Multi-criteria Mapping and Scheduling of Workflow Applications onto Heterogeneous Platforms

Veronika Sonigo

GRAAL team, LIP
École Normale Supérieure de Lyon

Supervisors:
Anne Benoit, Harald Kosch, Yves Robert

Multi-criteria Mapping and Scheduling

Mapping and scheduling for parallel machines

Makespan minimization $\longrightarrow$ already a difficult problem

Mapping and scheduling for large-scale heterogeneous platforms

- Need new objectives
- Period, reliability, latency, cost, QoS, energy, etc
- Multi-criteria optimization
- Assess problem complexity (polynomial /NP-hard instances)
- Design practical heuristics for important application problems

New results and solutions for algorithmically challenging problems

Thesis outline - Today's Presentation

Replica Placement in Tree-Networks

- Without Constraints
- QoS Constraints
- Bandwidth Constraints

Pipeline Workflow Applications

- Mono-criterion Optimization
- Bi-criteria Optimization
- Example: JPEG-Encoder

In-network Stream Processing

- Single Application - Platform Creation
- Multiple Applications

Thesis outline - Today's Presentation

Replica Placement in Tree-Networks

- Without Constraints
- QoS Constraints
- Bandwidth Constraints

Pipeline Workflow Applications

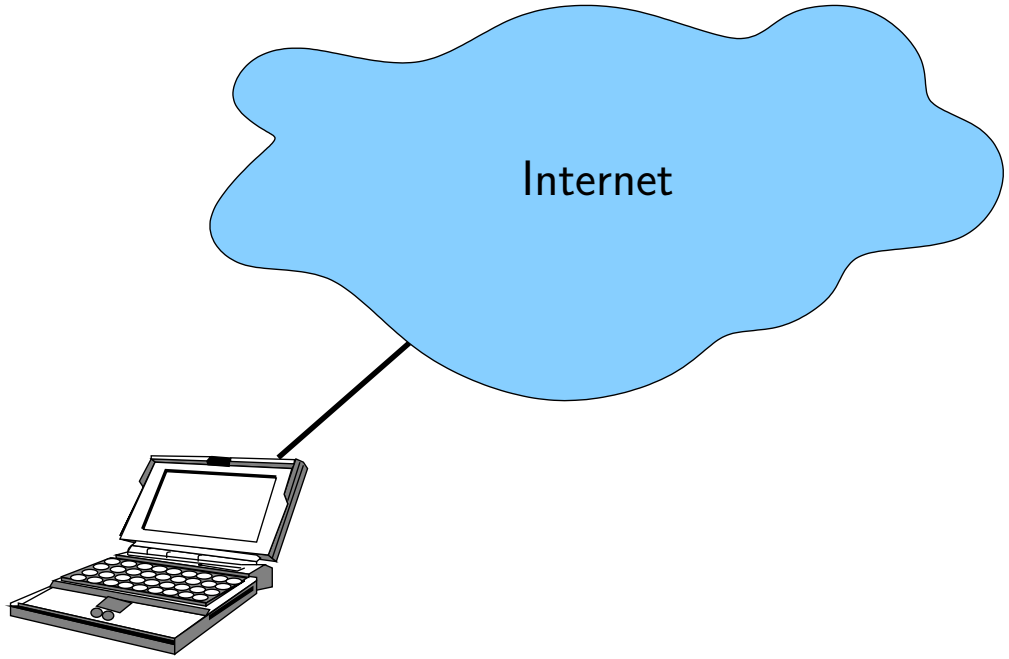
- Mono-criterion Optimization
- Bi-criteria Optimization
- Example: JPEG-Encoder

In-network Stream Processing

- Single Application - Platform Creation
- Multiple Applications

Outline of the Talk

Mimi alone at home

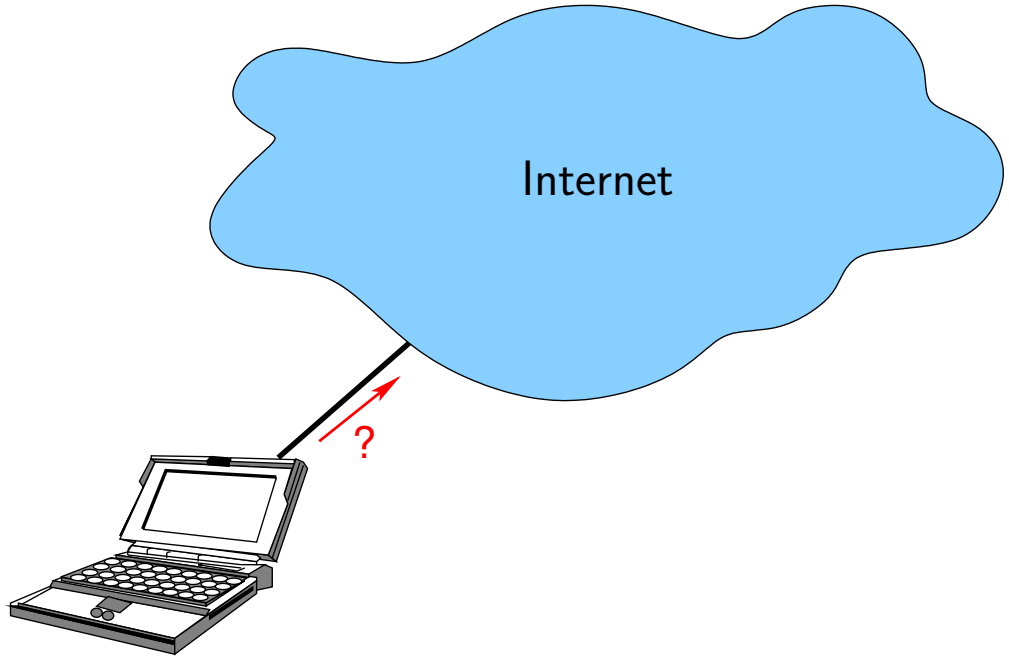


Problems and questions:

- Where to download from?
- How to deal with multiple users?
- Heterogeneity

Where to place the replicas?

