Parallel Computing

Introduction-level parallelism: several instructions can be in flight simultaneously inside a CPU

2.1 Introduction

Parallel computing means use multiple processor to do the same program but different and independent parts of the program.

Characteristics

a) sometimes algorithms need to be rewritten slightly to make them parallel

b) a parallel algorithm may not show perfect speedup

Processors are often connected through a network and moving data through this network takes time.

Data parallelism: the same operation is applied in parallel to many data elements.

Task parallelism: subprogram

2.2 Theoretical Concepts

Speedup $Sp = T_1/T_p$

T_1 is defined as the best time to solve the problem on s single processor.

T_p is the time it takes to solve the problem on the machine with p processors.

$E_p = S_p/p$ Efficiency

Embarrassingly parallel: in effect consisting of a number of completely independent calculations.

Cost-optimal: if the overhead is at most of the order of the running time of the sequential algorithm.

T_1 the time the computation takes on a single processor

T_p the time the computation takes with p processors

$T_{infinity}$ the time the computation takes if unlimited processors are available

$P_{infinity}$ the value of p for which $Tp = T_{infinity}$

Brent's theorem: The computation can be done in $Tp = t + (m – t)/p$, m is the total number of tasks, t the length of critical path.

Amdahl's law:

$Tp = T1*(Fs + Fp/P)$. Fs is the sequential fraction and Fp is the parallel fraction.

Weak Scalability: as problem size and number of processors grow in such a way that the amount of data per processor stays constant, the execution time also stays constant.

2.3 Parallel computers architectures

SISD: single instruction single data, this is the traditional CPU architecture, at any one time only a single instruction is executed, operating on a single data item.

SIMD single instruction multiple data, in this computer type there can be multiple processors, each operating on its own data item, but they are all executing the same instruction on that data item.

MISD multiple instruction single data. No architectures answering to this description exist.

MIMD multiple instruction multiple data, here multiple CPUs operate on multiple data items, each executing independent instructions. Most current parallel computers are of this type.

2.4 Different types of memory access

Symmetric Multi-Processors, Uniform Memory Access: between one memory location and another, any memory location is accessible to every processor, and the access times do not differ.

Non-Uniform Memory Access: the situation where a processor can access its own memory fast, and other processors' memory slower.

Logically and physically distributed memory, the processors have their own address space and can not directly see another processor's memory.

2.5 Granularity of parallelism

Data parallelism, instruction-level parallelism, task-level parallelism, conveniently parallel computing, medium-grain data parallelism, task granularity.

Exercise 2.5

1. No, x[i, j] depends on the previous term x[i, j-1].

2. No, x[i, j] depends on the term x[i-1, j] coming from the previous loop.

3. x[2, 1] = x[1, 1] + x[2, 0]; x[1, 2] = x [0, 2] + x[1, 1].

4. Once term x[i, i] is known, x[i, (i:N)] and x[(i:N), i] can be computed independently. So we can change the code:

```
For i in [1:N]:
    For j in [i:N]:
        x[i, j] = x[i-1,j] + x[i, j-1]
        x[j, i] = x[j-1,i] + x[j, i-1]
```

the two operations can be computed independently which makes the code to be a data parallelism one.

Exercise 2.6

Because if it happens, at least one of the processors has to be faster than the best performance when it is working alone which is contradictory.