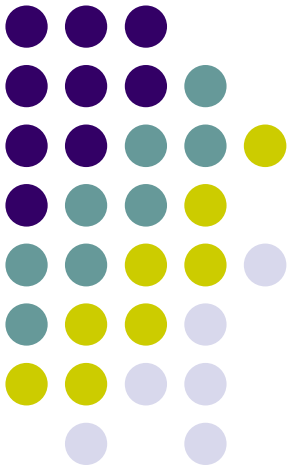
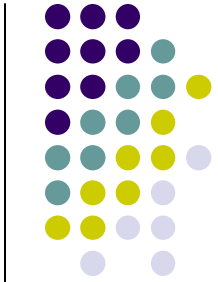
Parallel Prime Sieve

Finding Prime Numbers

David J. Wirian

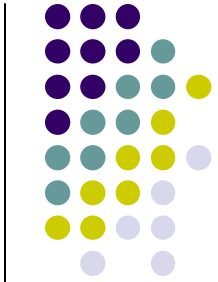


Outline



- Use of Prime Numbers
- Introduction of Sieve of Eratosthenes
- Using Parallel
- Data Decomposition
- Optimizations/Improvements

Use of Prime Numbers (1)



- Before 19th of century, there was little use for prime numbers.
- In the 19th of century, there was need for secrecy, especially during times of war.
- Messages and files needed to be encrypted and decrypted (related to Cryptography).
- Several public-key cryptography algorithm, such as RSA or Diffie Hellman key exchange are based on large prime numbers.
- It was found that using two prime numbers multiplied together makes a much better key than any old number, because it has only four factors. One, itself, and the two primes that it is a product of. This makes the code much harder to crack.
- Size of Prime Numbers used dictate how secure the encryption will be. e.g.

  5 digits in length (40-bit encryption) yields about 1.1 trillion possible results.

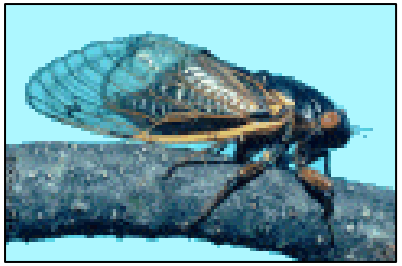
  7 digits in length (56-bit encryption) yields about 72 quadrillion possible results.

  16 digits in length (128-bit encryption) yields about 340,282,366,920,938,463,463,374,607,431,768,211,456 possible results

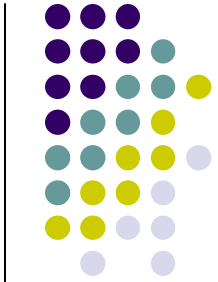
Use of Prime Numbers (2)

- However, the use of prime numbers is only in the computer world, and we still do not have any use for primes in the physical world.

There is a research about insects. Some insects in North America will live in the ground for a number of years, and come out after 13 or 17 years. By emerging at these times, it makes harder for predators to adapt and kill the insects and therefore more of them survive. E.g. Cicada Bug.

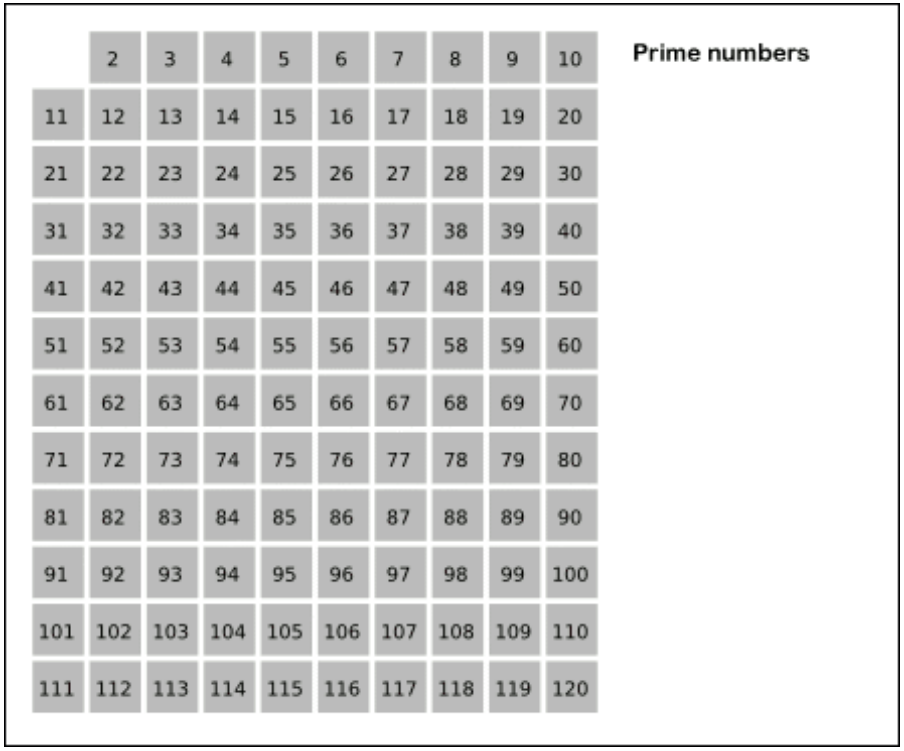


Sieve of Eratosthenes (1)



