Iterative algorithms
(on the impact of network models)

Frédéric Vivien

e-mail: Frederic.Vivien@ens-lyon.fr

Outline

Outline

The context: distributed heterogeneous platforms

New sources of problems

- Heterogeneity of processors (computational power, memory, etc.)
- Heterogeneity of communications links.
- Irregularity of interconnection network.
- Non dedicated platforms.

Targeted applications: iterative algorithms

- A set of data (typically, a matrix)
- Structure of the algorithms:
  - While the computation is not finished
    - Each processor exchange the "border" of its chunk of data with its neighbor processors

Question: how can we efficiently execute such an algorithm on such a platform?

Targeted applications: iterative algorithms

- A set of data (typically, a matrix)
- Structure of the algorithms:
  - While the computation is not finished
    - Each processor performs a computation on its chunk of data
    - Each processor exchange the "border" of its chunk of data with its neighbor processors

Question: how can we efficiently execute such an algorithm on such a platform?

Targeted applications: iterative algorithms

- A set of data (typically, a matrix)
- Structure of the algorithms:
  - While the computation is not finished
    - Each processor performs a computation on its chunk of data
    - Each processor exchange the “border” of its chunk of data with its neighbor processors

Question: how can we efficiently execute such an algorithm on such a platform?

Targeted applications: iterative algorithms

- A set of data (typically, a matrix)
- Structure of the algorithms:
  - While the computation is not finished
    - Each processor performs a computation on its chunk of data
    - Each processor exchange the "border" of its chunk of data with its neighbor processors

Question: how can we efficiently execute such an algorithm on such a platform?

Targeted applications: iterative algorithms

- A set of data (typically, a matrix)
- Structure of the algorithms:
  - While the computation is not finished
    - Each processor performs a computation on its chunk of data
    - Each processor exchange the “border” of its chunk of data with its neighbor processors

Question: how can we efficiently execute such an algorithm on such a platform?

Targeted applications: iterative algorithms

- A set of data (typically, a matrix)
- Structure of the algorithms:
  - While the computation is not finished
    - Each processor performs a computation on its chunk of data
    - Each processor exchange the “border” of its chunk of data with its neighbor processors

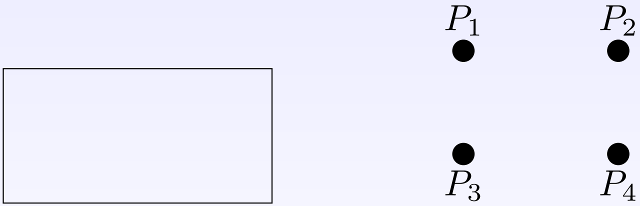
Question: how can we efficiently execute such an algorithm on such a platform?

The questions

- Which processors should be used ?
- What amount of data should we give them ?
- How do we cut the set of data ?

Before all, a simplification: slicing the data

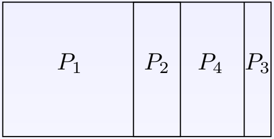
- Data: a 2-D array

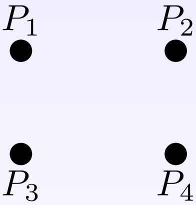


- Unidimensional cutting into vertical slices
- Consequences:
  1. Borders and neighbors are easily defined
  2. Constant volume of data exchanged between neighbors: D_c
  3. Processors are virtually organized into a ring

Before all, a simplification: slicing the data

- Data: a 2-D array

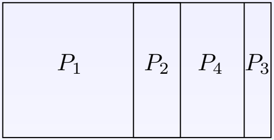


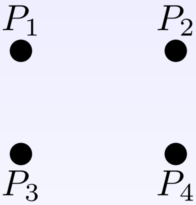


- Unidimensional cutting into vertical slices
- Consequences:
  1. Borders and neighbors are easily defined
  2. Constant volume of data exchanged between neighbors: D_c
  3. Processors are virtually organized into a ring

Before all, a simplification: slicing the data

- Data: a 2-D array

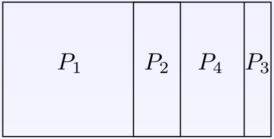


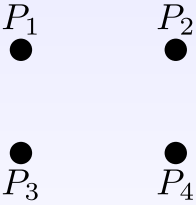


- Unidimensional cutting into vertical slices
- Consequences:
  1. Borders and neighbors are easily defined
  2. Constant volume of data exchanged between neighbors: D_c
  3. Processors are virtually organized into a ring

Before all, a simplification: slicing the data

- Data: a 2-D array

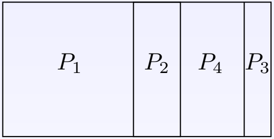


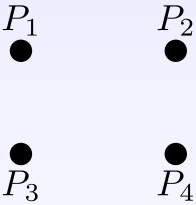


- Unidimensional cutting into vertical slices
- Consequences:
  1. Borders and neighbors are easily defined
  2. Constant volume of data exchanged between neighbors: D_c
  3. Processors are virtually organized into a ring

Before all, a simplification: slicing the data

- Data: a 2-D array

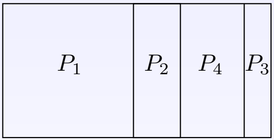


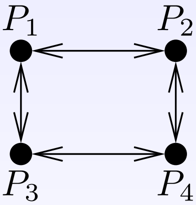


- Unidimensional cutting into vertical slices
- Consequences:
  1. Borders and neighbors are easily defined
  2. Constant volume of data exchanged between neighbors: D_c
  3. Processors are virtually organized into a ring

Before all, a simplification: slicing the data

- Data: a 2-D array





- Unidimensional cutting into vertical slices
- Consequences:
  1. Borders and neighbors are easily defined
  2. Constant volume of data exchanged between neighbors: D_c
  3. Processors are virtually organized into a ring

Notations

- Processors: $P_1, \ldots, P_p$
- Processor P_i executes a unit task in a time w_i
- Overall amount of work D_w;
  Share of P_i: $\alpha_i \cdot D_w$ processed in a time $\alpha_i \cdot D_w \cdot w_i$
  $(\alpha_i \geq 0, \sum_j \alpha_j = 1)$
- Cost of a unit-size communication from P_i to P_j: $c_{i,j}$
- Cost of a sending from P_i to its successor in the ring: $D_c.c_{i,\text{succ}(i)}$

Notations

- Processors: $P_1, \ldots, P_p$
- Processor P_i executes a unit task in a time w_i
- Overall amount of work D_w;
  Share of P_i: $\alpha_i \cdot D_w$ processed in a time $\alpha_i \cdot D_w \cdot w_i$
  $(\alpha_i \geq 0, \sum_j \alpha_j = 1)$
- Cost of a unit-size communication from P_i to P_j: $c_{i,j}$
- Cost of a sending from P_i to its successor in the ring: $D_c . c_{i,\text{succ}(i)}$

Notations

- Processors: $P_1, \ldots, P_p$
- Processor P_i executes a unit task in a time w_i
- Overall amount of work D_w;
  Share of P_i: $\alpha_i \cdot D_w$ processed in a time $\alpha_i \cdot D_w \cdot w_i$
  ($\alpha_i \geq 0$, $\sum_j \alpha_j = 1$)
- Cost of a unit-size communication from P_i to P_j: $c_{i,j}$
- Cost of a sending from P_i to its successor in the ring: $D_c.c_{i,\text{succ}(i)}$

Notations

- Processors: $P_1, \ldots, P_p$
- Processor P_i executes a unit task in a time w_i
- Overall amount of work D_w;
  Share of P_i: $\alpha_i \cdot D_w$ processed in a time $\alpha_i \cdot D_w \cdot w_i$
  ($\alpha_i \geq 0$, $\sum_j \alpha_j = 1$)
- Cost of a unit-size communication from P_i to P_j: $c_{i,j}$
- Cost of a sending from P_i to its successor in the ring: $D_c . c_{i,\text{succ}(i)}$

Notations

- Processors: $P_1, \ldots, P_p$
- Processor P_i executes a unit task in a time w_i
- Overall amount of work D_w;
  Share of P_i: $\alpha_i \cdot D_w$ processed in a time $\alpha_i \cdot D_w \cdot w_i$
  ($\alpha_i \geq 0$, $\sum_j \alpha_j = 1$)

- Cost of a unit-size communication from P_i to P_j: $c_{i,j}$
- Cost of a sending from P_i to its successor in the ring: $D_c.c_{i,\mathsf{succ}(i)}$

Communications: 1-port model

A processor can:

- send at most one message at any time;
- receive at most one message at any time;
- send and receive a message simultaneously.

Objective

1. Select q processors among p

So as to minimize:

$$\max_{1 \leq i \leq p} \left\{ \chi(i) \times \left(\alpha_i \cdot D_w \cdot w_i + D_c \cdot (c_{i,\mathsf{pred}(i)} + c_{i,\mathsf{succ}(i)}) \right) \right\}$$

with $\chi(i) = 1$ if P_i participates in the computation, and 0 otherwise

Objective

1. Select q processors among p
2. Order them into a ring

So as to minimize:

$$\max_{1 \leq i \leq p} \left\{ \chi(i) \times \left(\alpha_i \cdot D_w \cdot w_i + D_c \cdot (c_{i,\mathsf{pred}(i)} + c_{i,\mathsf{succ}(i)}) \right) \right\}$$

with $\chi(i) = 1$ if P_i participates in the computation, and 0 otherwise

Objective

1. Select q processors among p
2. Order them into a ring
3. Distribute the data among them

So as to minimize:

$$\max_{1 \le i \le p} \left\{ \chi(i) \times \left(\alpha_i \cdot D_w \cdot w_i + D_c \cdot (c_{i,\mathsf{pred}(i)} + c_{i,\mathsf{succ}(i)}) \right) \right\}$$

with $\chi(i) = 1$ if P_i participates in the computation, and 0 otherwise

Objective

1. Select q processors among p
2. Order them into a ring
3. Distribute the data among them

So as to minimize:

$$\max_{1 \leq i \leq p} \left\{ \chi(i) \times \left(\alpha_i \cdot D_w \cdot w_i + D_c \cdot (c_{i,\mathsf{pred}(i)} + c_{i,\mathsf{succ}(i)}) \right) \right\}$$

or

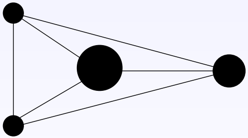
$$\max_{1 \leq i \leq p} \left\{ \chi(i) \times \max \left\{ \alpha_i D_w w_i + (c_{i,\mathsf{pred}(i)} + c_{i,\mathsf{succ}(i)}) D_c \, , \; (c_{\mathsf{pred}(i),i} + c_{\mathsf{succ}(i),i}) D_c \right\} \right\}$$

with $\chi(i) = 1$ if P_i participates in the computation, and 0 otherwise

Outline

Special hypotheses

1. There exists a communication link between any two processors
2. All links have the same characteristic
   $(\forall i, j \ c_{i,j} = c)$



