Foster's Methodology: Application Examples

Parallel and Distributed Computing

Department of Computer Science and Engineering (DEI)
Instituto Superior Técnico

October 19, 2011

Outline

- Foster's design methodology
  - partitioning
  - communication
  - agglomeration
  - mapping
- Application Examples
  - Boundary value problem
  - Finding the maximum
  - n-body problem

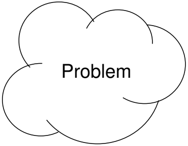
Distributed Memory Systems

Parallel programming of distributed-memory systems is significantly different from shared-memory systems essentially due to the very large overhead in terms of:

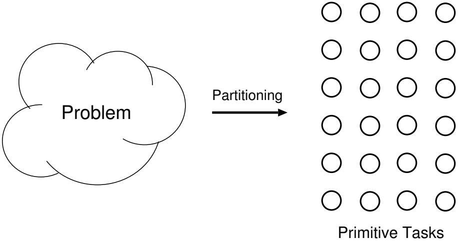
- communication
- task initialization / termination

$\Rightarrow$ efficient parallelization requires that these effects be taken into account from the start!

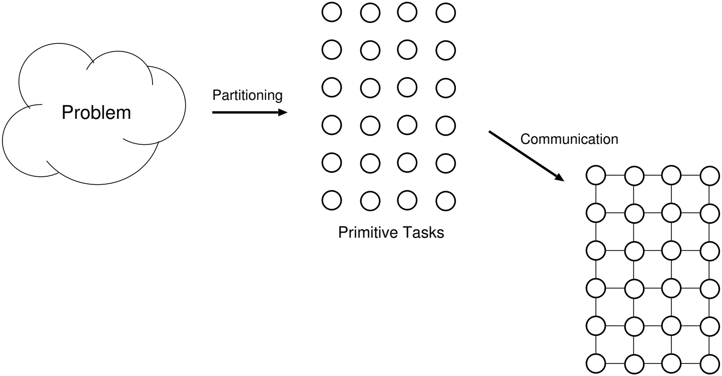
Foster's Design Methodology



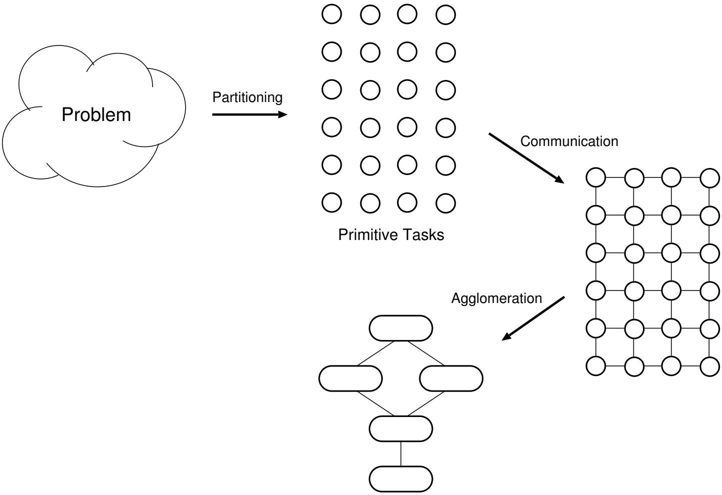
Foster's Design Methodology



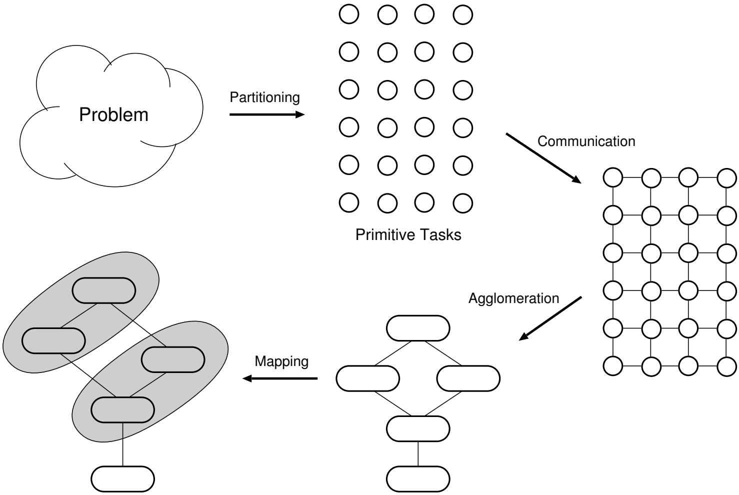
Foster's Design Methodology



Foster's Design Methodology



Foster's Design Methodology



Boundary Value Problem

Problem

Determine the evolution of the temperature of a rod in the following conditions:

