Distributed Memory Computing: algorithms

Erik Saule
esaule@uncc.edu

Intro to Parallel Programming

Learning Outcomes

At the end of this lecture, you will be able to

- Represent graphically an algorithm
- Derive communication patterns and load balance from data partitioning

Learning Outcomes

At the end of this lecture, you will be able to

- Represent graphically an algorithm
- Derive communication patterns and load balance from data partitioning

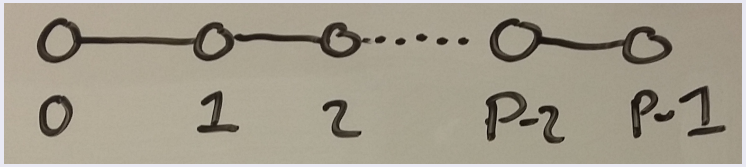
The assignment will ask you to

- Analyze distributed memory algorithms
- Explain why different algorithms are suited to different network topology
- Write your own distributed algorithm for two classic problems

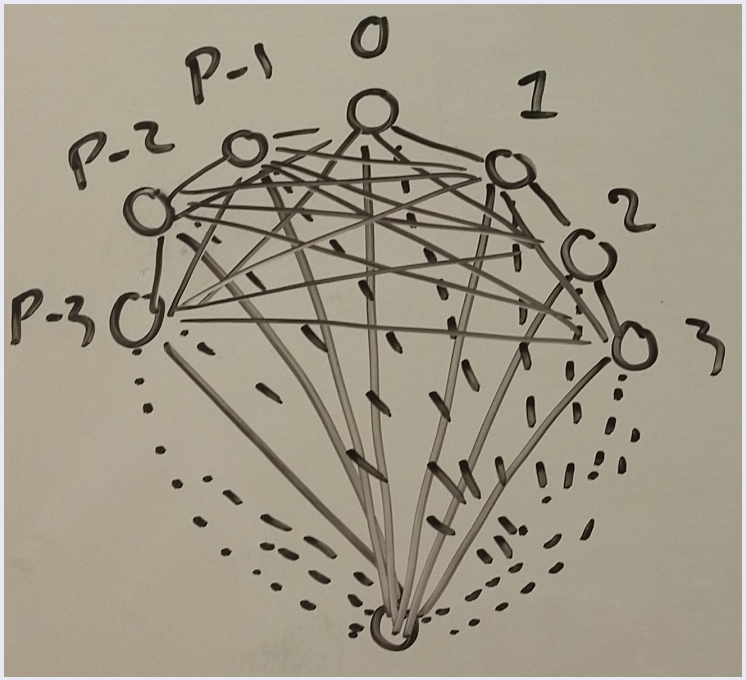
Outline

Network Topology Matters!

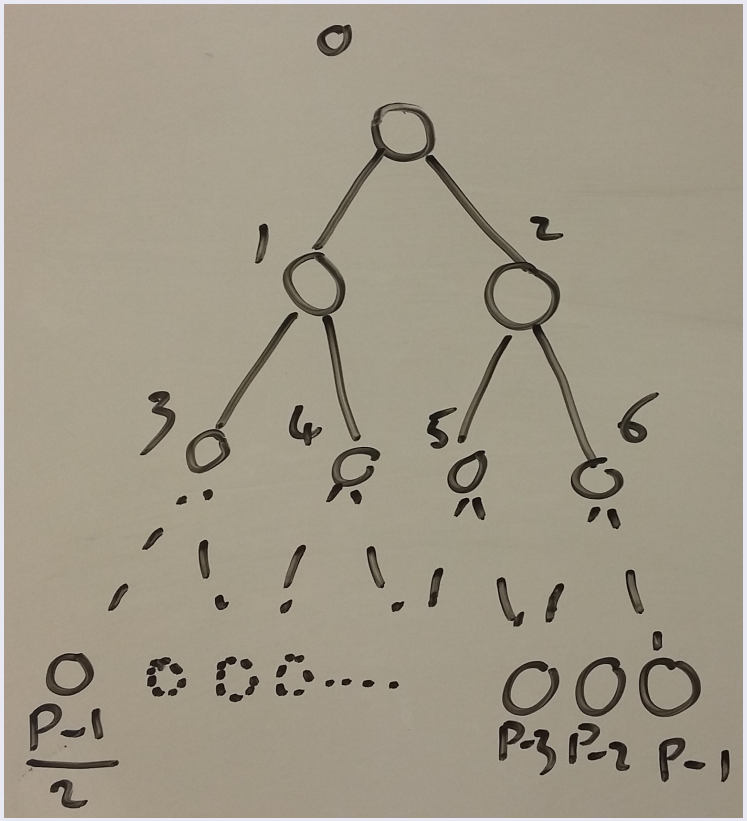
Chain



Clique



Hierarchical



Differences between shared and distributed memory

Limited memory per node

Often the entire dataset can not fit in the memory of a single node.