- A classical method of extracting prime numbers is by the sieve of Eratosthenes more than two thousand years ago (Bokhari, 1987).
- The 1st number of prime is 2 and it is kept.
- All multiples of this number are deleted as they cannot be prime.
- Repeat with each remaining number. The algorithm removes non primes, leaving only primes.

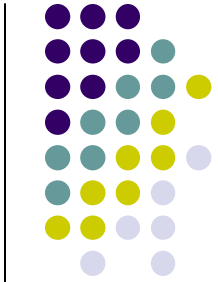
Sieve of Eratosthenes (2)



	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120

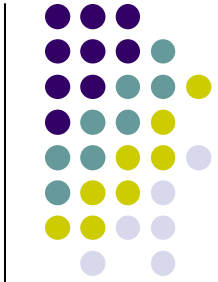
- Create a list of natural numbers, 2,3, 4, .., *n*, none of which is marked.
- Set *k* to 2, the first unmarked number on the list.
- Repeat: (key point)
  a. Mark all multiples of k between k^2 and *n*
  b. Find the smallest number greater than *k* that is unmarked. Set *k* to this new value. Until $k^2 > n$
- The unmarked numbers are primes.

Sieve of Eratosthenes (3)



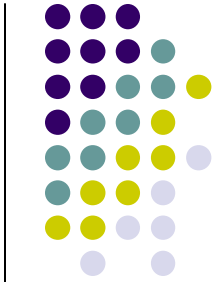
- However, the sieve of Eratosthenes is not practical for identifying large prime numbers with hundreds of digits.
- The algorithm has complexity O(n ln ln n), and n is exponential in the number of digits.
- A modified form of the sieve is needed.
- Use Parallel computation – the elements representing multiples of a particular prime *k* are marked as composite.

Using Parallel



- If a primitive task represents each integer, then two communication are needed to perform the repeat part each iteration of the repeat, until loop.
- Reduction needed each iteration in order to determine the new value of *k.*
- Then we broadcast to inform all the tasks of the new value of *k.*
- This will take many reduction and broadcast operations.
- New version of the parallel algorithm that requires less computation and less communication than original parallel algorithm.

Data Decomposition (1)



- Interleaved Data Decomposition:

  Suppose *p* is the number of processes.

  process 0 is responsible for the natural numbers 2, 2+*p,* 2+2*p*, …

  process 1 is responsible for the natural numbers 3, 3+*p,* 3+2*p*, …

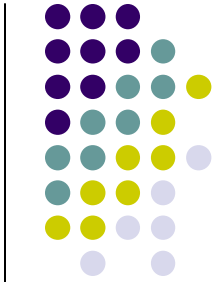
  and so on.

  (+) : easy to determine which process controls that index

  (process *i* mod *p)*

  (-) : lead imbalances among the processes, also still requires many reductions/broadcasts

Data Decomposition (2)



- Block Data Decomposition:
  Divide array into *p* contiguous blocks of equal size.

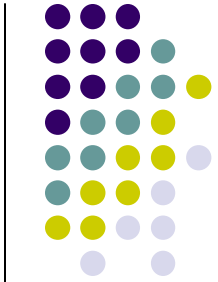
  Suppose *n* is the number of elements and *p* is the number of processes.
  The first element controlled by process *i* is

  $$[\, i\, n/p \,]$$

  The last element controlled by process *i* is the element immediately before the first element controlled by process *i* + 1:

  $$[(i\ + 1)\, n/p] - 1$$

Data Decomposition (3)

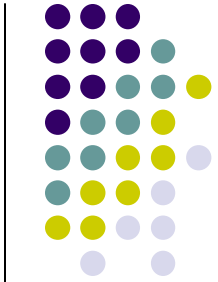


- How does block decomposition affect the implementation of the parallel algorithm ?

  First: The largest prime used to sieve up to n is $\sqrt{n}$, then finding next value of *k* requires no communications at all – it saves a reduction step.

  Second: Block decomposition speeds the marking of cells representing multiples of *k*, rather than check each array element to see if it represents an integer that is a multiple of *k*.