Consequences

- Either the most powerful processor performs all the work, or all the processors participate
- If all processors participate, all end their share of work simultaneously $\alpha_i D_w$ *rational values ???*
  $(\exists \tau, \quad \alpha_i D_w w_i = \tau$, so $1 = \sum_i \frac{\tau}{D_w w_i})$
- Time of the optimal solution:

$$T_{\text{step}} = \min \left\{ D_w w_{\min}, D_w \frac{1}{\sum_i \frac{1}{w_i}} + 2 D_c c \right\}$$

Consequences

- Either the most powerful processor performs all the work, or all the processors participate
- If all processors participate, all end their share of work simultaneously $\alpha_i D_w$ *rational values ???*
  $(\exists \tau, \quad \alpha_i D_w w_i = \tau$, so $1 = \sum_i \frac{\tau}{D_w w_i})$
- Time of the optimal solution:

$$T_{\text{step}} = \min \left\{ D_w w_{\min}, D_w \frac{1}{\sum_i \frac{1}{w_i}} + 2 D_c c \right\}$$

Consequences

- Either the most powerful processor performs all the work, or all the processors participate
- If all processors participate, all end their share of work simultaneously $\alpha_i D_w$ *rational values ???*
  ($\exists \tau, \quad \alpha_i D_w w_i = \tau$, so $1 = \sum_i \frac{\tau}{D_w w_i}$)
- Time of the optimal solution:

$$T_{\text{step}} = \min \left\{ D_w w_{\min}, D_w \frac{1}{\sum_i \frac{1}{w_i}} + 2 D_c c \right\}$$

Consequences

- Either the most powerful processor performs all the work, or all the processors participate
- If all processors participate, all end their share of work simultaneously $\alpha_i D_w$ *rational values ???*
  ($\exists \tau, \quad \alpha_i D_w w_i = \tau$, so $1 = \sum_i \frac{\tau}{D_w w_i}$)
- Time of the optimal solution:

$$T_{\text{step}} = \min\left\{ D_w w_{\min}, D_w \frac{1}{\sum_i \frac{1}{w_i}} + 2 D_c c \right\}$$

Consequences

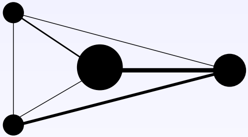
- Either the most powerful processor performs all the work, or all the processors participate
- If all processors participate, all end their share of work simultaneously $\alpha_i D_w$ *rational values ???*
  ($\exists \tau, \quad \alpha_i D_w w_i = \tau$, so $1 = \sum_i \frac{\tau}{D_w w_i}$)
- Time of the optimal solution:

$$T_{\textsf{step}} = \min \left\{ D_w w_{\min}, D_w \frac{1}{\sum_i \frac{1}{w_i}} + 2 D_c c \right\}$$

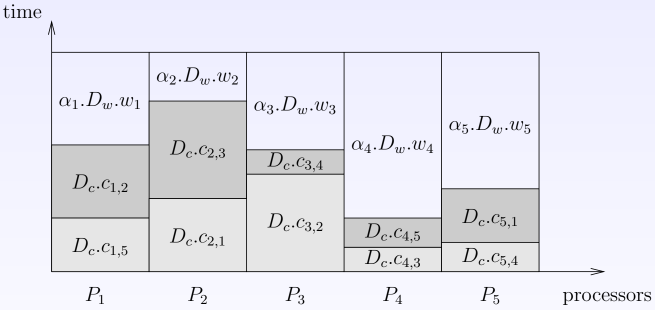
Outline

Special hypothesis

1. There exists a communication link between any two processors



All the processors participate: study (1)



All processors end simultaneously

All the processors participate: study (2)

- All processors end simultaneously

$$T_{\text{step}} = \alpha_i \cdot D_w \cdot w_i + D_c \cdot (c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)})$$

- $\sum_{i=1}^{p} \alpha_i = 1 \quad \Rightarrow \quad \sum_{i=1}^{p} \frac{T_{\text{step}} - D_c \cdot (c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)})}{D_w \cdot w_i} = 1.$

  Thus

$$\frac{T_{\text{step}}}{D_w \cdot w_{\text{cumul}}} = 1 + \frac{D_c}{D_w} \sum_{i=1}^{p} \frac{c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)}}{w_i}$$

where $w_{\text{cumul}} = \frac{1}{\sum_i \frac{1}{w_i}}$

All the processors participate: study (2)

- All processors end simultaneously

$$T_{\text{step}} = \alpha_i \cdot D_w \cdot w_i + D_c \cdot (c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)})$$

- $\displaystyle\sum_{i=1}^{p} \alpha_i \;=\; 1 \quad \Rightarrow \quad \sum_{i=1}^{p} \frac{T_{\text{step}} - D_c \cdot (c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)})}{D_w \cdot w_i} \;=\; 1.$

  Thus

$$\frac{T_{\text{step}}}{D_w \cdot w_{\text{cumul}}} = 1 + \frac{D_c}{D_w} \sum_{i=1}^{p} \frac{c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)}}{w_i}$$

where $w_{\text{cumul}} = \frac{1}{\sum_i \frac{1}{w_i}}$

All the processors participate: interpretation

$$\frac{T_{\mathsf{step}}}{D_w \cdot w_{\mathsf{cumul}}} = 1 + \frac{D_c}{D_w} \sum_{i=1}^{p} \frac{c_{i,\mathsf{succ}(i)} + c_{i,\mathsf{pred}(i)}}{w_i}$$

T_{step} is minimal when $\sum_{i=1}^{p} \frac{c_{i,\mathsf{succ}(i)} + c_{i,\mathsf{pred}(i)}}{w_i}$ is minimal

Look for an hamiltonian cycle of minimal weight in a graph where the edge from P_i to P_j has a weight of $d_{i,j} = \frac{c_{i,j}}{w_i} + \frac{c_{j,i}}{w_j}$

NP-complete problem

All the processors participate: interpretation

$$\frac{T_{\text{step}}}{D_w \cdot w_{\text{cumul}}} = 1 + \frac{D_c}{D_w} \sum_{i=1}^{p} \frac{c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)}}{w_i}$$

T_{step} is minimal when $\sum_{i=1}^{p} \frac{c_{i,\text{succ}(i)} + c_{i,\text{pred}(i)}}{w_i}$ is minimal

Look for an hamiltonian cycle of minimal weight in a graph where the edge from P_i to P_j has a weight of $d_{i,j} = \frac{c_{i,j}}{w_i} + \frac{c_{j,i}}{w_j}$

NP-complete problem

All the processors participate: interpretation

$$\frac{T_{\mathsf{step}}}{D_w \cdot w_{\mathsf{cumul}}} = 1 + \frac{D_c}{D_w} \sum_{i=1}^{p} \frac{c_{i,\mathsf{succ}(i)} + c_{i,\mathsf{pred}(i)}}{w_i}$$

T_{step} is minimal when $\sum_{i=1}^{p} \frac{c_{i,\mathsf{succ}(i)} + c_{i,\mathsf{pred}(i)}}{w_i}$ is minimal

Look for an hamiltonian cycle of minimal weight in a graph where the edge from P_i to P_j has a weight of $d_{i,j} = \frac{c_{i,j}}{w_i} + \frac{c_{j,i}}{w_j}$

NP-complete problem

All the processors participate: interpretation

$$\frac{T_{\textsf{step}}}{D_w \cdot w_{\textsf{cumul}}} = 1 + \frac{D_c}{D_w} \sum_{i=1}^{p} \frac{c_{i,\textsf{succ}(i)} + c_{i,\textsf{pred}(i)}}{w_i}$$

$T_{\textsf{step}}$ is minimal when $\displaystyle\sum_{i=1}^{p} \frac{c_{i,\textsf{succ}(i)} + c_{i,\textsf{pred}(i)}}{w_i}$ is minimal

Look for an hamiltonian cycle of minimal weight in a graph where the edge from P_i to P_j has a weight of $d_{i,j} = \frac{c_{i,j}}{w_i} + \frac{c_{j,i}}{w_j}$

NP-complete problem

All the processors participate: linear program

$$\textsc{Minimize} \ \sum_{i=1}^{p} \sum_{j=1}^{p} d_{i,j} \cdot x_{i,j},$$

SATISFYING THE (IN)EQUATIONS

$$\begin{cases} (1) \ \sum_{j=1}^{p} x_{i,j} = 1 & 1 \leq i \leq p \\ (2) \ \sum_{i=1}^{p} x_{i,j} = 1 & 1 \leq j \leq p \\ (3) \ x_{i,j} \in \{0,1\} & 1 \leq i,j \leq p \\ (4) \ u_i - u_j + p \cdot x_{i,j} \leq p - 1 & 2 \leq i,j \leq p, i \neq j \\ (5) \ u_i \text{ integer}, u_i \geq 0 & 2 \leq i \leq p \end{cases}$$

$x_{i,j} = 1$ if, and only if, the edge from P_i to P_j is used

General case : linear program

Best ring made of q processors

MINIMIZE T SATISFYING THE (IN)EQUATIONS

$$
\begin{cases}
(1)\ x_{i,j} \in \{0,1\} & 1 \leq i,j \leq p \\
(2)\ \sum_{i=1}^{p} x_{i,j} \leq 1 & 1 \leq j \leq p \\
(3)\ \sum_{i=1}^{p} \sum_{j=1}^{p} x_{i,j} = q & \\
(4)\ \sum_{i=1}^{p} x_{i,j} = \sum_{i=1}^{p} x_{j,i} & 1 \leq j \leq p \\
\\
(5)\ \sum_{i=1}^{p} \alpha_i = 1 & \\
(6)\ \alpha_i \leq \sum_{j=1}^{p} x_{i,j} & 1 \leq i \leq p \\
(7)\ \alpha_i \cdot w_i + \frac{D_c}{D_w} \sum_{j=1}^{p} (x_{i,j} c_{i,j} + x_{j,i} c_{j,i}) \leq T & 1 \leq i \leq p \\
\\
(8)\ \sum_{i=1}^{p} y_i = 1 & \\
(9)\ -p \cdot y_i - p \cdot y_j + u_i - u_j + q \cdot x_{i,j} \leq q - 1 & 1 \leq i,j \leq p, i \neq j \\
(10)\ y_i \in \{0,1\} & 1 \leq i \leq p \\
(11)\ u_i \text{ integer}, u_i \geq 0 & 1 \leq i \leq p
\end{cases}
$$

Linear programming

- Problems with rational variables: can be solved in polynomial time (in the size of the problem).
- Problems with integer variables: solved in exponential time in the worst case.
- No relaxation in rationals seems possible here...