Mimi alone at home

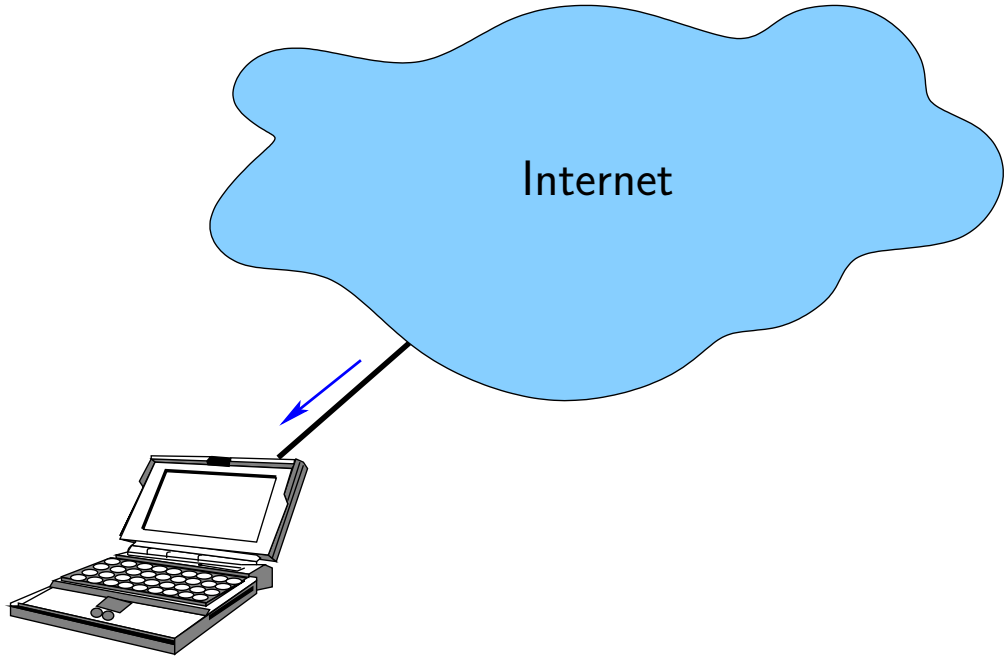


Problems and questions:

- Where to download from?
- How to deal with multiple users?
- Heterogeneity

Where to place the replicas?

Mimi alone at home

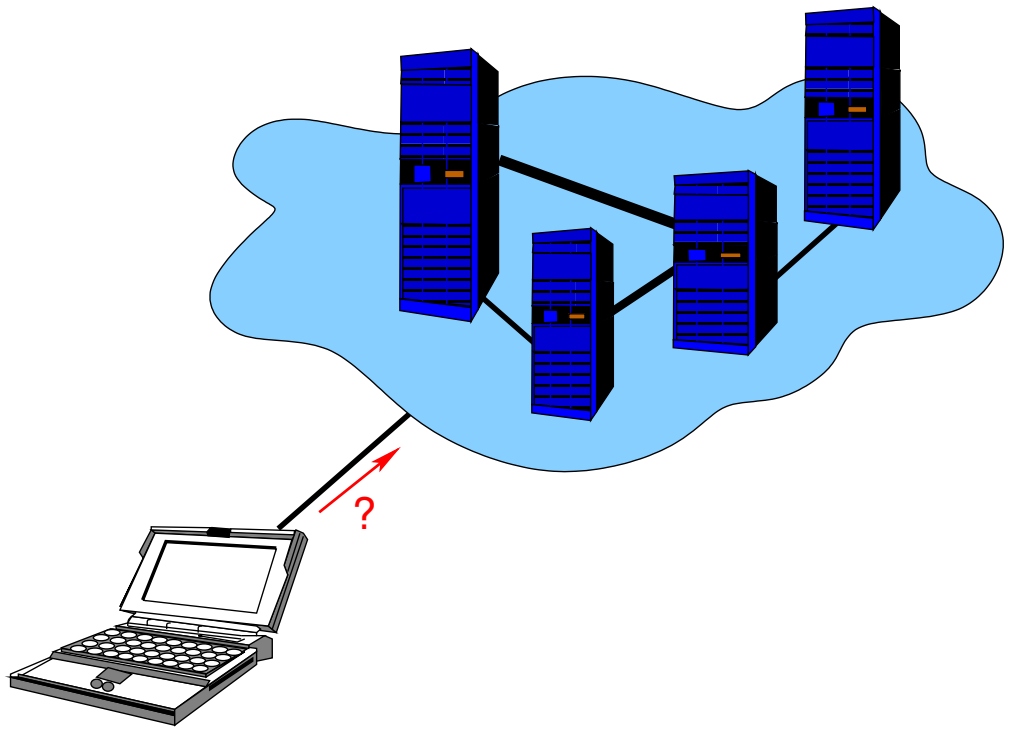


Problems and questions:

- Where to download from?
- How to deal with multiple users?
- Heterogeneity

Where to place the replicas?

