In what follows we consider $t_c = t_w = t_s = 1$. In homework 8, I provided a solution that ignored the computation work. Considering the computation leads to an interesting result. In the wavefront as described in the homework, we have n^2, P processors, and $2\sqrt{P}-1$ steps are required to complete the entire wavefront. From this we can see that

$$\begin{aligned} T_1 &= n^2, \text{and} \\ T_p &= (2\sqrt{P}-1)\frac{n}{\sqrt{P}}\frac{n}{\sqrt{P}} \\ &= (2\sqrt{P}-1)\frac{n^2}{P} \end{aligned}$$

We now compute PT_p due to the computational time, and get

$$\begin{aligned} PT_p &= (2\sqrt{P}-1)\frac{Pn^2}{P} \\ &= (2\sqrt{P}-1)n^2 \end{aligned}$$

We now form the isoefficiency relation for this term, getting

$$\begin{aligned} T_O &= PT_p - n^2 \\ &= (2\sqrt{P}-1)n^2 - n^2 \\ &= 2\sqrt{P}n^2 - n^2 - n^2 \\ &= 2\sqrt{P}n^2 - 2n^2 \\ W &= KPT_p \\ & 2Kn^2(\sqrt{P}-1) \end{aligned}$$

Substituting n^2 for W and dividing yields

$$\begin{aligned} n^2 &= 2Kn^2(\sqrt{P}-1) \\ 1 &= 2K\sqrt{P} - 2K \end{aligned}$$

and the question that immediately arises is, what are we to make of this?

One immediate observation is that the part of T_O that is based on the computation is larger than T_1 (which we will call $T_{O,c}$ for short) – that is, the time to merely do the work of the serial computation is growing with the number of processors! In the other calculations we have looked at, $T_{O,c} = T_1$. The reason for this is that even not counting communication (which we did not do above) some processors are idle while others are doing normal work. The consequence of this is that efficiency *must* drop as we go to larger numbers of processors, and it is impossible to perform an isoefficient scaling of the problem. That is, it is impossible to scale the program to more processors and maintain efficiency.

To confirm this, let compute the efficiency of a program where $T_1 = f(n)$ and $T_{p,c} = k\frac{f(n)}{P}$ for some (not necessarily constant) k, and where $T_{p,c}$ is the

parallel time due to computation for p processors. We can show that $k \geq 0$ for most calculations. If $k < 0$, then the amount of work needed to perform the computation on the parallel processor is less than the amount of work needed on one processor, and the speedup will be superlinear and the efficiency E will be greater than one.

The speedup is:

$$\begin{aligned} S_p &= \frac{T_1}{T_{p,c}} \\ &= \frac{f(n)}{k\frac{f(n)}{P}} \\ &= \frac{Pf(n)}{kf(n)} \\ &= \frac{P}{k} \end{aligned}$$

and observe that if $k < a, S_P > P$, i.e. is superlinear.

The efficiency is then

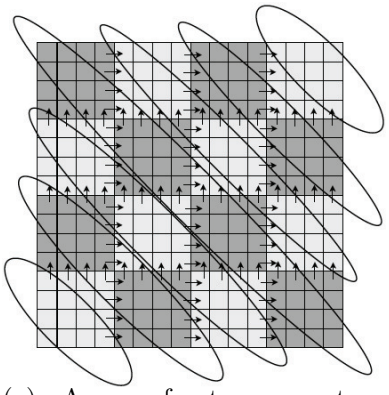
$$\begin{aligned} E &= \frac{\frac{P}{k}}{P} \\ &= \frac{\frac{P}{k}}{P} \\ &= \frac{1}{k} \end{aligned}$$

Now let $k = g(P)$, i.e. k is a function of the number of processors. If $g(P)$ is monotonically increasing in the number of processors, then each additional processor added to the computation reduces the value $\frac{P}{k}$ and reduces the efficiency. Thus there can be no isoefficiency scaling since the the effiency *must* drop in the number of processors. This is the case we observed with our wavefront.

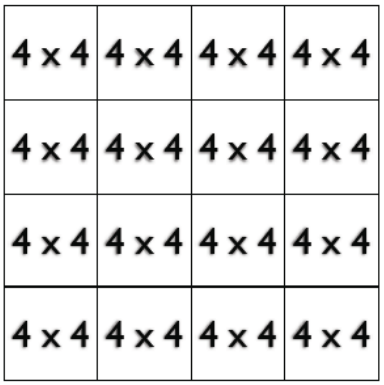
Another interesting question is can we do a wavefront better? And the answer, fortunately, is yes. In Figure 1(a) the figure from homework seven is reproduced showing the time that each of the sixteen submatrices. Figure 1(b) shows the sixteen 4×4 sub-matrices that make up a 32×32 matrix on which the computation is to take place. In Figure 1(c) it is shown how the computation on the 16 submatrices can be performed on an array of four processors.

Let's compute the resulting speedup and isoefficiency. The serial time remains n^2. Let $n = P$ and divide the matrix into $P \times P$ submatrices, as shown in Figure 1(b). Then each block requires $\frac{n}{P} \cdot \frac{n}{P}$, or $\frac{n^2}{P^2}$ work. However, the dependences must still be honored, and as seen in Figure 1, $2P - 1$ steps are needed to perform the entire computation, with each step taking $\frac{n^2}{P^2}$ time. Thus

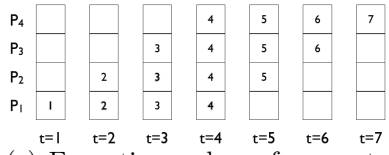
$$\begin{aligned} T_P &= (2P-1)\frac{n^2}{P^2} \\ &= \frac{n^2(2P-1)}{P^2} \end{aligned}$$



(a) A wavefront on a two-dimensional array



(b) The layout of a 32×32 matrix onto a 4×4 processor grid



(c) Execution orders of computation of 16 4×4 sub-matrices on a linear array of four processors in seven time steps. Numbers in the boxes represent the stage of the computation, illustrated in (a), that is executing on the processors. Note that each processor must contain the elements to perform the work to perform the computation on four of the sixteen blocks.

Figure 1: A better way to do wavefronts

$$\begin{aligned} S_P &= \frac{n^2}{\frac{n^2(2P-1)}{P^2}} \\ &= \frac{P^2}{(2P-1)} \\ &\approx \frac{P}{2} \end{aligned}$$

which is consistent with Figure 1(c) in which 1/2 of the processors are busy on average. Let us know compute the isoefficiency of this program.

$$\begin{aligned} PT_P &= P(2P-1)\frac{n^2}{P^2} \\ &= \frac{(2P^2n^2 - Pn^2}{P^2} \\ &= 2n^2 - \frac{n^2}{P} \\ PT_P - T_1 &= 2n^2 - \frac{n^2}{P} - n^2 \\ &= n^2 - \frac{n^2}{P} \\ W &= K\left(n^2 - \frac{P-1}{P}\right) \\ &= K\left(n^2 - \frac{P-1}{P}\right) \end{aligned}$$