Linear programming

- Problems with rational variables: can be solved in polynomial time (in the size of the problem).
- Problems with integer variables: solved in exponential time in the worst case.
- No relaxation in rationals seems possible here...

Linear programming

- Problems with rational variables: can be solved in polynomial time (in the size of the problem).
- Problems with integer variables: solved in exponential time in the worst case.
- No relaxation in rationals seems possible here...

And, in practice ?

All processors participate. One can use a heuristic to solve the traveling salesman problem (as Lin-Kernighan's one)

No guarantee, but excellent results in practice.

General case.

1. Exhaustive search: feasible until a dozen of processors. . .
2. Greedy heuristic: initially we take the best pair of processors; for a given ring we try to insert any unused processor in between any pair of neighbor processors in the ring. . .

And, in practice ?

All processors participate. One can use a heuristic to solve the traveling salesman problem (as Lin-Kernighan's one)
No guarantee, but excellent results in practice.

General case.

1. Exhaustive search: feasible until a dozen of processors. . .
2. Greedy heuristic: initially we take the best pair of processors; for a given ring we try to insert any unused processor in between any pair of neighbor processors in the ring. . .

And, in practice ?

All processors participate. One can use a heuristic to solve the traveling salesman problem (as Lin-Kernighan's one)
No guarantee, but excellent results in practice.

General case.

1. Exhaustive search: feasible until a dozen of processors. . .
2. Greedy heuristic: initially we take the best pair of processors; for a given ring we try to insert any unused processor in between any pair of neighbor processors in the ring. . .

And, in practice ?

All processors participate. One can use a heuristic to solve the traveling salesman problem (as Lin-Kernighan's one)
No guarantee, but excellent results in practice.

General case.

1. Exhaustive search: feasible until a dozen of processors. . .
2. Greedy heuristic: initially we take the best pair of processors; for a given ring we try to insert any unused processor in between any pair of neighbor processors in the ring. . .

And, in practice ?

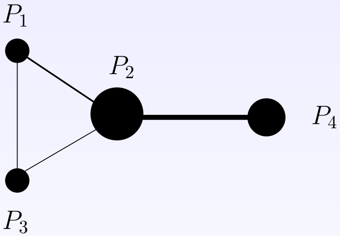
All processors participate. One can use a heuristic to solve the traveling salesman problem (as Lin-Kernighan's one)
No guarantee, but excellent results in practice.

General case.

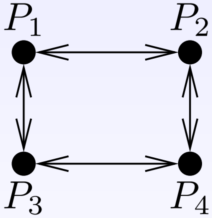
1. Exhaustive search: feasible until a dozen of processors...
2. Greedy heuristic: initially we take the best pair of processors; for a given ring we try to insert any unused processor in between any pair of neighbor processors in the ring...

Outline

New difficulty: communication links sharing



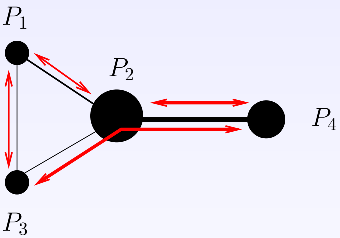
Heterogeneous platform



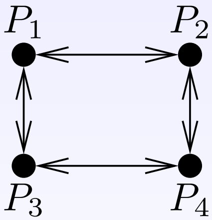
Virtual ring

We must take communication link sharing into account.

New difficulty: communication links sharing



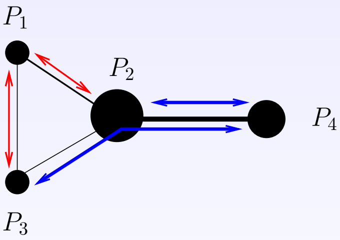
Heterogeneous platform



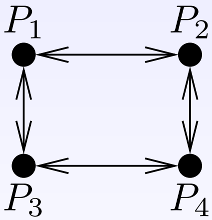
Virtual ring

We must take communication link sharing into account.

New difficulty: communication links sharing



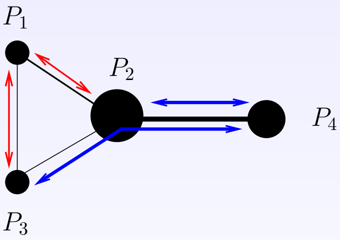
Heterogeneous platform



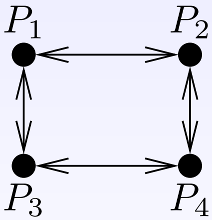
Virtual ring

We must take communication link sharing into account.

New difficulty: communication links sharing



Heterogeneous platform



Virtual ring

We must take communication link sharing into account.

New notations

- A set of communications links: $e_1, \ldots, e_n$
- Bandwidth of link e_m: b_{e_m}
- There is a path $\mathcal{S}_i$ from P_i to $P_{\text{succ}(i)}$ in the network
  - $\mathcal{S}_i$ uses a fraction $s_{i,m}$ of the bandwidth b_{e_m} of link e_m
  - P_i needs a time $D_c \cdot \frac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ to send to its successor a message of size D_c
  - Constraints on the bandwidth of e_m: $\sum_{1 \le i \le p} s_{i,m} \le b_{e_m}$
- Symmetrically, there is a path $\mathcal{P}_i$ from P_i to $P_{\text{pred}(i)}$ in the network, which uses a fraction $p_{i,m}$ of the bandwidth b_{e_m} of link e_m

New notations

- A set of communications links: $e_1, \ldots, e_n$
- Bandwidth of link e_m: b_{e_m}
- There is a path $\mathcal{S}_i$ from P_i to $P_{\text{succ}(i)}$ in the network
  - $\mathcal{S}_i$ uses a fraction $s_{i,m}$ of the bandwidth b_{e_m} of link e_m
  - P_i needs a time $D_c \cdot \dfrac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ to send to its successor a message of size D_c
  - Constraints on the bandwidth of e_m: $\sum_{1 \le i \le p} s_{i,m} \le b_{e_m}$
- Symmetrically, there is a path $\mathcal{P}_i$ from P_i to $P_{\text{pred}(i)}$ in the network, which uses a fraction $p_{i,m}$ of the bandwidth b_{e_m} of link e_m

New notations

- A set of communications links: $e_1, \ldots, e_n$
- Bandwidth of link e_m: b_{e_m}
- There is a path $\mathcal{S}_i$ from P_i to $P_{\mathsf{succ}(i)}$ in the network
  - $\mathcal{S}_i$ uses a fraction $s_{i,m}$ of the bandwidth b_{e_m} of link e_m
  - P_i needs a time $D_c \cdot \frac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ to send to its successor a message of size D_c
  - Constraints on the bandwidth of e_m: $\sum_{1 \le i \le p} s_{i,m} \le b_{e_m}$
- Symmetrically, there is a path $\mathcal{P}_i$ from P_i to $P_{\mathsf{pred}(i)}$ in the network, which uses a fraction $p_{i,m}$ of the bandwidth b_{e_m} of link e_m

New notations

- A set of communications links: $e_1, \ldots, e_n$
- Bandwidth of link e_m: b_{e_m}
- There is a path $\mathcal{S}_i$ from P_i to $P_{\mathsf{succ}(i)}$ in the network
    - $\mathcal{S}_i$ uses a fraction $s_{i,m}$ of the bandwidth b_{e_m} of link e_m
    - P_i needs a time $D_c \cdot \frac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ to send to its successor a message of size D_c
    - Constraints on the bandwidth of e_m: $\sum_{1 \leq i \leq p} s_{i,m} \leq b_{e_m}$
- Symmetrically, there is a path $\mathcal{P}_i$ from P_i to $P_{\mathsf{pred}(i)}$ in the network, which uses a fraction $p_{i,m}$ of the bandwidth b_{e_m} of link e_m

New notations

- A set of communications links: $e_1, \ldots, e_n$
- Bandwidth of link e_m: b_{e_m}
- There is a path $\mathcal{S}_i$ from P_i to $P_{\mathsf{succ}(i)}$ in the network
  - $\mathcal{S}_i$ uses a fraction $s_{i,m}$ of the bandwidth b_{e_m} of link e_m
  - P_i needs a time $D_c \cdot \frac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ to send to its successor a message of size D_c
  - Constraints on the bandwidth of e_m: $\sum_{1 \leq i \leq p} s_{i,m} \leq b_{e_m}$
- Symmetrically, there is a path $\mathcal{P}_i$ from P_i to $P_{\mathsf{pred}(i)}$ in the network, which uses a fraction $p_{i,m}$ of the bandwidth b_{e_m} of link e_m

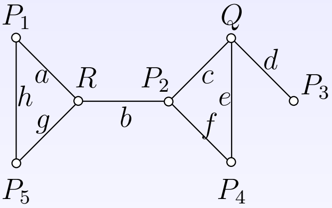
New notations

- A set of communications links: $e_1, \ldots, e_n$
- Bandwidth of link e_m: b_{e_m}
- There is a path $\mathcal{S}_i$ from P_i to $P_{\mathsf{succ}(i)}$ in the network
  - $\mathcal{S}_i$ uses a fraction $s_{i,m}$ of the bandwidth b_{e_m} of link e_m
  - P_i needs a time $D_c \cdot \dfrac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ to send to its successor a message of size D_c
  - Constraints on the bandwidth of e_m: $\displaystyle\sum_{1 \leq i \leq p} s_{i,m} \leq b_{e_m}$
- Symmetrically, there is a path $\mathcal{P}_i$ from P_i to $P_{\mathsf{pred}(i)}$ in the network, which uses a fraction $p_{i,m}$ of the bandwidth b_{e_m} of link e_m

