Iterative Algorithms (on the Impact of Network Models)

Master 2 Research Tutorial: High-Performance Architectures

Arnaud Legrand et Jean-François Méhaut

ID laboratory, arnaud.legrand@imag.fr

November 29, 2006

Outline

Outline

1. The problem
2. Fully homogeneous network
3. Heterogeneous network (complete)
4. Heterogeneous network (general case)
5. Non dedicated platforms
6. Conclusion

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)

Targeted applications: iterative algorithms

- A set of data (typically, a matrix)
- Structure of the algorithms:

Targeted applications: iterative algorithms

- A set of data (typically, a matrix)
- Structure of the algorithms:
  1. Each processor performs a computation on its chunk of data

Targeted applications: iterative algorithms

- A set of data (typically, a matrix)
- Structure of the algorithms:
  1. Each processor performs a computation on its chunk of data
  2. Each processor exchange the "border" of its chunk of data with its neighbor processors

Targeted applications: iterative algorithms

- A set of data (typically, a matrix)
- Structure of the algorithms:
  1. Each processor performs a computation on its chunk of data
  2. Each processor exchange the "border" of its chunk of data with its neighbor processors
  3. We go back at Step 1

Targeted applications: iterative algorithms

- A set of data (typically, a matrix)
- Structure of the algorithms:
  1. Each processor performs a computation on its chunk of data
  2. Each processor exchange the "border" of its chunk of data with its neighbor processors
  3. We go back at Step 1

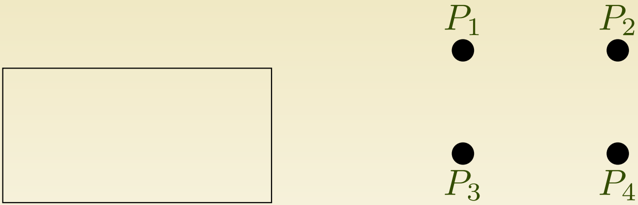
Question: how can we efficiently execute such an algorithm on such a platform?

The questions

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

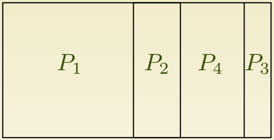
Before all, a simplification: slicing the data

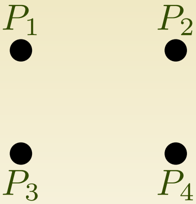
- Data: a 2-D array



Before all, a simplification: slicing the data

- Data: a 2-D array

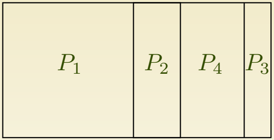


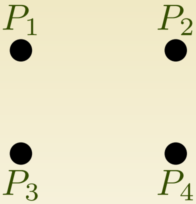


- Unidimensional cutting into vertical slices

Before all, a simplification: slicing the data

- Data: a 2-D array

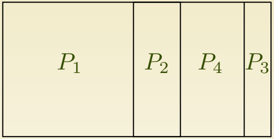


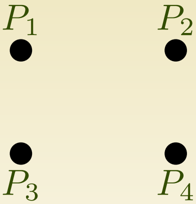


- Unidimensional cutting into vertical slices
- Consequences:

Before all, a simplification: slicing the data

- Data: a 2-D array

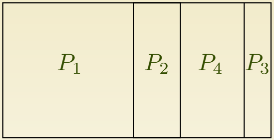


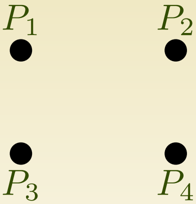


- Unidimensional cutting into vertical slices
- Consequences:
  1. Borders and neighbors are easily defined

Before all, a simplification: slicing the data

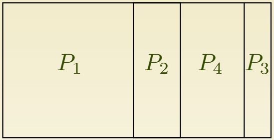
- Data: a 2-D array

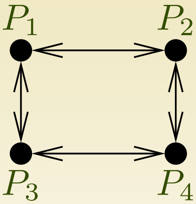




- Unidimensional cutting into vertical slices
- Consequences:
  1. Borders and neighbors are easily defined
  2. Constant volume of data exchanged between neighbors: D_c

