Argonne NATIONAL LABORATORY

Parallel Programming

Marc Snir

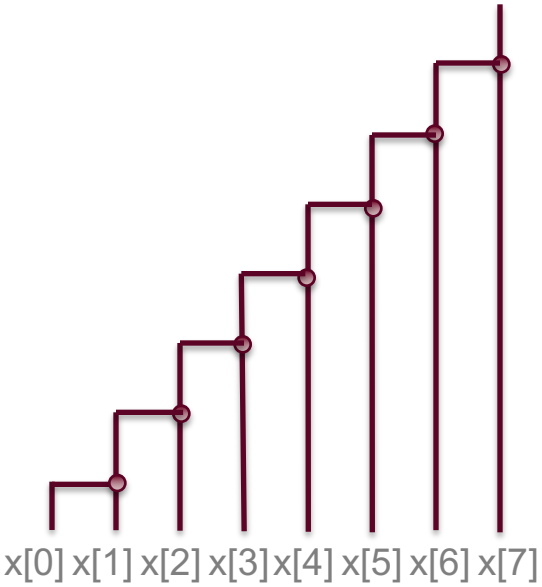
U. of Illinois at Urbana-Champaign & Argonne National Lab

U.S. DEPARTMENT OF ENERGY

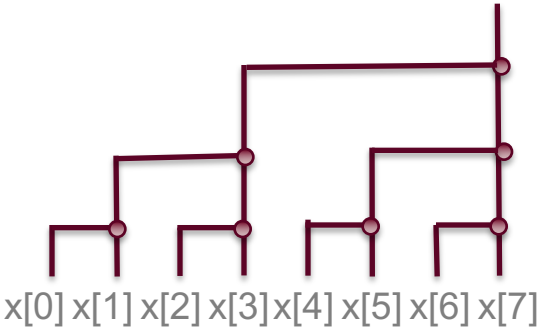
Summing n numbers

```
for(i=1; i++; i<n)
    x[i] += x[i-1];
```

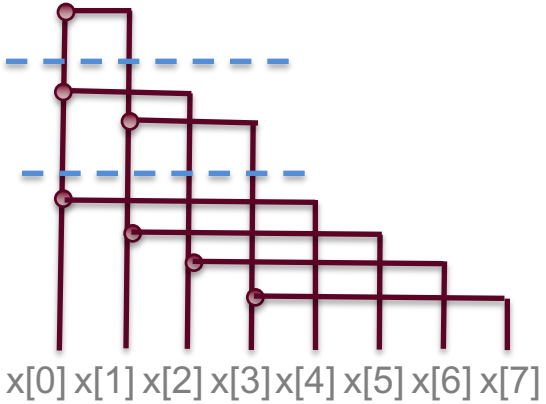
n-1 steps



Summing n numbers in parallel



$\lceil \log n \rceil$ steps



How does one code this algorithm?

OpenMP

Assume $n=2^k$

Sequential code

```
for(i=n/2; i>1; i /=2)
   for(j=0; j<i; j++)
         x[j] += x[j+i];
```

Parallel code

```
for(i=n/2; i>1; i /=2)
  #pragma omp for
  for(j=0; j<i; j++)
         x[j] += x[j+i];
```

- Code is C (or C++) with added pragma statements
- A C compiler will ignore the pragmas (as comments) and will compile second code same as first code
- An OpenMP compiler will understand that pragmas mean that inner loop iterates can be executed in parallel
- If only one thread execute the code, then the code is executed sequentially
- If multiple threads execute the code then
  - execution starts with one thread running
  - when the parallel for is encountered, the other threads start grapping iterates for execution
  - sequential execution resumes when all iterates have executed

