Because we have n computers, we can assign task n_i to a normal computer as soon as it finishes s_i on the supercomputer. We can write our total time T as

$$T(n) = max(s_1 + n_1,\ s_1 + s_2 + n_2,\ \ldots,\ (\sum_{i=1}^{n} s_i) + n_n)$$

Prove by Contradiction

Based on our algorithm, we sort the jobs by n_i in descending order

Let's say maximum is at index k, we have $T(n) = (\sum_{i=1}^{k} s_i) + n_k$

1. Assume we can make $T(n)$ smaller by executing job k later, say index h, $h > k$

Since n is a sorted array, we know $n_k \geq n_h$, the array becomes unsorted

We have

$T'(n) = max(\ldots,\ (\sum_{i=1}^{h} s_i) - s_k + s_k + n_k,\ \ldots)$

Since $h > k$,

$(\sum_{i=1}^{h} s_i) + n_k > (\sum_{i=1}^{k} s_i) + n_k$

which means $T'(n) > T(n)$

Therefore, moving job k backward will make T(n) larger

Which contradict with our assumption

2. Assume we can make $T(n)$ smaller by executing job k earlier, say index h, $h < k$

Since n is a sorted array, we know $n_h \geq n_k$, the array becomes unsorted

We have

$T'(n) = max(\ldots,\ (\sum_{i=1}^{k} s_i) + n_h,\ \ldots)$

Since $h > k$, we know $n_h \geq n_k$

$(\sum_{i=1}^{k} s_i) + n_h \geq (\sum_{i=1}^{k} s_i) + n_k$

which means $T'(n) \geq T(n)$

Therefore, moving job k forward will make T(n) equal or larger

Which contradict with our assumption

3. Assume we can make $T(n)$ smaller by switching any jobs j and l, $j < k < l$
Since n is a sorted array, we know $n_j \geq n_k \geq n_l$, the array becomes unsorted
We have

$$T'(n) = max(\dots,\ (\sum_{i=1}^{l} s_i) + n_j,\ \dots)$$

Since $j < k < l$, we know $n_j \geq n_k \geq n_l$

$$(\sum_{i=1}^{l} s_i) + n_j \geq (\sum_{i=1}^{k} s_i) + n_k$$

which means $T'(n) \geq T(n)$
Therefore, switching any jobs before and after k will make T(n) larger
Which contradict with our assumption

4. Assume we can make $T(n)$ smaller by switching any jobs j and l, $j < l < k$ or $k < j < l$
This will not affect k's term Thus $T'(n) >= T(n)$

Since we know with a sorted array, any action to make it unsorted will not make $T(n)$ smaller, and we also know k is an arbitrary value from 1 to n, thus the optimal solution for us is to keep it sorted in decending order.