2014 Spring Q6: Parallel Programming - MPI

Eitan Lees

July 27, 2016

A character array s contains the text of a book, and consists of exactly 10,000,000 characters. The only characters are lowercase alphabetic characters ('a' through 'z') and blanks. We may regard the text as a sequence of words of various lengths, where words are strings of non-blanks separated by one or more blanks.

a Describe an algorithm to count the number of words in **s**.
Answer: *To count the number of words in **s** we need to increment a counter if the current position is a character and the previous position is a space . Also a correction for the first word needs to be made where you check if your first position is a character.*

b We wish to use parallel processing to count the number of words in s. We are going to try to do parallel processing without using MPI. Suppose we have 10 separate computers, we cut the text into 10 parts, we run the same program on each computer, and we take the partial sums from each computer and add them up by hand. Even though the computation might be faster, the result will probably be slightly incorrect. Why is the sum of the partial word counts probably not the exact number of words in the text?
Answer: *If the file is split in the middle of a word our algorithm will over count the number of words. The first processor would correctly identify the starting word of the file, but the rest would not be able to.*

c Now we want to use MPI to carry out the task. Again, we plan to use 10 processes, and each process will only have access to 1/10 of the data. The fact that the MPI processes can communicate is very important, and means that we can avoid the error noticed in the previous section. Focus on a single MPI process with identifier ID, and explain what communication is necessary in order for this process to verify or adjust the partial word count it performs.
Answer: *To correct for this error each processor can request the status of the final position of the previous processor. If the starting position of the current process and the ending position of the previous process are both characters then the count needs to be decreased by 1.*

d Once each process has done its work, the partial counts need to be combined into a single result. Give a statement in C, C++, or Fortran, including

all the arguments, that implements this function call. Indicate which of the arguments to the function represent the actual variables containing the partial and final count information.
Answer: *To total up the local counts you would use* `MPI_REDUCE`*. All of the local counts should be sent to process 0, be summed and saved as a new variable.*

```
MPI_Reduce ( &local_count, &global_count, 1, MPI_INT, MPI_SUM,
0, MPI_COMM_WORLD );
```

would perform these tasks. Here ***local_count*** *represents the partial sums from each process and* ***global_count*** *is the final count.*