

Arnaud LEGRAND, H el ene RENARD, Yves ROBERT, Fr ed eric VIVIEN

LIP laboratory at  cole Normale Sup erieure de Lyon

www.ens-lyon.fr/~hrenard

Helene.Renard@ens-lyon.fr

Mapping and load-balancing iterative computations on heterogeneous clusters

GRAAL project : jointly operated by CNRS, ENS Lyon and INRIA

Algorithm design and scheduling techniques for heterogeneous clusters and grids

Outline

- 1 Framework**
- 2 Complexity
- 3 Heuristics
- 4 Simulation results
- 5 Related Work
- 6 Conclusion

Outline

- 1 Framework**
- 2 Complexity**
- 3 Heuristics
- 4 Simulation results
- 5 Related Work
- 6 Conclusion

Outline

- 1 Framework**
- 2 Complexity**
- 3 Heuristics**
- 4 Simulation results
- 5 Related Work
- 6 Conclusion

Outline

- 1 Framework
- 2 Complexity
- 3 Heuristics
- 4 Simulation results
- 5 Related Work
- 6 Conclusion

Outline

- 1 Framework
- 2 Complexity
- 3 Heuristics
- 4 Simulation results
- 5 Related Work
- 6 Conclusion

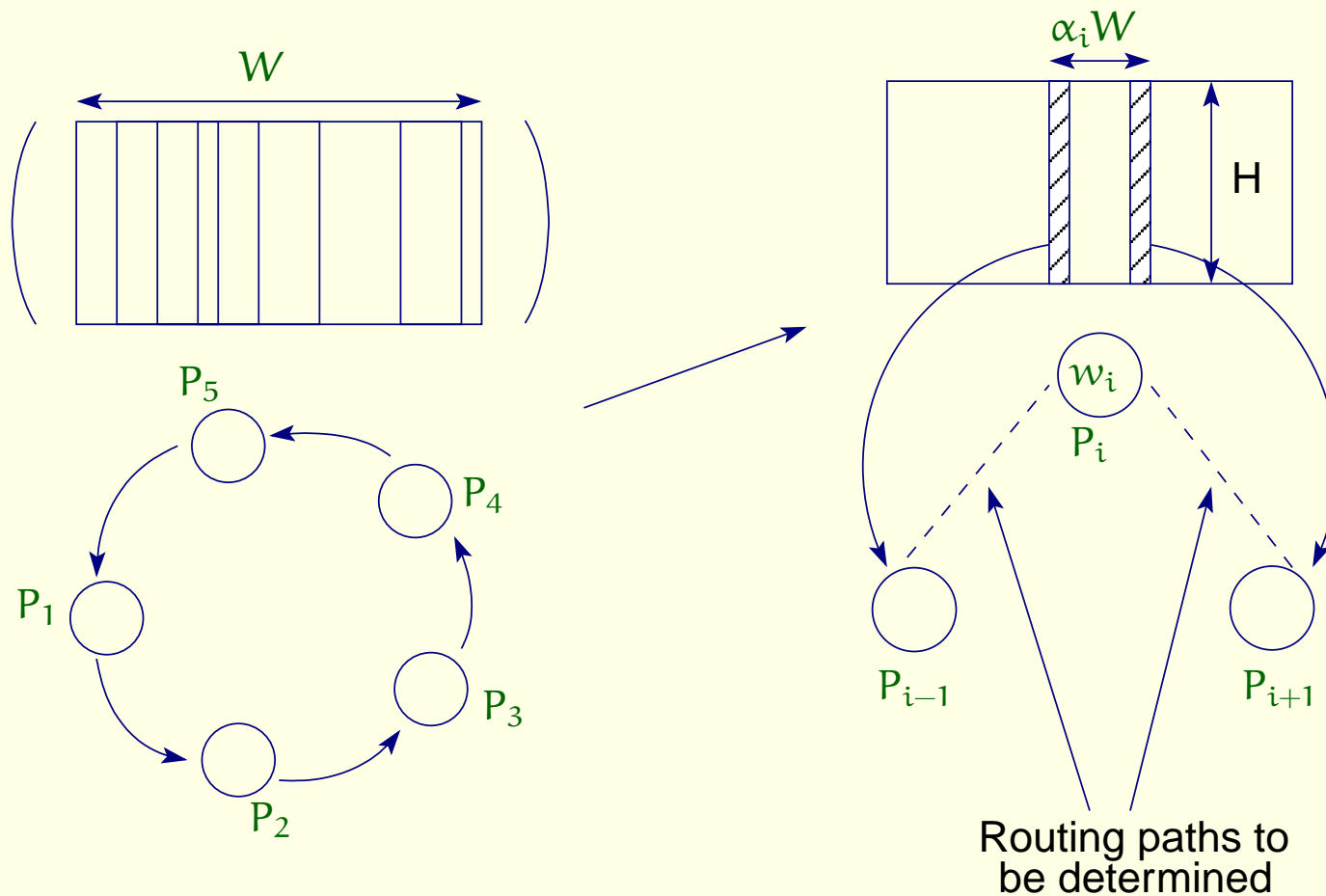
Outline

- 1 Framework
- 2 Complexity
- 3 Heuristics
- 4 Simulation results
- 5 Related Work
- 6 Conclusion