New notations

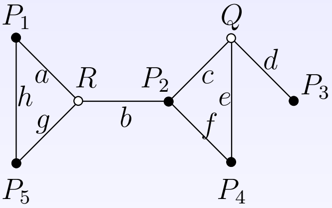
- A set of communications links: $e_1, \ldots, e_n$
- Bandwidth of link e_m: b_{e_m}
- There is a path $\mathcal{S}_i$ from P_i to $P_{\mathsf{succ}(i)}$ in the network
  - $\mathcal{S}_i$ uses a fraction $s_{i,m}$ of the bandwidth b_{e_m} of link e_m
  - P_i needs a time $D_c \cdot \frac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ to send to its successor a message of size D_c
  - Constraints on the bandwidth of e_m: $\sum_{1 \leq i \leq p} s_{i,m} \leq b_{e_m}$
- Symmetrically, there is a path $\mathcal{P}_i$ from P_i to $P_{\mathsf{pred}(i)}$ in the network, which uses a fraction $p_{i,m}$ of the bandwidth b_{e_m} of link e_m

Toy example: choosing the ring



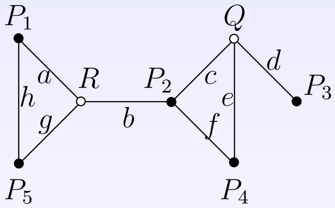
- 7 processors and 8 bidirectional communications links
- We choose a ring of 5 processors:
  $P_1 \to P_2 \to P_3 \to P_4 \to P_5$ (we use neither Q, nor R)

Toy example: choosing the ring



- 7 processors and 8 bidirectional communications links
- We choose a ring of 5 processors:
  $P_1 \to P_2 \to P_3 \to P_4 \to P_5$ (we use neither Q, nor R)

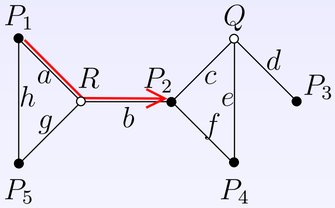
Toy example: choosing the paths



From P_1 to P_2, we use the links a and b: $\mathcal{S}_1 = \{a, b\}$.
From P_2 to P_1, we use the links b, g and h: $\mathcal{P}_2 = \{b, g, h\}$.

From P_1: to P_2, $\mathcal{S}_1 = \{a, b\}$ and to P_5, $\mathcal{P}_1 = \{h\}$
From P_2: to P_3, $\mathcal{S}_2 = \{c, d\}$ and to P_1, $\mathcal{P}_2 = \{b, g, h\}$
From P_3: to P_4, $\mathcal{S}_3 = \{d, e\}$ and to P_2, $\mathcal{P}_3 = \{d, e, f\}$
From P_4: to P_5, $\mathcal{S}_4 = \{f, b, g\}$ and to P_3, $\mathcal{P}_4 = \{e, d\}$
From P_5: to P_1, $\mathcal{S}_5 = \{h\}$ and to P_4, $\mathcal{P}_5 = \{g, b, f\}$

Toy example: choosing the paths

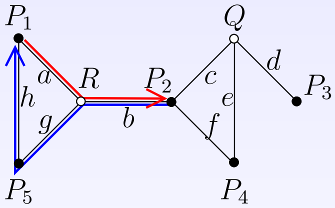


From P_1 to P_2, we use the links a and b: $\mathcal{S}_1 = \{a, b\}$.

From P_2 to P_1, we use the links b, g and h: $\mathcal{P}_2 = \{b, g, h\}$.

- From P_1: to P_2, $\mathcal{S}_1 = \{a, b\}$ and to P_5, $\mathcal{P}_1 = \{h\}$
- From P_2: to P_3, $\mathcal{S}_2 = \{c, d\}$ and to P_1, $\mathcal{P}_2 = \{b, g, h\}$
- From P_3: to P_4, $\mathcal{S}_3 = \{d, e\}$ and to P_2, $\mathcal{P}_3 = \{d, e, f\}$
- From P_4: to P_5, $\mathcal{S}_4 = \{f, b, g\}$ and to P_3, $\mathcal{P}_4 = \{e, d\}$
- From P_5: to P_1, $\mathcal{S}_5 = \{h\}$ and to P_4, $\mathcal{P}_5 = \{g, b, f\}$

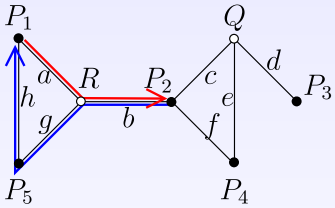
Toy example: choosing the paths



From P_1 to P_2, we use the links a and b: $\mathcal{S}_1 = \{a, b\}$.
From P_2 to P_1, we use the links b, g and h: $\mathcal{P}_2 = \{b, g, h\}$.

- From P_1: to P_2, $\mathcal{S}_1 = \{a, b\}$ and to P_5, $\mathcal{P}_1 = \{h\}$
- From P_2: to P_3, $\mathcal{S}_2 = \{c, d\}$ and to P_1, $\mathcal{P}_2 = \{b, g, h\}$
- From P_3: to P_4, $\mathcal{S}_3 = \{d, e\}$ and to P_2, $\mathcal{P}_3 = \{d, e, f\}$
- From P_4: to P_5, $\mathcal{S}_4 = \{f, b, g\}$ and to P_3, $\mathcal{P}_4 = \{e, d\}$
- From P_5: to P_1, $\mathcal{S}_5 = \{h\}$ and to P_4, $\mathcal{P}_5 = \{g, b, f\}$

Toy example: choosing the paths



From P_1 to P_2, we use the links a and b: $\mathcal{S}_1 = \{a, b\}$.
From P_2 to P_1, we use the links b, g and h: $\mathcal{P}_2 = \{b, g, h\}$.

From P_1: to P_2, $\mathcal{S}_1 = \{a, b\}$ and to P_5, $\mathcal{P}_1 = \{h\}$
From P_2: to P_3, $\mathcal{S}_2 = \{c, d\}$ and to P_1, $\mathcal{P}_2 = \{b, g, h\}$
From P_3: to P_4, $\mathcal{S}_3 = \{d, e\}$ and to P_2, $\mathcal{P}_3 = \{d, e, f\}$
From P_4: to P_5, $\mathcal{S}_4 = \{f, b, g\}$ and to P_3, $\mathcal{P}_4 = \{e, d\}$
From P_5: to P_1, $\mathcal{S}_5 = \{h\}$ and to P_4, $\mathcal{P}_5 = \{g, b, f\}$

Toy example: bandwidth sharing

From P_1 to P_2 we use links a and b: $c_{1,2} = \frac{1}{\min(s_{1,a}, s_{1,b})}$.
From P_1 to P_5 we use the link h: $c_{1,5} = \frac{1}{p_{1,h}}$.

Set of all sharing constraints:

Link a: $s_{1,a} \leq b_a$
Link b: $s_{1,b} + s_{4,b} + p_{2,b} + p_{5,b} \leq b_b$
Link c: $s_{2,c} \leq b_c$
Link d: $s_{2,d} + s_{3,d} + p_{3,d} + p_{4,d} \leq b_d$
Link e: $s_{3,e} + p_{3,e} + p_{4,e} \leq b_e$
Link f: $s_{4,f} + p_{3,f} + p_{5,f} \leq b_f$
Link g: $s_{4,g} + p_{2,g} + p_{5,g} \leq b_g$
Link h: $s_{5,h} + p_{1,h} + p_{2,h} \leq b_h$

Toy example: bandwidth sharing

From P_1 to P_2 we use links a and b: $c_{1,2} = \frac{1}{\min(s_{1,a}, s_{1,b})}$.
From P_1 to P_5 we use the link h: $c_{1,5} = \frac{1}{p_{1,h}}$.

Set of all sharing constraints:

Link a: $s_{1,a} \leq b_a$

Link b: $s_{1,b} + s_{4,b} + p_{2,b} + p_{5,b} \leq b_b$

Link c: $s_{2,c} \leq b_c$

Link d: $s_{2,d} + s_{3,d} + p_{3,d} + p_{4,d} \leq b_d$

Link e: $s_{3,e} + p_{3,e} + p_{4,e} \leq b_e$

Link f: $s_{4,f} + p_{3,f} + p_{5,f} \leq b_f$

Link g: $s_{4,g} + p_{2,g} + p_{5,g} \leq b_g$

Link h: $s_{5,h} + p_{1,h} + p_{2,h} \leq b_h$

Toy example: final quadratic system

MINIMIZE $\max_{1 \le i \le 5} \left(\alpha_i \cdot D_w \cdot w_i + D_c \cdot (c_{i,i-1} + c_{i,i+1})\right)$ UNDER THE CONSTRAINTS

$$
\begin{cases}
\sum_{i=1}^{5} \alpha_i = 1 & & \\
s_{1,a} \le b_a & s_{1,b} + s_{4,b} + p_{2,b} + p_{5,b} \le b_b & s_{2,c} \le b_c \\
s_{2,d} + s_{3,d} + p_{3,d} + p_{4,d} \le b_d & s_{3,e} + p_{3,e} + p_{4,e} \le b_e & s_{4,f} + p_{3,f} + p_{5,f} \le b_f \\
s_{4,g} + p_{2,g} + p_{5,g} \le b_g & s_{5,h} + p_{1,h} + p_{2,h} \le b_h & \\
s_{1,a} \cdot c_{1,2} \ge 1 & s_{1,b} \cdot c_{1,2} \ge 1 & p_{1,h} \cdot c_{1,5} \ge 1 \\
s_{2,c} \cdot c_{2,3} \ge 1 & s_{2,d} \cdot c_{2,3} \ge 1 & p_{2,b} \cdot c_{2,1} \ge 1 \\
p_{2,g} \cdot c_{2,1} \ge 1 & p_{2,h} \cdot c_{2,1} \ge 1 & s_{3,d} \cdot c_{3,4} \ge 1 \\
s_{3,e} \cdot c_{3,4} \ge 1 & p_{3,d} \cdot c_{3,2} \ge 1 & p_{3,e} \cdot c_{3,2} \ge 1 \\
p_{3,f} \cdot c_{3,2} \ge 1 & s_{4,f} \cdot c_{4,5} \ge 1 & s_{4,b} \cdot c_{4,5} \ge 1 \\
s_{4,g} \cdot c_{4,5} \ge 1 & p_{4,e} \cdot c_{4,3} \ge 1 & p_{4,d} \cdot c_{4,3} \ge 1 \\
s_{5,h} \cdot c_{5,1} \ge 1 & p_{5,g} \cdot c_{5,4} \ge 1 & p_{5,b} \cdot c_{5,4} \ge 1 \\
p_{5,f} \cdot c_{5,4} \ge 1 & &
\end{cases}
$$

Toy example: the moral

The problem sums up to a quadratic system if

1. The processors are selected;
2. The processors are ordered into a ring;
3. The communication paths between the processors are known.

