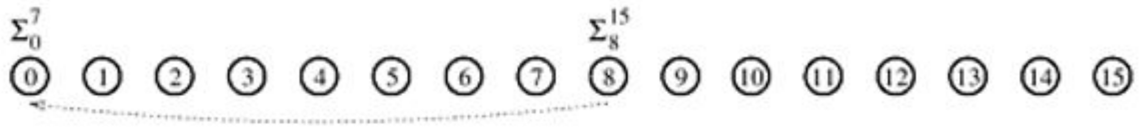


(d) Fourth communication step

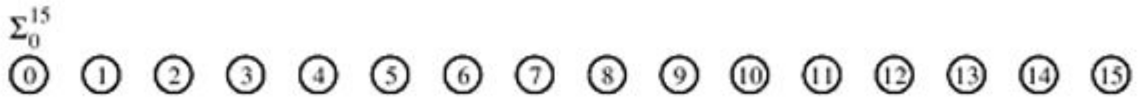
Performance Metrics

- **Speedup Example:**

  Adding ‘n’ number using ‘n’ processing elements.



(d) Fourth communication step



(e) Accumulation of the sum at processing element 0 after the final communicatior

Performance Metrics

- **Speedup Example:**

  Adding ‘n’ number using ‘n’ processing elements.

Serial execution time : $T_s = O(n)$

Performance Metrics

- **Speedup Example:**

  Adding 'n' number using 'n' processing elements.

Serial execution time	:	$T_s = O(n)$
Parallel execution time	:	$T_p = O(log_2 n))$

Performance Metrics

- **Speedup Example:**

  Adding 'n' number using 'n' processing elements.

Serial execution time : $T_s = O(n)$

Parallel execution time : $T_p = O(\log_2 n))$

Speed up : $S = \frac{T_s}{T_p} = O(\frac{n}{\log_2 n})$

Performance Metrics

- **Speedup - Amdhal's Law:**

  In computer architecture Amdhal's law is a formula which gives theoretical speedup in latency of the execution task of a fixed workload that can be expected of a system whose resources are improved.

$$T = (1 - p)T + pT$$

Performance Metrics

- **Speedup - Amdhal's Law:**

  In computer architecture Amdhal's law is a formula which gives theoretical speedup in latency of the execution task of a fixed workload that can be expected of a system whose resources are improved.

$$T = (1 - p)T + pT$$

$$S = \frac{T_s}{T_p}$$

Performance Metrics

- **Speedup - Amdhal's Law:**

  In computer architecture Amdhal's law is a formula which gives theoretical speedup in latency of the execution task of a fixed workload that can be expected of a system whose resources are improved.

$$T = (1 - p)T + pT$$

$$S = \frac{T_s}{T_p}$$

$$S = \frac{1}{(1 - p) + p}$$

Performance Metrics

- **Speedup - Amdhal's Law:**

  In computer architecture Amdhal's law is a formula which gives theoretical speedup in latency of the execution task of a fixed workload that can be expected of a system whose resources are improved.

$$T = (1 - p)T + pT$$

$$S = \frac{T_s}{T_p}$$

$$S = \frac{1}{(1 - p) + p}$$

$$S = \frac{1}{(1 - p) + \frac{p}{s}}$$

Performance Metrics

- **Speedup - Amdhal's Law (Example):**

  A program needs 20 hours using a single processor core.

Performance Metrics

- **Speedup - Amdhal's Law (Example):**

  A program needs 20 hours using a single processor core.
  A particular part in the program requires 1 hour to complete and cannot be parallelized.

Performance Metrics

- **Speedup - Amdhal's Law (Example):**

  A program needs 20 hours using a single processor core.
  A particular part in the program requires 1 hour to complete and cannot be parallelized.
  Remaining part (i.e. 20 - 1 = 19 hour work) can be parallelized.

