The Sieve of Eratosthenes using MPI

Devin Delfino

Comp 401: Senior Seminar

4/16/2015

Outline

1. Parallel Programming and MPI
2. The Sequential Algorithm
3. The Parallel Algorithm
4. Sequential vs. Parallel Comparison

Outline

1. Parallel Programming and MPI
2. The Sequential Algorithm
3. The Parallel Algorithm
4. Sequential vs. Parallel Comparison

Introduction

- Parallel Computing is the use of multiple computers or processors to reduce the time needed to solve a single computational problem.
- A task is a single program including local memory and a collection of input/output ports.
- A channel is a message queue between two tasks used for communication

Ian Foster's Design Methodology

1. Partitioning - the process of dividing the computations and data into pieces.
2. Communication - channels between tasks allow communication between them
   - Local - a task's computation requires values from a small number of other tasks
   - Global - many tasks must contribute values to perform a computation
3. Agglomeration - grouping tasks in order to improve performance and reduce overhead.
4. Mapping - assigning processes or tasks to specific processors or computers

Message Passing Interface (MPI)

- The most popular message-passing library standard
- There are many free implementations of MPI libraries, including OpenMPI and MPICH
- Integrates sequential language with functions that allow processes to communicate with each other

Outline

1. Parallel Programming and MPI
2. The Sequential Algorithm
3. The Parallel Algorithm
4. Sequential vs. Parallel Comparison

The Sequential Algorithm

The Sieve of Eratosthenes

Summary: Finds all primes between 2 and n, inclusive

Create a list of natural numbers 2, 3, ... , n, none of which are marked
Set k equal to the first prime number, 2
while $k^2 \leq n$ **do**
  Mark all multiples of k between k^2 and n
  Set k to the smallest unmarked number greater than the current k
end while

Outline

1. Parallel Programming and MPI
2. The Sequential Algorithm
3. The Parallel Algorithm
4. Sequential vs. Parallel Comparison

The Parallel Algorithm: Block Decomposition

- n is the size of array and p is the number of processors
- Divide the array into p contiguous blocks of roughly equal size
- If n is divisible by p, then there will be p blocks of size n/p. Otherwise, there will be p blocks of with sizes of either $\lfloor n/p \rfloor$ or $\lceil n/p \rceil$.
- Common data block computations include finding the first/last element controlled by a given process

The Parallel Algorithm: Block Decomposition, cont.

- n Suppose is the size of array and p is the number of processors
- First element controlled by process i:

$$\lfloor in/p \rfloor$$

- Last element controlled by process i:

$$\lfloor (i+1)n/p \rfloor - 1$$

Block Decomposition Example, $n = 121$, $p = 3$

Process 0:

- First index: $\lfloor (0)(121)/3 \rfloor = 0$
- Last index: $\lfloor (0+1)121/3 \rfloor - 1 = \lfloor 40\frac{1}{3} \rfloor - 1 = 39$
- Size: last - first + 1 $= \lfloor n/p \rfloor = 40$.

Process 1:

- First index: $\lfloor (1)(121)/3 \rfloor = \lfloor 40\frac{1}{3} \rfloor = 40$
- Last index: $\lfloor (1+1)121/3 \rfloor - 1 = \lfloor 80\frac{2}{3} \rfloor - 1 = 79$
- Size: last - first + 1 $= 40 = \lfloor n/p \rfloor$.

Process 2:

- First index: $\lfloor (2)(121)/3 \rfloor = \lfloor 80\frac{2}{3} \rfloor = 80$
- Last index: $\lfloor (2+1)121/3 \rfloor - 1 = \lfloor 121 \rfloor - 1 = 120$
- Size: last - first + 1 $= 41 = \lceil n/p \rceil$.

Developing the Algorithm, Step 1

1. Create a list of natural numbers 2, 3, ... , n, none of which are marked
    - Each task handles a specific block of the entire array
    - Use the formulas to determine the numbers represented by the first and last elements of the block along with its size
    - Keep in mind the difference between the local index for the block and the global index for the entire array
    - To minimize communication between tasks, make task 0 responsible for finding the next value of k
    - This can be done by ensuring $n/p > \sqrt{n}$

2. Set k equal to the first prime number, 2
   - k is set to the first prime number for all tasks
   - After each iteration, each task must be told the next value of k

Developing the Algorithm, Step 3

While $k^2 \leq n$

3a. Mark all multiples of k between k^2 and n

- Must determine the first multiple of k in the given block (if it's greater than k^2)
- From the first multiple of k, mark every k^{th} element

3b. Set k to the smallest unmarked number greater than the current k

- The smallest unmarked number greater than the current k is always part of the block belonging to task 0
- Task 0 finds the next value of k, and broadcasts it to the rest of the tasks

Outline

1. Parallel Programming and MPI
2. The Sequential Algorithm
3. The Parallel Algorithm
4. Sequential vs. Parallel Comparison

Sequential vs. Parallel

References

- MICHAEL J. QUINN.
  Parallel Programming in C with MPI and OpenMP.
  McGraw-Hill Science/Engineering/Math, 2003.