Mimi alone at home

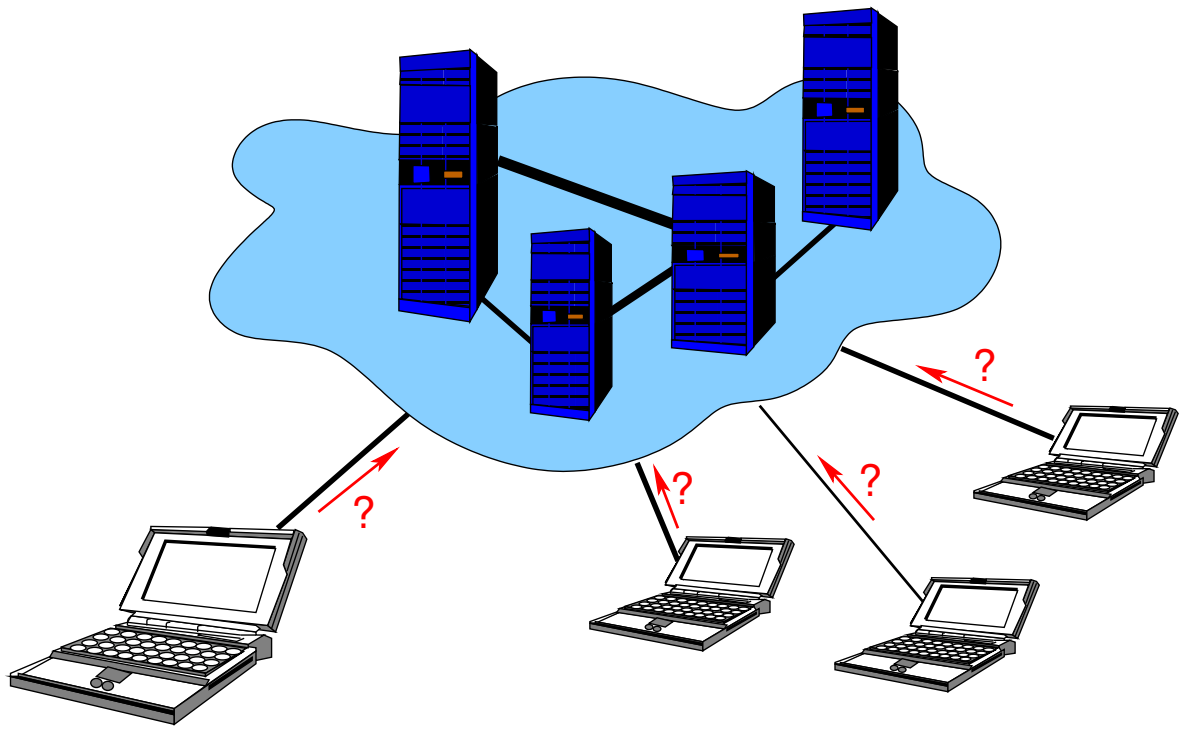


Problems and questions:

- Where to download from?
- How to deal with multiple users?
- Heterogeneity

Where to place the replicas?