In other words: a quadratic system if the ring is known.

If the ring is known:

- Complete graph: closed-form expression;
- General graph: quadratic system.

Toy example: the moral

The problem sums up to a quadratic system if

1. The processors are selected;
2. The processors are ordered into a ring;
3. The communication paths between the processors are known.

In other words: a quadratic system if the ring is known.

If the ring is known:

- Complete graph: closed-form expression;
- General graph: quadratic system.

Toy example: the moral

The problem sums up to a quadratic system if

1. The processors are selected;
2. The processors are ordered into a ring;
3. The communication paths between the processors are known.

In other words: a quadratic system if the ring is known.

If the ring is known:

- Complete graph: closed-form expression;
- General graph: quadratic system.

Toy example: the moral

The problem sums up to a quadratic system if

1. The processors are selected;
2. The processors are ordered into a ring;
3. The communication paths between the processors are known.

In other words: a quadratic system if the ring is known.

If the ring is known:

- Complete graph: closed-form expression;
- General graph: quadratic system.

Toy example: the moral

The problem sums up to a quadratic system if

1. The processors are selected;
2. The processors are ordered into a ring;
3. The communication paths between the processors are known.

In other words: a quadratic system if the ring is known.

If the ring is known:

- Complete graph: closed-form expression;
- General graph: quadratic system.

And, in practice ?

We adapt our greedy heuristic:

1. Initially: best pair of processors
2. For each processor P_k (not already included in the ring)
    - For each pair (P_i, P_j) of neighbors in the ring
        1. We build the graph of the unused bandwidths (Without considering the paths between P_i and P_j)
        2. We compute the shortest paths (in terms of bandwidth) between P_k and P_i and P_j
        3. We evaluate the solution
3. We keep the best solution found at step 2 and we start again

+ refinements (*max-min fairness*, quadratic solving)

Is this meaningful ?

- No guarantee, neither theoretical, nor practical
- Simple solution:
    1. we build the complete graph whose edges are labeled with the bandwidths of the best communication paths
    2. we apply the heuristic for complete graphs
    3. we allocate the bandwidths

Is this meaningful ?

- No guarantee, neither theoretical, nor practical
- Simple solution:
  1. we build the complete graph whose edges are labeled with the bandwidths of the best communication paths
  2. we apply the heuristic for complete graphs
  3. we allocate the bandwidths

Is this meaningful ?

- No guarantee, neither theoretical, nor practical
- Simple solution:
  1. we build the complete graph whose edges are labeled with the bandwidths of the best communication paths
  2. we apply the heuristic for complete graphs
  3. we allocate the bandwidths

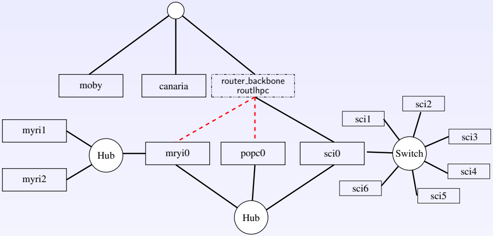
Is this meaningful ?

- No guarantee, neither theoretical, nor practical
- Simple solution:
  1. we build the complete graph whose edges are labeled with the bandwidths of the best communication paths
  2. we apply the heuristic for complete graphs
  3. we allocate the bandwidths

Is this meaningful ?

- No guarantee, neither theoretical, nor practical
- Simple solution:
  1. we build the complete graph whose edges are labeled with the bandwidths of the best communication paths
  2. we apply the heuristic for complete graphs
  3. we allocate the bandwidths

An example of an actual platform (Lyon)



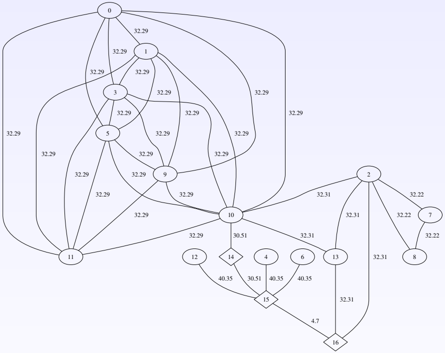
Topology

P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
0.0206	0.0206	0.0206	0.0206	0.0291	0.0206	0.0087	0.0206	0.0206

P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}
0.0206	0.0206	0.0206	0.0291	0.0451	0	0	0

Processors processing times (in seconds par megaflop)

Describing Lyon's platform



Abstracting Lyon's platform.

Results

First heuristic building the ring without taking link sharing into account

Second heuristic taking into account link sharing (and with quadratic programing)

Ratio D_c/D_w	H1		H2		Gain
0.64	0.008738	(1)	0.008738	(1)	0%
0.064	0.018837	(13)	0.006639	(14)	64.75%
0.0064	0.003819	(13)	0.001975	(14)	48.28%

Ratio D_c/D_w	H1		H2		Gain
0.64	0.005825	(1)	0.005825	(1)	0 %
0.064	0.027919	(8)	0.004865	(6)	82.57%
0.0064	0.007218	(13)	0.001608	(8)	77.72%

Table: T_{step}/D_w for each heuristic on Lyon's and Strasbourg's platforms (the numbers in parentheses show the size of the rings built).

Outline

New difficulties

The available processing power of each processor changes over time

The available bandwidth of each communication link changes over time

$\Rightarrow$ Need to reconsider the allocation previously done

$\Rightarrow$ Introduce dynamicity in a static approach

A possible approach

- If the actual performance is “too much” different from the characteristics used to build the solution
  - If the actual performance is “very” different
    - We compute a new ring
    - We redistribute data from the old ring to the new one
  - If the actual performance is “a little” different
    - We compute a new load-balancing in the existing ring
    - We redistribute the data in the ring

A possible approach

- If the actual performance is “too much” different from the characteristics used to build the solution

  Actual criterion defining “too much” ?

  - If the actual performance is “very” different
    - We compute a new ring
    - We redistribute data from the old ring to the new one

      Actual criterion defining “very” ?

      Cost of the redistribution ?
  - If the actual performance is “a little” different
    - We compute a new load-balancing in the existing ring
    - We redistribute the data in the ring

      How to efficiently do the redistribution ?

Principle of the load-balancing

Principle: the ring is modified only if this is profitable.

- T_{step}: length of an iteration *before* load-balancing;
- T'_{step}: length of an iteration *after* load-balancing;
- $T_{\text{redistribution}}$: cost of the redistribution;
- n_{iter}: number of remaining iterations

Condition: $$T_{\text{redistribution}} + n_{\text{iter}} \times T'_{\text{step}} \leq n_{\text{iter}} \times T_{\text{step}}$$

Load-balancing on a ring

- Homogeneous unidirectional ring
- Heterogeneous unidirectional ring
- Homogeneous bidirectional ring
- Heterogeneous bidirectional ring

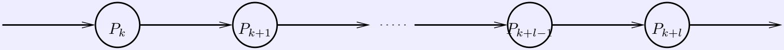
Notations

- $C_{k,l}$ the set of the processors from P_k to P_l:

$$C_{k,l} = P_k, P_{k+1}, \ldots, P_l$$

- $c_{i,i+1}$: time needed by processor P_i to send a data item to processor P_{i+1} (next one in the ring).
- Initially, processor P_i holds L_i data items (atomic).
  After redistribution, P_i will hold $L_i - \delta_i$ data items.
  δ_i is the unbalance of processor P_i.
  $\delta_{k,l}$: unbalance of the set $C_{k,l}$: $\delta_{k,l} = \sum_{i=k}^{l} \delta_i$.
  Conservation law for the data: $\sum_i \delta_i = 0$
  We assume that each processor at least one data item before and after the redistribution: $L_i \geq 1$ and $L_i \geq 1 + \delta_i$.

Lower bound on the length of the redistribution



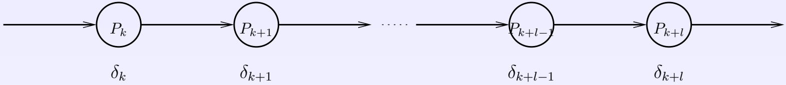
Homogeneous communication time: c.

P_k can only send messages to P_{k+1}.

P_l needs a time $\delta_{k,l} \times c$ to send $\delta_{k,l}$ data (if $\delta_{k,l} > 0$).

Lower bound: $$\left(\max_{1 \leq k \leq n,\ 0 \leq l \leq n-1} \delta_{k,k+l} \right) \times c$$

Lower bound on the length of the redistribution



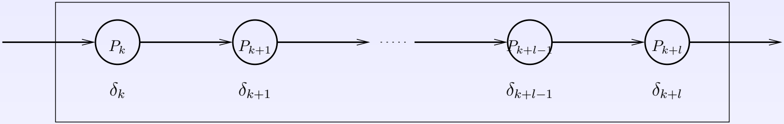
Homogeneous communication time: c.

P_k can only send messages to P_{k+1}.

P_l needs a time $\delta_{k,l} \times c$ to send $\delta_{k,l}$ data (if $\delta_{k,l} > 0$).

$$\text{Lower bound:} \quad \left(\max_{1 \le k \le n,\ 0 \le l \le n-1} \delta_{k,k+l} \right) \times c$$

Lower bound on the length of the redistribution



$$\delta_{k,k+l} = \delta_k + \delta_{k+1} + ... + \delta_{k+l-1} + \delta_{k+l}$$

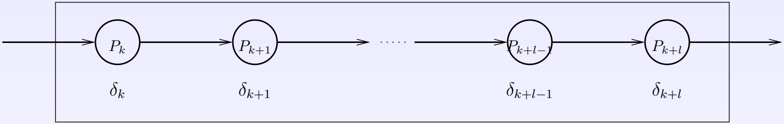
Homogeneous communication time: c.

P_k can only send messages to P_{k+1}.

P_l needs a time $\delta_{k,l} \times c$ to send $\delta_{k,l}$ data (if $\delta_{k,l} > 0$).

