HPP; Sieve of Eratosthenes

Casper Smet

April 2020

1 Concept

1.1 Revision

Algorithm 1: Parallel Sieve

```
Result: Amount of prime numbers under N; total sum
shape = receive shape from master thread;
ei = receive equivalent integer from master thread;
if N is equally dividable by number of processes then
    shape = N // number of processes;
    ei = shape × process number;
else
    greater parts = 0..(number of processes - (N % number of processes);
    if process number in greater parts then
        shape = N // number of processes + 1;
        ei = shape × process number;
    else
        shape = N // number of processes + 1;
        ei = $(shape + 1) \times *len(greaterparts) + shape \times (processnumber - len(greaterparts))$;
    end
end
arr = array filled with True in shape of shape
for $k = 2;\ k \leq \sqrt{N};\ k++;$ do
    if $k \geq ei$ then
        index = k - ei;
        if array[index] == True then
            arr[$k \times 2 :: k$] = False;
        end
    else
        if $ei \% k == 0$ then
            index = 0
        else
            index = $(floor(\frac{ei}{k}) + 1) \times k - ei$
        end
        arr[index::k] = False
    end
end
if $ei = 0$ then
    arr[0] = False;
    arr[1] = False;
end
partial sum = sum arr total sum = gather and sum reduce partial sum
```

1.2 Original

The two master thread "Algorithm" are only executed once, the worker thread as many times as there are threads. The thread used as the master thread executes the worker thread "algorithm" in the exact same way as the other threads.

```
Algorithm 2: Master thread initialisation
  Result: Each thread is initialised with integer equivalent to arr[0] and
          a shape for arr
  shapes = shape of sieve roughly divided by amount of threads;
  equivalent integers = equivalent integer for each partial sieve at index:0;
  Send a shape to each thread;
  Send an equivalent integer to each thread;
```

```
Algorithm 3: Worker thread
  Result: Each thread creates and updates their partial sieve
  shape = receive shape from master thread;
  ei = receive equivalent integer from master thread;
  arr = array filled with True in shape of shape
  for $k = 2;\ k \leq \sqrt{N};\ k++$ do
      if $k \geq ei$ then
          index = k - ei;
          if $array[index] == True$ then
              arr[$k \times 2 :: k$] = False;
          end
      else
          if $ei\%k == 0$ then
              index = 0;
          else
              index = $(floor(\frac{ei}{k}) + 1) \times k - ei$;
          end
          arr[index::k] = False;
      end
  end
```

```
Algorithm 4: Master thread recombining
  Result: Number of prime numbers under N is revealed
  partial sieves = gather arr from each thread;
  sieve = recombined partial sieves;
  sieve[0] = False;
  sieve[1] = False;
  prime count = count all True values in sieve;
```