Jamal Alkurdi

CSC 4760

2/10/2025

HW #1

1. Devise an algorithm for the functions that calculate my\_first\_i and my\_last\_i in the global sum example Remember that each core should be assigned roughly the same number of elements of computations in the loop. H int: First consider the case when is evenly divisible by *p*. Note: You must also handle the case when is not evenly divisible by *p*.  
      
   Implement your formulas in C/C++. An example run should look like the following:  
     
   $ map 1234 5  
   core 0: my\_first\_i = 0, my\_last\_i = 245  
   core 1: my\_first\_i = 246, my\_last\_i = 491  
   core 2: my\_first\_i = 492, my\_last\_i = 737  
   core 3: my\_first\_i = 738, my\_last\_i = 983  
   core 4: my\_first\_i = 984, my\_last\_i = 1233  
     
   The first parameter is the problem size and the second parameter is the number of cores involved in the computation.  
   Note: if you can do a better job than the method used above, then go for it.  
   Your deliverable is the programs C/C++ source code that can be compiled and executed.
2. Consider the tree structured global sum. Answer the following questions.  
   a. What is the formula for the number of levels (or rounds) of computation for  
    elements in the array?  
   b. Notice that at each level, except the first, not all cores are involved in  
   the computation. How would you determine which cores would "drop out"?
3. Derive formulas for the number of receives and additions that core 0 carries out using  
   a. the original pseudocode for a global sum, and  
   b. the tree-structured global sum.

Make a table showing the numbers of receives/additions carried out by core 0 when the two sums are used with 2, 4, 8, . . . , 1024 cores.

Note: don't forget to include both the formula and the table for part b.

1. C++ file along with makefile included in the submitted compressed folder.

2.

a. log(p)2 where p = # of cores.

b. Cores that aren’t a multiple of 2^r +1 are dropped.

3.

a. Receives = p-1, addition = p-1.

b. Receives = log2(p), addition log2(p)

|  |  |  |
| --- | --- | --- |
| # of cores | Original | Tree |
| 2 | 2-1 = 1 | Log2 (2) = 1 |
| 4 | 4 – 1 = 3 | Log2(4) = 2 |
| 8 | 8 – 1 = 7 | Log2(8) = 3 |
| 16 | 15 | 4 |
| 32 | 31 | 5 |
| 64 | 64 | 6 |
| 128 | 127 | 7 |
| 256 | 255 | 8 |
| 512 | 511 | 9 |
| 1024 | 1023 | 10 |