Mimi alone at home

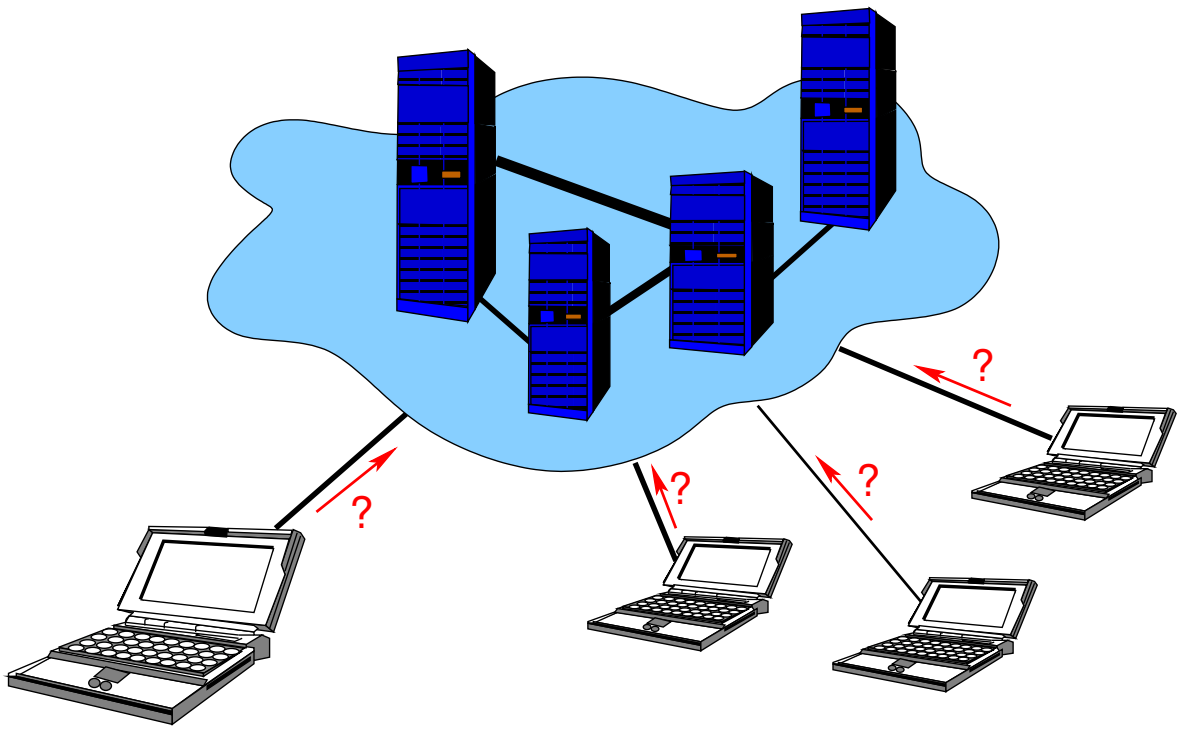


Problems and questions:

- Where to download from?
- How to deal with multiple users?
- Heterogeneity

Where to place the replicas?

Mimi alone at home

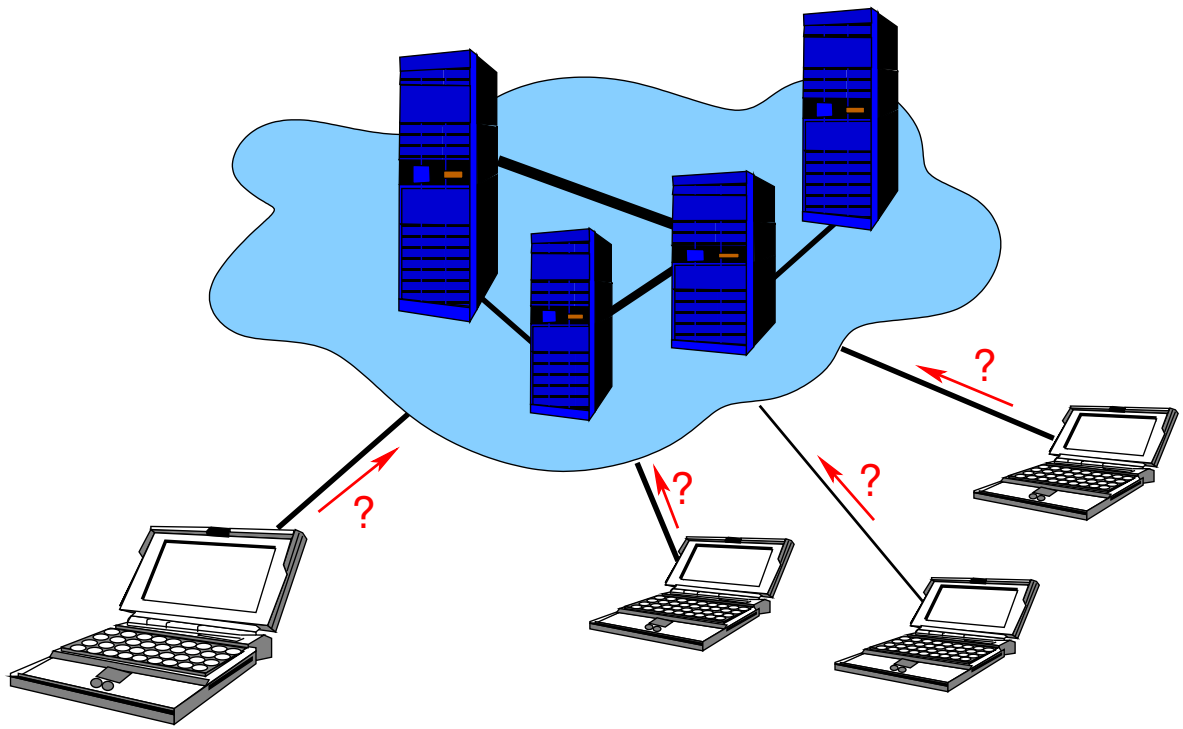


Problems and questions:

- Where to download from?
- How to deal with multiple users?
- Heterogeneity

Where to place the replicas?

Mimi alone at home



Problems and questions:

- Where to download from?
- How to deal with multiple users?
- Heterogeneity

Where to place the replicas?

Introduction and motivation

- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS or bandwidth constraints, known in advance
- Internal nodes may be provided with a replica;
  in this case they become servers
  and process requests (up to their capacity limit)

Research questions:

How many replicas required? Total replica cost?
Which locations? Quality of Service?

Introduction and motivation

- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS or bandwidth constraints, known in advance
- Internal nodes may be provided with a replica;
  in this case they become servers
  and process requests (up to their capacity limit)

Research questions:

How many replicas required? Total replica cost?
Which locations? Quality of Service?

Introduction and motivation

- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS or bandwidth constraints, known in advance
- Internal nodes may be provided with a replica;
  in this case they become servers
  and process requests (up to their capacity limit)

Research questions:

How many replicas required? Total replica cost?
Which locations? Quality of Service?

