Performance analysis

Jeremy Iverson

College of Saint Benedict & Saint John's University

plan for the day

- `reduction`
- theoretical analysis framework
- performing analysis
- experimental analysis
- conducting experiments

theoretical analysis framework

seeks to answer the following questions:

- how do we reason about parallel algorithms?
- how can we compare two algorithms and determine which is better?
- how do we measure improvement?

performance metrics

- execution time (T_s and T_p)
- speedup (S)
- efficiency (E)
- cost (C)

execution time

serial (T_s)

- time elapsed between beginning and end of execution

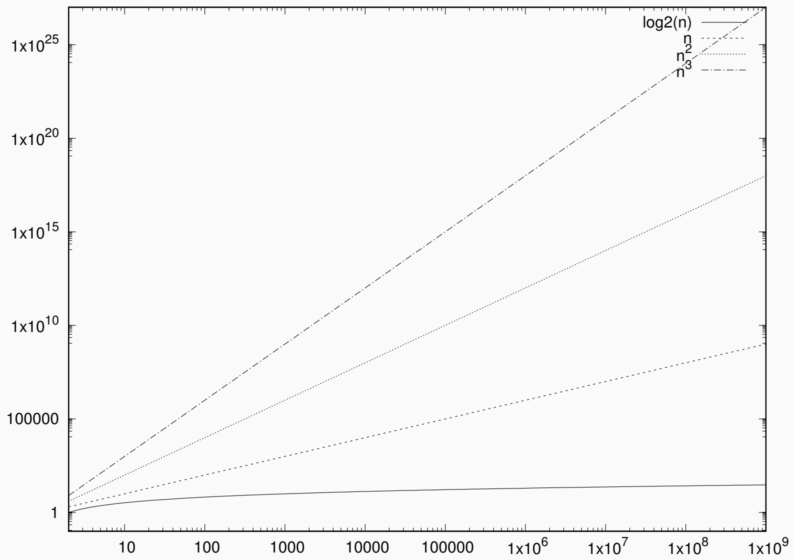
parallel (T_p)

- time elapsed between beginning of execution and the moment the last processing element finishes execution

- Axpy
- Reduction
- Dot-product
- Matrix-vector multiplication
- Matrix-matrix multiplication

1. For now, let's assume that $n = p$
2. Axpy — $T_s = \Theta(n)$ — $T_p = \Theta(1)$
3. Reduction — $T_s = \Theta(n)$ — $T_p = \Theta(\log n)$
4. Matrix-vector — $T_s = \Theta(n^2)$ — $T_p = \Theta(n)$
5. Matrix-matrix — $T_s = \Theta(n^3)$ — $T_p = \Theta(n^2)$
6. So how long to compute mat-mat for a 10000×10000 mat?

execution time



speedup

speedup ($S = T_s/T_p$)

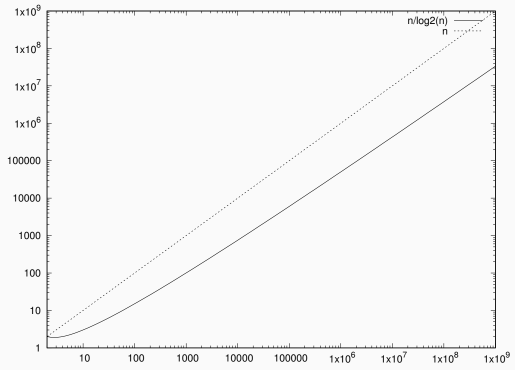
- the ratio of time taken to solve a problem on a single processing element to the time required to solve the same problem on a parallel computer with p processing elements

1. Axpy — $S = \Theta(\frac{n}{n/p}) = \Theta(p)$
2. Reduction — $S = \Theta(n/\log n)$
3. Matrix-vector — $S = \Theta(n^2/n) = \Theta(n)$
4. Matrix-matrix — $S = \Theta(n^3/n^2) = \Theta(n)$
5. What are the limits of speedup?

speedup

speedup ($S = T_s/T_p$)

- the ratio of 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 processing elements



1. Axpy — $S = \Theta(\frac{n}{n/p}) = \Theta(p)$
2. Reduction — $S = \Theta(n/\log n)$
3. Matrix-vector — $S = \Theta(n^2/n) = \Theta(n)$
4. Matrix-matrix — $S = \Theta(n^3/n^2) = \Theta(n)$
5. What are the limits of speedup?

Amdahl's law

Speedup is limited by the fraction of a parallel program that is serial.

If r is the fraction of the code which is serial, then

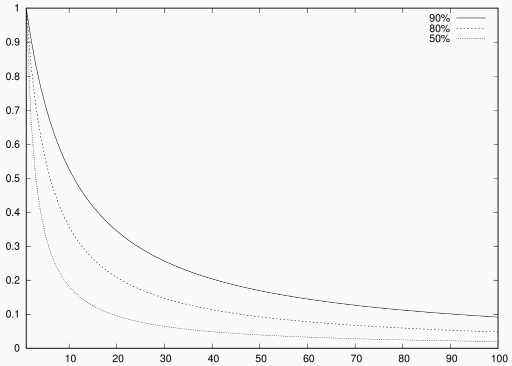
$$S = \frac{T_s}{T_p} = \frac{T_s}{(1-r)T_s/p + rT_s}$$

Amdahl's law

Speedup is limited by the fraction of a parallel program that is serial.

If r is the fraction of the code which is serial, then

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



Amdahl's law

Speedup is limited by the fraction of a parallel program that is serial.

