Finding Prime Numbers using MPI and C

Andrew Wantuch

Fall 2011

Presented on 12/8/11

CSE 633

Background Info

- A prime number is a natural number that has exactly two distinct natural number divisors, 1 and itself.
- There are infinitely many primes.
- Smallest prime is 2
- The largest known prime is $2^{43,112,609} - 1$

Some Primes

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61,
67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131,
137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193,
197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269,
271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349,
353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421,
431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491,
499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577,
587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647,
653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733,
739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823,
827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911,
919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997

Algorithm

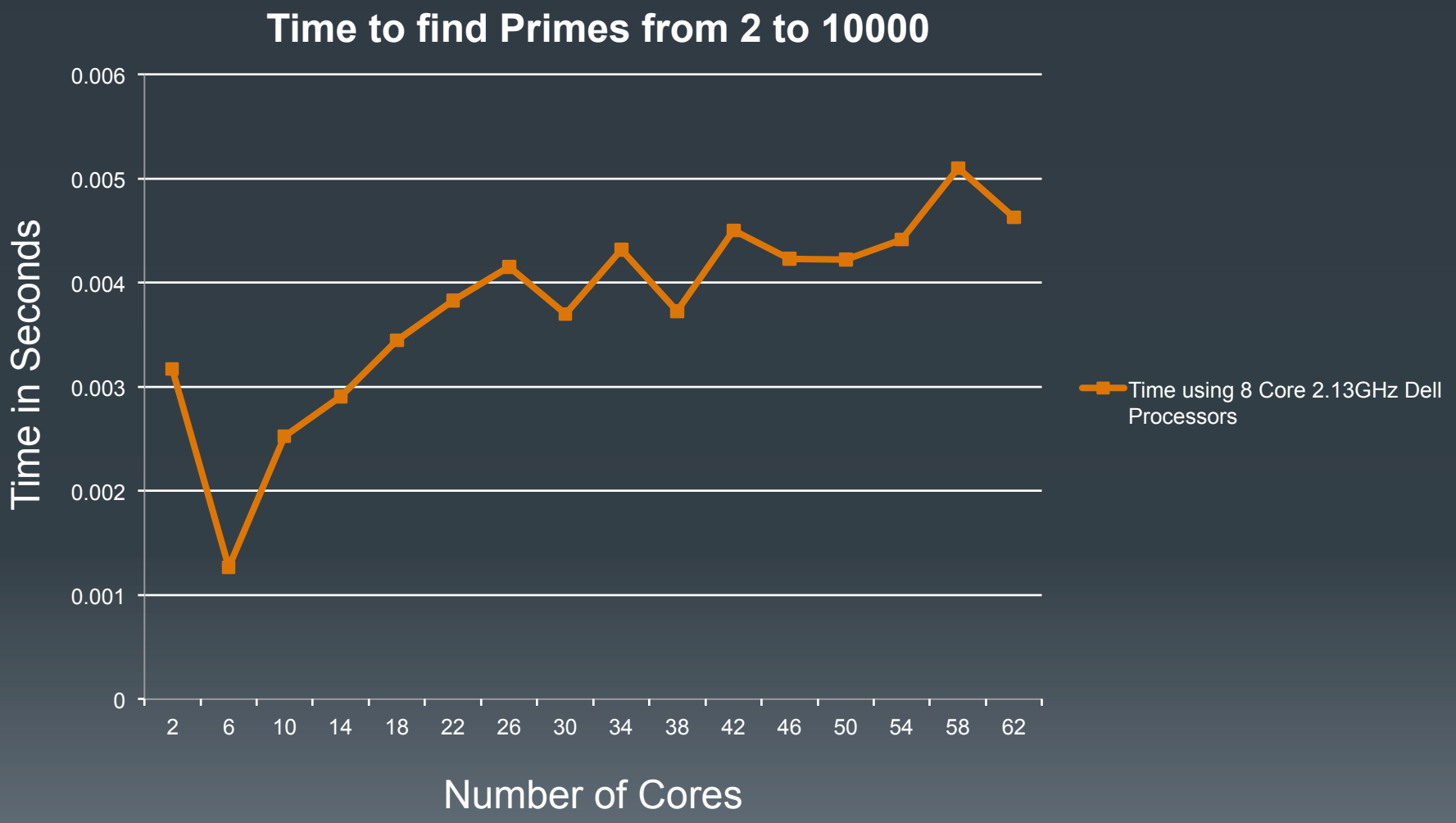
- (Almost) Brute Force
  - Small amount of optimization, but there are still much more efficient algorithms out there.
    - I'm worried more about learning the tradeoffs of parallel computing, so I'm not going to devote a lot of time to optimization.
- For each number n, check if it is divisible by every integer d s.t. $1 < d \leq \sqrt{n}$
- If not, n is prime.