- rod of length 1
- both ends of the rod at 0^{o}C
- uniform material
- insulated except at the ends
- initial temperature of a point at distance x is $100 \sin(\pi x)$

Boundary Value Problem

Problem

Determine the evolution of the temperature of a rod in the following conditions:

- rod of length 1
- both ends of the rod at 0^{o}C
- uniform material
- insulated except at the ends
- initial temperature of a point at distance x is $100 \sin(\pi x)$

A partial differential equation (PDE) governs the temperature of the rod in time. Typically too complicated to solve analytically.

Boundary Value Problem

Finite difference methods can be used to obtain an approximate solution to these complex problems
$\Rightarrow$ discretize space and time:

- unit-length rod divided into n sections of length h
- time from 0 to P divided into m periods of length k

We obtain a grid $n \times m$ of temperatures $T(x, t)$.

Boundary Value Problem

Finite difference methods can be used to obtain an approximate solution to these complex problems
$\Rightarrow$ discretize space and time:

- unit-length rod divided into n sections of length h
- time from 0 to P divided into m periods of length k

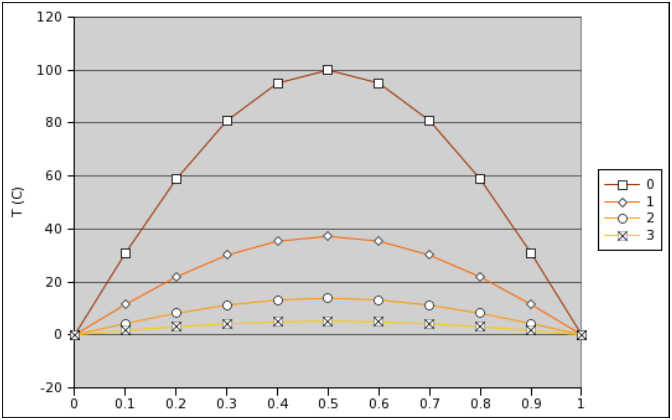
We obtain a grid $n \times m$ of temperatures $T(x, t)$.

Temperature over time is given by:

$$T(x,t) = r \cdot T(x-1, t-1) + (1-2r) \cdot T(x, t-1) + r \cdot T(x+1, t-1)$$

where $r = k/h^2$.

Boundary Value Problem



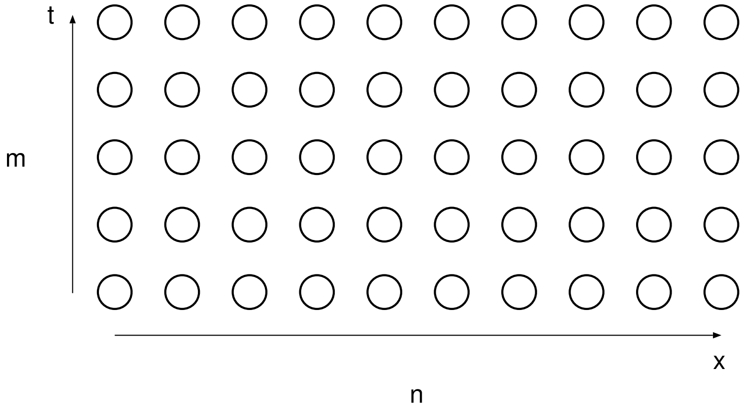
Boundary Value Problem

Partitioning:

Boundary Value Problem

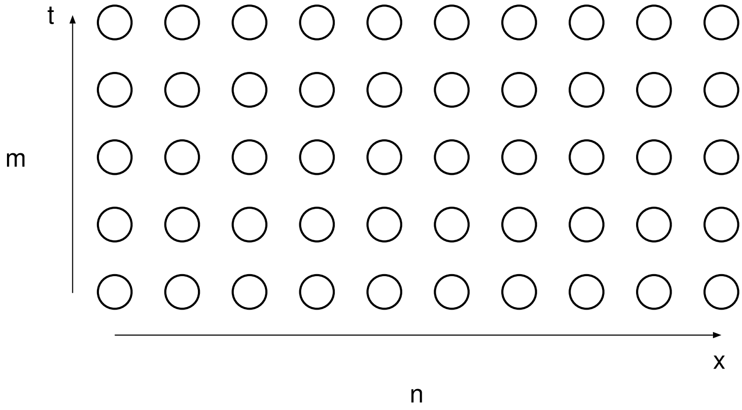
Partitioning:

Make each $T(x, t)$ computation a primitive task.
$\Rightarrow$ 2-dimensional domain decomposition

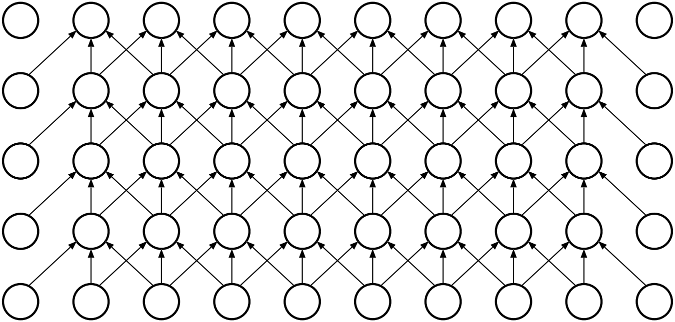


Boundary Value Problem

Communication:

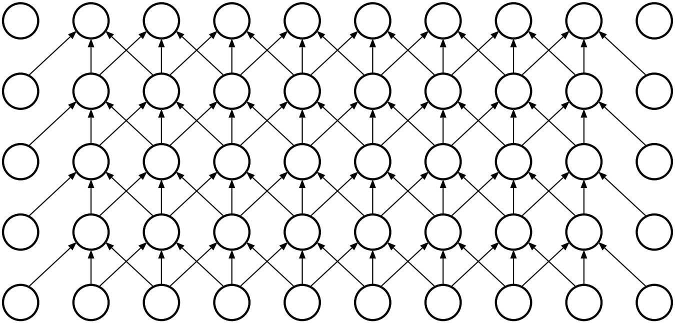


Communication:



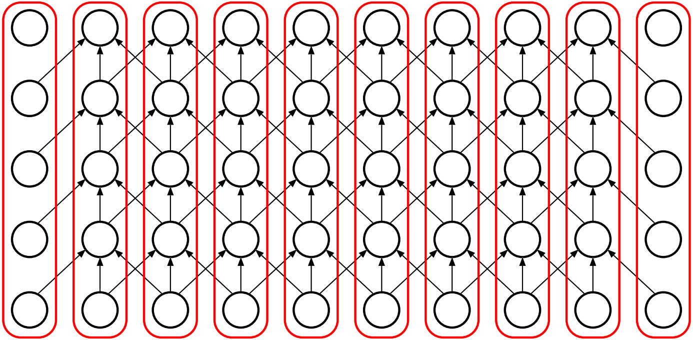
Boundary Value Problem

Agglomeration:



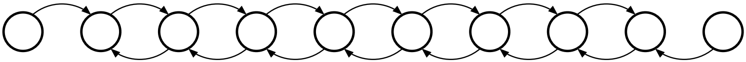
Boundary Value Problem

Agglomeration:



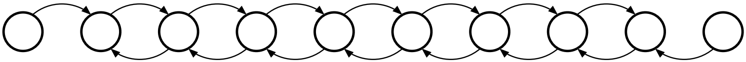
Boundary Value Problem

Agglomeration:



Boundary Value Problem

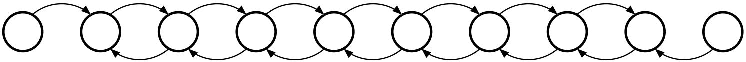
Agglomeration:



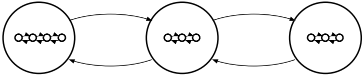
Mapping:

Boundary Value Problem

Agglomeration:



Mapping:



Boundary Value Problem

Analysis of execution time:

- C: time to compute
  $T(x,t) = rT(x-1,t-1) + (1-2r)T(x,t-1) + rT(x+1,t-1)$
- D: time to send/receive one $T(x,t)$ from another processor
- p: number of processors

Sequential algorithm:

Boundary Value Problem

Analysis of execution time:

- C: time to compute
  $T(x,t) = rT(x-1,t-1) + (1-2r)T(x,t-1) + rT(x+1,t-1)$
- D: time to send/receive one $T(x,t)$ from another processor
- p: number of processors

Sequential algorithm: mnC

Parallel algorithm:

computation per time instant:

Boundary Value Problem

Analysis of execution time:

- C: time to compute
  $T(x,t) = rT(x-1,t-1) + (1-2r)T(x,t-1) + rT(x+1,t-1)$
- D: time to send/receive one $T(x,t)$ from another processor
- p: number of processors

Sequential algorithm: mnC

Parallel algorithm:

computation per time instant: $\left\lceil \frac{n}{p} \right\rceil C$

communication per time instant:

Boundary Value Problem

Analysis of execution time:

- C: time to compute
  $T(x,t) = rT(x-1,t-1) + (1-2r)T(x,t-1) + rT(x+1,t-1)$
- D: time to send/receive one $T(x,t)$ from another processor
- p: number of processors

Sequential algorithm: mnC

Parallel algorithm:

computation per time instant: $\left\lceil \frac{n}{p} \right\rceil C$

communication per time instant: $2D$

(receiving is synchronous, sending asynchronous)

Total: $m(\left\lceil \frac{n}{p} \right\rceil C + 2D)$

Finding the Maximum

Problem

Determine the maximum over a set of n values.

This is a particular case of a reduction:

$$a_0 \oplus a_1 \oplus a_2 \oplus \cdots \oplus a_{n-1}$$

where $\oplus$ can be any associative binary operator.

$\Rightarrow$ a reduction always takes $\Theta(n)$ time on a sequential computer

Finding the Maximum

Partitioning:

Finding the Maximum

Partitioning:

Make each value a primitive task.
$\Rightarrow$ 1-dimensional domain decomposition

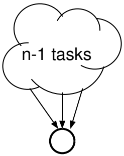
One task will compute final solution: root task

Finding the Maximum

Communication:

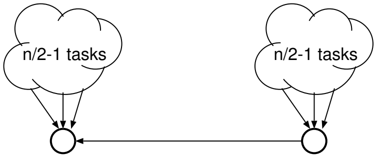
Finding the Maximum

Communication:



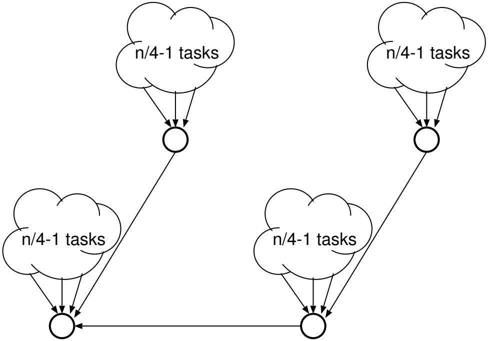
Finding the Maximum

Communication:



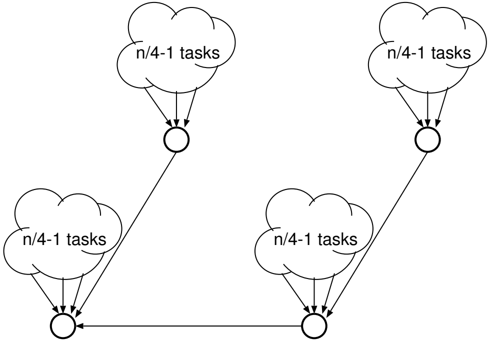
Finding the Maximum

Communication:



Finding the Maximum

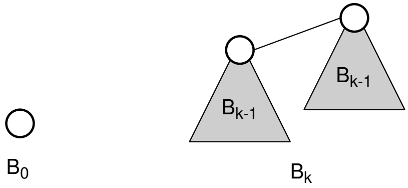
Communication:



