Linpack scaling and analysis

Michel Gonzalez

November 24, 2021

1 Exercises 1

1.1 Part 1

In this scenario the number of processor cores is $N_p = 1$, the benchmark time $T = T_o$, Number of words in the memory $M = M_o$. If the new processor is twice as fast as the old processor, then the new processor will have double the core clock speed compared to the old processor. Therefore the new processor will be capable to execute double the amount of cycles compared to the old processor. Thus, it will execute double the number of operations, assuming the number of cycles per operations remains the same.

Since the clock speed was double, it will take T/2 amount of time to run the same benchmark. The required time to run the new benchmark = T, thus the speed up is,

$$S_p = \frac{T/2}{T} \rightarrow S_p = \frac{1}{2}$$

This means that the amount of used memory should decrease. Thus, the amount of required memory is given by the ratio of the old memory size M_o and the speed up S_p.

$$\frac{M_o}{S_p} = \frac{M}{2}\ words$$

1.2 Part 2

In this scenario the number of processor core is $N_p = P$, the benchmark time is $T = T_o$ and the number of words per core is $M = M_o$. If the new processor has twice the number of cores as the old processor, then the new processor will have double the core clock speed compared to the old processor. Therefore the new processor will be capable to execute double the amount of cycles compared to the old processor. Thus, it will execute double the number of operations, assuming the number of cycles per operations remains the same.

As the number of cores of the new processor is double, so it will take $\frac{T}{2}$ amount of time to run the same benchmark. Since the required time to run the new benchmark is $T = T_o$ the speed up is still,

$$S_p = \frac{1}{2}$$

So again, the amount of used memory should decrease.Thus, the amount of required memory is given by the ratio of the old memory size M_o and the speed up S_p.

$$\frac{M_o}{S_p} = \frac{M}{2}\ words$$

2 Exercises 2

2.1 Part 1

$$\begin{bmatrix} a_{00} & a_{01} & \dots & a_{0n} \\ a_{10} & a_{11} & \dots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n0} & a_{n1} & \dots & a_{nn} \end{bmatrix}$$

For an n by n matrix, assuming that there are no pivots required to solve the matrix, the amount of divisions for the first iteration is $n-1$ and for the second it would be $n-2$, for the third $n-3$ and so on. Thus the total amount of division operations would be $n\frac{(n-1)}{2} \approx \frac{n^2}{2}$ operations. For the first iteration of one vector matrix dot product, it would require $(n-1)^2 mults$ and $(n-1)^2 adds$. For both, the next requirement is $(n-2)^2$ and so on. Thus it would take $\frac{n(n-1)(2n+5)}{6} \approx \frac{n^3}{3} add$ operations, and it would take $\frac{n(n^2+3n-1)}{3} \approx \frac{n^3}{3} mults \ \& \ divs$. Thus, it takes $m = \frac{2n^3}{3}$ tasks to complete a Gaussian elimination.

Now there is a critical path here since for each operation there is a deep dependency on the previous calculations of the numbers for each column of the resulting matrix. This means that the whole process can't be calculate in parallel. This is because it first needs the new values for the resultant matrix thus, it must serially go through $n-1 \approx n$ iterations in order to complete the process. Thus, length of the critical path is $t = n$. Using Brent's theorem I found the upper bound on parallel run time,

$$T_p \leq t + \frac{m-t}{p} \rightarrow T_p = n + \frac{\frac{2n^3}{3} - n}{p}$$

where p is the number of processors.

2.2 Part 2

In parallel the tasks would be split up so each core finds the corresponding vector from the vector-vector multiplications. Since the first iteration requires $\approx 2n^2$ operations per dot product, the theoretical max number of processors would be $2n^2$ processors.

With this we can find the upper bound for a Gaussian elimination arithmetic.

$$T_n = n + \frac{2n^3}{3*2n^2} - \frac{n}{n} \rightarrow T_n \approx \frac{4n}{3}$$

The efficiency is given by the ratio of the speed up value, which is the ratio of one processor and in this case the upper bound, and the max number of processors. The time it takes for a single processor to complete the vector-vector multiplication is $\approx 2n^2$ units of time assuming each operation takes a single unit of time. Therefore, the speedup value is,

$$S_n = \frac{2n^2}{\frac{4n}{3}} = \frac{3n}{2}$$

Thus, the efficiency is,

$$\varepsilon_n = \frac{\frac{3n}{2}}{2n^2} = \frac{3}{4n}$$

The efficiency decreases as $n \rightarrow \infty$.