Performance analysis

Jeremy Iverson

College of Saint Benedict & Saint John's University

Performance analysis

- 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_p)
- speedup (S)
- efficiency (E)
- cost (C)

Execution time

Serial (T_s)

- time elapsed between beginning and end of execution

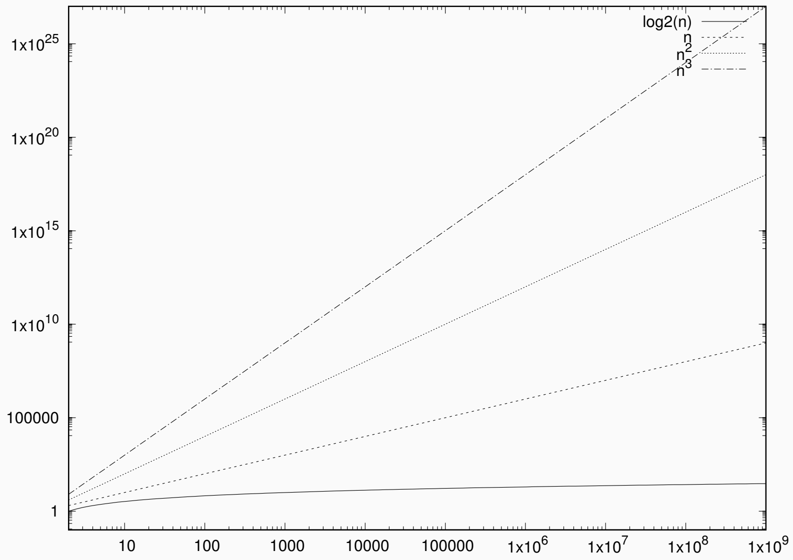
Parallel (T_p)

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

- Adding numbers
- Dot-product
- Matrix-vector multiplication
- Matrix-matrix multiplication

1. Adding numbers — $T_s = \Theta(n)$ — $T_p = \Theta(\log n)$
2. Matrix-vector — $T_s = \Theta(n^2)$ — $T_p = \Theta(n)$
3. Matrix-matrix — $T_s = \Theta(n^3)$ — $T_p = \Theta(n^2)$
4. So how long to compute mat-mat for a 10000×10000 mat?

Execution time



Speedup

Speedup ($S = T_s/T_p$)

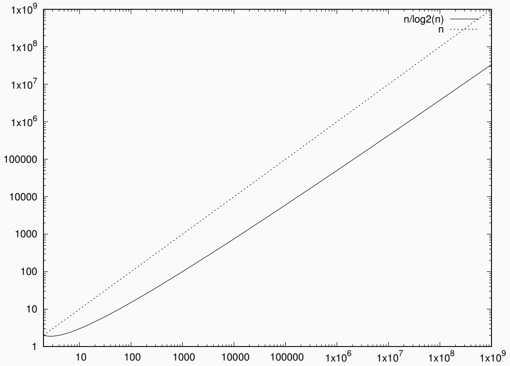
- the ratio of time taken to solve a problem on a single processing element to the time required to solve the same problem on a parallel computer with p processing elements

1. Adding numbers — $S = \Theta(n/\log n)$
2. Matrix-vector — $S = \Theta(n^2/n) = \Theta(n)$
3. Matrix-matrix — $S = \Theta(n^3/n^2) = \Theta(n)$
4. 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. Adding numbers — $S = \Theta(n/\log n)$
2. Matrix-vector — $S = \Theta(n^2/n) = \Theta(n)$
3. Matrix-matrix — $S = \Theta(n^3/n^2) = \Theta(n)$
4. What are the limits of speedup?