- *Resulting code will run more slowly than original code if n is small ☺*

OpenMP

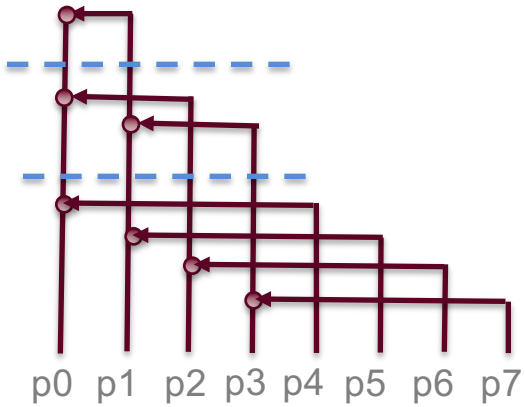
http://www.openmp.org

- Now OpenMP V4 – V4.5 to come soon
- A language used to program code that takes advantage of multicore procesors and of simultaneous multithreading
  - Multiple hardware threads run simultaneously
  - They all have access to shared memory
- Has extensions to take advantage of GPUs and vector instructions

How about using multiple processors (cluster, supercomputer)?

- Execution consists of one (or more) process per processor node
  - Each process executes the same code
  - The processes use messages to communicate

- Assume n= 2^k processors and one number per processor



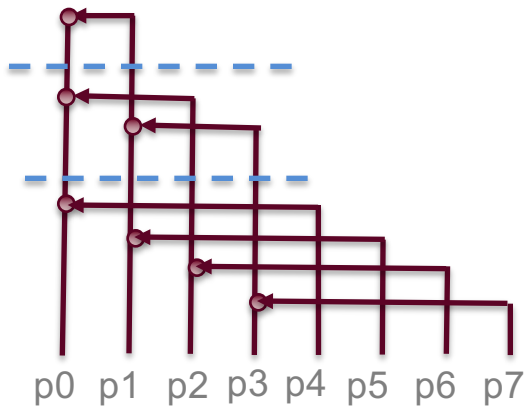
MPI Code (executed at each processor)

```
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
# processes are numbered with consecutive ranks (0…n-1)
for(i=n/2; i>1; i /=2)
   if (myrank<i) {
        MPI_Recv(&y, 1, MPI_DOUBLE, myrank+i, tag, MPI_COMM_WORLD, status);
        x +=y;
        }
   else if (myrank <2*j)
      MPI_Send(&x, 1, MPI_DOUBLE, myrank-i, tag, MPI_COMM_WORLD);
# a receive matches a send according the source rank, tag, communicator)
```

Subtle point

- Second message could arrive at processor 0 before the first message arrives
- Nevertheless, messages will be handled in the right order because of the matching rules
- In general, one may need to make sure that no process starts next iteration before all processes completed the previous iteration

MPI_Barrier(…)



MPI collective operations

- Replace previous code with

MPI_Reduce(&x, &sum, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD)

- Variable sum at process with rank 0 will be set to the sum of the variables x, one from each process.
  - Internally, the MPI library will execute something similar to the code on previous slide
  - Will handle any number of processes

Alltogher now

- p multicore processes; each process has 2^k numbers. Need to compute the sum of all of them

```
for(i=n/2; i>1; i /=2)
  #pragma omp for
  for(j=0; j<i; j++)
        x[j] += x[j+i];
MPI_Reduce(&x[0], &sum, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD)
```

MPI

http://www.mpi-forum.org

- Now MPI 3.1
- A library used to program code that runs on multiple processes
  - Each process runs a C (or C++, or OpenMP) code
  - The processes communicate using MPI calls

Is there more than MPI and OpenMP?

- GPU programming – CUDA is often needed
- Work on new languages
  - Data parallel computing: focus on distributed data structures and moving computation to data
    - Chapel: http://chapel.cray.com/,
    - Legion: http://legion.stanford.edu/
  - PGAS – Partitioned Global Address Space: Programs can use local references and global references (pointing to an address on another node)
    - UPC: https://upc-lang.org/

- The different world of high-end analytics – Hadoop, Spark…
  - focused on data that does not fit in memory and on coarser-level parallelism

(MS)

To know more, please take a course in parallel programming

Preferably, at the University of Illinois

Questions?