Chapter 1
In class examples

August 25, 2009

EGRE 426 Fall 09

Chapter 1 Examples

Definitions

Multioperation computer - a computer capable of performing more than one operation at a time.

$T_p(n)$ – Time to compute n terms using p processors

The ***speedup*** of performing some computation on a multioperation computer (with p processors or p function units) compared to a uniprocessor is given by

$$S_p = \frac{T_1}{T_p}$$

where T_1 is the time to perform the computaion on the uniprocessor and Tp is the time to perform the computation unsing p processors. Idealy p processors would be p times faster than a single processor. i.e. $T_p = {T_1}/{p}$. This is the besst possible case and in practice we would expect the speedup to be less than p.

Efficiency is the mesurament of how close we come to acheiving ideal speed up.

$$E_p = \frac{{T_1}/{p}}{T_p} = \frac{S_p}{p} \leq 1$$

Order: $f(x) = \mathbf{O}(\mathrm{g(x)})$ if there is a constant $r > 0$ such that $\lim_{x\to\infty}\left({f(x)}/{g(x)} \right) = r$.

Examples:

$$\left(5n^2 + 99n - 999\right) = \mathbf{O}(n^2) \text{ since } \lim_{n\to\infty}\left(\frac{5n^2 + 99n - 999}{n^2}\right) = 5 \geq 0$$

$$\left(5n^2 + 99n - 999\right) \neq \mathbf{O}(n^3) \text{ since } \lim_{n\to\infty}\left(\frac{5n^2 + 99n - 999}{n^3}\right) \to 0$$ ie. n^3 grows faster than n^2.

$$\left(5n^2 + 99n - 999\right) \neq \mathbf{O}(n) \text{ since } \lim_{n\to\infty}\left(\frac{5n^2 + 99n - 999}{n}\right) \to \infty$$

$$(n/\log(n)) \neq O(n) \text{ since } \lim_{n\to\infty}\left(\frac{\mathrm{n/log(n)}}{\mathrm{n}}\right) = \lim_{n\to\infty}\left(\frac{1}{\log(n)}\right) = 0$$ ie. n grows faster than n/log(n).

EGRE 426 Fall 09
Chapter 1 Examples

EXAMPLES

ASSUME: All operations take one unit of time. All instructions and data are available when needed. ie. We don't have to wait for memory or communication.

CONVENTIONAL UNIPROCESSOR

```
T    A1*B1  A2*B2  A3*B3  A4*B4 ... An*Bn
      \ /   |   |  |   |  |   |
1      *    |   |  |   |  |   |
             \ /   |   |  |   |
2             *    |   |  |   |
                    \ /   |   |
3                    *    |   |
                           \ /
4                           *
```

For 4 terms $T_1 = 4$

In general for n terms $T_1(n) = n$

T_1 is of order n.

Multiprocessor MIMD (unlimited processors)

```
T    A1*B1  A2*B2  A3*B3  A4*B4 ... An*Bn
      \ /    \ /    \ /    \ /       \ /
1      *      *      *      *         *
```

For n terms and at least n processors

$T_n = 1$ of order 1.

Seed up of using n processors verses a single processor is:

$$S = \frac{T_1}{T_n} = \frac{n}{1} = n = \mathbf{O}(n)$$

$$E = \frac{S}{n} = \frac{n}{n} = 1 = 100\%$$

Parallel processor SIMD (unlimited processors)

Same as above

Multifunction Computer (2 *)

```
T    A1*B1  A2*B2  A3*B3  A4*B4 ... An*Bn
      \ /    \ /   |   |  |   |
1      *      *    |   |  |   |
                    \ /    \ /
2                    *      *

3

4
```

For 4 terms T2 = 2

In general for n terms

$$T_2 = \begin{cases} n/2 \text{ if n is even} \\ (n+1)/2 \text{ if n is odd} \end{cases}$$

Better form

T2 = $\lceil n/2 \rceil$

Ceiling n/2 ie. $\lceil 5.5 \rceil = 6$, $\lceil 5.0 \rceil = 5$.