- 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, ..., P_p

Notations

- Processors: P_1, ..., P_p
- Processor P_i executes a unit task in a time w_i

Notations

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

Notations

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

Notations

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

Objective

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

Objective

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

Objective

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

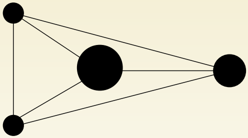
So as to minimize:

$$\max_{1 \leq i \leq p} \mathbb{I}\{i\}[\alpha_i.D_w.w_i + D_c.(c_{i,\mathsf{pred}(i)} + c_{i,\mathsf{succ}(i)})]$$

Where $\mathbb{I}\{i\}[x] = x$ if P_i participates in the computation, and 0 otherwise

Outline

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



Consequences

- Either the most powerful processor performs all the work, or all the processors participate

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

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 ???*

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 $1 = \sum_i \frac{\tau}{D_w.w_i})$

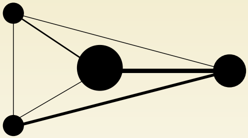
- Either the most powerful processor performs all the work, or all the processors participate
- If all processors participate, all end their share of work simultaneously $\alpha_i.D_w$ *rational values ???*
  ($\exists \tau, \quad \alpha_i.D_w.w_i = \tau$, so $1 = \sum_i \frac{\tau}{D_w.w_i}$)
- Time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

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

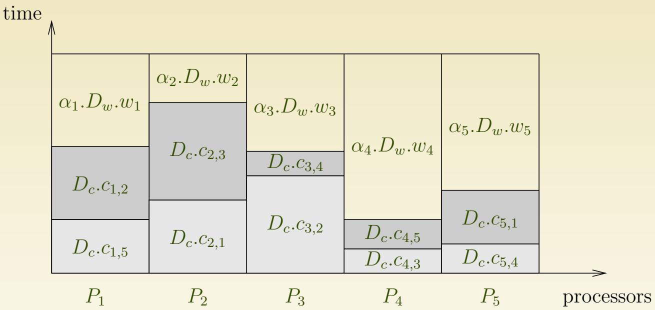
Outline

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

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

All the processors participate: study (2)

- All processors end simultaneously

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

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

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

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

All the processors participate: interpretation

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

All the processors participate: interpretation

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

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

All the processors participate: interpretation

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

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

All the processors participate: interpretation

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

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: linear program

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

$$\textsc{satisfying the (in)equations}$$

$$\left\{\begin{array}{lll} (1) & \sum_{j=1}^{p} x_{i,j} = 1 & 1 \le i \le p \\ (2) & \sum_{i=1}^{p} x_{i,j} = 1 & 1 \le j \le p \\ (3) & x_{i,j} \in \{0,1\} & 1 \le i,j \le p \\ (4) & u_i - u_j + p.x_{i,j} \le p-1 & 2 \le i,j \le p, i \ne j \\ (5) & u_i \text{ integer}, u_i \ge 0 & 2 \le i \le p \end{array}\right.$$

$x_{i,j} = 1$ 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

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

Linear programming

- Problems with rational variables: can be solved in polynomial time (in the size of the problem).

- 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.

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)

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.

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.

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...

And, in practice ?

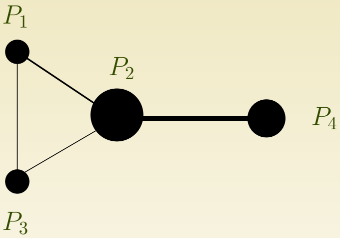
All processors participate. One can use a heuristic to solve the traveling salesman problem (as Lin-Kernighan's one)
No guarantee, but excellent results in practice.

General case.

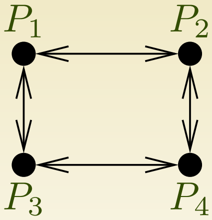
1. Exhaustive search: feasible until a dozen of processors...
2. Greedy heuristic: initially we take the best pair of processors; for a given ring we try to insert any unused processor in between any pair of neighbor processors in the ring...