Efficiency

Efficiency ($E = S/p$)

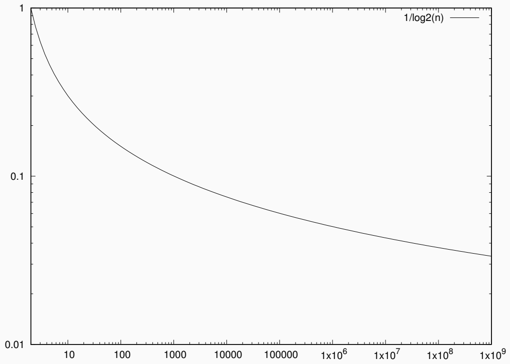
- the ratio of speedup to the number of processing elements — the fraction of time for which a processing element is usefully employed

1. Adding numbers — $E = \Theta(n/\log n)/n = \Theta(1/\log n)$
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. Adding numbers — $E = \Theta(n/\log n)/n = \Theta(1/\log n)$
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 ($C = pT_p$)

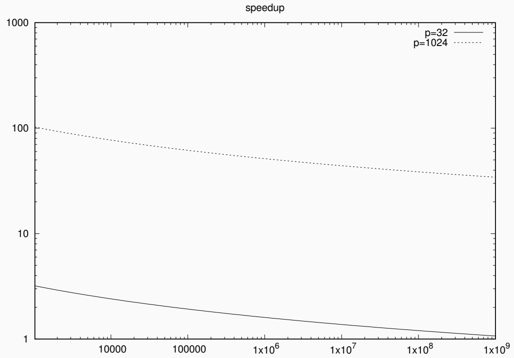
- the sum of the time spent by all processing elements solving the problem
- *cost-optimal* if $C = T_s$

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

Exercise — vector summation

$p = n$ — *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 — vector summation

$p = n$ — *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 — vector summation

$p = n$ — *not cost-optimal*

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

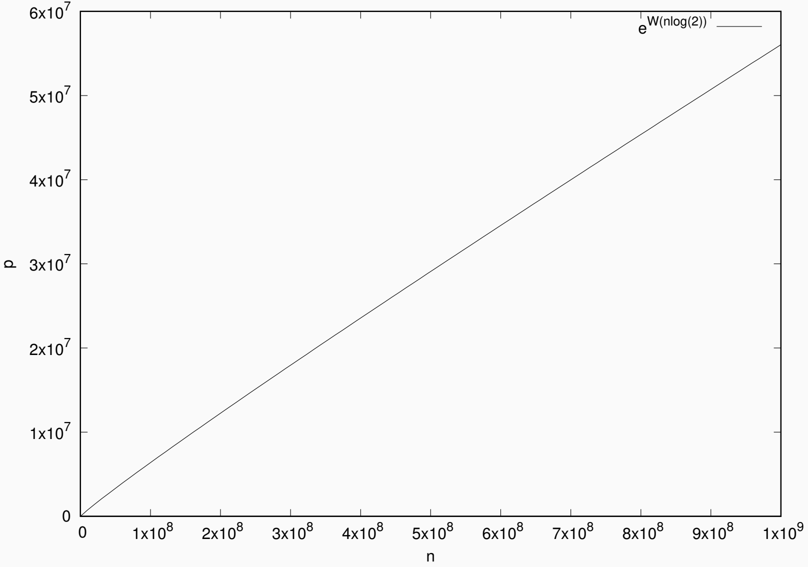
$p > n$ — too many processing elements, use less

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

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

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?

Exercise



1. give students introduction to Gnuplot

Counting sort

```
function CountingSort(inValues, outValues)
    count ← {0}
    for all v in inValues do
        count[v] ← count[v] + 1
    end for
    PrefixScan(count)
    for all v in inValues do
        outValues[count[v]] ← v
        count[v] ← count[v] + 1
    end for
end function
```


1. Which of these steps can be parallelized?
2. Assuming we have $p = n$, and we can efficiently parallelize the two for loops, what is our analysis?

Counting sort — analysis

$p = n$ — ?

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

1. T_p will be $\log |count|$ unless we use a dense data structure like a map in which case it will be $\log n$
2. n will always be $\leq |count|$