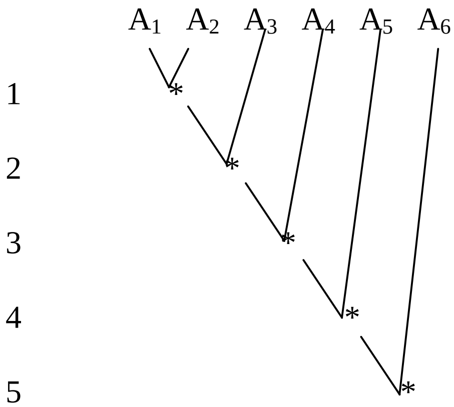
$$S = \frac{T_1}{T_2} = \frac{n}{\lceil n/2 \rceil} = \begin{Bmatrix} 2 \text{ for n even} \\ 2\dfrac{n}{n+1} \text{ for n odd} \end{Bmatrix} \longrightarrow 2 \text{ for large n , S = O(1).}$$

$$E = \frac{S_2}{2} = \frac{\lceil n/2 \rceil}{n} \Big/ 2 \approx 100\%$$

EGRE 426 Fall 09
Chapter 1 Examples

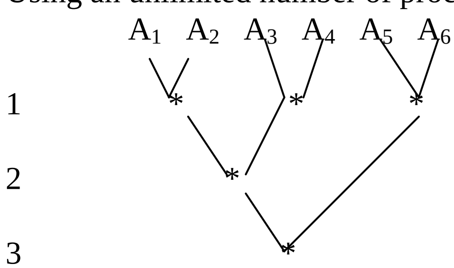
Now consider $P = \prod_{i=1}^{n} A_i = A_1 A_3 \ldots A_n$

Using a uniprocessor



In general $T_1 = n - 1$

Using an unlimited number of processors:



In general $T_\infty = \log_2(n)$

Why

Time	n - number of terms
1	$2 = 2^1$
2	$4 = 2^2$
3	$8 = 2^3$
...	...
k	$n = 2^k$

$2^k = n$

$k \log(2) = \log(n)$

$k = \frac{\log(n)}{\log(2)} = \log_2(n)$

But, when n is not a power of 2 we must round up to the next highest integer.
Therefor,

$k = \lceil \log_2(n) \rceil$

EGRE 426 Fall 09
Chapter 1 Examples

Supose we build a two operation (three operand) computer capable of performing A*B*C in a single operation. (IBM has a workstation that uses A*B+C as its fundamental floating point operation.)

Q. How long would it take to perform $P = \prod_{i=1}^{n} A_i = A_1 A_3 \ldots A_n$

Ans. log3(n)

Suppose we build a 32 bit adder using 4 input gate. Best time we can hope for.
$\log_4(64)$

EXAMPLE

Algorithm can effect speedup. (a). Only one processor can be used at a time. (b). Two processors can produce fastest result.

```
A ( B C D + E)     = A B C D + A E
\   \ /  |    |       \ / \      \ /
1\   *    /    /       *    *      X
  \   \ /     /         \ /        |
2  \   *     /           *         *
    \   \ /                \__ __/
3     \   +                    +
       \ |                  SAME FOR SIMD
4        +
```

$$T_1 = T_\infty = 4$$
$$T_2 = 3$$
$$S = \frac{T_1}{T_2} = \frac{4}{3} = 1\tfrac{1}{3}$$
$$E = \frac{T_1/n}{T_2} = \frac{4/2}{3} = \frac{2}{3} = 67\%$$

EGRE 426 Fall 09
Chapter 1 Examples

POLY.

Consider the evaluation of the polynominal.

$$F = \sum_{i=0}^{n} A_i X^i = A_0 + A_1 X + A_2 XX + A_3 XXX + \ldots$$

For a single operation computer the polynoninal can be evaluated as shown below.