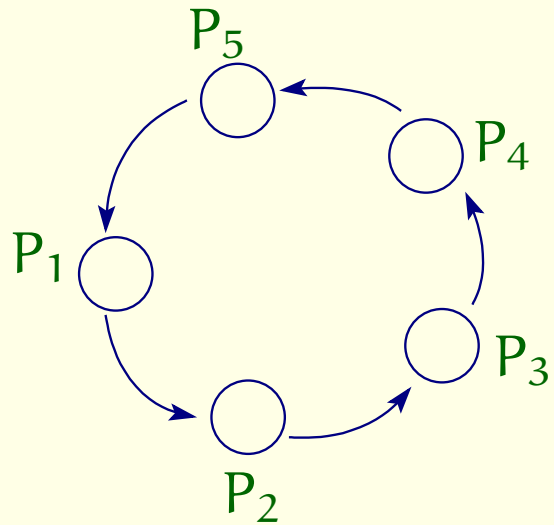
Framework

Target problem (1/3)

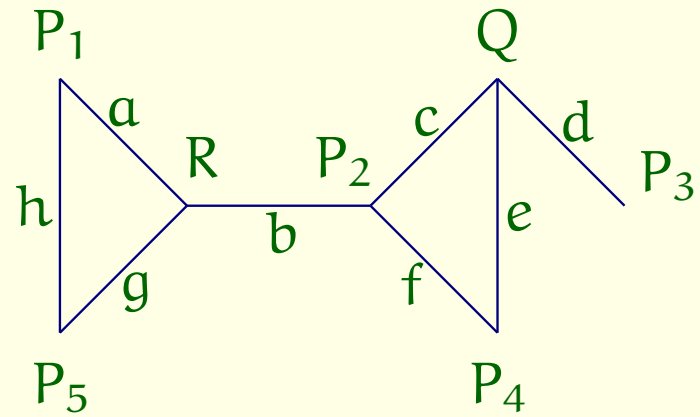
- Load balancing techniques for iterative algorithms
- Large rectangular data matrix split into vertical slices
- Search for a (virtual) processor ring embedded in target heterogeneous cluster



Target problem (2/3)



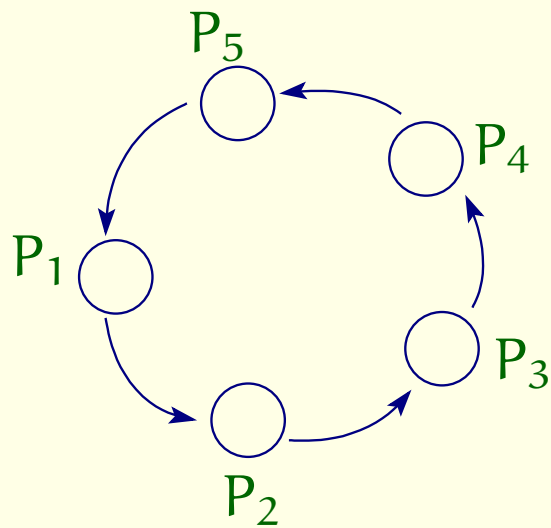
?



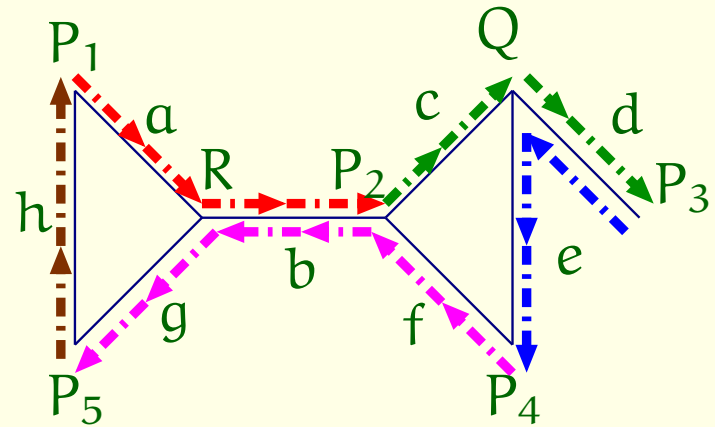
Questions

- resource selection
- ring embedding

Target problem (3/3)



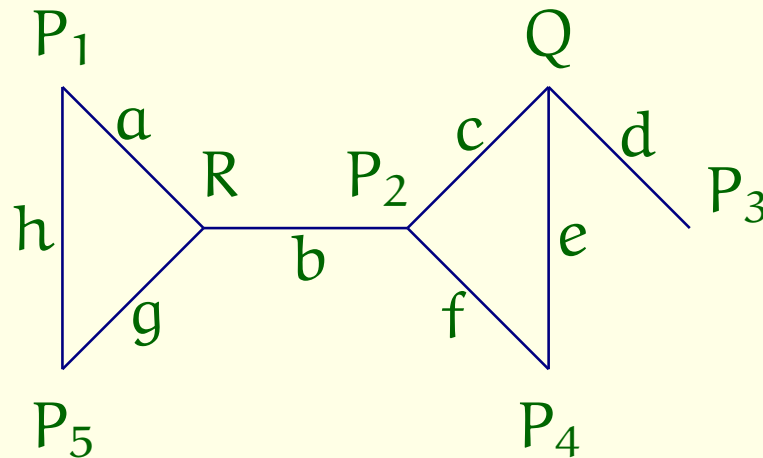
?