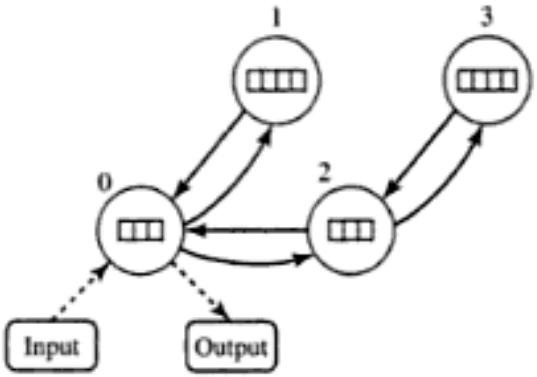
Function MPI_Bcast



In the case of parallel sieve algorithm, process 0 needs to broadcast a single integer, *k*, to all other processes.

After this function has executed, every process has an up-to-date value of *k* and is able to evaluate the termination condition in the repeat … until loop.

Analysis



The dotted arrows represents channels used for I/O.

The curved arrows represent channels used for the broadcast step

The straight solid arrows represent channels used for the reduction step

- Execution Time:

  Let X represents the time needed to mark a particular cell as being the multiple of a prime.

  Single data value is broadcast each iteration, then cost of broadcast is

  $\lambda [\log p]$, where λ is message latency

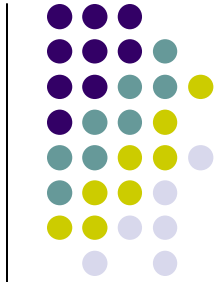
  Number of primes between 2 and n is

  $(n / \ln n)$

  number of loop iterations: $\sqrt{n} / \ln \sqrt{n}$

  Expected execution time:

  $X^{((n \ln \ln n)/p + (\sqrt{n} / \ln \sqrt{n}) \lambda[\log p])}$



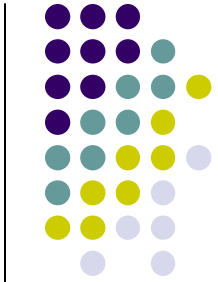
```
for (i = 0; i < size; i++)
    marked[i] = 0;
    if (!id) index = 0;
    prime = 2;
    do  {
        if (prime * prime > low_value)
            first = prime * prime - low_value;
        else {
            if (!(low_value % prime))
                first = 0;
            else
                first = prime - (low_value % prime);
         }
         for (i = first; i < size; i += prime)
                  marked[i] = 1;
         if (!id)  {
             while (marked[++index]);
             prime = index + 2;
         }
         MPI_Bcast (&prime,  1, MPI_INT, 0, MPI_COMM_WORLD);
    }
    while (prime * prime <= n);
    count = 0;
    for (i = 0; i < size; i++)
        if (!marked[i])
            count++;
    MPI_Reduce (&count, &global_count, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
```

Sieving

Process 0 finds next prime by locating the next unmarked location in array

Process 0 broadcast the value of next prime to other processes

Improvements



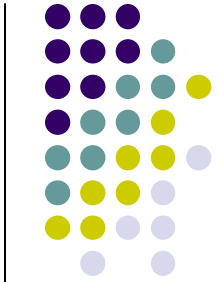
- Delete even integers – since 2 is the only even number in prime numbers
- Eliminate broadcast – improves speed of parallel algorithm

  Before finding prime numbers from 3 through n, each task will use the sequential algorithm to find the primes from 3 through $\sqrt{n}$.

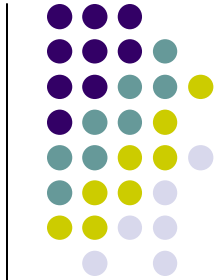
  Once this has been done, each task now has its own private copy of an array containing all the primes between 3 and $\sqrt{n}$.
- Reorganize Loops

  Each process is marking widely dispersed elements of a very large array, leading to poor cache hit rate.

References



- http://en.wikipedia.org/wiki/Sieve_of_Eratosthenes
- Quinn, M.J. Parallel Programming in C with MPI and OpenMP. pg 115-134. McGraw-Hill Professional, 2004.
- Tabak, E.G. Why do cicadas have prime life-spans?. CIMS. 2001
- B. Wilkinson and M. Allen, Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers, Prentice Hall.
- http://mathforum.org
- Hwang, S., Chung, K., Kim, D. Load Balanced Parallel Prime Number Generator with Sieve of Eratosthenes on Cluster Computers. IEEE 2007.
- Sorenson, J., Parberry, I. Two Fast Parallel Prime Number Sieves. Academy Press, Inc. 1994.



End of Presentation

Thank You

Any Question ?