Programming Assignment 4

Collatz conjecture

HOTPO (halve or triple plus one)

Start with an initial seed integer n.

- If n, is even, replace it by n/2.
- Else If n is odd, replace it by 3n+1

Continue …

HOTPO (halve or triple plus one)

Start with an initial seed integer n.

- If n, is even, replace it by n/2.
- Else If n is odd, replace it by 3n+1

Continue … one repeating cycle is

4,2,1,4,2,1,4,2,1,…

HOTPO (halve or triple plus one)

Start with an initial seed integer n.

- If n, is even, replace it by n/2.
- Else If n is odd, replace it by 3n+1

Do all integers eventually reach this same cycle?

6,3,10,5,16,8,4,2,1

256,128,64,32,16,8,4,2,1

19,58,29,88,44,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1

HOTPO (halve or triple plus one)

Start with an initial seed integer n.

- If n, is even, replace it by n/2.
- Else If n is odd, replace it by 3n+1

Do all integers eventually reach this same cycle? Collatz conjecture, says yes.

6,3,10,5,16,8,4,2,1

https://www.dcode.fr/collatz-conjecture

256,128,64,32,16,8,4,2,1

19,58,29,88,44,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1

Simple sketch of algorithm to explore Collatz conjecture

- Loop over all seed integers n up to Nmax:

```
      for (j = 1; j < Nmax; j++)
      {
         n=j;
         for (i = 1; i < Imax; i++)
         {
           replace n by HOTPO(n);
           if (n==1) break;
         }
      }
```

- Apply HOTPO to n repeatedly until reach 1, or exceed some max number of iterations.

Simple code to explore Collatz conjecture

- Loop over all seed integers n up to Nmax:

```
for (j = 1; j < Nmax; j++)
{
   n=j;
   for (i = 1; i < Imax; i++)
   {
     replace n by HOTPO(n);
     if (n==1) break;
   }
}
```

- Apply HOTPO to n repeatedly until reach 1, or exceed some max number of iterations.
- Outer iterations may have widely different runtimes.

MPI Options

- Each process does some of the outer loop.
  - Divide up interval [1,Nmax] by chunks (like HW3 part C)
  - Divide up interval round robin (like HW2)
  - Worker/Manager with manager giving out a single n value at a time
  - Worker/Manager with manager giving out a block of n values at a time.

OpenMP Options

- Each process does some of the outer loop.
  - Need to carefully think through which variables inside the loop should be private.

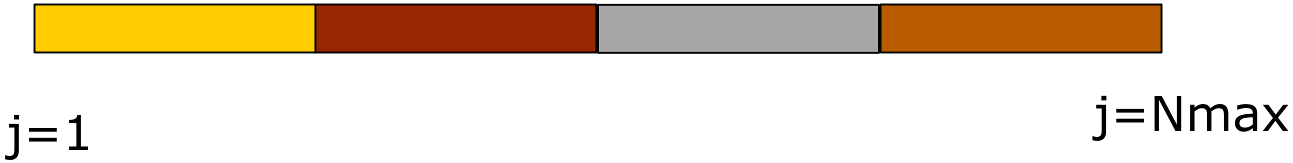
```
#pragma omp parallel for
for (j = 1; j < Nmax; j++)
{
   n=j;
   for (i = 1; i < Imax; i++)
   {
     replace n by HOTPO(n);
     if (n==1) break;
   }
}
```

OpenMP Options

- Each process does some of the outer loop.
  - Some starting seed values will require many inner iterations, some will not.
  - Run time may be effected by how the outer loop is scheduled.
  - Good load balance means each thread has similar work load, so all finish at nearly the same time.

Static schedule: usually the default

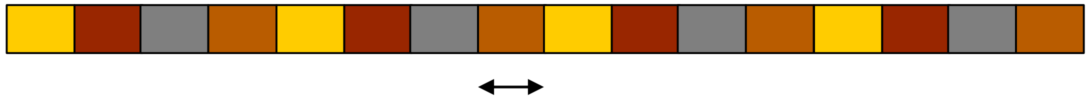
```
#pragma omp parallel for schedule(static)
```



j=1

j=Nmax

```
#pragma omp parallel for schedule(static,chunk_size)
```



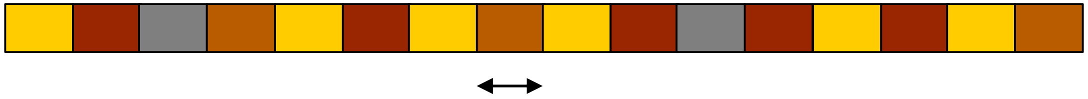
chunk_size

specifying chunk_size is optional

Has low overhead, but may have poor load balance if individual iterations have widely different run times.

Dynamic schedule: first thread to finish, gets next chunk

```
#pragma omp parallel for schedule(dynamic,chunk_size)
```



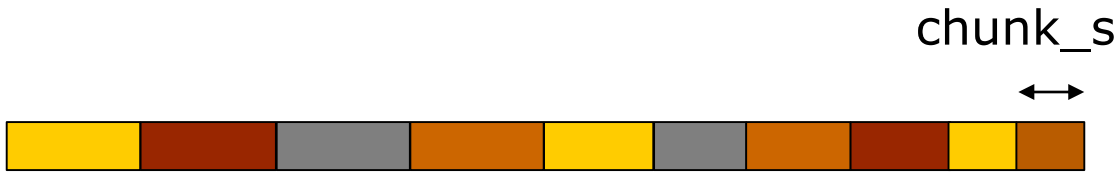
chunk_size

specifying chunk_size is optional

Has higher overhead, especially with small chunk_size, but may have better load balance if iterations have widely different run times.

Guided schedule: first thread to finish but chunks decrease in size

```
#pragma omp parallel for schedule(guided,chunk_size)
```



specifying chunk_size is optional

Compromise:
large chunks (low overhead) with small chunks (good load balance)

Added requirements

- Suppose we want to record the highest n ever reached?

```
for (j = 1; j < Nmax; j++)
{
   n=j;
   for (i = 1; i < Imax; i++)
   {
     replace n by HOTPO(n);
     if (n > High) High = n;
     if (n==1) break;
   }
}
```

Added requirements

- Suppose we want to record the highest n ever reached?

```
for (j = 1; j < Nmax; j++)
{
   n=j;
   for (i = 1; i < Imax; i++)
   {
     replace n by HOTPO(n);
     if (n > High) High = n;
     if (n==1) break;
   }
}
```

High is a reduction variable,
we want the max over all threads.

- http://www.techdarting.com/2013/06/openmp-min-max-reduction-code.html

Assignment 4: OpenMP implementation of HOTPO

- I've posted details in

  CANVAS->Files->Assignment 4
- Assignment will be due April 11, the last meeting before first project presentations.