Questions

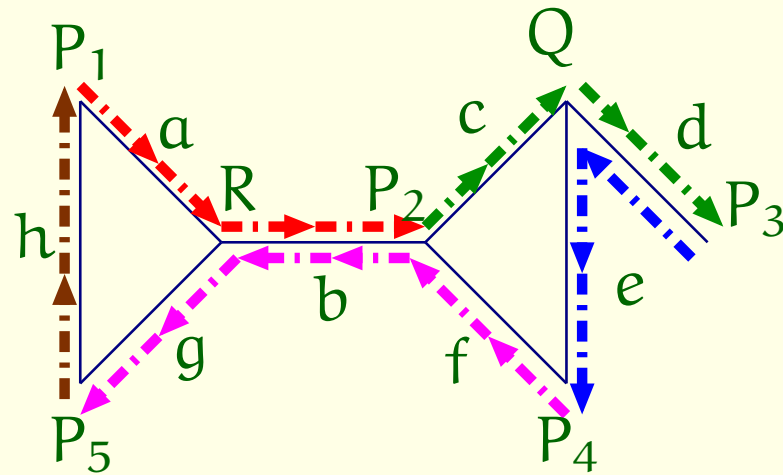
- path construction (for successive processors in the ring)
- bandwidth assignment to all paths
- chunk sizes

Toy example : choosing the ring (1/4)



- 7 processors and 8 bidirectional communication links
- Solution 5-processor ring $P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow P_4 \rightarrow P_5$, leaving out Q and R
- links labeled with letters from a to h
- b_x denotes the bandwidth of link x

Toy example : choosing the paths (2/4)



From P_1 to P_2 , we choose to use links a and b so that $\mathcal{S}_1 = \{a, b\}$. But from P_2 to P_1 , we may use links b , g and h , so that $\mathcal{P}_2 = \{b, g, h\}$.

- 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 : assigning path costs (3/4)

For P_1 , because $\mathcal{S}_1 = \{a, b\}$, we get $c_{1,2} = \frac{1}{\min(s_{1,a}, s_{1,b})}$; and because $\mathcal{P}_1 = \{h\}$, we get $c_{1,5} = \frac{1}{p_{1,h}}$.

The list of all the equations that must be satisfied :

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 : quadratic programming (4/4)

minimize $\max_{1 \leq i \leq 5} (\alpha_i \cdot W \cdot w_i + H \cdot (c_{i,i-1} + c_{i,i+1}))$ subject to

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

Modeling the platform graph (1/2)

1. Computing costs

- directed graph $G = (P, E)$
- P_i = computing resource, with (relative) cycle-time w_i

2. Communication costs

- $e \in E$ = graph edge = communication link, with bandwidth b_e
- L/b_e time-units to transfer a message of size L through edge e
- bandwidth sharing :
first message uses $2b_e/3 \Rightarrow$ second message uses no more than $b_e/3$

3. Routing

- freely decide how to route messages between processor pairs
- message along path $p = (e_1, e_2, \dots, e_m, \dots, e_k)$:
 - allocate fraction f_m of available bandwidth b_{e_m}
 - time L/b to route the message, where $b = \min_{1 \leq m \leq k} f_m$
- total bandwidth capacity of each link cannot be exceeded

Modeling the platform graph (2/2)

4. Application parameters : computations

- W = total size of the work
- P_i executes fraction $\alpha_i.W$ where $\alpha_i \geq 0$ and $\sum_{i=1}^p \alpha_i = 1$
- $\alpha_j = 0$ if P_j is not involved in the computation

5. Application parameters : communications in the ring

- P_i sends a message of length H to its successor $\text{succ}(i)$
- P_i sends a message of length H to its predecessor $\text{pred}(i)$
- \mathcal{S}_i communication path from P_i to $P_{\text{succ}(i)}$, to be determined
- $s_{i,m}$ fraction of bandwidth b_{e_m} of physical link $e_m \in \mathcal{S}_i$
- $c_{i,\text{succ}(i)} = \frac{1}{\min_{e_m \in \mathcal{S}_i} s_{i,m}}$
- P_i requires $H.c_{i,\text{succ}(i)}$ time-units to send its message to $P_{\text{succ}(i)}$

6. Objective function

$$T_{\text{step}} = \max_{1 \leq i \leq p} \mathbb{I}\{i\} [\alpha_i.W.w_i + H.(c_{i,\text{pred}(i)} + c_{i,\text{succ}(i)})]$$

SHARED RING optimization problem (1/2)

Definition SHARED RING(p, w_i, E, b_{e_m}, W, H) : minimize

$$T_{\text{step}} = \min_{1 \leq q \leq p} \min_{\sigma \in \Theta_{q,p}} \max_{1 \leq i \leq q} (\alpha_{\sigma(i)} \cdot W \cdot w_{\sigma(i)} + H \cdot (c_{\sigma(i), \sigma(i-1 \bmod q)} + c_{\sigma(i), \sigma(i+1 \bmod q)}))$$
$$\sum_{i=1}^q \alpha_{\sigma(i)} = 1$$

- $\Theta_{q,p}$ = one-to-one functions $\sigma : [1..q] \rightarrow [1..p]$ to index the q selected processors that form the ring.
- $2q$ communicating paths :
 - path \mathcal{S}_i from $P_{\sigma(i)}$ to its successor $P_{\text{succ}(\sigma(i))} = P_{\sigma(i+1 \bmod q)}$
 - path \mathcal{P}_i from $P_{\sigma(i)}$ to its predecessor $P_{\text{pred}(\sigma(i))} = P_{\sigma(i-1 \bmod q)}$
- For each link e_m
 - $s_{\sigma(i),m}$ fraction of bandwidth b_{e_m} allocated to path $\mathcal{S}_{\sigma(i)}$
 - $p_{\sigma(i),m}$ fraction of bandwidth b_{e_m} allocated to path $\mathcal{P}_{\sigma(i)}$

SHARED RING optimization problem (2/2)