Lower bound: $$\left(\max_{1 \le k \le n,\ 0 \le l \le n-1} \delta_{k,k+l} \right) \times c$$

Lower bound on the length of the redistribution



$$\delta_{k,k+l} = \delta_k + \delta_{k+1} + ... + \delta_{k+l-1} + \delta_{k+l}$$

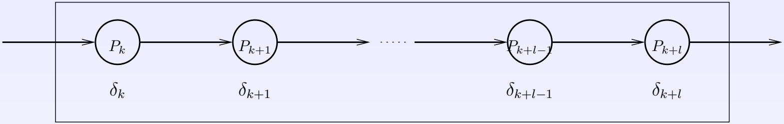
Homogeneous communication time: c.

P_k can only send messages to P_{k+1}.

P_l needs a time $\delta_{k,l} \times c$ to send $\delta_{k,l}$ data (if $\delta_{k,l} > 0$).

Lower bound: $$\left(\max_{1 \leq k \leq n,\ 0 \leq l \leq n-1} \delta_{k,k+l} \right) \times c$$

Lower bound on the length of the redistribution



$$\delta_{k,k+l} = \delta_k + \delta_{k+1} + ... + \delta_{k+l-1} + \delta_{k+l}$$

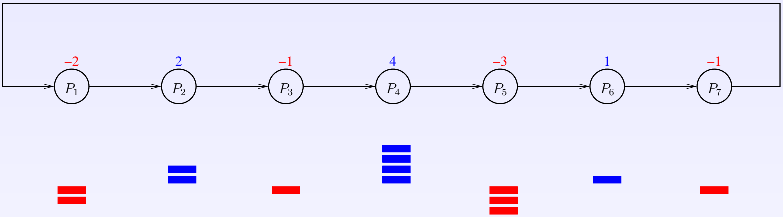
Homogeneous communication time: c.

P_k can only send messages to P_{k+1}.

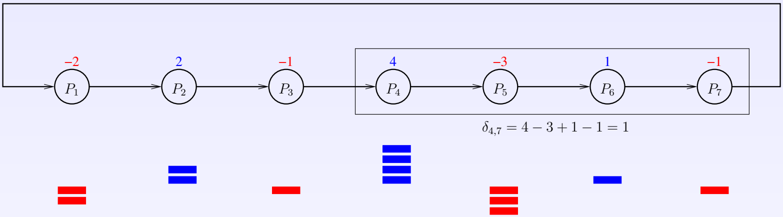
P_l needs a time $\delta_{k,l} \times c$ to send $\delta_{k,l}$ data (if $\delta_{k,l} > 0$).

Lower bound: $$\left(\max_{1 \leq k \leq n,\ 0 \leq l \leq n-1} \delta_{k,k+l} \right) \times c$$

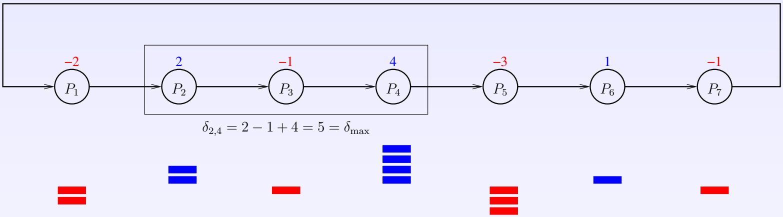
Redistribution algorithm



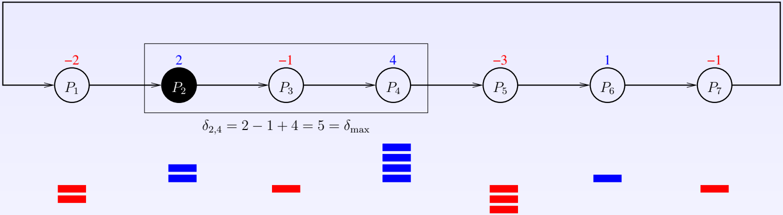
Redistribution algorithm



Redistribution algorithm



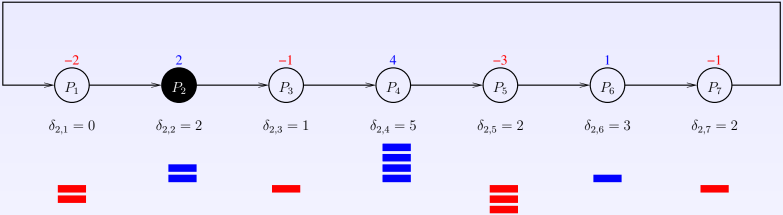
Redistribution algorithm



$\delta_{\mathsf{max}} = 5$

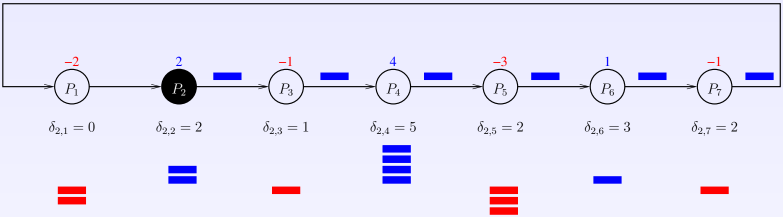
The redistribution algorithm is defined by the first processor of a "chain" of processors whose unbalance is maximal.

Redistribution algorithm



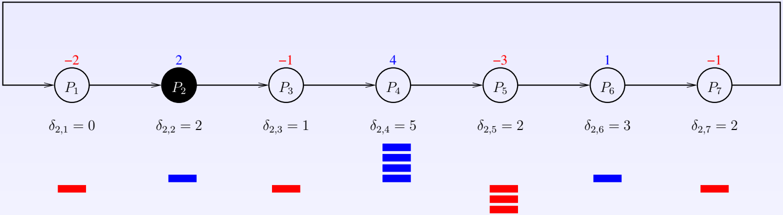
During the algorithm execution processor P_i sends $\delta_{2,i}$ data.

Redistribution algorithm



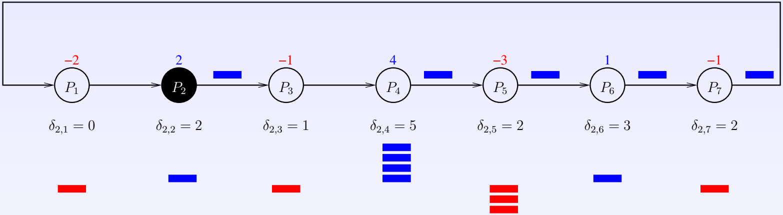
At step 1, P_i sends a data item if and only if $\delta_{2,i} \geq 1$

Redistribution algorithm



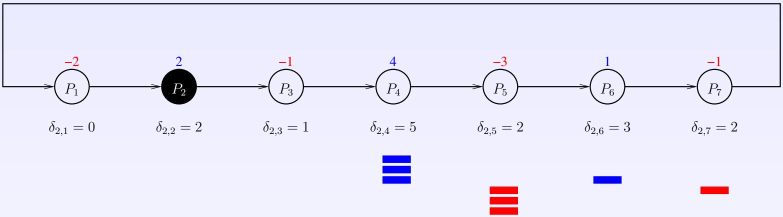
At step 1, P_i sends a data item if and only if $\delta_{2,i} \geq 1$

Redistribution algorithm



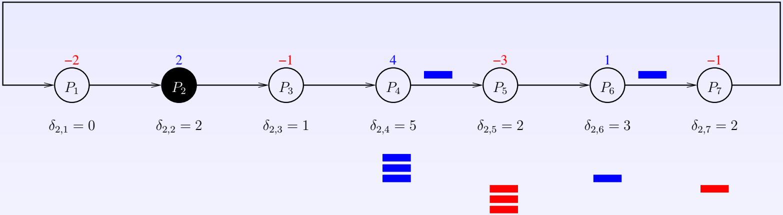
At step 2, P_i sends a data item if and only if $\delta_{2,i} \geq 2$

Redistribution algorithm



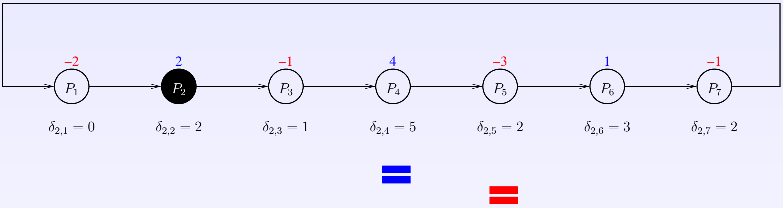
At step 2, P_i sends a data item if and only if $\delta_{2,i} \geq 2$

Redistribution algorithm



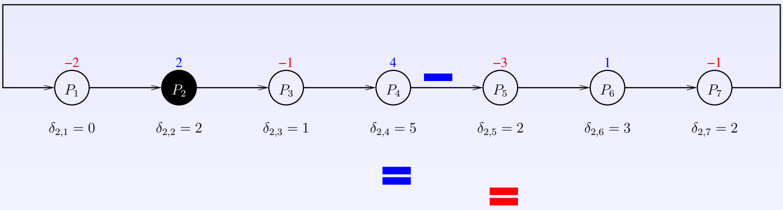
At step 3, P_i sends a data item if and only if $\delta_{2,i} \geq 3$

Redistribution algorithm



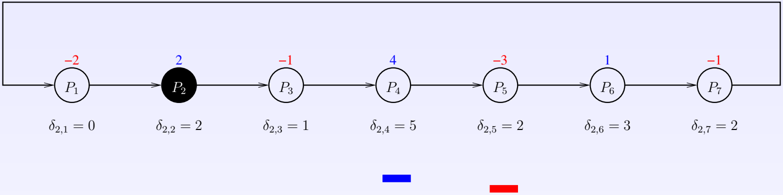
At step 3, P_i sends a data item if and only if $\delta_{2,i} \geq 3$

Redistribution algorithm



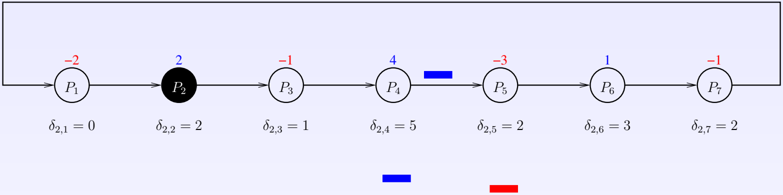
At step 4, P_i sends a data item if and only if $\delta_{2,i} \geq 4$

Redistribution algorithm



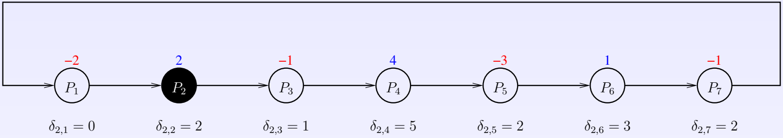
At step 4, P_i sends a data item if and only if $\delta_{2,i} \geq 4$

Redistribution algorithm



At step 5, P_i sends a data item if and only if $\delta_{2,i} \geq 5$

Redistribution algorithm



At step 5, P_i sends a data item if and only if $\delta_{2,i} \geq 5$

Homogeneous unidirectional ring: formal algorithm

Let $\delta_{\mathsf{max}} = (\max_{1 \leq k \leq n, 0 \leq l \leq n-1} |\delta_{k,k+l}|)$
Let `start` and `end` be two indices such that the slice $C_{\texttt{start},\texttt{end}}$ is of maximal imbalance: $\delta_{\texttt{start},\texttt{end}} = \delta_{\mathsf{max}}$.
for $s = 1$ to δ_{max} **do**
    for all $l = 0$ to $n - 1$ **do**
        if $\delta_{\texttt{start},\texttt{start}+l} \geq s$ **then**
            $P_{\texttt{start}+l}$ sends to $P_{\texttt{start}+l+1}$ a data item during the time interval $[(s-1) \times c, s \times c[$

Theorem

This redistribution algorithm is optimal.