Introduction and motivation

- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS or bandwidth constraints, known in advance
- Internal nodes may be provided with a replica;
  in this case they become servers
  and process requests (up to their capacity limit)

Research questions:

How many replicas required? Total replica cost?
Which locations? Quality of Service?

Introduction and motivation

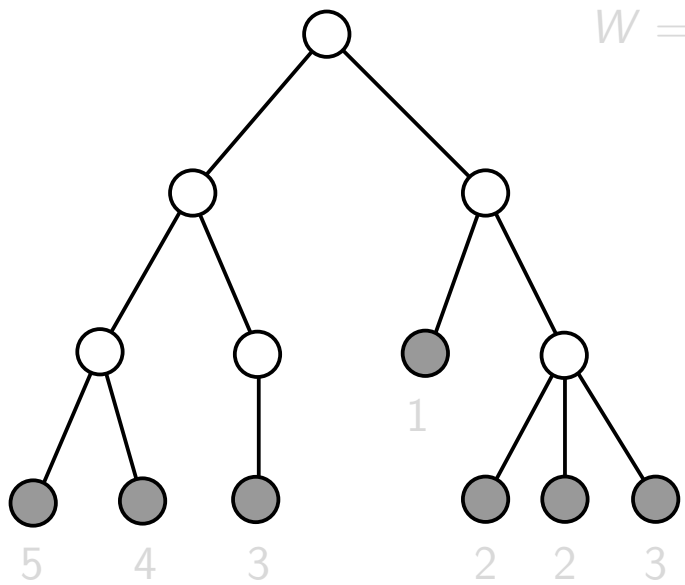
- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS or bandwidth constraints, known in advance
- Internal nodes may be provided with a replica;
  in this case they become servers
  and process requests (up to their capacity limit)

Research questions:

How many replicas required? Total replica cost?
Which locations? Quality of Service?

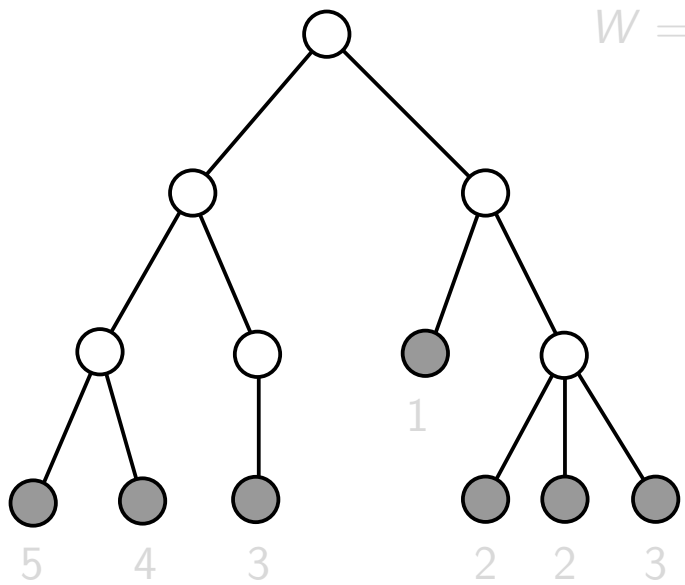
Rule of the game

- Handle all client requests, and minimize cost of replicas
- → Replica Placement problem
- Several policies to assign replicas



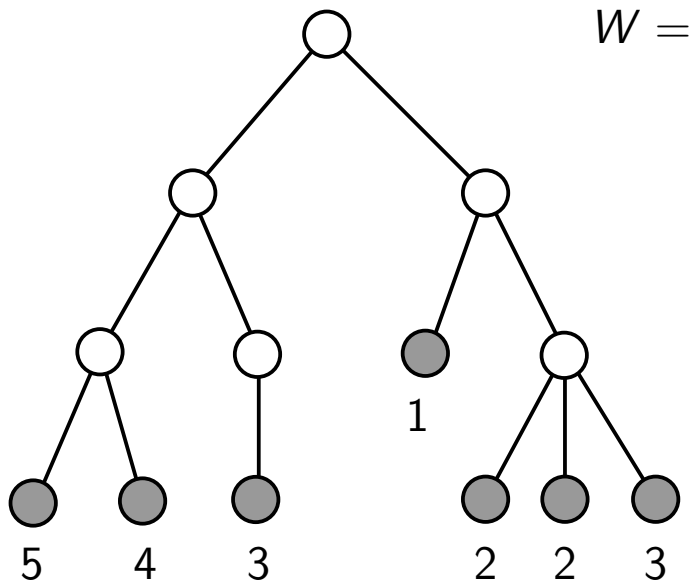
Rule of the game

- Handle all client requests, and minimize cost of replicas
- $\rightarrow$ Replica Placement problem
- Several policies to assign replicas