METHOD 1.

```
      F = A0 + A1*X + A2*X*X + A3*X*X*X + A4*X*X*X*X
           \   \ /   |  | |    | | | |    | | | | |
1           \   *    |  | |    | | | |    | | | | |
             \ /     |  | |    | | | |    | | | | |
2             +      |  | |    | | | |    | | | | |
               \     |  \ /    | | | |    | | | | |
3               \    \   *     | | | |    | | | | |
                 \    \ /      | | | |    | | | | |
4                 \    *       | | | |    | | | | |
                   \ _/        | | | |    | | | | |
5                   +          | | | |    | | | | |
                     \         | | \ /    | | | | |
6                     \        | |  *     | | | | |
                       \       \ /  |     | | | | |
7                       \       *   |     | | | | |
                         \       \ /      | | | | |
8                         \   ____*       | | | | |
                           \ /            | | | | |
9                           +             | | | | |
                             \            | | | \ /
10                            \           | | |  *
                               \          | \ /  |
11                              \         |  *   |
                                 \        |   \ /
12                                \       |  __*
                                   \      \ /
13                                  \__   _*
                                       \ /
14                                      +
```

For $n = 4$, $T_1 = 14$

In general

T_1 = time for n adds + time for (1+2+3+4+...+n) multiplies

= n + 1 + 2 + 3 + 4 +...+ n

= n + n(n+1)/2

= n(n+3)/2 = $\mathbf{O}(n^2)$

METHOD 2.
This can be done faster by not recomputing known terms. ie. Using a better compiler.

```
F = A0 + A1*X + A2*X*X + A3*X*X*X + A4*X*X*X*X
     \   \ /   |  |  |    |  |        |
1     \   *    |  |  |    |  |        |
       \ /     |  |  |    |  |        |
2       +      |  |  |    |  |        |

         \     |  \ /    |  |        |
3         \    \   *     |  |  _     |  _   _
           \    \ /      |  |  |     |  |   |
4           \    *       |  |  |     |  |   |
             \ _/        |  |  |     |  |   |
5             +          |  |  /     |  |   |
               \         |  \ /      |  |   |
6               \        |  *       |  |   |
                 \       \ /        |  |   |
7                 \__   __*         |  |   |
                     \ /            |  |   |
8                     +             |  \   /
                       \            |   \ /
9                       \           |  __*
                         \          \ /
10                        \_____  ___*
                                \ /
11                               +
```

For $n = 4$, $T_2 = 11$
In general

$$T = n \text{ times for } n \text{ adds} + n \text{ times for multiplying } A_i \text{ and } X^n$$
$$+ (n-1) \text{ times for multiplying } X \text{ and } X^{i-1}$$
$$= n + n + n - 1$$
$$= 3n - 1 \text{ of order n.}$$

METHOD 3.

A new algorithm makes the solution even faster.

```
F = A0 + A1*X + A2*X*X + A3*X*X*X + A4*X*X*X*X
  = A0 + X * (A1 + X * (A2 + X * (A3 + X * (A4 ... ))))
     |   |     |   |     |   |     |    \ /
 1   |   |     |   |     |   |     \   __*
     |   |     |   |     |   |      \ /
 2   |   |     |   |     |   |   ____+
     |   |     |   |     |    \ /
 3   |   |     |   |     \   __*
     |   |     |   |      \ /
 4   |   |     |   \   ____+
     |   |     |    \ /
 5   |   |     \   __*
     |   |      \ /
 6   |   \   ____+
     |    \ /
 7   \   __*
      \ /
 8     +
```

For n = 4, T = 8.
In general, T = n adds + n multiplications = 2n of order n.
This new algorithm produces a speed up over METHOD 1 of

$$S = \frac{T_1}{T_3} = \frac{n(n+3)/2}{2n} = \frac{n}{4} + \frac{3}{4} \to \frac{n}{4}$$ for large n.