Memory accesses

One node can only access part of the total memory allocated to the task.

Communication Cost

Splitting the computation is no longer the only objective. Minimizing communication becomes a primary problem.

Numerical Integration

```
numerical (a, b, inten, N, p, P) {
  if (p == 0) {
    integral = 0.;
  } else {
    recv integral from p-1;
  }
  //assumes N/P is integer
  begin = p*N/P;
  end = (p+1)*N/P;
  for (i = begin; i< end; ++i ) {
    integral += (b-a)/N
      * f (a+(i.5)*(b-a)/N, inten);
  }

  if (p != P-1) {
    send integral to p+1;
  }
  else {
    print (integral);
```

Data Partitioning

Why?

Often nodes can not store *all* the data.
Even if they could, having all nodes obtain all data is probably not scalable.

The question is not who *does* what at each time, but also who *knows* what.

The owner computes rule

The process that stores a particular data item is responsible for all computations associated with it.

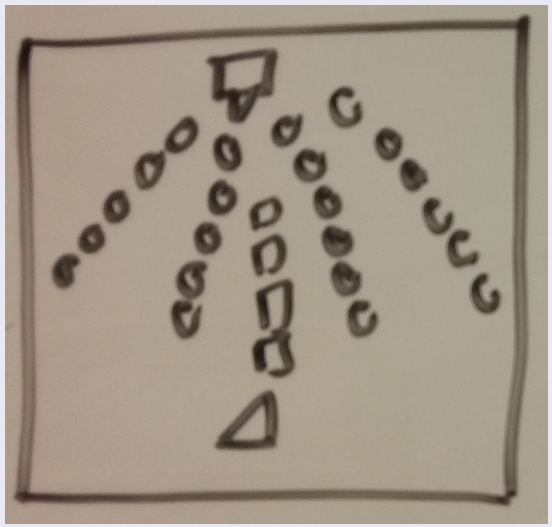
Consequence

- Memory balance: Each node should be able to store the data it needs.
- Load balance: is determined by the partitioning.
- Communication: is necessary to access data not available on the local node.

An example: Detecting collisions

Problem

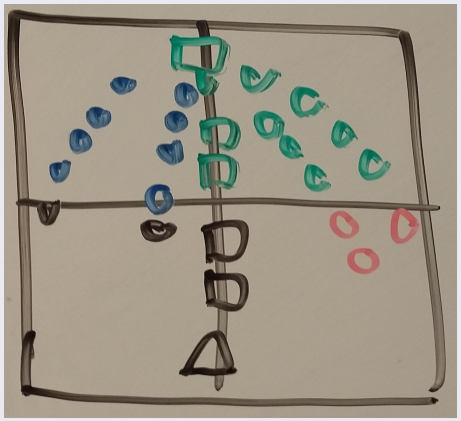
In a shooting game, how to detect collisions using a distributed memory machine?



Need to make sure there is no collision between pairs of objects.