Continue recursively: Binomial Tree

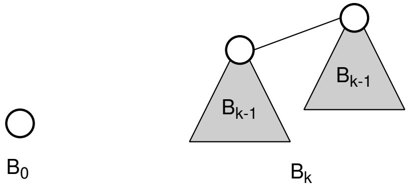
Binomial Trees

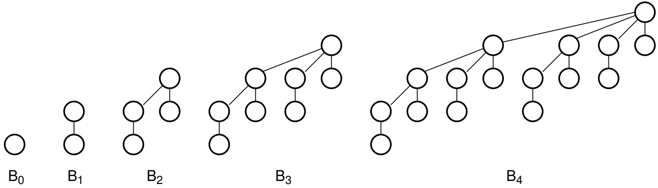
Recursive definition of Binomial Tree, with $n = 2^k$ nodes:



Binomial Trees

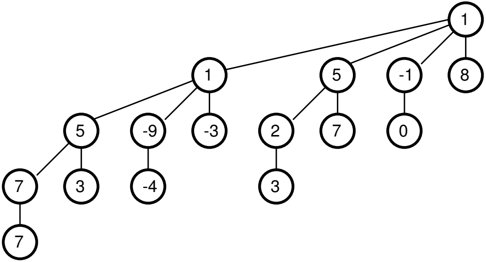
Recursive definition of Binomial Tree, with $n = 2^k$ nodes:





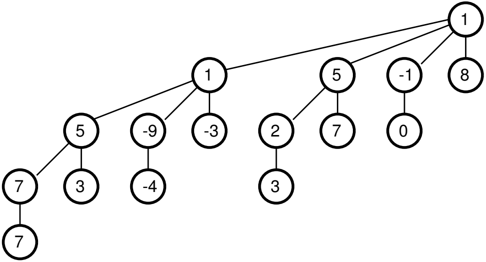
Finding the Maximum

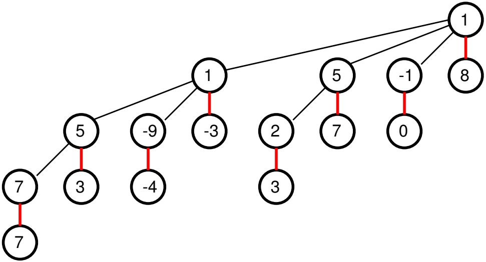
Illustrative example:



Finding the Maximum

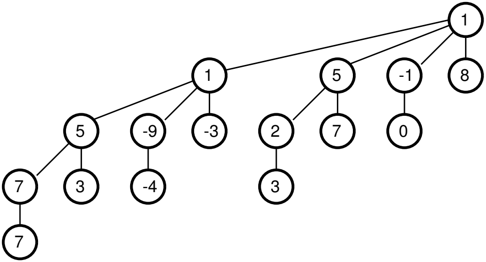
Illustrative example:

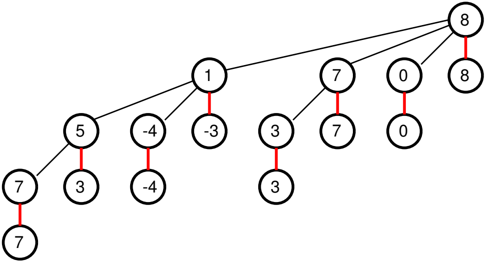




Finding the Maximum

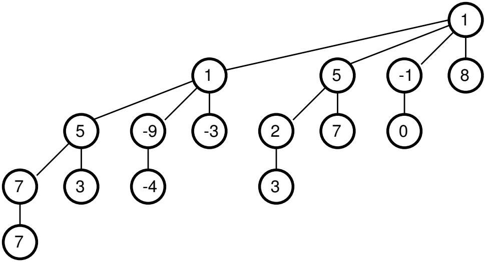
Illustrative example:

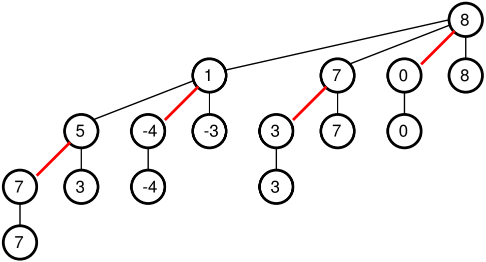




Finding the Maximum

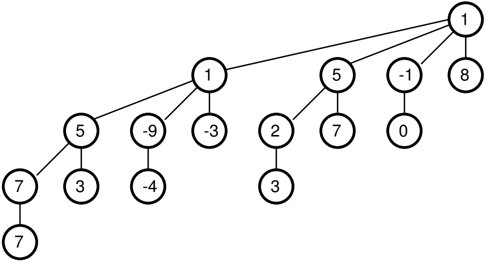
Illustrative example:

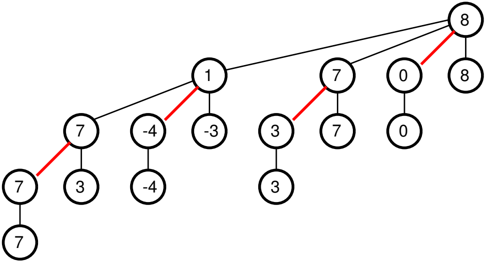




Finding the Maximum

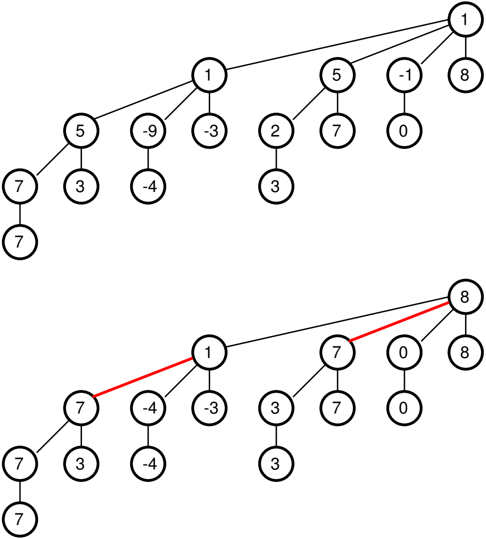
Illustrative example:





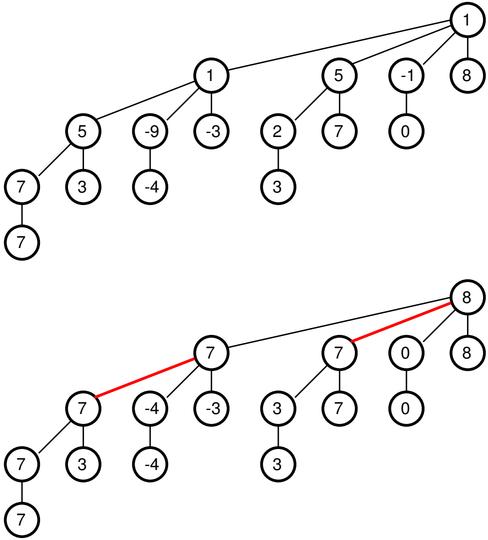
Finding the Maximum

Illustrative example:



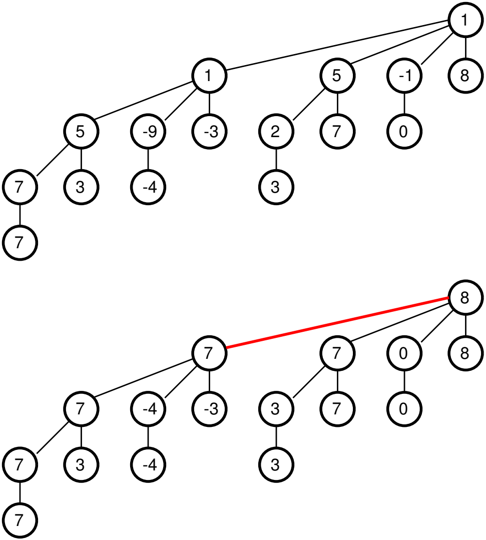
Finding the Maximum

Illustrative example:



Finding the Maximum

Illustrative example:



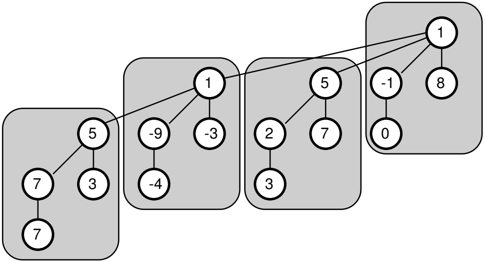
Finding the Maximum

Agglomeration:

Finding the Maximum

Agglomeration:

