**Problem 1:**

double r[threadCount];

s = 0;

#pragma omp parallel for

for (i=0; i < n; i++) {

r[omp\_get\_thread\_num( )] = r[omp\_get\_thread\_num( )] + a[i];

}

for (i=0; i < threadCount; i++) { s += [i]; }

Description of Method:

There is an entry in the array r for each thread. Each thread is given a subset of n numbers and computes a local sum of those numbers. After the local sums have been completed, all of the local sums are added together.

Each core will perform n/P units of work

There are P communications (occur during final sum)

General Equations:

Solution:

**Problem 2:**

#pragma omp parallel for reduction(+:r)

for (i=0; i < n; i++) {

r = r + a[i];

}

Description of Method:

Sum locally in core where possible, then use a binary reduction to combine the local sums from each node.

**Problem 3:**

#pragma omp parallel for simd reduction(+:r)

for (i=0; i < n; i++) { r = r + a[i]; }

Description of Method:   
 Same as 2, but with the assumption that four adds occur at a time

**Problem 4:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **p** | **1** | **2** | **3** | **4** | **256** |
| **Method 1** | 1 | 4 | 9 | 16 | 65,536 |
| **Method 2** | 0 | 2 | 4.75 | 8 | 2048 |
| **Method 3** | 0 | .5 | 1.188 | 2 | 512 |