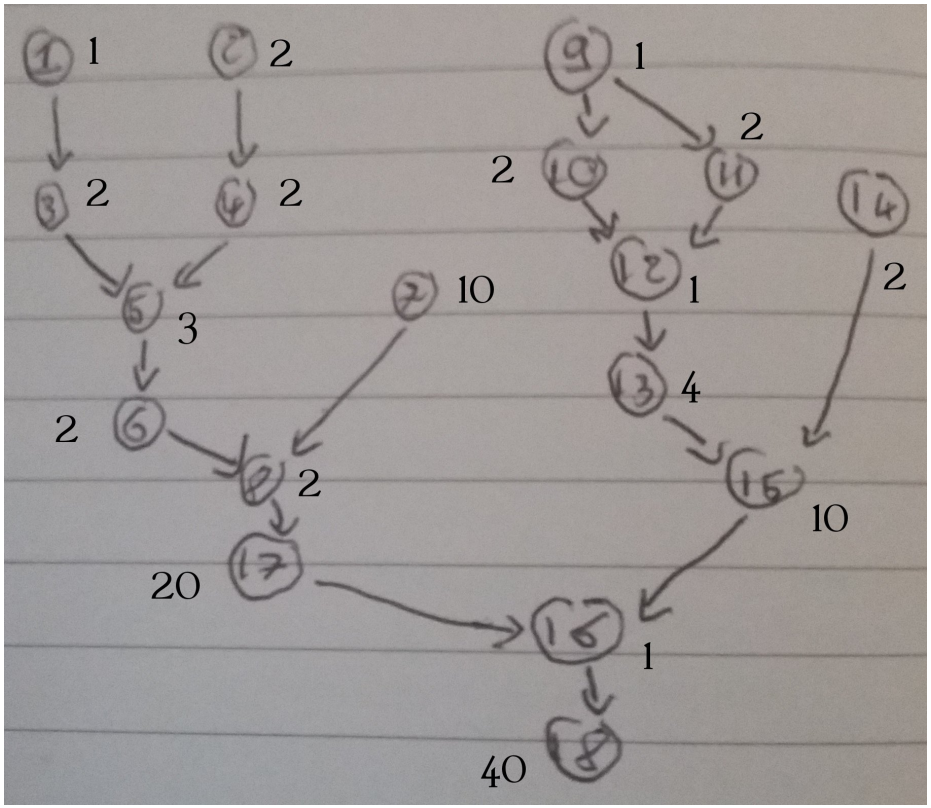
Object decomposition

Spatial decomposition



Outline

Shared memory model



Work

Total amount of work that is to perform on the application.

Width

How many threads can work at once.

Critical Path

Length of the longest chain of task to execute.

Distributed memory model

Task Allocation

Each task becomes allocated:

- to a particular node.
- or to a particular link of the network.

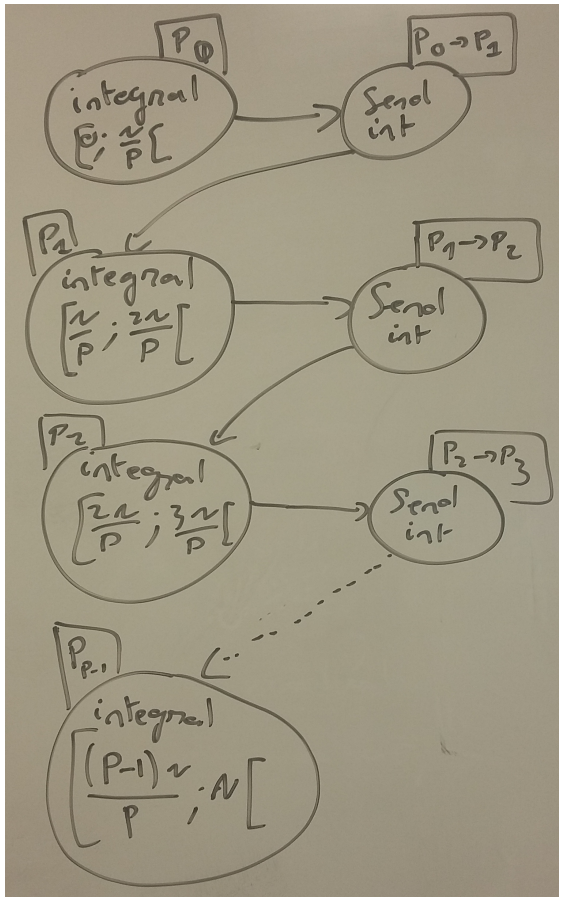
Metrics

- Load balance: Do all nodes have an equal amount of work? Usually measured as $\frac{maxload}{avgload}$.
- Memory requirement: What is the maximum amount of memory one node needs?
- Most loaded link: Which link transports the maximum amount of data?
- Most communicative node: Which node is involved in the most communication?
- Critical Path: What is the longest chain of events?