Equations :

$$\left\{ \begin{array}{ll} s_{\sigma(i),m} \geq 0, p_{\sigma(i),m} \geq 0, & 1 \leq i \leq q, \quad 1 \leq m \leq E \\ c_{\sigma(i),\text{succ}(\sigma(i))} = \frac{1}{\min_{e_m \in \mathcal{S}_{\sigma(i)}} s_{\sigma(i),m}} & 1 \leq i \leq q \\ c_{\sigma(i),\text{pred}(\sigma(i))} = \frac{1}{\min_{e_m \in \mathcal{P}_{\sigma(i)}} p_{\sigma(i),m}} & 1 \leq i \leq q \\ \sum_{i=1}^q (s_{\sigma(i),m} + p_{\sigma(i),m}) \leq b_{e_m} & 1 \leq m \leq E \end{array} \right.$$

Remark The optimal solution will involve all processors as soon as the ratio $\frac{W}{H}$ is large enough

Doubt 😞 Extracting the “best” ring = difficult combinatorial problem

Variants

1 Start-up overheads

Only large-scale applications to be deployed

⇒ start-up overheads can be neglected

2 Bi-directional links

To model a bidirectional link of bandwidth b :

- assign bandwidth fraction f_{path} to each communication path requesting b , regardless its orientation

- constraint $\sum f_{\text{path}} \leq b$

⇒ uni / bi - directional links can simultaneously exist in the network

3 Multiple links

Model G as a multi-graph rather than as a simple graph.

4 Backbone links

Replace constraint $\sum f_{\text{path}} \leq b$ by $f_{\text{path}} \leq b$ for each path using the link

Variants

1 Start-up overheads

Only large-scale applications to be deployed

⇒ start-up overheads can be neglected

2 Bi-directional links

To model a bidirectional link of bandwidth b :

- assign bandwidth fraction f_{path} to each communication path requesting b , regardless its orientation

- constraint $\sum f_{\text{path}} \leq b$

⇒ uni / bi - directional links can simultaneously exist in the network

3 Multiple links

Model G as a multi-graph rather than as a simple graph.

4 Backbone links

Replace constraint $\sum f_{\text{path}} \leq b$ by $f_{\text{path}} \leq b$ for each path using the link

Variants

1 Start-up overheads

Only large-scale applications to be deployed

⇒ start-up overheads can be neglected

2 Bi-directional links

To model a bidirectional link of bandwidth b :

- assign bandwidth fraction f_{path} to each communication path requesting b , regardless its orientation

- constraint $\sum f_{\text{path}} \leq b$

⇒ uni / bi - directional links can simultaneously exist in the network

3 Multiple links

Model G as a multi-graph rather than as a simple graph.

4 Backbone links

Replace constraint $\sum f_{\text{path}} \leq b$ by $f_{\text{path}} \leq b$ for each path using the link

Variants

1 Start-up overheads

Only large-scale applications to be deployed

⇒ start-up overheads can be neglected

2 Bi-directional links

To model a bidirectional link of bandwidth b :

- assign bandwidth fraction f_{path} to each communication path requesting b , regardless its orientation

- constraint $\sum f_{\text{path}} \leq b$

⇒ uni / bi - directional links can simultaneously exist in the network

3 Multiple links

Model G as a multi-graph rather than as a simple graph.

4 Backbone links

Replace constraint $\sum f_{\text{path}} \leq b$ by $f_{\text{path}} \leq b$ for each path using the link

SLICERING optimization problem

Definition SLICERING

Same problem as before, but fixed routing and bandwidth assignment

⇔ (virtual) complete graph of dedicated links

Simulation

Given a “real” network, how can we compute (estimate) communication costs

$c_{i,j}$?

- take the actual network as input
 - compute shortest paths (in terms of bandwidths) between all processor pairs
- ⇒ (fake) fully connected network

Complexity

Complexity

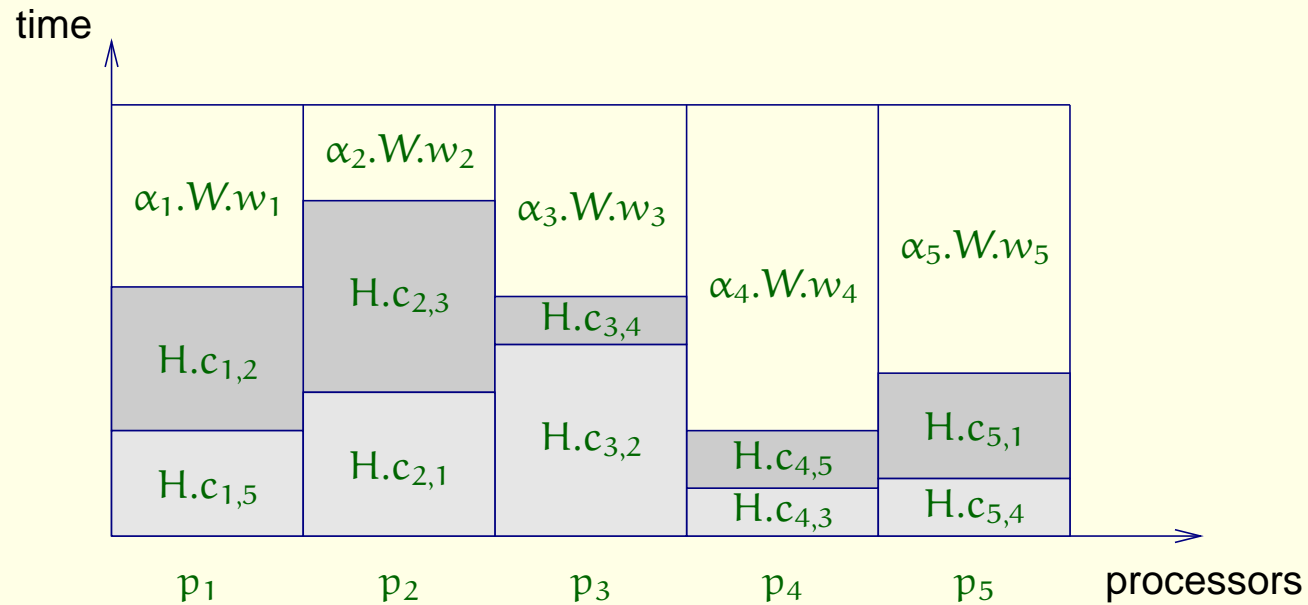
Proposition The decision problem associated to SHARED RING is NP-complete