Outline

New difficulty: communication links sharing

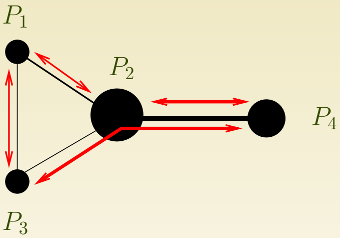


Heterogeneous platform

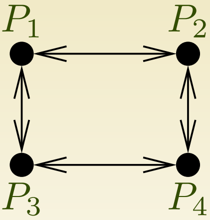


Virtual ring

New difficulty: communication links sharing

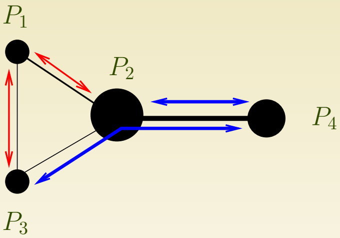


Heterogeneous platform

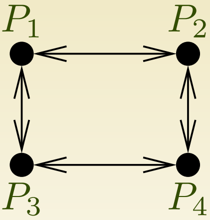


Virtual ring

New difficulty: communication links sharing

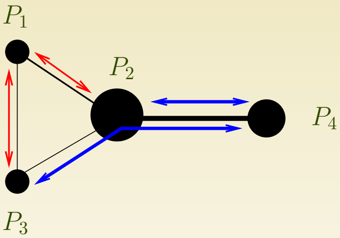


Heterogeneous platform

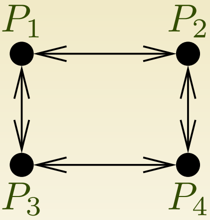


Virtual ring

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, ..., e_n

New notations

- A set of communications links: e_1, ..., e_n
- Bandwidth of link e_m: b_{e_m}

New notations

- A set of communications links: e_1, ..., 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

New notations

- A set of communications links: e_1, ..., 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

New notations

- A set of communications links: e_1, ..., 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 $D_c.\dfrac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ to send to its successor a message of size D_c

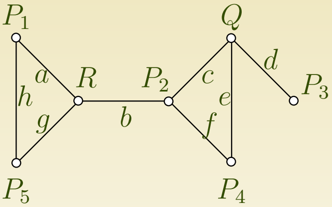
New notations

- A set of communications links: e_1, ..., 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 $D_c.\dfrac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ to send to its successor a message of size D_c
  - Constraints on the bandwidth of e_m: $\displaystyle\sum_{1 \leq i \leq p} s_{i,m} \leq b_{e_m}$

New notations

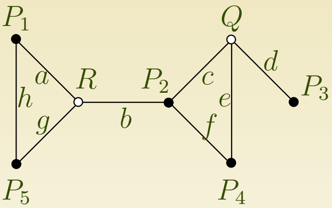
- A set of communications links: e_1, ..., e_n
- Bandwidth of link e_m: b_{e_m}
- There is a path $\mathcal{S}_i$ from P_i to $P_{\mathsf{succ}(i)}$ in the network
  - $\mathcal{S}_i$ uses a fraction $s_{i,m}$ of the bandwidth b_{e_m} of link e_m
  - P_i needs a time $D_c.\dfrac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$ 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

Toy example: choosing the ring



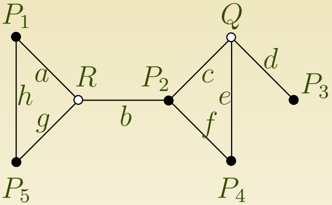
- 7 processors and 8 bidirectional communications links

Toy example: choosing the ring

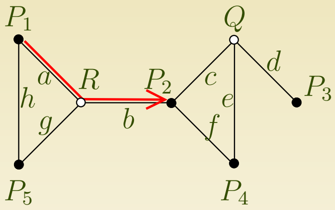


- 7 processors and 8 bidirectional communications links
- We choose a ring of 5 processors:
  $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_5$ (we use neither Q, nor R)

Toy example: choosing the paths

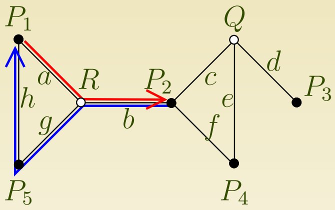


Toy example: choosing the paths



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

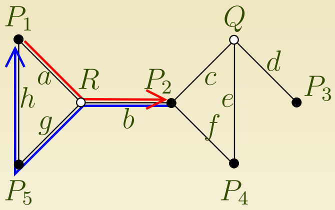
Toy example: choosing the paths



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: $\mathcal{P}_2 = \{b, g, h\}$.