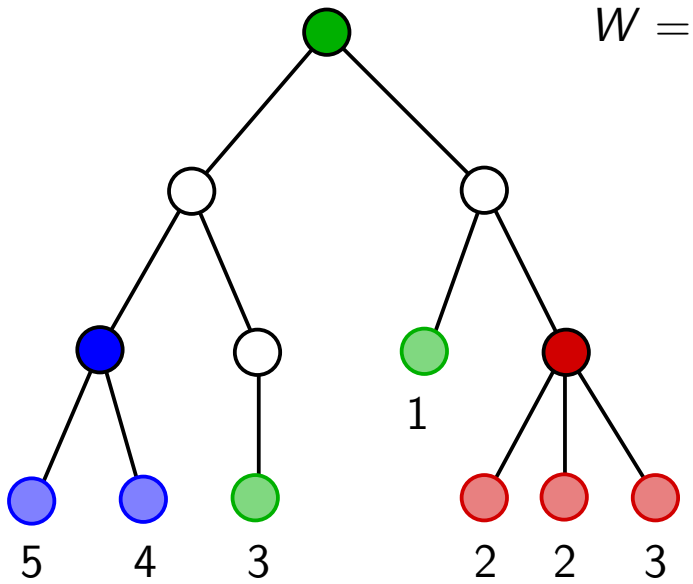
Rule of the game

- Handle all client requests, and minimize cost of replicas
- $\rightarrow$ Replica Placement problem
- Several policies to assign replicas



Rule of the game

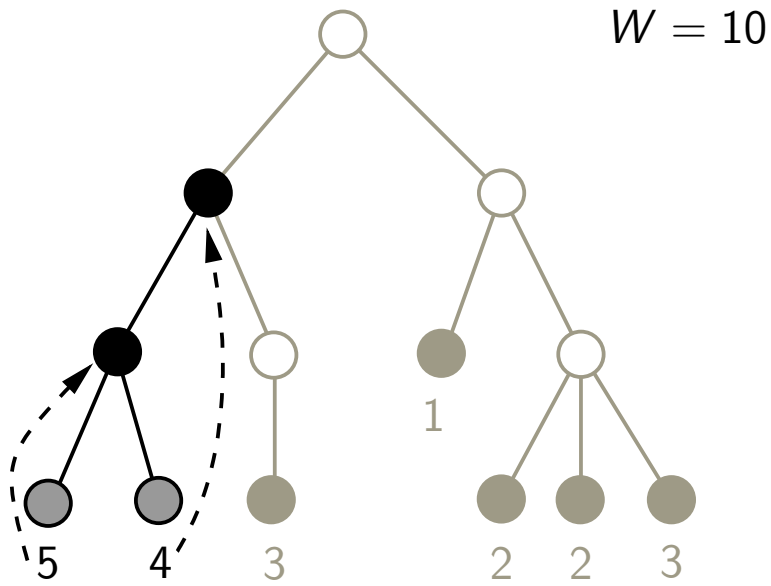
- Handle all client requests, and minimize cost of replicas
- $\rightarrow$ Replica Placement problem
- Several policies to assign replicas



Closest

Rule of the game

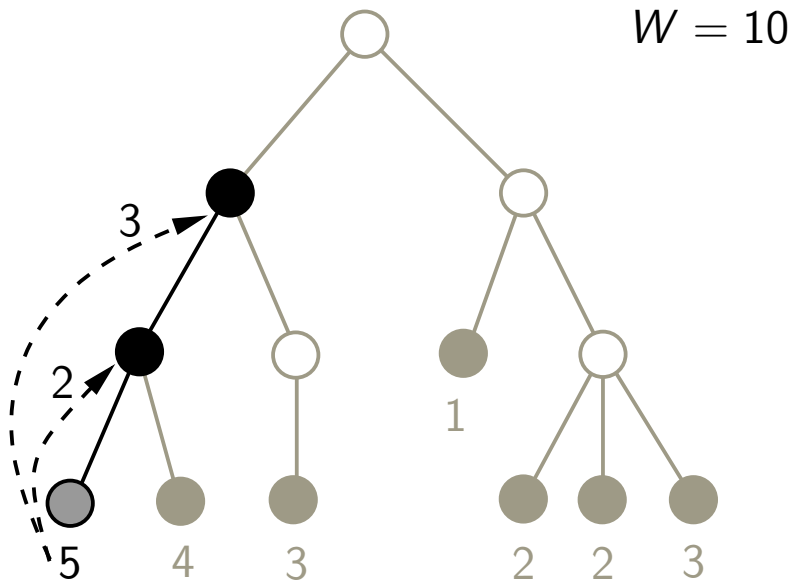
- Handle all client requests, and minimize cost of replicas
- $\rightarrow$ Replica Placement problem
- Several policies to assign replicas



Upwards

Rule of the game

- Handle all client requests, and minimize cost of replicas
- $\rightarrow$ Replica Placement problem
- Several policies to assign replicas



Multiple

Major contributions

Theory
- New access policies
- Problem complexity
- LP-based optimal solution to cost of Replica Placement

Practice
- Heuristics for each policy
- Experiments to assess impact of new policies
- Experiments to assess impact of QoS on different policies

Major contributions

Theory
- New access policies
- Problem complexity
- LP-based optimal solution to cost of Replica Placement

Practice
- Heuristics for each policy
- Experiments to assess impact of new policies
- Experiments to assess impact of QoS on different policies

Definitions and notations

- Distribution tree $\mathcal{T}$, clients $\mathcal{C}$ (leaf nodes), internal nodes $\mathcal{N}$
- **Client** $i \in \mathcal{C}$:
  - Sends r_i requests per time unit (number of accesses to a single object database)
  - Quality of service q_i (response time)
- **Node** $j \in \mathcal{N}$:
  - Can contain the object database replica (server) or not
  - Processing capacity W_j
  - Storage cost sc_j
- **Tree edge:** $l \in \mathcal{L}$ (communication link between nodes)
  - Communication time $comm_l$
  - Bandwidth limit BW_l