- reduction from Hamiltonian Cycle
- same result for SLICER RING (despite simplification)

Heuristics

Ring construction (1/2)

$$T_{\text{step}} = \alpha_i \cdot W \cdot w_i + H \cdot (c_{i,i-1} + c_{i,i+1})$$



Summary of computation and communication times with $q = 5$ processors

– $\frac{T_{\text{step}}}{W \cdot w_{\text{cumul}}} = 1 + \frac{H}{W} \sum_{i=1}^q \frac{c_{i,i-1} + c_{i,i+1}}{w_i}$ where $w_{\text{cumul}} = \frac{1}{\sum_{i=1}^q \frac{1}{w_i}}$

– T_{step} minimal when $\sum_{i=1}^q \frac{c_{i,i-1} + c_{i,i+1}}{w_i}$ minimal (with all processors)

Ring construction (2/2)

1. Select best processor pair
2. Greedily include new node in the current solution ring :
 - for each pair of consecutive processors (P_j, P_k) in the ring
 - for each non participating P_i
 - compute cost of inserting P_i between P_j and P_k :
 - (a) update w_{cumul} (adding P_i decreases the value)
 - (b) in $\sum_{s=1}^r \frac{c_{\sigma(s), \sigma(s-1)} + c_{\sigma(s), \sigma(s+1)}}{w_{\sigma(s)}}$, suppress old two paths and add $\frac{c_{k,j} + c_{k,i}}{w_k}$, $\frac{c_{j,k}}{w_j}$ and $\frac{c_{i,k}}{w_i}$
 - retain pair and processor that minimize insertion cost
 - store new value of T_{step}

Bandwidth allocation

- we have already constructed $2r$ communicating paths
- a certain fraction of the initial bandwidth has been allocated to these paths

To build the new four paths involving P_k :

- $G = (V, E, b)$, each edge labeled with remaining bandwidth
- $b(e_m)$ not initial bandwidth of e_m , but what is left by the $2r$ paths
- re-inject bandwidth fractions used by the two paths between P_i and P_j
- shortest path algorithm (in terms of bandwidth) to determine the four paths, from P_k to P_i and P_j and vice-versa

Strategy :

- independently compute four paths of maximal bandwidth, using a standard shortest path algorithm in G .
- if some paths happen to share some links, do not change the paths ; use a brute force method to compute bandwidth fractions

Refinements

- greedily grow the ring by peeling off the bandwidths to insert new processors
- so far, no re-evaluation of already allocated bandwidth fractions

1. **Max-min fairness :**

traditional bandwidth-sharing algorithm, which is designed to maximize the minimum bandwidth allocated to a path

2. **Quadratic resolution :**

using the KINSOL software. Once we have a ring and all the communicating paths, the program to minimize T_{step} is **quadratic** in the unknowns α_i , $s_{i,j}$ and $p_{i,j}$

Heuristic for SLICERING

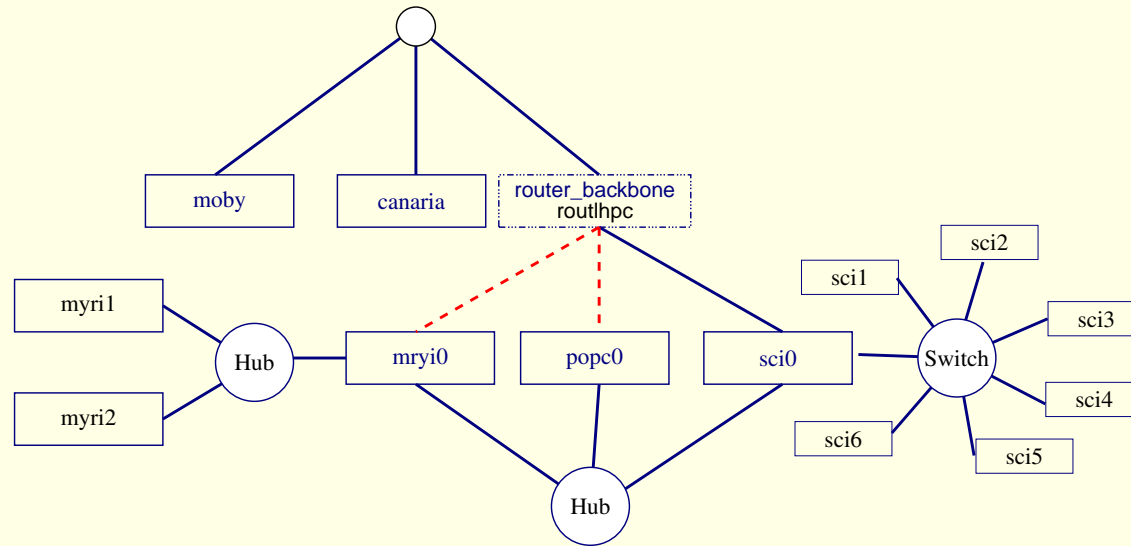
- Similar greedy heuristic :
 - compute bandwidths of the (virtual) fully-connected network
 - grow a solution ring greedily
- **Assessing the impact of link sharing**
 - the SLICERING heuristic returns the same solution as would the SHARED-RING heuristic if given the fully-connected network as input
 - keep this ring and the communicating paths, and re-compute chunk sizes and bandwidths to get a feasible solution
 - compare the latter with the solution returned by SHARED-RING

