

Performance metrics for parallel systems

V. Sundararajan
Thanks to Dr. S.S. Kadam
C-DAC, Pune

Purpose

- To determine best parallel algorithm
- Evaluate hardware platforms
- Examine the benefits from parallelism

Execution time

- **Serial runtime of a program**
 - Time elapsed between the beginning and end of execution on a sequential computer
 - Usually denoted by T_s
- **Parallel runtime**
 - Time elapsed from start of the parallel computation to end of execution by the last processing element (PE)
 - Usually denoted by T_p

Total parallel overhead

- Total time spent by all the PEs over and above that required by the fastest known sequential algorithm on a single PE for solving the same problem
- Represented by an overhead function (T_o)
 - Total time spent in solving a problem using p PEs is pT_p
 - Time spent for performing useful work is T_s
 - The remainder is overhead T_o given by
$$T_o = pT_p - T_s$$

Speedup

- Measures performance gain achieved by parallelizing a given application over sequential implementation
 - Captures relative benefit of solving a problem in parallel
- Defined as ratio of time taken to solve a problem on a single PE to time required to solve the same problem on a parallel computer with p PEs, that is

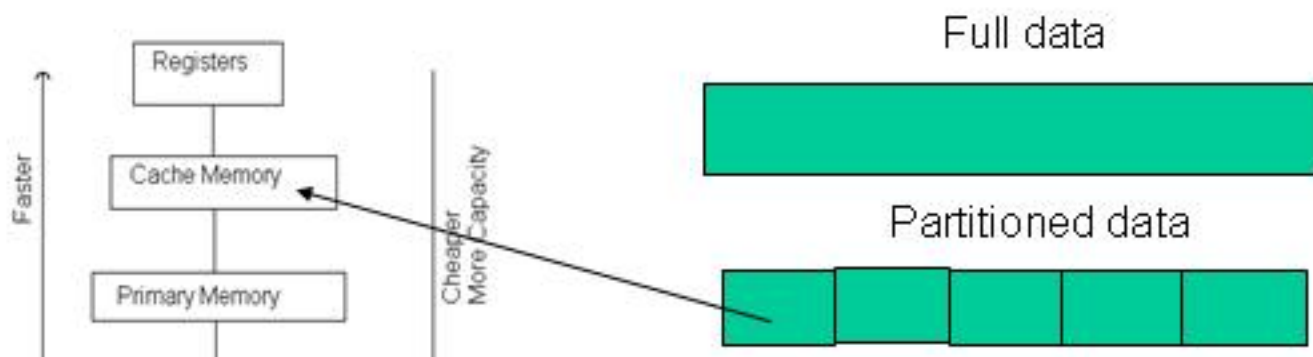
$$S = T_s/T_p$$

Speedup: Some observations

- Serial runtime of the best sequential algorithm should be considered while computing the speedup ratio
- The p PEs used are assumed to be identical to the one used by the sequential algorithm
- Speedup, in theory, cannot exceed number of PEs, p
 - Speedup greater than p possible only if each PE spends less than T_S/p time solving the problem
 - Single PE could then be time-slided to achieve faster serial program
 - Which contradicts assumption of fastest serial program as basis for speedup

Super-linear speedup

- A phenomenon where a speedup greater than p is sometimes observed in certain parallel applications
- Happens when
 - Work performed by a serial algorithm is greater than its parallel formulation
 - The hardware features put the serial implementation at a disadvantage
 - Data for a problem too large to fit into cache of single PE
 - In parallel implementation, individual data-partitions maybe small enough to fit individual caches of the PEs



- **Measures fraction of time for which a PE is usefully employed**
- **Defined as ratio of the speedup to the number of PEs**
$$E = S/p = T_S / (T_p * p)$$
- **In ideal parallel systems**
 - Speedup is equal to p , and
 - Efficiency is equal to one
- **In practice**
 - Speedup is less than p , and
 - Efficiency is between zero and one

Scalability of parallel systems

- **The efficiency of a parallel program can be written as**

$$E = \frac{S}{p} = \frac{T_S}{pT_P}$$

$$T_O = pT_P - T_S$$

or
$$E = \frac{1}{1 + \frac{T_O}{T_S}}$$

- **The total overhead function T_O is an increasing function of p**
- **For a given problem size**
 - Value of T_S remains constant
 - As we increase number of PEs, T_O increases
 - The overall efficiency of the parallel program goes down
- **This is the case for all parallel programs**

- Total overhead function T_o is a function of both problem size T_s and the number of processing elements p
- In many cases, T_o grows sub-linearly with respect to T_s
 - In such cases, efficiency increases if problem size is increased keeping number of PEs constant
 - One can simultaneously increase the problem size and number of processors to keep efficiency constant
 - Such systems are called *scalable* parallel systems

Amdahl's law

- If f is fraction of sequential part in the program, Amdahl's law states that the
Speedup = $1/[f + (1-f) / P]$
- $S \leq 1 / [f + (1-f)/p]$ f - fraction of sequential part, S - speedup, p - no of processors

Example, if $f = 0.1$, $P = 10$ PEs

$$S = \frac{1}{0.1 + (0.9) / 10}$$

$$= \frac{1}{0.1 + (0.09)} \cong 5$$

As $P \longrightarrow \infty$ $S \longrightarrow 10$

Observations of the Amdahl's law

- **Small number of sequential operations can significantly limit speedup achievable by a parallel computer**
 - **This is one of the stronger arguments against parallel computers**
- **Amdahl's law arguments serve as a way of determining whether an algorithm is a good candidate for parallelisation**
 - **This argument doesn't take in to account the problem size**
 - **In most applications as data size increases, the sequential part diminishes**

Gustafson's law

Rather than assuming that the problem size is fixed, assume that the parallel execution time is fixed. In increasing the problem size, Gustafson also makes the case that the serial section of the code does not increase as the problem size.

Scaled Speedup Factor

The scaled speedup factor becomes

$$S_s(n) = (s + np) / (s+p) = s + np = n + (1 - n)s$$

called *Gustafson's law*.

Example

Suppose a serial section of 5% and 20 processors; the speedup according to the formula is $0.05 + 0.95(20) = 19.05$ instead of 10.26 according to Amdahl's law. (Note, however, the different assumptions.)

- **Performance metrics for parallel systems**

Introduction to Parallel Computing, Ananth Grama, et. al., Pearson Education, Ltd., Second Edition, 2004