1. Amdahl's law. (28%)

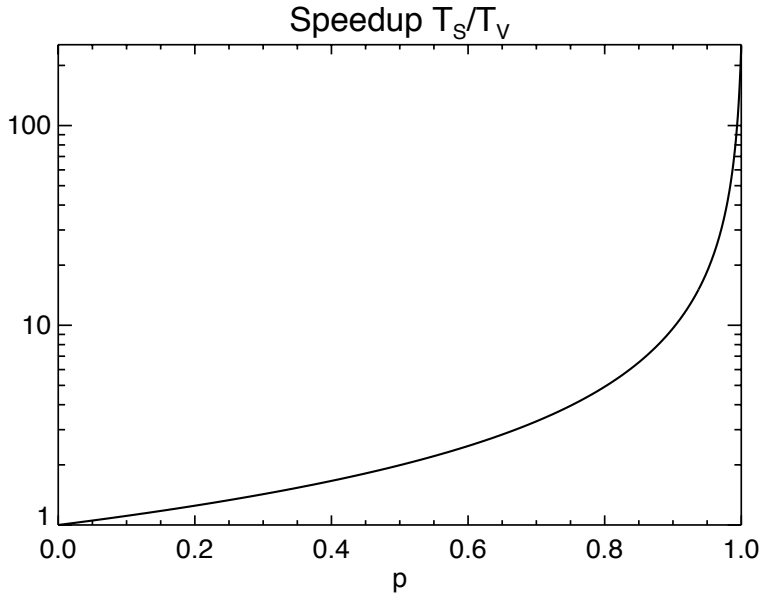
Consider parallel processing with common memory. The number of parallel processor is 254. Explain why Amdahl's law for this situation is the same as for vector processing. Using Amdahl's law, plot the speed-up in execution time (on a log scale) as a function of the fraction p of the program that is parallelized. How much of the program (in terms of operations) must be parallelized to achieve speedups of 3, 10, and 250?

The problem is equivalent to vector processing because all processors have access to the same memory such that all operations of an explicit matrix operation (multiplication, addition, or subtraction) can be executed by any processor without the need of exchange of information between processors. The equivalent "vector" speed is $V = 254S$ for this case

$\tau_V = \frac{N}{S}(1-P) + \frac{N}{V}P$ **with** $V/S = 254$**.**

Speed up for vector computers (see class): $r = \frac{\tau_S}{\tau_V} = \left(1 - P + \frac{S}{V}P\right)^{-1}$.

The following figure show the plot of the speed up r as a function of the fraction p of the program which is vectorized.



For a given speedup the fraction of a program that has to be parallelized is given by

$$1 = r[1-(1-S/V)P] \quad => \quad P = \frac{1-1/r}{1-S/V}$$

To achieve speedups of 3, 50 and 60:

Fraction of operations that can remain scalar is

$$1-P=\frac{1/r-S/V}{1-S/V}$$

Speedup of 3:

$$1-P_3=\frac{1/3-1/254}{1-1/254}=\frac{254/3-1}{254-1}=0.331$$

Speedup of 10:

$$1-P_{50}=\frac{1/10-1/254}{1-1/254}=\frac{25.4-1}{254-1}=0.096$$

Speedup of 60:

$$1-P_{250}=\frac{1/250-1/254}{1-1/254}=\frac{254/250-1}{254-1}=0.000063$$

in other words about 66.9%, 91.4% and 99.9937% of all operations must be parallelized for a speedup of 3, 10, and 250 respectively.

2. Parallel Processing. (44%)

The total execution time of a fully parallelized simulation code of size N (no. of elements or grid points) on N_p parallel processors is

$$\tau_{tot}=2d\frac{N^{2/3}}{N_p^{\alpha}}\tau_c+\frac{N}{FN_p}\tau_s$$

where $N^{2/3}/N_p^{\alpha}$ is the number of elements which require inter processor communication, τ_c is the communication time required for one element, F is the execution efficiency for an individual processor, and τ_s is the time for the computation of an individual element. Assume $\tau_c=2\tau_s=10^{-6}$, $F=0.95$, and $N=10^9$.

a) $\alpha=2/3$, $1/2$, and 0 (for a 3D, 2D, and 1D decomposition of a 3-D domain).

