Scalability

Ned Nedialkov

McMaster University
Canada

SE/CS 4F03
January 2016

Outline

Speedup

- Given a problem, let T_s be the time for the fastest (practical) sequential algorithm to solve it
- Let T_p be the time to solve it in parallel on p processes
- The speedup is defined as

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

- Theoretically, S cannot exceed p
- In practice, it is possible: superlinear speedup
  E.g. the data of a serial program may not fit into cache, but could fit in a parallel version

Efficiency

- Efficiency is a measure of the fraction of time for which a processing element is usefully employed
- $$E = \frac{S}{p} = \frac{T_s}{pT_p}$$
- E is between 0 and 1

Illustration: speedup and efficiency

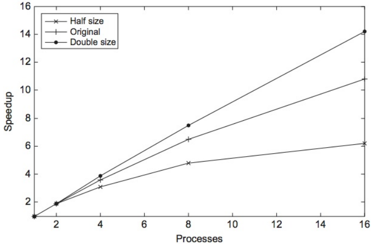
p	1	2	4	8	16
S	1.0	1.9	3.6	6.5	10.8
$E = S/p$	1.0	0.95	0.90	0.81	0.68

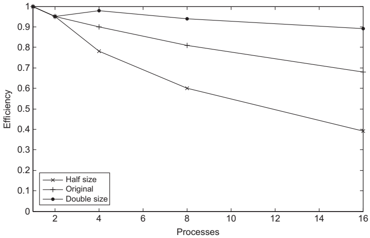
Source: P. Pacheco, An Introduction to Parallel Programming, 2011

Illustration: speedup and efficiency

	p	1	2	4	8	16
half size	S	1.0	1.9	3.1	4.8	6.2
	$E = S/p$	1.0	0.95	0.78	0.60	0.39
original size	S	1.0	1.9	3.6	6.5	10.8
	$E = S/p$	1.0	0.95	0.90	0.81	0.68
double size	S	1.0	1.9	3.9	7.5	14.2
	$E = S/p$	1.0	0.95	0.98	0.94	0.89

Source: P. Pacheco, An Introduction to Parallel Programming, 2011





Figures from P. Pacheco, An Introduction to Parallel Programming, 2011

Amdahl's law

- Example
  - a program needs 10 hours on a single processor
  - one hour is sequential, the rest 9 hours can be parallelized
  - cannot run in less than an hour
  - speedup is at most 10
- Speedup is limited by the sequential part
- Note: the problem size is fixed!

Amdahl's law cont.

- T_s serial time
- T_p parallel time
- Speedup is $S = \frac{T_s}{T_p}$
- Let r be the fraction of statements that can be executed in parallel
- Then $(1 - r)$ is the fraction of statements that is inherently serial
- Time for serial part: $(1 - r)T_s$
- $T_p = \frac{rT_s}{p} + (1 - r)T_s$

Amdahl's law cont.

- Then

$$S(p) = \frac{T_s}{T_p} = \frac{T_s}{\frac{rT_s}{p} + (1-r)T_s}$$
$$= \frac{1}{\frac{r}{p} + (1-r)} = \frac{p}{r + (1-r)p}$$

- $dS/dp > 0$, $S(p)$ increases as $p \to \infty$
- $\lim_{p\to\infty} S(p) = \frac{1}{1-r}$
- $S(p) \leq \frac{1}{1-r}$
- Speedup cannot exceed $\frac{1}{1-r}$

Amdahl's law cont.

- $S \leq \frac{1}{1-r}$
- E.g.
  - $r = 0.5$, $S(p) \leq 2$
  - $r = 0.75$, $S(p) \leq 4$
  - $r = 0.99$, $S(p) \leq 100$

Gustafson's law

- We are not interested in solving a fixed problem in the shortest possible period of time
- We are interested in increasing the problem size and number of processors and solving in about the same amount of time

Gustafson's law cont.

- Assume *a* is the sequential time and *b* is the parallel time, one any of *p* processors
  Then

$$T_p = a + b$$

- Assume that the work to be done in parallel varies linearly with *p*, number of processors
- That is, *b* work on each processor times *p* processors is
  $p \times b$
  This contributes to the serial part
- $T_s = a + p \times b$

Gustafson's law cont.

Let $\alpha = a/(a+b)$
Then

$$S(p) = \frac{a+pb}{a+b} = \alpha + p(1-\alpha) = p - \alpha(p-1)$$

- If α small, $S(p)$ approaches p as p increases

- Amdahl's law: we keep the size fixed, but increase the number of processors
- Gustafson's law: we increase both problem size and number of processors

Sources of overhead

- Interprocess communication, most significant part
- Idling
  - load imbalance: a problem is not subdivided equally
  - synchronization
  - serial component in a program: a process may execute a sequential part, while the remaining processing are waiting
- Excess computation
  - the fastest serial algorithm may be difficult or impossible to parallelize
  - a poorer algorithm may be easier to parallelize
  - excess computation = (computation performed by the parallel algorithm) − (computation performed by the best serial algorithm)

Scalability

- In general, an algorithm is scalable if we can handle increasing problem sizes
- Strongly scalable
  - problem size is fixed
  - we increase number of processes/threads
  - the efficiency is about the same
- Weakly scalable
  - we increase problem size and number of processes/threads at the same rate
  - the efficiency is about the same