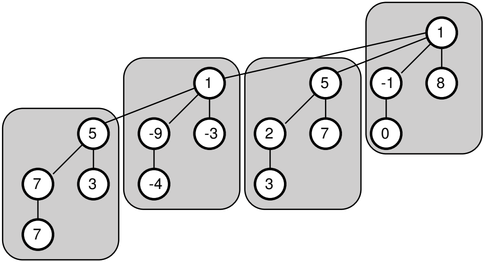
Group n leafs of the tree:



Finding the Maximum

Agglomeration:

Group n leafs of the tree:

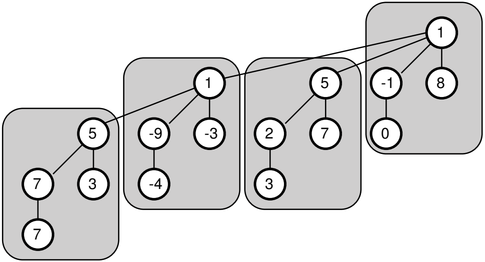


Mapping:

Finding the Maximum

Agglomeration:

Group n leafs of the tree:



Mapping:

The same (actually, in the agglomeration phase, use n such that you end up with p tasks).

Finding the Maximum

Analysis of execution time:

- C: time to perform the binary operation (maximum)
- D: time to send/receive one value from another processor
- p: number of processors

Sequential algorithm:

Finding the Maximum

Analysis of execution time:

- C: time to perform the binary operation (maximum)
- D: time to send/receive one value from another processor
- p: number of processors

Sequential algorithm: $(n-1)C$

Parallel algorithm:

computation in the leafs:

Finding the Maximum

Analysis of execution time:

C: time to perform the binary operation (maximum)

D: time to send/receive one value from another processor

p: number of processors

Sequential algorithm: $(n-1)C$

Parallel algorithm:

computation in the leafs: $\left(\left\lceil \frac{n}{p} \right\rceil - 1\right)C$

computation up the tree:

Finding the Maximum

Analysis of execution time:

- C: time to perform the binary operation (maximum)
- D: time to send/receive one value from another processor
- p: number of processors

Sequential algorithm: $(n-1)C$

Parallel algorithm:

computation in the leafs: $\left(\left\lceil \frac{n}{p} \right\rceil - 1\right)C$

computation up the tree: $\lceil \log p \rceil (C + D)$

Total: $\left(\left\lceil \frac{n}{p} \right\rceil - 1\right)C + \lceil \log p \rceil (C + D)$

n-Body Problem

Problem

Simulate the motion of n particles of varying masses in two dimensions.

- compute new positions and velocities
- consider gravitational interactions only

Straightforward sequential algorithms solve this problem in $\Theta(n^2)$ time (however, better time complexity algorithms exist)

n-Body Problem

Partitioning:

n-Body Problem

Partitioning:

Make each particle a primitive task.
$\Rightarrow$ 1-dimensional domain decomposition

Communication:

n-Body Problem

Partitioning:

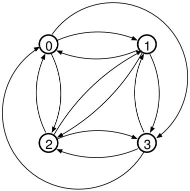
Make each particle a primitive task.
⇒ 1-dimensional domain decomposition

Communication:

All tasks need to communicate with all tasks!

Gather operation: one task receive a dataset from all tasks.

All-gather operation: all tasks receive a dataset from all tasks.



n-Body Problem

Partitioning:

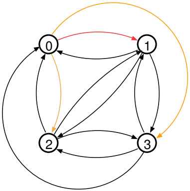
Make each particle a primitive task.
⇒ 1-dimensional domain decomposition

Communication:

All tasks need to communicate with all tasks!

Gather operation: one task receive a dataset from all tasks.

All-gather operation: all tasks receive a dataset from all tasks.



n-Body Problem

Partitioning:

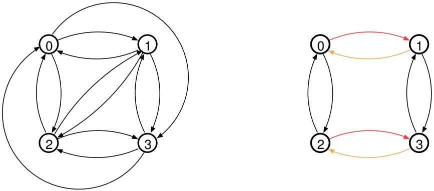
Make each particle a primitive task.
⇒ 1-dimensional domain decomposition

Communication:

All tasks need to communicate with all tasks!

Gather operation: one task receive a dataset from all tasks.

