1.

Assume we have cores labeled $0, 1, 2, \ldots (p-1)$, each index denoted by the variable my_rank. We would likely to denote n integers between these p cores. Let $k = \lceil \frac{n}{p} \rceil$, note that $k * p$ need not equal n if p does not evenly divide n, in this case, all cores will be given equal work of k items with the exception of the last core, who will only take the remaining.

Thus, given core with index my_rank...

case my_rank$< (p-1)$:

Note that in the loop my_last_i is exclusive, while my_first_i is inclusive, therefore, my_last_i should be equal to my_first_i for the next core.

my_first_i $= i \times k$

my_last_i $= (i+1) \times k$

Note that we are adding an extra 1 to my_rank to convert from the 0 based indecies given by the core id to the 1 based indecies I used for my algebra. case my_rank$= (p-1)$:

This is the last core, we don't want to dip out of bounds of our array, thus simply terminate it with value n. As the array is 0 based, and my_last_i is exclusive, this is the value we want and not $n-1$.

my_first_i $= i \times k$

my_last_i $= n$

2.

Note that each subsequent core takes twice as long as the previous (given equal amounts of work), thus if we desire to have them take equal amounts of time, each subsequent core should do half as much work.

Thus we want $\Sigma_{i=0}^{p-1} \frac{k}{2^i} = n$ for some constant k, the amount of work performed by the initial unit (note that $\frac{k}{2^0} = k$). Doing some simple algebra we find $k = n/(\Sigma_{i=0}^{p-1} \frac{1}{2^i})$. Note that $k \in \mathbb{R}$, we will take the ceiling as before at each step. Then, my_first_i $= \lceil \sum_{i=0}^{\text{my\_rank}} \frac{k}{2^i} \rceil$ and my_last_i $= \lceil \sum_{i=0}^{\text{my\_rank}+1} \frac{k}{2^i} \rceil$ with the exception for my_rank$= (p-1)$ then my_last_i = n. For example, if we had an infinite number of cores, $k = \frac{1}{2}$.