Numerical Integration

```
numerical (a, b, inten, N, p, P) {
  if (p == 0) {
    integral = 0.;
  } else {
    recv integral from p-1;
  }
  //assumes N/P is integer
  begin = p*N/P;
  end = (p+1)*N/P;
  for (i = begin; i< end; ++i ) {
    integral += (b-a)/N
      * f (a+(i.5)*(b-a)/N, inten);
  }

  if (p != P-1) {
    send integral to p+1;
  }
  else {
    print (integral);
  }
}
```



Metrics

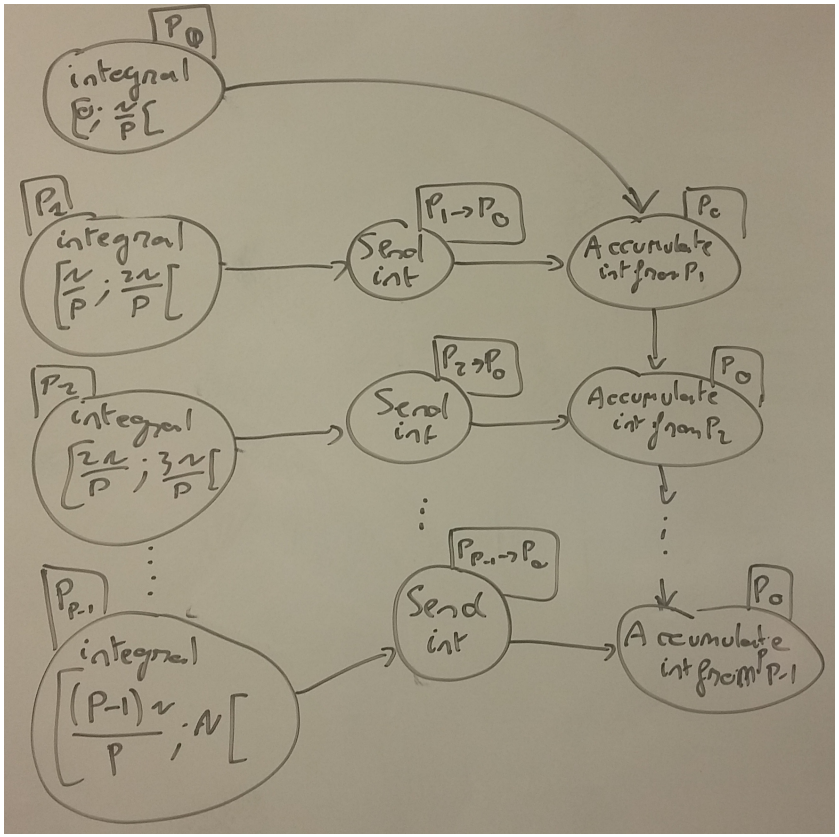
- Load balance:
- Memory:
- Load of links:
- Network load of nodes:
- Critical path:

Numerical Integration - reworked

```
numerical (a, b, inten, N, p, P) {
  integral = 0.;

  //assumes N/P is integer
  begin = p*N/P;
  end = (p+1)*N/P;
  for (i = begin; i< end; ++i ) {
    integral += (b-a)/N
      * f (a+(i.5)*(b-a)/N, inten);
  }

  if (p != 0) {
    send integral to 0;
  }
  else {
    for (i=1; i<P; ++i) {
      recv integralp from i;
      integral += integralp;
    }
  }
}
```



Metrics

- Load balance:
- Memory:
- Load of links:
- Network load of nodes:
- Critical path:

Outline

External

Books:

- "Parallel and Distributed Computation: numerical methods". by D. Bertsekas and J. Tsisiklis. (Chapter 1 is on network topology and schedules.)
- "Introduction to Parallel Computing: Design and analysis of Algorithms". V. Kumar, A. Grama, A. Gupta, G. Karypis. (First four chapters.)

Distributed memory workstealing:

- KAAPI: `https://hal.inria.fr/hal-00684843/en`
- Habanero-UPC++: `http://habanero-rice.github.io/habanero-upc/`
- hartley PhD: `http://library.ohio-state.edu/record=b6960693~S7`

Library for partitioning:

- zoltan: `http://www.cs.sandia.gov/zoltan/`
- scotch: `https://www.labri.fr/perso/pelegrin/scotch/`
- spart: `http://bmi.osu.edu/hpc/software/spart/`
- metis: `http://glaros.dtc.umn.edu/gkhome/metis/metis/overview`
- patoh: `http://bmi.osu.edu/umit/software.html`