Total communication time: $\tau_{comm}=4d\frac{N^{2/3}}{N_p^{\alpha}}\tau_s$

Total computation time: $\tau_{comp}=\frac{N}{FN_p}\tau_s$

Ratio R of the total communication time and the total computation time (for $N_p\in[1,10000]$) with $b=2$, $F=0.95$, and $N=10^9$:

$$R=\frac{\tau_{comm}}{\tau_{comp}}=4d\frac{FN_p^{1-\alpha}}{N^{1/3}} \tag{1}$$

The following plots show R (top - *this is the only plot required in the homework*),

the speedup (Amdahl's law, middle),

$$r=\frac{N\tau_s}{\tau_{tot}}=\left[4d\frac{N^{1/3}}{N_p^{\alpha}}+\frac{1}{FN_p}\right]^{-1}$$

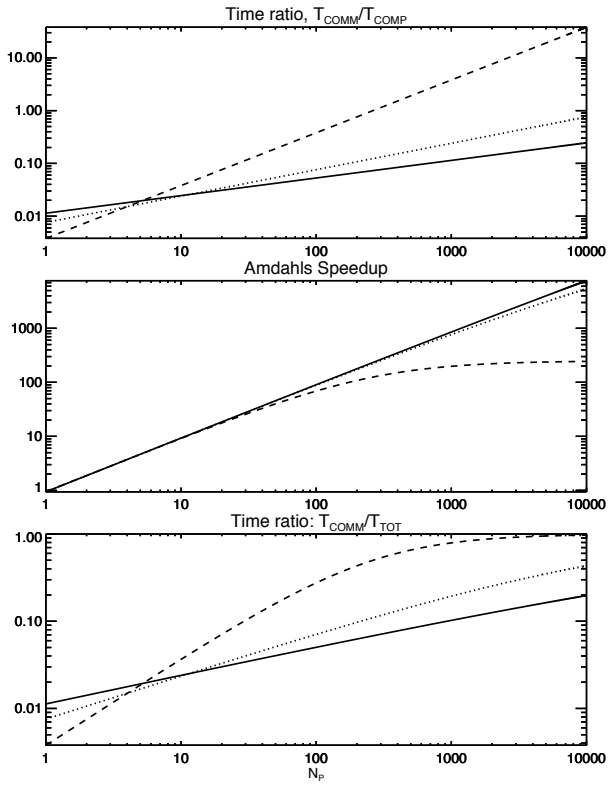
and the total communication time devided by the total execution time (because part a could be misunderstood to plot this quantity)

$$\frac{\tau_{comm}}{\tau_{tot}} = [1 + 1/R]^{-1}$$

The dashed line corresponds to the 1-D decomposition, the dotted line to the 2-D decomposition and the solid line to the 3D decomposition.

The time ratio plot for R (total communication to computation time) demonstrates that the 1-D decomposition becomes less efficient for processor numbers larger than a few 100. While the 2-D and 3-D decompositions are much better, the plot indicates still some difference between the 2-D and 3-D decompositions when the processor number is close to 10000. Note that the actual results are theoretical particularly for small numbers of processors because a practical implementation requires $N_p^{1/2}$ and $N_p^{1/3}$ to be integer or $N_p = k^6$ for any integer k.

The speedup (Amdahl's law) illustrates a similar property for the 3 domain decompositions. While the 3D decomposition results in an almost straight line in the double log plot 2D composition deviates already slightly from this and the 1D decomposition actually saturates. This means that the 1D case has a maximum execution speed for about 10000 processors and will not benefit from a further increase in processors. Finally the ratio of communication to total execution time illustrates that almost all time is spend on communication in the 1D decomposition for processor numbers of a few 1000 while this fraction remains quite low for the 2D and even lower for the 3D decomposition.



b) Number of processors for each α such that half of the total execution time is spend on communication (indication of inefficiency), i.e., $r = 1$.

