Sreepathy Jayanand

CED17I038

High Performance Computing LAB - 6

Analysis of parallelization fraction

**Processor Specifications:**

* Intel i5 - 5350U
* Base clock speed - 1.80GHz
* Max clock speed - 2.90GHz
* Physical Cores - 2
* Threads / Core - 2
* Logical Cores - 4
* L1 cache size (per core) - 64 KB
* L2 cache size (per core) - 256 KB
* L3 cache size - 3 MB
* Main memory - 8GB

**Question 1**: Addition of N numbers (Reduction)

Input size : 4 x 1e8

Segmentation fault : 1 x 1e9

Let the total processes = total,

n = array size

Strategy: Each processor does the partial addition of (n / total) indices of the array.

Each process has a separate partial sum variable which has the cumulative sum of the indices that processes were assigned to.

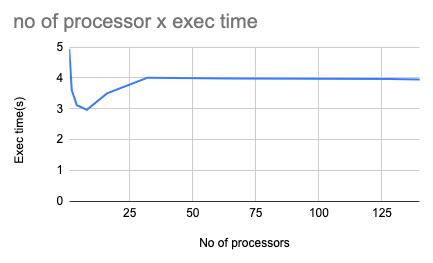
Now the total sum = ∑psum for all processes.

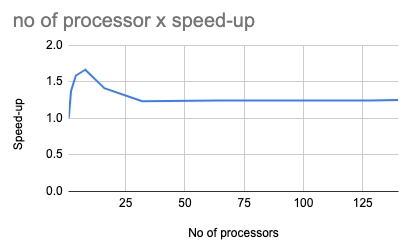
Since total\_sum is only one variable and since each process tries to update it simultaneously there is an inherent race condition and hence we use reduction clause to take care of that.

This ensures that when 1 process is reading unless it updates no other process can read “total\_ sum” variable. This ensures atomicity of the update operation.

Table 1:

| No of processors | Execution time(s) |
| --- | --- |
| 1 | 4.949186 |
| 2 | 3.601761 |
| 4 | 3.125625 |
| 8 | 2.976664 |
| 16 | 3.505750 |
| 32 | 4.018541 |
| 64 | 3.986381 |
| 128 | 3.970641 |
| 140 | 3.952952 |





The most efficient processor size for this program is 8

According to Amdahl’s law :

Speed up = 1/((1-p) + p/N)

Or

Parallelization factor = (N - N/S.U)/(N - 1)

So here N = 8;

Speed Up = 4.94/2.97 = 1.68 for N = 8

PF = (8 - 8/1.68)/7 = .46

**Question 2**: Vector dot product(Reduction)

Input size : 4 x 1e8 per vector

Segmentation fault : 1 x 1e9 per vector

Let the total processes = total,

n = array size of the vector

Strategy: Each processor does the partial addition of the multiplication of (n / total) indices of both the arrays.

Each process has a separate partial sum variable which has the cumulative sum of products the indices that processes were assigned to.

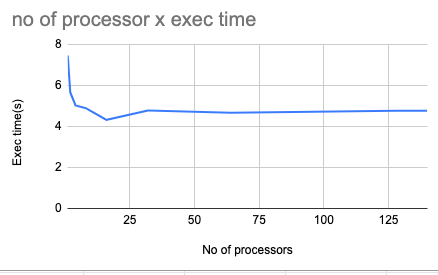
Now the total sum = ∑psum for all processes.

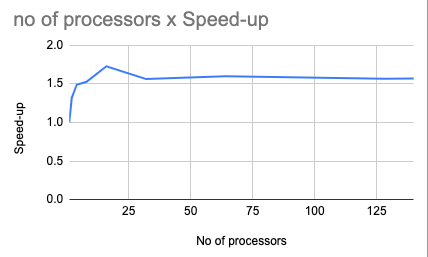
Since total\_sum is only one variable and since each process tries to update it simultaneously there is an inherent race condition and hence we use reduction clause to take care of that.

This ensures that when 1 process is reading unless it updates no other process can read “total\_ sum” variable. This ensures atomicity of the update operation.

Table 2:

| No of processors | Execution time(s) |
| --- | --- |
| 1 | 7.474893 |
| 2 | 5.670463 |
| 4 | 5.037293 |
| 8 | 4.908580 |
| 16 | 4.337245 |
| 32 | 4.793618 |
| 64 | 4.692683 |
| 128 | 4.783163 |
| 140 | 4.775917 |





The most efficient processor size for this program is 16

According to Amdahl’s law :

Speed up = 1/((1-p) + p/N)

Or

Parallelization factor = (N - N/S.U)/(N - 1)

So here N = 16;

Speed Up = 7.47/4.33 = 1.72 for N = 16

PF = (16 - 16/1.72)/15 = .44