Homogeneous unidirectional ring: formal algorithm

Let $\delta_{\text{max}} = (\max_{1 \leq k \leq n, 0 \leq l \leq n-1} |\delta_{k,k+l}|)$

Let `start` and `end` be two indices such that the slice $C_{\texttt{start},\texttt{end}}$ is of maximal imbalance: $\delta_{\texttt{start},\texttt{end}} = \delta_{\text{max}}$.

for $s = 1$ to δ_{max} **do**

    for all $l = 0$ to $n - 1$ **do**

        if $\delta_{\texttt{start},\texttt{start}+l} \geq s$ **then**

            $P_{\texttt{start}+l}$ sends to $P_{\texttt{start}+l+1}$ a data item during the time interval $[(s-1) \times c, s \times c[$

Theorem

This redistribution algorithm is optimal.

Heterogeneous unidirectional ring: lower bound

Processor P_i needs a time $c_{i,i+1}$ to send a data to processor P_{i+1}.

Principle of the lower bound : same as for the homogeneous case.

P_l needs a time $\delta_{k,l} \times c_{l,l+1}$ to send $\delta_{k,l}$ data items to P_{l+1} (if $\delta_{k,l} > 0$).

Lower bound: $$\max_{1 \leq k \leq n,\ 0 \leq l \leq n-1} \delta_{k,k+l} \times c_{k+l,k+l+1}$$

Heterogeneous unidirectional ring: lower bound

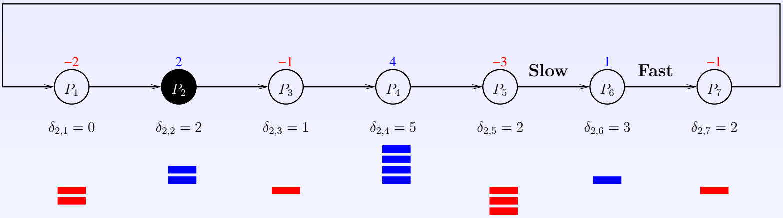
Processor P_i needs a time $c_{i,i+1}$ to send a data to processor P_{i+1}.

Principle of the lower bound : same as for the homogeneous case.

P_l needs a time $\delta_{k,l} \times c_{l,l+1}$ to send $\delta_{k,l}$ data items to P_{l+1} (if $\delta_{k,l} > 0$).

Lower bound: $$\max_{1 \leq k \leq n,\ 0 \leq l \leq n-1} \delta_{k,k+l} \times c_{k+l,k+l+1}$$

Consequences of the heterogeneity of communications

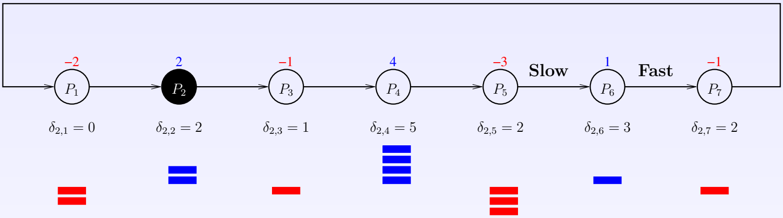


P_6 can have to receive some data items from P_5 to complete sending all the necessary data items to P_7.

We cannot express with a simple closed-form expression the time needed by P_6 to complete its share of the work.

The redistribution algorithm is asynchronous.

Consequences of the heterogeneity of communications



P_6 can have to receive some data items from P_5 to complete sending all the necessary data items to P_7.

We cannot express with a simple closed-form expression the time needed by P_6 to complete its share of the work.

The redistribution algorithm is asynchronous.

The redistribution algorithm

This is just an asynchronous version of the previous algorithm.

1: Let $\delta_{\mathsf{max}} = (\max_{1 \leq k \leq n, 0 \leq l \leq n-1} |\delta_{k,k+l}|)$

2: Let `start` and `end` be two indices such that the slice $C_{\texttt{start},\texttt{end}}$ is of maximal unbalance: $\delta_{\texttt{start},\texttt{end}} = \delta_{\mathsf{max}}$.

3: **for all** $l = 0$ to $n - 1$ **do**

4: $P_{\texttt{start}+l}$ sends $\delta_{\texttt{start},\texttt{start}+l}$ data items one by one and as soon as possible to processor $P_{\texttt{start}+l+1}$

Optimality

Obvious by construction

Lemma

The execution time of the redistribution algorithm is

$$\max_{0 \leq l \leq n-1} \delta_{start,start+l} \times c_{start+l,start+l+1}.$$

In other words, there is no propagation delay, whatever the initial distribution of the data, and whatever the communication speeds. . .

Optimality

Obvious by construction

Lemma

The execution time of the redistribution algorithm is

$$\max_{0 \leq l \leq n-1} \delta_{start,start+l} \times c_{start+l,start+l+1}.$$

In other words, there is no propagation delay, whatever the initial distribution of the data, and whatever the communication speeds. . .

Optimality

Obvious by construction

Lemma

The execution time of the redistribution algorithm is

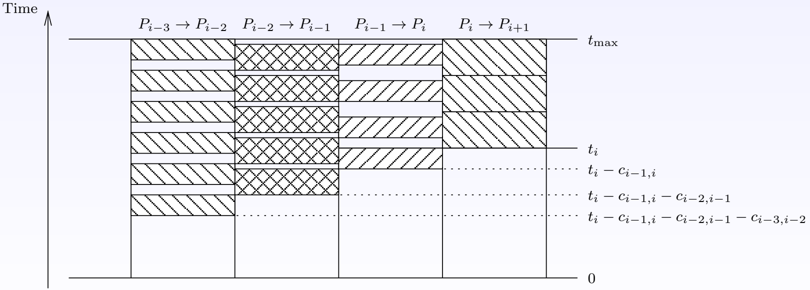
$$\max_{0 \leq l \leq n-1} \delta_{start,start+l} \times c_{start+l,start+l+1}.$$

In other words, there is no propagation delay, whatever the initial distribution of the data, and whatever the communication speeds...

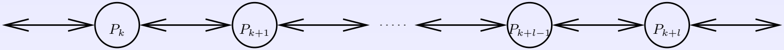
Optimality : principle of the proof

The execution time of the algorithm is

$$\max_{0 \leq l \leq n-1} \delta_{\texttt{start},\texttt{start}+l} \times c_{\texttt{start}+l,\texttt{start}+l+1}.$$



Homogeneous bidirectional ring : framework

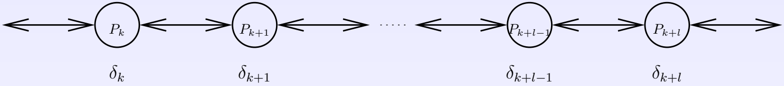


Homogeneous communication time: c.

Bidirectional communications

Lower bound: $$\max\left\{\max_{1\leq i\leq n}|\delta_i|, \max_{1\leq i\leq n, 1\leq l\leq n-1}\left\lceil\frac{|\delta_{i,i+l}|}{2}\right\rceil\right\}\times c$$

Homogeneous bidirectional ring : lower bound

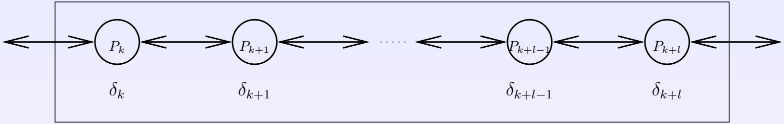


Homogeneous communication time: c.

Bidirectional communications

Lower bound: $\max \left\{ \max_{1 \leq i \leq n} |\delta_i|, \max_{1 \leq i \leq n, 1 \leq l \leq n-1} \left\lceil \frac{|\delta_{i,i+l}|}{2} \right\rceil \right\} \times c$

Homogeneous bidirectional ring : lower bound



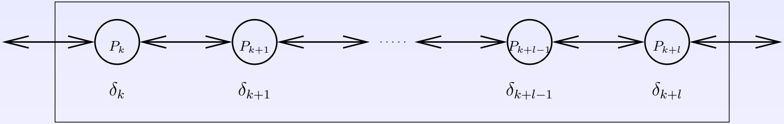
$$\delta_{k,k+l} = \delta_k + \delta_{k+1} + ... + \delta_{k+l-1} + \delta_{k+l}$$

Homogeneous communication time: c.

Bidirectional communications

Lower bound: $$\max\left\{\max_{1\leq i\leq n}|\delta_i|, \max_{1\leq i\leq n, 1\leq l\leq n-1}\left\lceil\frac{|\delta_{i,i+l}|}{2}\right\rceil\right\}\times c$$

Homogeneous bidirectional ring : lower bound



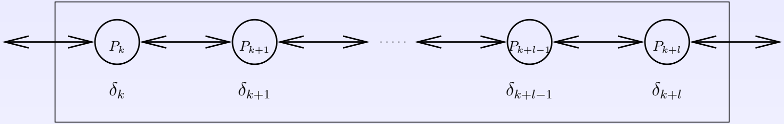
$$\delta_{k,k+l} = \delta_k + \delta_{k+1} + ... + \delta_{k+l-1} + \delta_{k+l}$$

Homogeneous communication time: c.

We need a time $\left\lceil \frac{\delta_{k,k+l}}{2} \right\rceil \times c$ to send $\delta_{k,k+l}$ data items of the processor "chain" $P_k, \ldots, P_{k+l}$(if $\delta_{k,l} > 0$).