All-gather operation: all tasks receive a dataset from all tasks.



n-Body Problem

Partitioning:

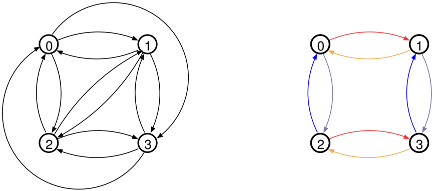
Make each particle a primitive task.
⇒ 1-dimensional domain decomposition

Communication:

All tasks need to communicate with all tasks!

Gather operation: one task receive a dataset from all tasks.

All-gather operation: all tasks receive a dataset from all tasks.



Implement communication using a hypercube topology!

n-Body Problem

Agglomeration / Mapping:

All primitive tasks have the same computation cost and communication pattern

$\Rightarrow$ no particular strategy for agglomeration required

Agglomerate n/p primitive tasks, so that we have one task per processor.

n-Body Problem

Analysis of execution time (per time instant):

- C: time to compute new particle position and velocity
- D: time to initiate message
- B: (bandwidth) number of data units that can be sent in one unit of time
- p: number of processors

Sequential algorithm: $Cn(n-1)$
Parallel algorithm:

Computation time:

n-Body Problem

Analysis of execution time (per time instant):

- C: time to compute new particle position and velocity
- D: time to initiate message
- B: (bandwidth) number of data units that can be sent in one unit of time
- p: number of processors

Sequential algorithm: $Cn(n-1)$

Parallel algorithm:

Computation time: $C\frac{n}{p}(n-1)$

Communication time:

n-Body Problem

Analysis of execution time (per time instant):

- C: time to compute new particle position and velocity
- D: time to initiate message
- B: (bandwidth) number of data units that can be sent in one unit of time
- p: number of processors

Sequential algorithm: $Cn(n-1)$
Parallel algorithm:

Computation time: $C\frac{n}{p}(n-1)$

Communication time:

one message with k data units:

n-Body Problem

Analysis of execution time (per time instant):

- C: time to compute new particle position and velocity
- D: time to initiate message
- B: (bandwidth) number of data units that can be sent in one unit of time
- p: number of processors

Sequential algorithm: $Cn(n-1)$
Parallel algorithm:

Computation time: $C\frac{n}{p}(n-1)$

Communication time:

one message with k data units: $D + \frac{k}{B}$
depth of tree:

n-Body Problem

Analysis of execution time (per time instant):

- C: time to compute new particle position and velocity
- D: time to initiate message
- B: (bandwidth) number of data units that can be sent in one unit of time
- p: number of processors

Sequential algorithm: $Cn(n-1)$
Parallel algorithm:

Computation time: $C\frac{n}{p}(n-1)$

Communication time:

one message with k data units: $D + \frac{k}{B}$
depth of tree: $\log p$
length of message at depth i:

n-Body Problem

Analysis of execution time (per time instant):

- C: time to compute new particle position and velocity
- D: time to initiate message
- B: (bandwidth) number of data units that can be sent in one unit of time
- p: number of processors

Sequential algorithm: $Cn(n-1)$

Parallel algorithm:

Computation time: $C\frac{n}{p}(n-1)$

Communication time:

one message with k data units: $D + \frac{k}{B}$

depth of tree: $\log p$

length of message at depth i: $2^{i-1}\frac{n}{p}$

total communication time:

n-Body Problem

Analysis of execution time (per time instant):

- C: time to compute new particle position and velocity
- D: time to initiate message
- B: (bandwidth) number of data units that can be sent in one unit of time
- p: number of processors

Sequential algorithm: $Cn(n-1)$
Parallel algorithm:

Computation time: $C\frac{n}{p}(n-1)$

Communication time:

one message with k data units: $D + \frac{k}{B}$
depth of tree: $\log p$
length of message at depth i: $2^{i-1}\frac{n}{p}$
total communication time: $\sum_{i=1}^{\log p}\left(D + \frac{2^{i-1}n}{pB}\right) = D\log p + \frac{n(p-1)}{pB}$

Total: $D\log p + \frac{n}{p}\left(\frac{p-1}{B} + C(n-1)\right)$

Review

- Foster's design methodology
  - partitioning
  - communication
  - agglomeration
  - mapping
- Application Examples
  - Boundary value problem
  - Finding the maximum
  - n-body problem

Next Class

- MPI