Performance Metrics

- **Speedup - Amdhal's Law (Example):**

  A program needs 20 hours using a single processor core.
  A particular part in the program requires 1 hour to complete and cannot be parallelized.
  Remaining part (i.e. 20 - 1 = 19 hour work) can be parallelized.

$$S = \frac{1}{(1-p) + \frac{p}{s}}$$

Performance Metrics

- **Speedup - Amdhal's Law (Example):**

  A program needs 20 hours using a single processor core.
  A particular part in the program requires 1 hour to complete and cannot be parallelized.
  Remaining part (i.e. 20 - 1 = 19 hour work) can be parallelized.

$$S = \frac{1}{(1-p) + \frac{p}{s}}$$

$$S = \frac{20}{1 + \frac{19}{s}} \; or \; S = \frac{1}{(1-0.95) + \frac{0.95}{s}}$$

Performance Metrics

- **Speedup - Amdhal's Law (Example):**

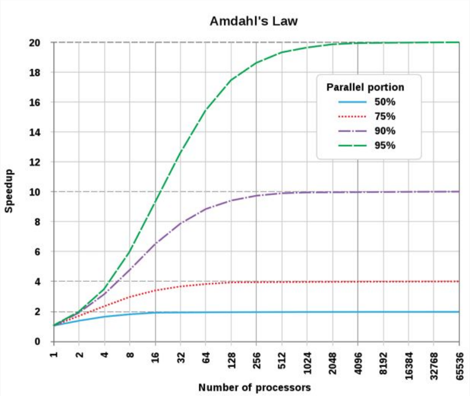
  A program needs 20 hours using a single processor core.
  A particular part in the program requires 1 hour to complete and cannot be parallelized.
  Remaining part (i.e. 20 - 1 = 19 hour work) can be parallelized.

$$S = \frac{20}{1 + \frac{19}{s}} \ or\ S = \frac{1}{(1 - 0.95) + \frac{0.95}{s}} \quad S = \frac{20}{1 + 0} \ or\ S = \frac{1}{0.05 + 0} \ or\ S = 20$$

Performance Metrics

- **Speedup - Amdhal's Law (Example):**

  A program needs 20 hours using a single processor core.
  A particular part in the program requires 1 hour to complete and cannot be parallelized.
  Remaining part (i.e. 20 - 1 = 19 hour work) can be parallelized.



$$S = \frac{20}{1+0} \; or \; S = \frac{1}{0.05+0} \; or \; S = 20$$

Performance Metrics

- **Superlinear Speedup:**

  Caches
  Exploratory decomposition

Performance Metrics

- **Superlinear Speedup:**

  Caches
  Exploratory decomposition

For example:
A problem of size ‘W’ | Problem is memory bound
64kb cache on one processor
Cache latency 2ns | DRAM latency 100ns
Remote DRAM latency 400ns
Takes 1 FLOP per memory access

Performance Metrics

- **Superlinear Speedup:**

  Caches
  Exploratory decomposition

For example:
A problem of size ‘W’ | Problem is memory bound
64kb cache on one processor
Cache latency 2ns | DRAM latency 100ns
Remote DRAM latency 400ns
Takes 1 FLOP per memory access

Single processor system:
Problem size = W |
Memory access time (cache hit 80%):
2 x 0.8 +100 x 0.2 = 21.6ns
Processing rate:
1000 / 21.6 = 46.3 MFLOPS

Performance Metrics

- **Superlinear Speedup:**

  Caches
  Exploratory decomposition

For example:
A problem of size 'W' | Problem is memory bound
64kb cache on one processor
Cache latency 2ns | DRAM latency 100ns
Remote DRAM latency 400ns
Takes 1 FLOP per memory access

Single processor system:
Problem size = W
Memory access time (cache hit 80%):
2 x 0.8 +100 x 0.2 = 21.6ns
Processing rate:
1000 / 21.6 = 46.3 MFLOPS

Two processor system:
Problem size = W / 2
Memory access time (cache hit 90%):
2 x 0.9 +100 x 0.08 + 400 x 0.02 = 17.8 ns
Processing rate (**2** procs simultaneously):
(**2** x 1000) / 17.8 = 112.36 MFLOPS

Performance Metrics

- **Efficiency:**

  Only an ideal parallel system containing ‘p’ processors can deliver a speedup equal to ‘p’. In practice, ideal behaviour is not achieved because while executing a parallel program, the processing elements cannot devote 100% of their time to the computation of the algorithm.