From (1): $N_p^{1-\alpha} = N^{1/3}/(4dF)$

Therefore the **parallelization becomes inefficient for**

$$N_{p2} = \begin{cases} 6.7\cdot 10^5 & for \quad \alpha = 2/3 \quad or \quad 3-Ddecomposition \\ 1.7\cdot 10^4 & for \quad \alpha = 1/2 \quad or \quad 2-Ddecomposition \\ 263 & for \quad \alpha = 0 \quad or \quad 1-Ddecomposition \end{cases}$$

These numbers illustrate the same properties that the plots already revealed. Half of the execution time is spend on communication for only 263 processors in the 1D case. Any further increase in processor numbers consumes an increasing fraction of communication time. this is much better for the 2D composition with about 17000 processors and the plots also indicate some slowdown for about 10000 processors. Only the 3D decomposition can be pushed to about a million processors before becoming slightly inefficient.

Since here $\tau_{comm} = \tau_{comp}$ the total execution time for these 3 cases are

$$\tau_{tot,2} = 2\tau_{comp,2} = \frac{2N\tau_s}{FN_{p2}} = \frac{1.05\cdot 10^3}{N_{p2}} = \begin{cases} 1.6\cdot 10^{-3} & 3-Ddecomposition \\ 6.2\cdot 10^{-2} & 2-Ddecomposition \\ 4s & 1-Ddecomposition \end{cases}$$

3. Amdahl's law for parallel processing. (28%)

Consider a computer environment with N_n parallel nodes where each node has N_m parallel computing cores such that the total number of cores is $N_p = N_n N_m$. Each node has shared memory for the cores within the node. However, between nodes memory is distributed. Derive Amdahl's law for the speedup in this processing environment. What is the maximum speedup? For a fixed number of total processors is there an optimum distribution (configuration) for the N_n and N_m processors (for $N_m > 1$) in terms of the most efficient execution and what would this distribution be?

With the result from problem 1 we can regard execution within a node basically like a vector operation (shared memory) with Amdahl's law from problem 1. Execution time for distributed memory with N_n nodes

$$\tau_{tot} = N(1-P_n)\,\tilde{\tau}_n + \frac{N}{FN_n}P_n\tau_n + N_s\tau_{comm}$$

where τ_n is the elementary node execution time, i.e., the time a node takes for a single operation. $\tilde{\tau}_n$ represents the elementary node execution time for operations that cannot be parallelized between nodes such that these in all likelihood can also not be parallelized/vectorized within a node. Therefore $\tilde{\tau}_n = \tau_s = 1/S$ with S being the speed for a single core! Further P_n represents the fraction of the program that can be parallelized between nodes.

Now we need to evaluate τ_n. For each node with shared memory and $\tilde{N}$ operations the execution time (vector form of Amdahl's law):

$$\tau_{Node} = \frac{\tilde{N}}{S}(1-P_m) + \frac{\tilde{N}}{N_m S}P_m$$

such that the elementary node execution time (average for a single operation) is

$$\tau_n = \frac{\tau_{node}}{\tilde{N}} = \frac{1}{S}(1-P_m) + \frac{1}{N_m S}P_m$$

substitution of τ_n yields the total time as

$$\begin{aligned}\tau_{tot} &= \frac{N}{S}(1-P_n)+\frac{N}{FN_n}P_n\left[\frac{1}{S}(1-P_m)+\frac{1}{N_mS}P_m\right]+N_s\tau_{comm}\\ \frac{S\tau_{tot}}{N} &= (1-P_n)+\frac{1}{FN_n}P_n(1-P_m)+\frac{1}{FN_nN_m}P_nP_m+\frac{SN_s\tau_{comm}}{N}\end{aligned}$$

Considering $N_p = N_nN_m$ fixed the equation shows that larger numbers of nodes N_n are preferable to larger numbers of processors per node. This appears counter intuitive, however, any number of operations that cannot be parallelized within the node level (with shared memory) may be parallelized on the higher distributed memory level such that larger number of nodes will lead to faster execution of these operations. However, in practice it may well be that operations that cannot be parallelized on the node level have been duplicated on each node effectively increasing the number of operations in order to generate identical code on each node. A second reason not identified by the above result is the possibility of reduced need for communication on the distributed memory level if larger node numbers are used.

The related speedup is the ratio of total execution time on a single processor/core and execution in the parallel environment

$$R = \frac{N\tau_s}{\tau_{tot}} = \left[(1-P_n)+\frac{1}{FN_n}P_n(1-P_m)+\frac{1}{FN_nN_m}P_nP_m+\frac{SN_s\tau_{comm}}{N}\right]^{-1}$$