Noah Wiley
CSCI 176 - Parallel Processing
Professor Park
02/17/26

CSCI 176 - Homework 2

Question 1
Given:

$T_{serial} = n$
$T_{parallel} = \frac{n}{p} + log_2(p)$

The goal is to determine how the problem size n must grow the number of processes p increases by a factor of k, while keeping the efficiency constant. We will also evaluate what happens when doubling from 8 to 16 processes and determine whether the program is scalable.

Efficiency measures how well processors are being used and is defined as $E = \frac{Speedup}{p}$

We can first compute the speedup:
$S(p) = \frac{T_{serial}}{T_{parallel}} = \frac{n}{\frac{n}{p}+log_2(p)}$

Divide by p to obtain efficiency:
$E = \frac{1}{p} * \frac{n}{\frac{n}{p}+log_2(p)}$

Simplifying:
$E = \frac{n}{n+plog_2(p)}$
This formula shows that efficiency depends on both problem size $n$and the overhead term $p\ log_2(p)$. To maintain constant efficiency as we increase the number of processes, the numerator and denominator must grow proportionally. This means that when we replace p with kp, the problem size must also increase appropriately.

Let: $p = kp$

$\frac{n}{n+p\ log_2(p)} = \frac{n}{n+kplog_2(kp)}$

Solving for n gives:

$$n = \frac{k\ log_2(kp)}{log_2(p)} n$$

Now suppose we double the number of processes from 8 to 16

$$p = 8,\ k = 2$$

$$n = \frac{2\ log_2(16)}{log_2(8)} n$$

$$log_2(16) = 4,\ log_2(8) = 3$$

$$n = \frac{2*4}{3} n = \frac{8}{3} n$$

$$n = 2.67n$$

When doubling from 8 to 16 processes, the problem size must increase by approximately 2.67 times to maintain the same efficiency.

Is the program scalable?

From the efficiency formula: $E = \frac{n}{n+plog_2(p)}$, we see that to keep efficiency constant, the problem size must grow proportionally to:

$n \propto p\ log_2(p)$, since this growth is only slightly super-linear, the program can maintain good efficiency as the number processes increases. So Yes, the program is scalable.

Question 2

A program is said to achieve linear speedup if its speedup is equal to the number of processors used. In other words, if we denote speedup by $S(p)$, then linear speedup means $S(p) = p$
This implies that doubling the number of processors cuts the execution time in half, tripling the processors reduces time to one-third, and so on. Linear speedup represents ideal parallel performance because every processor contributes fully to reducing execution time. Strong scalability refers to how well a parallel performs when the problem size remains fixed and the number of processors increases. A program is strongly scalable if its speedup grows proportionally with the number of processors while solving the same problem size.. In other words, as we increase p, the execution time decreases proportionally. If a program achieves true linear speedup, this its efficiency is $E = \frac{S(p)}{p} = \frac{p}{p} = 1$. This means efficiency remains constant at 100% as the number of processors increases. Since strong scalability requires that performance improves proportionally with the number of processors for a fixed problem size, a program that achieves linear speedup satisfies this condition exactly. Therefore, yes, a program that obtains true linear speedup is strongly scalable, because it maintains constant efficiency and proportional performance improvement as processors increase.