If r is the fraction of the code which is serial, then

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

In general, we cannot get a speed up better than

$$\frac{1}{r}$$

efficiency

efficiency ($E = S/p$)

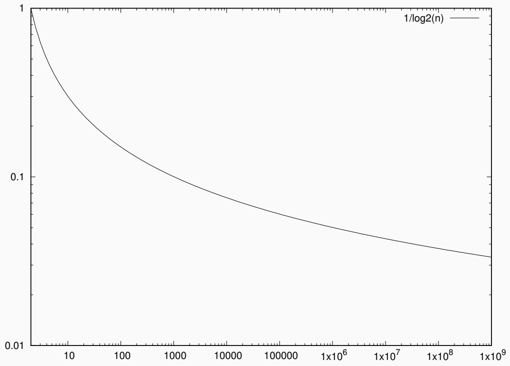
- the ratio of speedup to the number of processing elements — the fraction of time for which a processing element is usefully employed

1. Reduction — $E = \Theta(n/\log n)/n = \Theta(1/\log n)$
2. Matrix-vector — $E = \Theta(n)/n = \Theta(1)$
3. Matrix-matrix — $E = \Theta(n)/n = \Theta(1)$
4. Why do you think that the efficiency of adding numbers is less than matrix-vector or matrix-matrix?
5. What are the limits of efficiency?

efficiency

efficiency ($E = S/p$)

- the ratio of speedup to the number of processing elements — the fraction of time for which a processing element is usefully employed



1. Reduction — $E = \Theta(n/\log n)/n = \Theta(1/\log n)$
2. Matrix-vector — $E = \Theta(n)/n = \Theta(1)$
3. Matrix-matrix — $E = \Theta(n)/n = \Theta(1)$
4. Why do you think that the efficiency of adding numbers is less than matrix-vector or matrix-matrix?
5. What are the limits of efficiency?

cost

cost ($C = pT_p$)

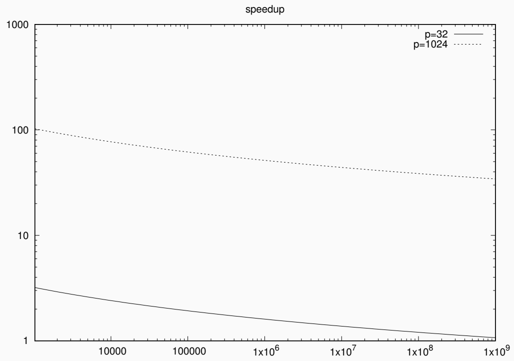
- the sum of the time spent by all processing elements solving the problem
- *cost-optimal* if $C = T_s$

1. Axpy — $C = \Theta(n)$
2. Reduction — $C = \Theta(n \log n)$
3. Matrix-vector — $C = n \times \Theta(n) = \Theta(n^2)$
4. Matrix-matrix — $C = n \times \Theta(n^2) = \Theta(n^3)$
5. Reduction is not cost-optimal, the others are

cost

cost ($C = pT_p$)

- the sum of the time spent by all processing elements solving the problem
- *cost-optimal* if $C = T_s$



1. Axpy — $C = \Theta(n)$
2. Reduction — $C = \Theta(n \log n)$
3. Matrix-vector — $C = n \times \Theta(n) = \Theta(n^2)$
4. Matrix-matrix — $C = n \times \Theta(n^2) = \Theta(n^3)$
5. Reduction is not cost-optimal, the others are

exercise — axpy

- $T_p = ?$
- $S = ?$
- $E = ?$
- $C = ?$

exercise — axpy

$p = n$ — *cost-optimal*

- $T_p = \Theta(n/p)$
- $S = \Theta(\frac{n}{n/p} = p)$
- $E = \Theta(1)$
- $C = \Theta(n)$

exercise — reduction

$p = n$ — *not cost-optimal*

- $T_p = \Theta(\log n)$
- $S = \Theta(\frac{n}{\log n})$
- $E = \Theta(\frac{1}{\log n})$
- $C = \Theta(n \log n)$

$p > n$ — too many processing elements, use less

$p < n$ — ?

exercise — reduction

$p = n$ — *not cost-optimal*

- $T_p = \Theta(\log n)$
- $S = \Theta(\frac{n}{\log n})$
- $E = \Theta(\frac{1}{\log n})$
- $C = \Theta(n \log n)$

$p > n$ — too many processing elements, use less

$p < n$ — *not cost-optimal?*

- $T_p = \Theta(\frac{n}{p} + \log p)$
- $S = \Theta(\frac{n}{\frac{n}{p} + \log p})$
- $E = \Theta(\frac{n}{n + \log p})$
- $C = \Theta(n + p \log p)$

exercise — reduction

$p = n$ — *not cost-optimal*

- $T_p = \Theta(\log n)$
- $S = \Theta(\frac{n}{\log n})$
- $E = \Theta(\frac{1}{\log n})$
- $C = \Theta(n \log n)$

$p > n$ — too many processing elements, use less

$p < n$ — *cost-optimal iff* $n = \Omega(p \log p)$

- $T_p = \Theta(\frac{n}{p} + \log p)$
- $S = \Theta(\frac{n}{\frac{n}{p} + \log p})$
- $E = \Theta(\frac{n}{n + \log p})$
- $C = \Theta(n + p \log p)$

1. if you are given a problem size, what is the maximum number of processing elements that can be used in a cost optimal way?
2. what are the limits of efficiency...how about speedup?