Plan

- Use MPI to distribute workload.
  - If there are n workers, the i^{th} node starts with the $2i-1^{th}$ value.
  - Worker then checks every 2n values from its start position until all numbers have been checked.
  - Worker sends found primes to master node as soon as they are found.
    - Not the best way if finding small primes. That's boring and there is a relatively small number of small primes compared to large ones, so who cares.

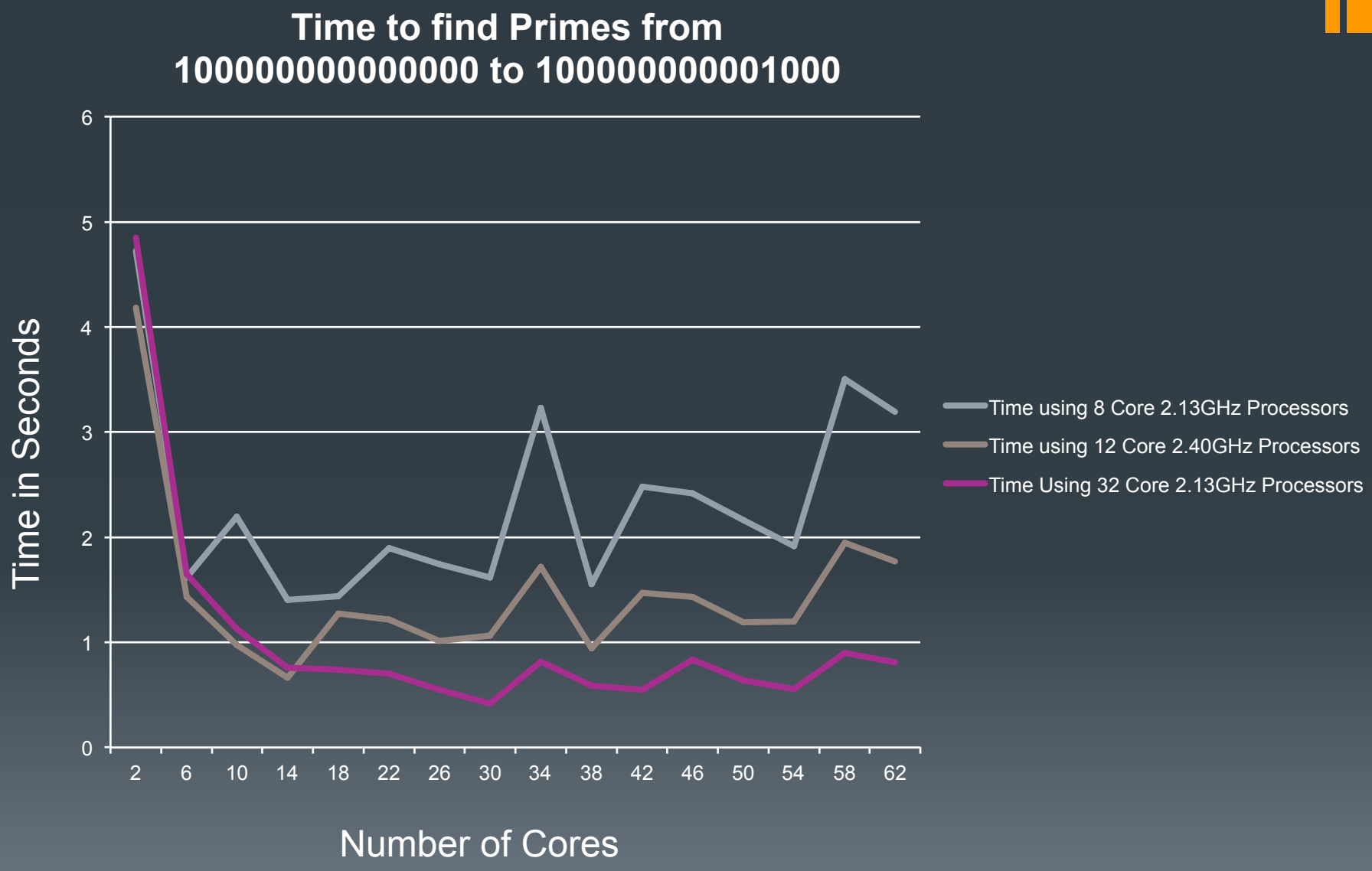
Experiment Details

- Processors are connected using InfiniBand.
- Times given are averages of 3 runs.
- Time is in seconds.
- Each of the two experiments finds primes in a different range of numbers.
  - One finds small numbers
  - One finds large numbers


Time to find Primes from 2 to 10000
Time in Seconds
0.006
0.005
0.004
0.003
0.002
0.001
0
2
6
10
14
18
22
26
30
34
38
42
46
50
54
58
62
Number of Cores
Time using 8 Core 2.13GHz Dell Processors

Observations

- Jobs with very small numbers are bound by communication time.
  - Since sequential runtime is so small, the time to send found primes to the head node makes the program take longer with more nodes, and makes adding processors slow down the program's runtime.
- Parallel execution of these computations is impractical.
  - Speedup is observed by using a small cluster, but it doesn't scale well at all.
  - You are better off with one processor than even a remotely large cluster.


Time to find Primes from 100000000000000 to 100000000001000
Time in Seconds
0
1
2
3
4
5
6
2
6
10
14
18
22
26
30
34
38
42
46
50
54
58
62
Number of Cores
Time using 8 Core 2.13GHz Processors
Time using 12 Core 2.40GHz Processors
Time Using 32 Core 2.13GHz Processors

Observations