Definitions and notations

- Distribution tree $\mathcal{T}$, clients $\mathcal{C}$ (leaf nodes), internal nodes $\mathcal{N}$
- **Client** $i \in \mathcal{C}$:
  - Sends r_i requests per time unit (number of accesses to a single object database)
  - Quality of service q_i (response time)
- **Node** $j \in \mathcal{N}$:
  - Can contain the object database replica (server) or not
  - Processing capacity W_j
  - Storage cost sc_j
- **Tree edge:** $l \in \mathcal{L}$ (communication link between nodes)
  - Communication time $comm_l$
  - Bandwidth limit BW_l

Definitions and notations

- Distribution tree $\mathcal{T}$, clients $\mathcal{C}$ (leaf nodes), internal nodes $\mathcal{N}$
- **Client** $i \in \mathcal{C}$:
  - Sends r_i requests per time unit (number of accesses to a single object database)
  - Quality of service q_i (response time)
- **Node** $j \in \mathcal{N}$:
  - Can contain the object database replica (server) or not
  - Processing capacity W_j
  - Storage cost sc_j
- **Tree edge:** $l \in \mathcal{L}$ (communication link between nodes)
  - Communication time $comm_l$
  - Bandwidth limit BW_l

Definitions and notations

- Distribution tree $\mathcal{T}$, clients $\mathcal{C}$ (leaf nodes), internal nodes $\mathcal{N}$
- **Client** $i \in \mathcal{C}$:
  - Sends r_i requests per time unit (number of accesses to a single object database)
  - Quality of service q_i (response time)
- **Node** $j \in \mathcal{N}$:
  - Can contain the object database replica (server) or not
  - Processing capacity W_j
  - Storage cost sc_j
- **Tree edge:** $l \in \mathcal{L}$ (communication link between nodes)
  - Communication time $comm_l$
  - Bandwidth limit BW_l

Problem instances

- Minimize $\sum_{s \in R} sc_s$ under the constraints:
- **Server capacity** – $\forall s \in R, \sum_{i \in \mathcal{C} | s \in \text{Servers}(i)} r_{i,s} \leq W_s$
- **QoS** – $\forall i \in \mathcal{C}, \forall s \in \text{Servers}(i), \sum_{l \in \text{path}[i \to s]} \text{comm}_l \leq q_i$.
- **Link capacity** – $\forall l \in \mathcal{L} \sum_{i \in \mathcal{C}, s \in \text{Servers}(i) | l \in \text{path}[i \to s]} r_{i,s} \leq BW_l$
- Restrict to case where $sc_s = W_s$:
  Replica Counting problem on homogeneous platforms,
  Replica Cost problem with heterogeneous servers.

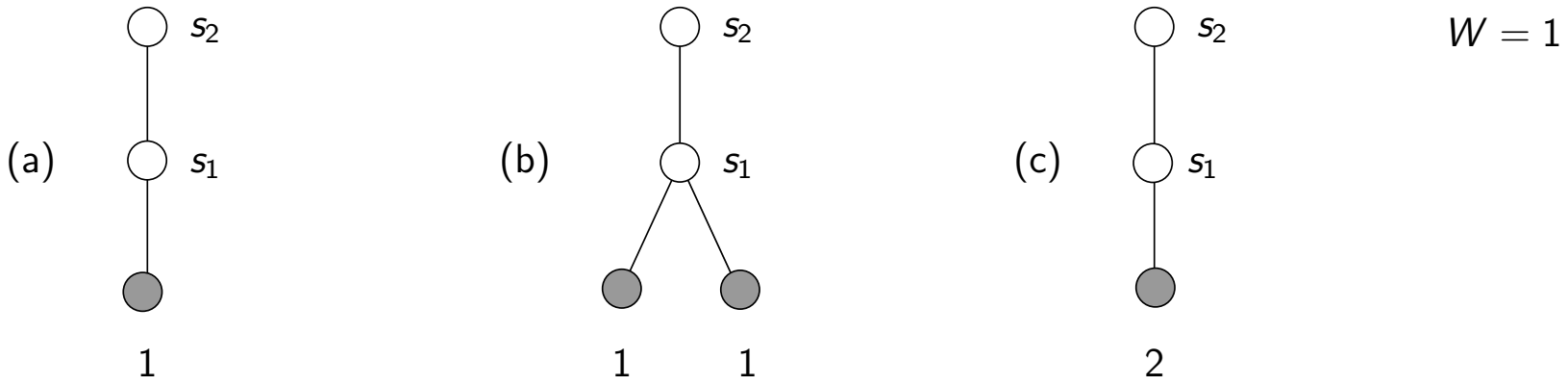
Problem instances

- Minimize $\sum_{s \in R} sc_s$ under the constraints:
- **Server capacity** – $\forall s \in R, \sum_{i \in \mathcal{C} | s \in \text{Servers}(i)} r_{i,s} \leq W_s$
- **QoS** – $\forall i \in \mathcal{C}, \forall s \in \text{Servers}(i), \sum_{l \in \text{path}[i \rightarrow s]} \text{comm}_l \leq q_i$.
- **Link capacity** – $\forall l \in \mathcal{L} \sum_{i \in \mathcal{C}, s \in \text{Servers}(i) | l \in \text{path}[i \rightarrow s]} r_{i,s} \leq BW_l$
- Restrict to case where $sc_s = W_s$:
  Replica Counting problem on homogeneous platforms,
  Replica Cost problem with heterogeneous servers.