Heuristic for SLICERING

- Similar greedy heuristic :
 - compute bandwidths of the (virtual) fully-connected network
 - grow a solution ring greedily
- **Assessing the impact of link sharing**
 - the SLICERING heuristic returns the same solution as would the SHARED-RING heuristic if given the fully-connected network as input
 - keep this ring and the communicating paths, and re-compute chunk sizes and bandwidths to get a feasible solution
 - compare the latter with the solution returned by SHARED-RING

Simulation results

Platform description (Lyon)

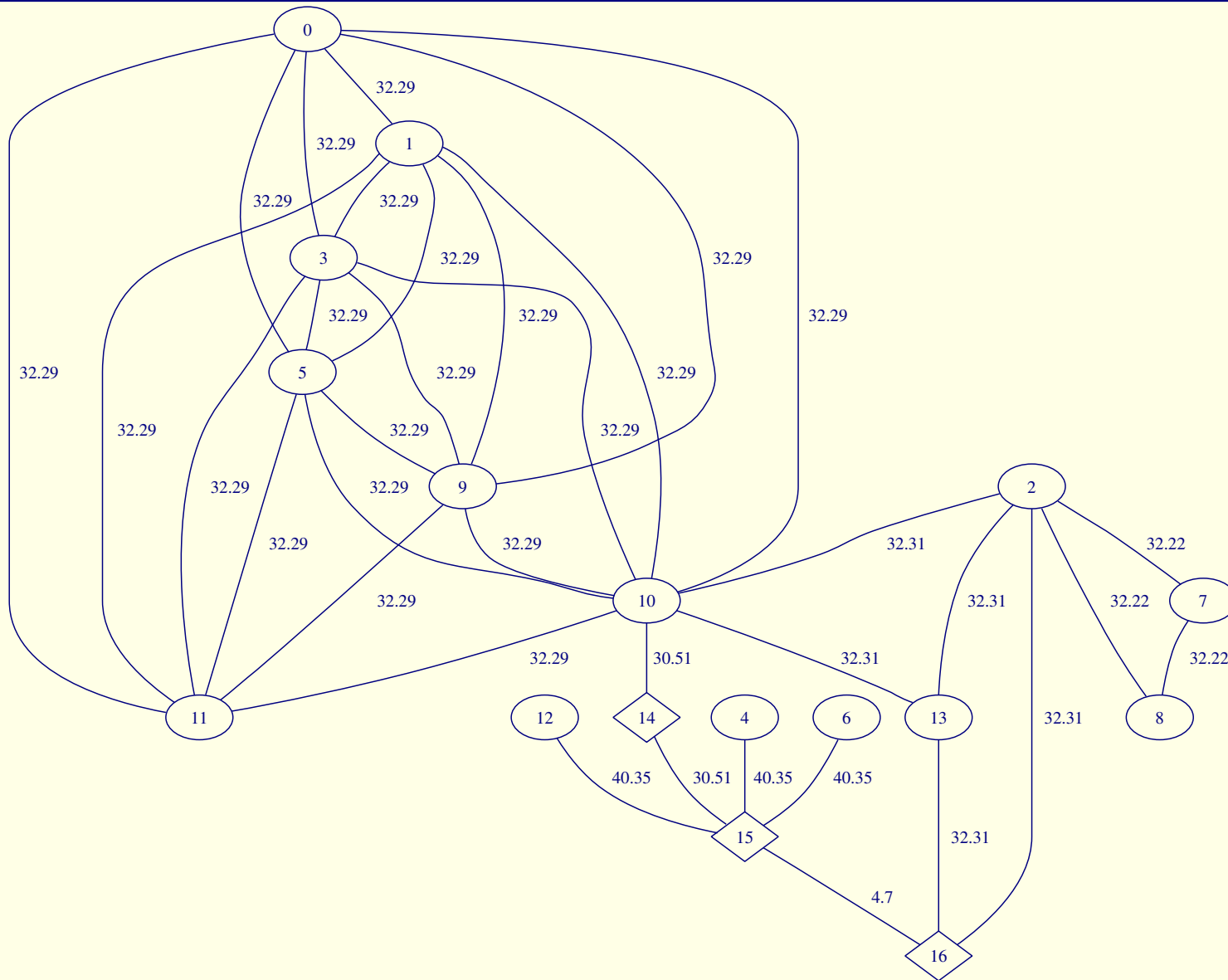


Topology of the Lyon platform

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	

