Amdahl's Law

Exercise 1

Assume 1% of the runtime of a program is not parallelizable. This program is run on 61 cores of a Intel Xeon Phi. Under the assumption that the program runs at the same speed on all of those cores, and there are no additional overheads, what is the parallel speedup?

Solution

Amdahl's law assumes that a program consists of a serial part and a parallelizable part. The fraction of the program which is serial can be denoted as B — so the parallel fraction becomes $1 - B$. If there is no additional overhead due to parallelization, the speedup can therefore be expressed as

$$S(n) = \frac{1}{B + \frac{1}{n}(1 - B)}$$

For the given value of $B = 0.01$ we get $S(61) = 38.125$.

Exercise 2

Assume that the program invokes a broadcast operation. This broadcast adds overhead, depending on the number of cores involved. There are two broadcst implementations available. One adds a parallel overhead of $0.0001n$, the other one $0.0005 \cdot log(n)$. For which number of cores do you get the highest speedup for both implementations?

Solution

$$S_1(n) = \frac{1}{0.001 + \frac{1}{n}0.999 + 0.0001n}$$
$$S_2(n) = \frac{1}{0.001 + \frac{1}{n}0.999 + 0.0005log(n)}$$

We can get the maximum of these terms if we minimize the term in denominator.

$$\frac{d}{dn}0.001 + \frac{1}{n}0.999 + 0.0001n = 0 \leftrightarrow 0.0001 - \frac{0.999}{n^2} = 0 \leftrightarrow n \approx 100$$

$$\frac{d}{dn}0.001 + \frac{1}{n}0.999 + 0.0005log(n) = 0 \leftrightarrow \frac{0.005n0.999}{n^2} = 0 \leftrightarrow n = 1998$$

Exercise 3

By Amdahls law, it does not make much sense to run a programm on millions of cores, if there is only a small fraction of sequential code (which is often inevitable, i.e., reading input data). Why do people build such systems anyway?

Solution

Amdahl's law Assumes that the problem size is kept constant — by adding more processors the same problem gets solved faster. In HPC it is often the case that bigger computers are used to solve bigger problems, not to solve old problems faster. If the sequential part of the program does not increase when increasing the input (or increases sublinearly) we can run on a large number of cores.

PRAM

Exercise 1

We can find the minimum from an unordered collection of n natural numbers by performing a reduction along a binary tree: In each round, each processor compares two elements, and the smaller element gets to the next round, the bigger one is discarded. What is the work and depth of this algorithm?

Solution

The dependency graph of this computation is a tree with $log_2(n)$ levels. Therefore the longest path, which is equal to the depth/span has length $log_2(n)$. The tree contains $2n-1$ nodes, which is equal to the work.

Exercise 2

Develop an Algorithm which can find the minimum in an unordered collection of n natural numbers in $O(1)$ time on a CRCW-PRAM machine.

Solution

Assume the inmput list is stored in the array input. We use n^2 processors, labelled $p(i,j)$ with $0 \leq p, j < n$. Each processor $p(i,j)$ performs the comparison input[i] ¡ input[j]. If the result is false then i can not be the smallest element, and tmp[i] is set to false (all elements of tmp are initially set to true). Then n processors check the different values of tmp — only one element tmp[x] will be true, that means input[x] is the smallest element.