Lower bound: $\max\left\{ \max_{1 \le i \le n} |\delta_i|, \max_{1 \le i \le n, 1 \le l \le n-1} \left\lceil \frac{|\delta_{i,i+l}|}{2} \right\rceil \right\} \times c$

Homogeneous bidirectional ring : lower bound



$$\delta_{k,k+l} = \delta_k + \delta_{k+1} + ... + \delta_{k+l-1} + \delta_{k+l}$$

Homogeneous communication time: c.

We need a time $\left\lceil \frac{\delta_{k,k+l}}{2} \right\rceil \times c$ to send $\delta_{k,k+l}$ data items of the processor "chain" $P_k, \ldots, P_{k+l}$(if $\delta_{k,l} > 0$).

Lower bound: $$\max \left\{ \max_{1 \leq i \leq n} |\delta_i|, \max_{1 \leq i \leq n, 1 \leq l \leq n-1} \left\lceil \frac{|\delta_{i,i+l}|}{2} \right\rceil \right\} \times c$$

Homogeneous bidirectional ring: principle of the algorithm

1. Each non trivial set $C_{k,l}$ such that $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\mathsf{max}}$ and $\delta_{k,l} \geq 0$ must send two data items at each step, one by each of its two extremities.
2. Each non trivial set $C_{k,l}$ such that $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\mathsf{max}}$ and $\delta_{k,l} \leq 0$ must receive two data items at each step, one by each of its two extremities.
3. Once the communications required by the two previous cases are defined, we take care of P_i such that $|\delta_i| = \delta_{\max}$.
   If P_i is already implied in a communication: everything is already set up.
   Otherwise, we have the choice of the processor to which P_i sends (case $\delta_i \geq 0$) or from which P_i receives (case $\delta_i \leq 0$) a data item.
   For the sake of simplicity: all these communications are in the same direction "from P_i to P_{i+1}".

Homogeneous bidirectional ring: principle of the algorithm

1. Each non trivial set $C_{k,l}$ such that $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\max}$ and $\delta_{k,l} \geq 0$ must send two data items at each step, one by each of its two extremities.
2. Each non trivial set $C_{k,l}$ such that $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\max}$ and $\delta_{k,l} \leq 0$ must receive two data items at each step, one by each of its two extremities.
3. Once the communications required by the two previous cases are defined, we take care of P_i such that $|\delta_i| = \delta_{\max}$.

   If P_i is already implied in a communication: everything is already set up.

   Otherwise, we have the choice of the processor to which P_i sends (case $\delta_i \geq 0$) or from which P_i receives (case $\delta_i \leq 0$) a data item.

   For the sake of simplicity: all these communications are in the same direction "from P_i to P_{i+1}".

Homogeneous bidirectional ring: principle of the algorithm

1. Each non trivial set $C_{k,l}$ such that $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\mathsf{max}}$ and $\delta_{k,l} \geq 0$ must send two data items at each step, one by each of its two extremities.
2. Each non trivial set $C_{k,l}$ such that $\left\lceil \frac{|\delta_{k,l}|}{2} \right\rceil = \delta_{\mathsf{max}}$ and $\delta_{k,l} \leq 0$ must receive two data items at each step, one by each of its two extremities.
3. Once the communications required by the two previous cases are defined, we take care of P_i such that $|\delta_i| = \delta_{\max}$.
   If P_i is already implied in a communication: everything is already set up.
   Otherwise, we have the choice of the processor to which P_i sends (case $\delta_i \geq 0$) or from which P_i receives (case $\delta_i \leq 0$) a data item.
   For the sake of simplicity: all these communications are in the same direction "from P_i to P_{i+1}".

Homogeneous bidirectional ring: optimality

Difficulties:

- Particular cases (taking care of the termination)
- Proof of the correctness of the algorithm (the optimality is then obvious)

Heterogeneous bidirectional ring: bound

The length τ of any redistribution satisfies:

$$\tau \geq \max \begin{cases} \displaystyle\max_{1 \leq k \leq n,\ \delta_k > 0} \delta_k \min\{c_{k,k-1}, c_{k,k+1}\} \\ \displaystyle\max_{1 \leq k \leq n,\ \delta_k < 0} -\delta_k \min\{c_{k-1,k}, c_{k+1,k}\} \\ \displaystyle\max_{\substack{1 \leq k \leq n, \\ 1 \leq l \leq n-2, \\ \delta_{k,k+l} > 0}} \min_{0 \leq i \leq \delta_{k,k+l}} \max\{i \cdot c_{k,k-1}, (\delta_{k,k+l} - i) \cdot c_{k+l,k+l+1}\} \\ \displaystyle\max_{\substack{1 \leq k \leq n, \\ 1 \leq l \leq n-2, \\ \delta_{k,k+l} < 0}} \min_{0 \leq i \leq -\delta_{k,k+l}} \max\{i \cdot c_{k-1,k}, -(\delta_{k,k+l} + i) \cdot c_{k+l+1,k+l}\} \end{cases}$$

Heterogeneous bidirectional ring: bound

The length τ of any redistribution satisfies:

$$\tau \geq \max \begin{cases} \max\limits_{1 \leq k \leq n,\ \delta_k > 0} \delta_k \min\{c_{k,k-1}, c_{k,k+1}\} \\ \max\limits_{1 \leq k \leq n,\ \delta_k < 0} -\delta_k \min\{c_{k-1,k}, c_{k+1,k}\} \\ \max\limits_{\substack{1 \leq k \leq n, \\ 1 \leq l \leq n-2, \\ \delta_{k,k+l} > 0}} \min\limits_{0 \leq i \leq \delta_{k,k+l}} \max\{i \cdot c_{k,k-1}, (\delta_{k,k+l} - i) \cdot c_{k+l,k+l+1}\} \\ \max\limits_{\substack{1 \leq k \leq n, \\ 1 \leq l \leq n-2, \\ \delta_{k,k+l} < 0}} \min\limits_{0 \leq i \leq -\delta_{k,k+l}} \max\{i \cdot c_{k-1,k}, -(\delta_{k,k+l} + i) \cdot c_{k+l+1,k+l}\} \end{cases}$$

Heterogeneous bidirectional ring: bound

The length τ of any redistribution satisfies:

$$\tau \geq \max \begin{cases} \max\limits_{1 \leq k \leq n,\ \delta_k > 0} \delta_k \min\{c_{k,k-1}, c_{k,k+1}\} \\ \max\limits_{1 \leq k \leq n,\ \delta_k < 0} -\delta_k \min\{c_{k-1,k}, c_{k+1,k}\} \\ \max\limits_{\substack{1 \leq k \leq n, \\ 1 \leq l \leq n-2, \\ \delta_{k,k+l} > 0}} \min\limits_{0 \leq i \leq \delta_{k,k+l}} \max\{i \cdot c_{k,k-1}, (\delta_{k,k+l} - i) \cdot c_{k+l,k+l+1}\} \\ \max\limits_{\substack{1 \leq k \leq n, \\ 1 \leq l \leq n-2, \\ \delta_{k,k+l} < 0}} \min\limits_{0 \leq i \leq -\delta_{k,k+l}} \max\{i \cdot c_{k-1,k}, -(\delta_{k,k+l} + i) \cdot c_{k+l+1,k+l}\} \end{cases}$$

Heterogeneous bidirectional ring: bound

The length τ of any redistribution satisfies:

$$\tau \geq \max \begin{cases} \max\limits_{1 \leq k \leq n,\ \delta_k > 0} \delta_k \min\{c_{k,k-1}, c_{k,k+1}\} \\ \max\limits_{1 \leq k \leq n,\ \delta_k < 0} -\delta_k \min\{c_{k-1,k}, c_{k+1,k}\} \\ \max\limits_{\substack{1 \leq k \leq n, \\ 1 \leq l \leq n-2, \\ \delta_{k,k+l} > 0}} \min\limits_{0 \leq i \leq \delta_{k,k+l}} \max\{i \cdot c_{k,k-1}, (\delta_{k,k+l} - i) \cdot c_{k+l,k+l+1}\} \\ \max\limits_{\substack{1 \leq k \leq n, \\ 1 \leq l \leq n-2, \\ \delta_{k,k+l} < 0}} \min\limits_{0 \leq i \leq -\delta_{k,k+l}} \max\{i \cdot c_{k-1,k}, -(\delta_{k,k+l} + i) \cdot c_{k+l+1,k+l}\} \end{cases}$$

Heterogeneous bidirectional ring: "light" redistributions (1)

Definition: we say that a redistribution is "light" if each processor initially holds all the data items it needs to send during the execution of the algorithm.

$\mathcal{S}_{i,j}$: amount of data sent by P_i to its neighbor P_j.

$$\text{MINIMIZE } \tau, \text{ SUBJECT TO}$$
$$\begin{cases} \mathcal{S}_{i,i+1} \geq 0 & 1 \leq i \leq n \\ \mathcal{S}_{i,i-1} \geq 0 & 1 \leq i \leq n \\ \mathcal{S}_{i,i+1} + \mathcal{S}_{i,i-1} - \mathcal{S}_{i+1,i} - \mathcal{S}_{i-1,i} = \delta_i & 1 \leq i \leq n \\ \mathcal{S}_{i,i+1} c_{i,i+1} + \mathcal{S}_{i,i-1} c_{i,i-1} \leq \tau & 1 \leq i \leq n \\ \mathcal{S}_{i+1,i} c_{i+1,i} + \mathcal{S}_{i-1,i} c_{i-1,i} \leq \tau & 1 \leq i \leq n \end{cases}$$

Heterogeneous bidirectional ring: “light” redistributions (2)

1. Any integral solution is feasible.

   Ex.: P_i sends its $\mathcal{S}_{i,i+1}$ data to P_{i+1} starting at time 0. Once this communication is completed, P_i sends $\mathcal{S}_{i,i-1}$ data to P_{i-1} as soon as it is possible under the one port model.

2. If we solve the system in rational, one of the two natural rounding in integer defines an optimal integral solution.

Heterogeneous bidirectional ring: general case

Any idea anybody ?

Outline

Conclusion

"Regular" parallelism was already complicated, now we have:

- Processors with different characteristics
- Communications links with different characteristics
- Irregular interconnection networks
- Resources whose characteristics evolve over time

We need to use a realistic model of networks. . . but a more realistic model may lead to a more complicated problem.