Sieve of Eratosthenes

- Find the prime numbers less than or equal to some positive integer n
  - Begin with a list of natural numbers 2, 3, 4, …, n
  - Remove composite numbers from the list by striking multiples of 2, 3, 5, and successive primes
  - After each striking, next unmarked natural number is prime
  - Sieve terminates after multiples of largest prime less than or equal to   have been struck from the list
- Sequential algorithm
  - Boolean array containing numbers being sieved,integer corresponding largest prime found so far, integer keeping track of multiples of current prime

A Control-Parallel Approach

- *Control parallelism* refers to applying different operations to different data elements simultaneously
  - Shared-memory MIMD, Distributed-memory MIMD
- Control-parallel Sieve
  - Each processor works with a different prime, and is responsible for striking multiples of that prime and identifying a new prime number
  - Each processor starts marking…
  - Shared memory contain boolean array containing numbers being sieved,integer corresponding largest prime found so far
  - PE's local memories contain integer keeping track of multiples of its current prime (each working with different prime)

A Control-Parallel Approach (cont.)

- Problems and inefficiencies:
  - Processor accesses variable containing current prime, searches for next unmarked value, then updates variable containing current prime
    - Two processors could do this at once
  - Processor could end up sieving multiples of a composite number
- How much speedup can we get?
  - Suppose n = 1000
  - Sequential algorithm
    - Multiples of 2: ((1000–4)+1)/2=997/2=498
    - Multiples of 3: ((1000–9)+1)/3=992/3=330
    - Sum = 1411 (number of "steps")
  - 2 PEs gives speedup 1411/706=2.00
  - 3 PEs gives speedup 1411/499=2.83
  - 4 PEs is same,so upper bound is 2.83

A Data-Parallel Approach

- *Data parallelism* refers to using multiple PEs to apply the same operation to different data elements simultaneously
  - Shared-memory MIMD, Distributed-memory MIMD, Distributed-memory SIMD
- Data-parallel Sieve
  - Each processor works with a same prime, and is responsible for striking multiples of that prime from a segment of the array of natural numbers
  - Assume we have p processors, where p << sqrt(n)
    - Each processor gets no more than ceiling(n/p) natural numbers
    - All primes less than sqrt(n), as well as first prime greater than sqrt(n) are in list controlled by first processor

A Data-Parallel Approach (cont.)

- Data-parallel Sieve (cont.)
  - Algorithm
    - Processor 1 finds next prime, broadcasts it to all PEs
    - Each PE goes through their part of the array, striking multiples of that prime (performing same operation)
    - Continues until first processor reaches a prime greater than sqrt(n)
- How much speedup can we get?
  - Suppose n = 1,000,000
  - There are 168 primes less than 1,000, the largest of which is 997
  - Maximum execution time = (ceil(ceil(1,000,000/50)/2)+ ceil(ceil(1,000,000/50)/3)+ ceil(ceil(1,000,000/50)/5)…)etime
  - Communication time = 168(50–1)ctime

A Data-Parallel Approach (cont.)

- How much speedup can we get? (cont.)
  - Speedup is not directly proportional to the number of PEs — it's highest at 11 PEs
    - Computation time is inversely proportional to the number of processors used
    - Communication time increases linearly
    - After 11 processors, increase in communication time is higher than decrease in computation time, and total execution time increases
  - How about I/O time?
    - Have to output 78,498 primes!
    - I/O time is constant because output must be performed sequentially
    - This sequential code limits the speedup
    - Amdahl's law says that the fraction of operations that must be performed sequentially limits the maximum speedup possible (more on this later in the course)