Performance Metrics

- **Efficiency:**

  Only an ideal parallel system containing ‘p’ processors can deliver a speedup equal to ‘p’.
  In practice, ideal behaviour is not achieved because while executing a parallel program, the processing elements cannot devote 100% of their time to the computation of the algorithm.
  Efficiency is the measure of the fraction of time for which a processing element is usefully employed; it is defined as the ratio of speedup to the number of processing elements.
  In an ideal parallel system, speedup is equal to ‘p’ and efficiency is equal to 1.

Performance Metrics

- **Efficiency:**

  Only an ideal parallel system containing ‘p’ processors can deliver a speedup equal to ‘p’. In practice, ideal behaviour is not achieved because while executing a parallel program, the processing elements cannot devote 100% of their time to the computation of the algorithm. Efficiency is the measure of the fraction of time for which a processing element is usefully employed; it is defined as the ratio of speedup to the number of processing elements. In an ideal parallel system, speedup is equal to ‘p’ and efficiency is equal to 1.

$$E$$

Performance Metrics

- **Efficiency:**

  Only an ideal parallel system containing ‘p’ processors can deliver a speedup equal to ‘p’. In practice, ideal behaviour is not achieved because while executing a parallel program, the processing elements cannot devote 100% of their time to the computation of the algorithm. Efficiency is the measure of the fraction of time for which a processing element is usefully employed; it is defined as the ratio of speedup to the number of processing elements.
  In an ideal parallel system, speedup is equal to ‘p’ and efficiency is equal to 1.

E

$$E = \frac{S}{p} \; or \; \frac{T_s}{pT_p}$$

Performance Metrics

- **Cost:**

  Cost reflects the sum of work that each processing element spends solving the problem.
  It is defined as the product of the parallel runtime and the number of processing elements used.

Performance Metrics

- **Cost:**

  Cost reflects the sum of work that each processing element spends solving the problem.
  It is defined as the product of the parallel runtime and the number of processing elements used.

$$C \qquad\qquad C = pT_p$$

Performance Metrics

- **Cost:**

  Cost reflects the sum of work that each processing element spends solving the problem.
  It is defined as the product of the parallel runtime and the number of processing elements used.

  $$C \qquad\qquad C = pT_p$$

  Efficiency can be expressed as the ratio of the execution time of the fastest known sequential algorithm to the cost of solving the same problem with ‘p’ processing elements.

Performance Metrics

- **Cost:**

  Cost reflects the sum of work that each processing element spends solving the problem.
  It is defined as the product of the parallel runtime and the number of processing elements used.

  C $$C = pT_p$$

  Efficiency can be expressed as the ratio of the execution time of the fastest known sequential algorithm to the cost of solving the same problem with ‘p’ processing elements.

  $$E = \frac{T_s}{C}$$

Performance Metrics

- Very often, programs are designed and tested for smaller problems on fewer processing elements.
- The real problems are intended to solve are much larger, and the machines contain larger number of processing elements.
- Code development is simplified by using scaled-down versions of the machine and the problem, their performance and correctness is much more difficult to establish based on scaled down systems.
- We have to apply difference techniques for evaluating the scalability of the parallel programs using analytical tools.

"One of the best programming skills you can have is knowing when to walk away for a while."

Questions?

Thanks!

Contact:

Vineet More
vineet.more.bfab@gmail.com
[Make sure to use HPCAP23 as the subject line for your queries]

LinkedIn Handle:
https://www.linkedin.com/in/vineet-more-c-programmer/