The speed up of METHOD 3 over METHOD 2 is:

$$S = \frac{T_2}{T_3} = \frac{(3n-1)}{2n} = \frac{3}{2} - \frac{1}{n} \to \frac{3}{2}$$ for large n.

EGRE 426 Fall 09

Chapter 1 Examples

Assume we have a parallel processor that can perform an unlimited number of additions and multiplications simultaneously.
Using the provious algorithm:

```
F = A0 + A1*X + A2*X*X + A3*X*X*X + A4*X*X*X*X
         = A0 + X * (A1 + X * (A2 + X * (A3 + X * (A4 ... ))))
            |   |     |   |     |   |     |    \ /
        1   |   |     |   |     |   |     \   __*
            |   |     |   |     |   |      \ /
        2   |   |     |   |     |   |   ____+
            |   |     |   |     |    \ /
        3   |   |     |   |     \   __*
            |   |     |   |      \ /
        4   |   |     |   \   ____+
            |   |     |    \ /
        5   |   |     \   __*
            |   |      \ /
        6   |   \   ____+
            |    \ /
        7   \   __*
             \ /
        8     +
```

For n = 4, T = 7.
In general, T = n adds + n multiplications = 2n of order n.
No improvement over uniprocessor!

Returning to the original algorithm.

```
F = A0 + A1*X + A2*X*X + A3*X*X*X + A4*X*X*X*X
         = A0 + X * (A1 + X * (A2 + X * (A3 + X * (A4 ... ))))
           \     \ /    \ / |    \ / \ /    \ / \ / |
        1   \     *      *  |     *   *      *   *  |
             \ __/        \ /      \ /        \ /   |
        2     +            *        *          *    |
               \_____ ____/        /            \__ /
        3            +            /                *
                      \________ _/                /
        4                     +                  /
                                \________ ______/
        5                               +
```

For n = 4, T = 5
It is difficult to find a general solution for the time as a function of n. I have obtained a solution, but have not proved that it is correct. However, it is easy to obtain a good least uper bounds on the time. This can be done by first doing all adds then doing all multiplies. Then T ≤ time to do all adds + time to do all multiplies or

$$T(n) \le \lceil \log_2(n+1) \rceil + \lceil \log_2(n+1) \rceil = 2\lceil \log_2(n+1) \rceil$$

For example when n=9 $T(9) \le \lceil \log_2(10) \rceil = 2 \times \lceil 3.3219 \rceil = 2 \times 4 = 8$

The exact answer is for n=9 is T = 7. Example: Consider a multiply add unit capable of computing a*b+c in one unit of time.

EGRE 426 Fall 09
Chapter 1 Examples

Show how to compute: $F(n) = \sum_{i=0}^{n} A_i X^i$ using the multiply add unit.

Consider the case when n = 4.

```
F = A0 + A1*X + A2*X*X + A3*X*X*X + A4*X*X*X*X
  = A0 + X * (A1 + X * (A2 + X * (A3 + X * (A4 ... ))))
     |   |     \    \    \    \     \__ \ /
 1   |   |      \   |     \__  \ _______+*
      \  |       \  |        | |/
 2     \ |        \ |        + *
        \ \        \ \______/
 3       \ \  ______+*
          \ \/
 4         +*
```

It appears that in general the time to compute $F(n) = \sum_{i=0}^{n} A_i X^i$ is given by T(n) = n.

Proof: Assume T(n) = n is the time to compute $F(n) = \sum_{i=0}^{n} A_i X^i$,and show that it follows that T(n+1) = n+1.

$$F(n+1) = \sum_{i=0}^{n+1} A_i X^i = A_0 + X \sum_{i=0}^{n} A_{i+1} X^i$$

Once $\sum_{i=0}^{n} A_{i+1} X^i$, has been computed the remainder can be computer in one unit of time.

Therefore, T(n+1) = T(n) + 1 = n+1.

Since we can easily show that T(1) = 1, it follows that T(1+1) or T(2) = 2. Since T(2) = 2, T(3) = 2+1. etc. for all values of n.