- Jobs with larger numbers as input are bound by sequential computation time for a small number of processors, but eventually adding processors causes communication time to take over.
  - Sequential runtime with large numbers is much larger, so it scales much better than with small numbers as input.
- Inter-node communication has a much larger effect on runtime than intra-node communication.
- With infinitely large numbers, communication times would be negligible.
- Unlike with job requiring very little sequential computation and a lot of communication, this job achieved speedup with large numbers of processors.

Interesting Findings

- 2 is prime.
- 3 is prime.
- 4 is not prime.

Interesting Findings Continued

- The same program can be either practical or impractical depending on the input.
  - Large values for input are good.
  - Small values for input are bad.

General Things I Learned About

- How to write programs using MPI in C.
- The difference of difficulty involved with getting a parallel program to work properly compared to a sequential program.
- Tradeoffs associated with using more/less processors.
- Tradeoffs associated with using different types of nodes.
- Different factors that affect whether or not a job will likely parallelize well.
  - Sequential Runtime and Communication Time
- I found a bunch of numbers that I didn't know were prime.
  - (Who knew 7 was prime?!)

Future Work

- Use bignums to find REALLY big primes
  - Hope to find a new one ☺
- Change program's structure so that all processors work together to find out if numbers are prime one at a time, instead of all checking primality of different numbers.
  - Works better for very very large numbers.
- Implement using CUDA

Any Questions?