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: $c_{1,5} = \frac{1}{p_{1,h}}$.

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:

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

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

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

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

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

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

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

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

Toy example: final quadratic system

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

$$
\left\{
\begin{array}{lll}
\sum_{i=1}^{5} \alpha_i = 1 & & \\
s_{1,a} \leq b_a & s_{1,b} + s_{4,b} + p_{2,b} + p_{5,b} \leq b_b & s_{2,c} \leq b_c \\
s_{2,d} + s_{3,d} + p_{3,d} + p_{4,d} \leq b_d & s_{3,e} + p_{3,e} + p_{4,e} \leq b_e & s_{4,f} + p_{3,f} + p_{5,f} \leq b_f \\
s_{4,g} + p_{2,g} + p_{5,g} \leq b_g & s_{5,h} + p_{1,h} + p_{2,h} \leq b_h & \\
s_{1,a}.c_{1,2} \geq 1 & s_{1,b}.c_{1,2} \geq 1 & p_{1,h}.c_{1,5} \geq 1 \\
s_{2,c}.c_{2,3} \geq 1 & s_{2,d}.c_{2,3} \geq 1 & p_{2,b}.c_{2,1} \geq 1 \\
p_{2,g}.c_{2,1} \geq 1 & p_{2,h}.c_{2,1} \geq 1 & s_{3,d}.c_{3,4} \geq 1 \\
s_{3,e}.c_{3,4} \geq 1 & p_{3,d}.c_{3,2} \geq 1 & p_{3,e}.c_{3,2} \geq 1 \\
p_{3,f}.c_{3,2} \geq 1 & s_{4,f}.c_{4,5} \geq 1 & s_{4,b}.c_{4,5} \geq 1 \\
s_{4,g}.c_{4,5} \geq 1 & p_{4,e}.c_{4,3} \geq 1 & p_{4,d}.c_{4,3} \geq 1 \\
s_{5,h}.c_{5,1} \geq 1 & p_{5,g}.c_{5,4} \geq 1 & p_{5,b}.c_{5,4} \geq 1 \\
p_{5,f}.c_{5,4} \geq 1 & &
\end{array}
\right.
$$

Toy example: the moral

The problem sums up to a quadratic system if

1. The processors are selected;

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;

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.

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.

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

Is this meaningful ?

- No guarantee, neither theoretical, nor practical
- Simple solution:

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

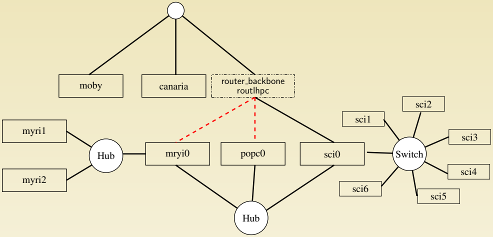
Is this meaningful ?

- 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

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)



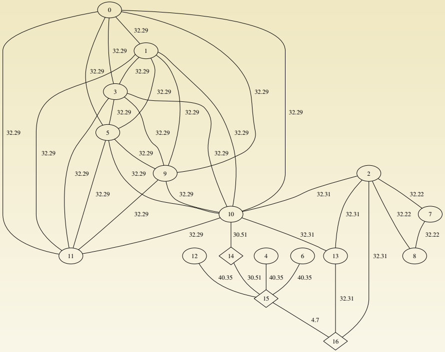
Topology

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

Modeling such a problem is hard and I won't go furthermore into the details.

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.