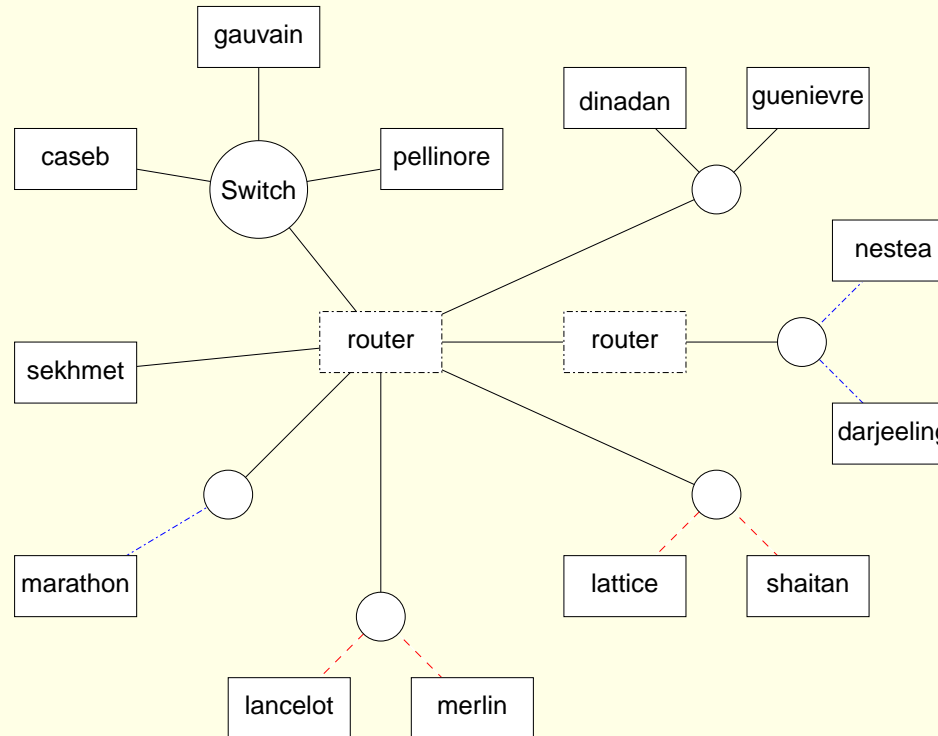
Processor cycle-times (in seconds per megaflop) for the Lyon platform

Platform description (Lyon)



Abstraction of the Lyon platform.

Platform description (Strasbourg)

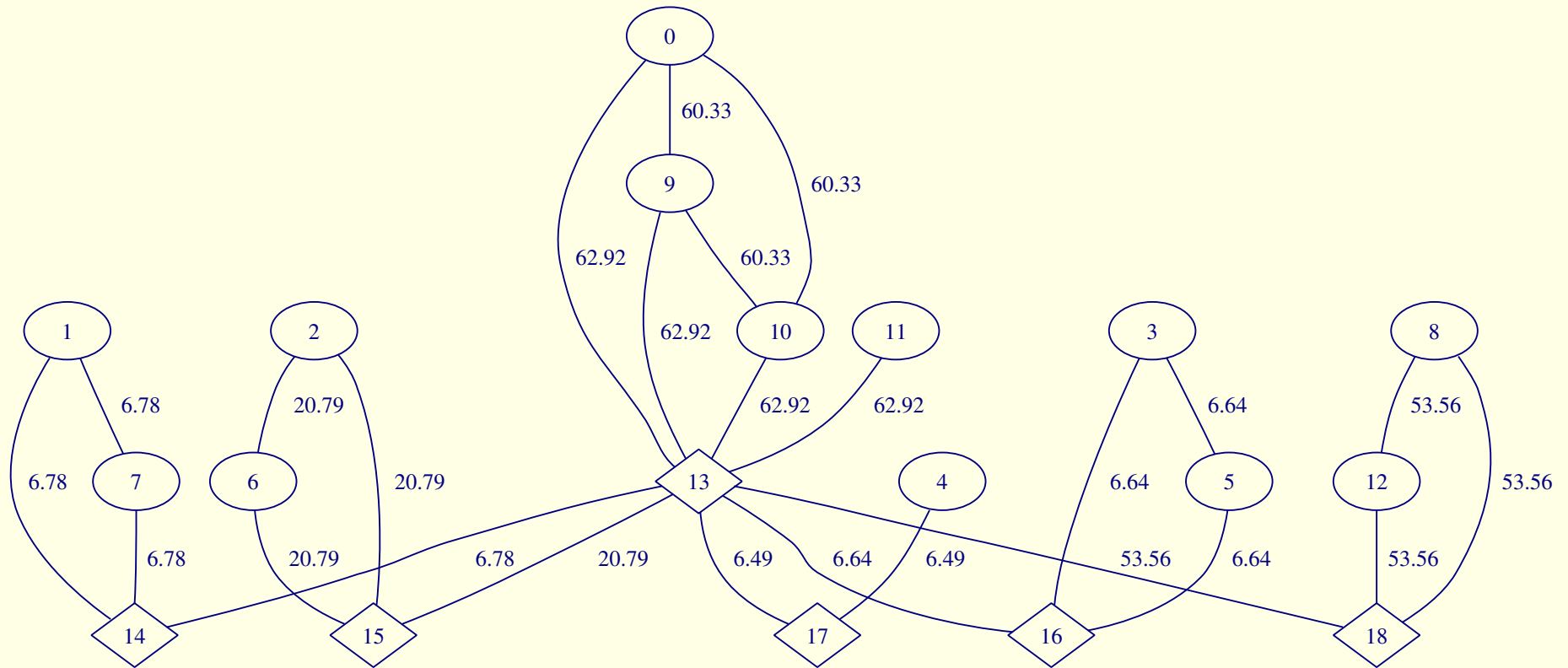


Topology of the Strasbourg platform

P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
0.0087	0.0072	0.0087	0.0131	0.016	0.0058	0.0087	0.0262	0.0102	0.0131
P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}	P_{16}	P_{17}	P_{18}	
0.0072	0.0058	0.0072	0	0	0	0	0	0	

Processor cycle-times (in seconds per megaflop) for the Strasbourg platform.

Platform description (Strasbourg)



Abstraction of the Strasbourg platform.

