Speedup & Efficiency

Kudang B. Seminar

Definitions of *Speedup*

$$S_N = Ts / Tp$$

- S_N = Speedup of N processors
- Ts = The time taken to run the *fastest serial algorithm* on one processor
- Tp = The time taken by a parallel algorithm on N processors

Definitions of *Efficiency*

$$E_N = S_N / N$$

- E_N = Efficiency of N processors
- S_N = Speedup of N processors
- N = the number of parallel processors

Factors that limit speedup

- Software Overhead
- Load Balancing
- Communication Overhead
- Amdahl's Law

Software Overhead

- Even with a completely equivalent algorithm, software overhead arises in the concurrent implementation. (e.g. there may be additional index calculations necessitated by the manner in which data are "split up" among processors.)
- There is generally more lines of code to be executed in the parallel program than the sequential program

Load Balancing

- Speedup is generally limited by the speed of the slowest node. So an important consideration is to ensure that each node performs the same amount of work. i.e. the system is load balanced.

Communication Overhead

- Assuming that communication and calculation cannot be overlapped, then any time spent communicating the data between processors directly degrades the speedup.
- Because of this, a goal of the parallel algorithm designer should be make the grain size (relative amount of work done between synchronizations - communications) as large as possible, while keeping all the processors busy.
- The effect of communication on speedup is reduced, in relative terms, as the grain size increases.

machine dependant ratio (MDR)

$$MDR = tcomm / tcalc$$

- **Tcomm :** time to transfer a single word between two nodes
- **tcalc** : the time to perform some floating point calculation

Amdahls Law

Speedup of a parallel algorithm is effectively limited by the number of operations which must be performed sequentially, i.e its Serial Fraction

Tseq = S + P and Tpar = S + P/N

S_N = Tseq/Tpar = (S+P) / (S+P/N)

S : amount of time spent (by one processor) on serial parts of the program
P : amount of time spent (by one processor) on parts of the program that could be done in parallel

Amdahls Law (continued)

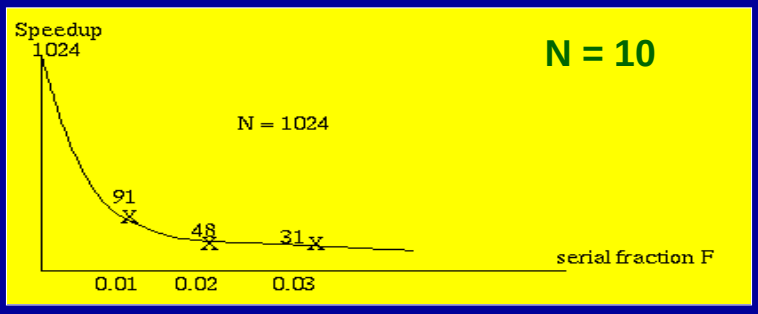
If we define the serial fraction F to be.
F = S / Tseq ==> P = (1 - F)Tseq

Then, $S_N = 1 / (F + (1 - F)/N)$

Therefore,

If F = 0 i.e. no serial part then S_N = N (the ideal value)
If F = 1 i.e. completely serial then S_N = 1 no matter how many processors are used

Amdahls Law (continued)



N = 10

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

N = 1024

Conclusions of Amdhal's Law

- **Tells us that the serial fraction F places a severe constraint on the speedup as the number of processors increase.**
- **Since most parallel programs contain a certain amount of sequential code, a possible conclusion of Amdahls Law is that it is not cost effective to build systems with large numbers of processors because sufficient speedup will never be produced.**
- **Amdahls Law is valid for problems in which the serial fraction F does not vary with the problem size i.e. as the problem increases the time Tseq and S increase keeping F = S / Tseq constant.**

Using F (serial Fraction to Measure Performance)

Since $S_N = 1 / (F + (1 - F)/N)$ then

$$F = (1/ S_N - 1/N) / (1 - 1/N)$$

The value of F is useful because equation (0) i.e. Tpar = S + P/N is idealised; it assumes all processors compute for the same amount of time i.e. perfectly load balanced. The overhead in communication and synchronization of processors is not included.

Load balancing effects are likely to result in an irregular change in F as N increases.