Results (1/2)

First heuristic slice-ring modified to be feasible

Second heuristic greedy SHARED_RING, quadratic programming refinement

H/W ratio	H1 : slice-ring	H2 : shared-ring	Improvement
0.1	3.17	3.15	0.63%
1	3.46	3.22	6.94%
10	10.39	3.9	62.46%

T_{step}/W for each heuristic on the Lyon platform

H/W ratio	H1 : slice-ring	H2 : shared-ring	Improvement
0.1	7.32	7.26	0.82%
1	9.65	7.53	21.97%
10	19.24	10.26	46.67%

T_{step}/W for each heuristic on the Strasbourg platform

Results (2/2)

- 1 When the impact of communication costs is low :
 - main goal = balance computations
 - both heuristics are equivalent
- 2 When the communication-to-computation ratio becomes more important :
 - effect of link contention becomes clear
 - solution returned by second heuristic much better
- 3 😊 An accurate modeling of communications has a dramatic impact on the performance of the load-balancing strategies

Results (2/2)

- 1 When the impact of communication costs is low :
 - main goal = balance computations
 - both heuristics are equivalent
- 2 When the communication-to-computation ratio becomes more important :
 - effect of link contention becomes clear
 - solution returned by second heuristic much better
- 3 😊 An accurate modeling of communications has a dramatic impact on the performance of the load-balancing strategies

Results (2/2)

- 1 When the impact of communication costs is low :
 - main goal = balance computations
 - both heuristics are equivalent
- 2 When the communication-to-computation ratio becomes more important :
 - effect of link contention becomes clear
 - solution returned by second heuristic much better
- 3 😊 An accurate modeling of communications has a dramatic impact on the performance of the load-balancing strategies

Related work

Related work

- Load balancing strategies for homogeneous / heterogeneous clusters
 - Vast majority : dynamic strategies
 - “use the past to predict the future”
 - data exchanged only between neighbor processors
 - difficult in the context of library oriented approach
 - Static strategies = less general
 - Matrix multiplication (LU and QR)
 - Static partitioning schemes to map 2D-matrices
- ⇒ Distributing independent chunks of work :
- linear arrays of heterogeneous processors : **easy**
 - two-dimensional processor grids : **difficult**
- ⇒ Relaxing the geometrical constraints induced by 2D grids ?
- irregular partitionings
 - good load-balancing but much more difficult to implement

Related work

- Load balancing strategies for homogeneous / heterogeneous clusters
 - Vast majority : dynamic strategies
 - “use the past to predict the future”
 - data exchanged only between neighbor processors
 - difficult in the context of library oriented approach
 - Static strategies = less general
 - Matrix multiplication (LU and QR)
 - Static partitioning schemes to map 2D-matrices
- ⇒ Distributing independent chunks of work :
- linear arrays of heterogeneous processors : **easy**
 - two-dimensional processor grids : **difficult**
- ⇒ Relaxing the geometrical constraints induced by 2D grids ?
- irregular partitionings
 - good load-balancing but much more difficult to implement

Related work

- Load balancing strategies for homogeneous / heterogeneous clusters
 - Vast majority : dynamic strategies
 - “use the past to predict the future”
 - data exchanged only between neighbor processors
 - difficult in the context of library oriented approach
 - Static strategies = less general
 - Matrix multiplication (LU and QR)
 - Static partitioning schemes to map 2D-matrices
- ⇒ Distributing independent chunks of work :
- linear arrays of heterogeneous processors : **easy**
 - two-dimensional processor grids : **difficult**
- ⇒ Relaxing the geometrical constraints induced by 2D grids ?
- irregular partitionings
 - good load-balancing but much more difficult to implement

Related work

- Load balancing strategies for homogeneous / heterogeneous clusters
 - Vast majority : dynamic strategies
 - “use the past to predict the future”
 - data exchanged only between neighbor processors
 - difficult in the context of library oriented approach
 - Static strategies = less general
 - Matrix multiplication (LU and QR)
 - Static partitioning schemes to map 2D-matrices
- ⇒ Distributing independent chunks of work :
- linear arrays of heterogeneous processors : **easy**
 - two-dimensional processor grids : **difficult**
- ⇒ Relaxing the geometrical constraints induced by 2D grids ?
- irregular partitionings
 - good load-balancing but much more difficult to implement

Related work

- Load balancing strategies for homogeneous / heterogeneous clusters
 - Vast majority : dynamic strategies
 - “use the past to predict the future”
 - data exchanged only between neighbor processors
 - difficult in the context of library oriented approach
 - Static strategies = less general
 - Matrix multiplication (LU and QR)
 - Static partitioning schemes to map 2D-matrices
- ⇒ Distributing independent chunks of work :
- linear arrays of heterogeneous processors : **easy**
 - two-dimensional processor grids : **difficult**
- ⇒ Relaxing the geometrical constraints induced by 2D grids ?
- irregular partitionings
 - good load-balancing but much more difficult to implement

Conclusion

- Difficulties with heterogeneous platforms : 😞
 - additional difficulty of balancing the load
 - data and computations not evenly distributed to processors
 - minimizing communication overhead = challenging task
- Main results : 😊
 - NP-completeness of the SHARED_RING problem
 - ⇒ provides evidence of the intrinsic difficulty of designing heterogeneous algorithms
 - design of efficient heuristics, that provide a pragmatic guidance to the designer of iterative scientific computations
- Future work : 😞 😊
 - more accurate platform/network models
 - adaption to changes in resource capacities
 - ⇒ still worthy to inject some static knowledge into dynamic schedulers ?

Max-min fairness

Goal : maximize $\min_i d_i$

Motivation

Each path receives the same bandwidth through bottleneck links

$$\forall p_i \in \mathcal{P}, \quad \exists e \in p_i, \quad \sum_{p_j \ni e} d_j = u(e) \text{ and } d_i = \max\{d_j, p_j \ni e\}$$