Problem instances

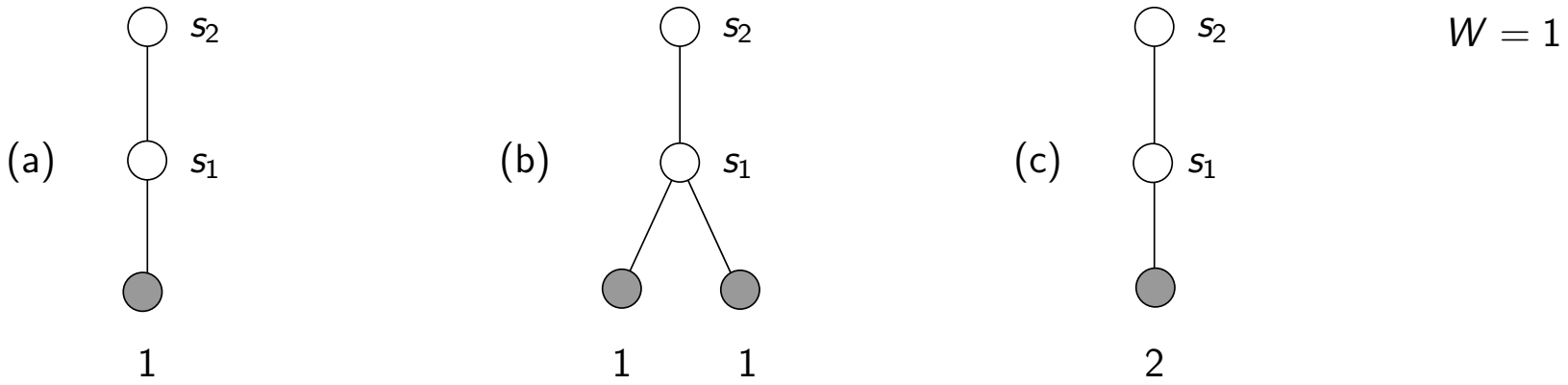
- Minimize $\sum_{s \in R} \text{sc}_s$ under the constraints:
- **Server capacity** – $\forall s \in R, \sum_{i \in \mathcal{C} | s \in \text{Servers}(i)} r_{i,s} \leq W_s$
- **QoS** – $\forall i \in \mathcal{C}, \forall s \in \text{Servers}(i), \sum_{l \in \text{path}[i \rightarrow s]} \text{comm}_l \leq q_i$.
- **Link capacity** – $\forall l \in \mathcal{L} \sum_{i \in \mathcal{C}, s \in \text{Servers}(i) | l \in \text{path}[i \rightarrow s]} r_{i,s} \leq BW_l$
- Restrict to case where $\text{sc}_s = W_s$:
  Replica Counting problem on homogeneous platforms,
  Replica Cost problem with heterogeneous servers.

Example: existence of a solution



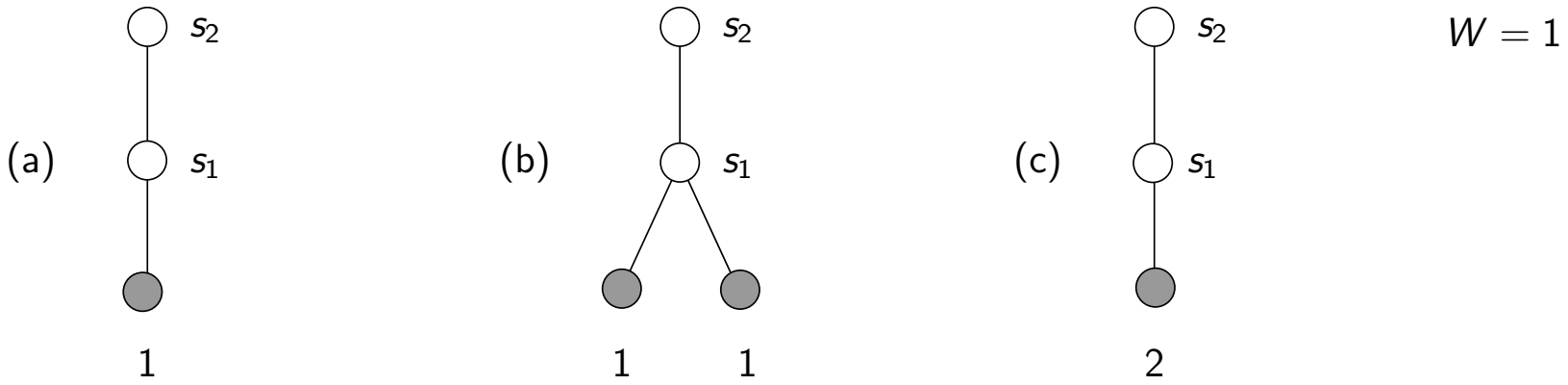
- (a): solution for all policies (*Closest*, *Upwards*, *Multiple*)
- (b): no solution with *Closest*
- (c): no solution with *Closest* nor *Upwards*

Example: existence of a solution



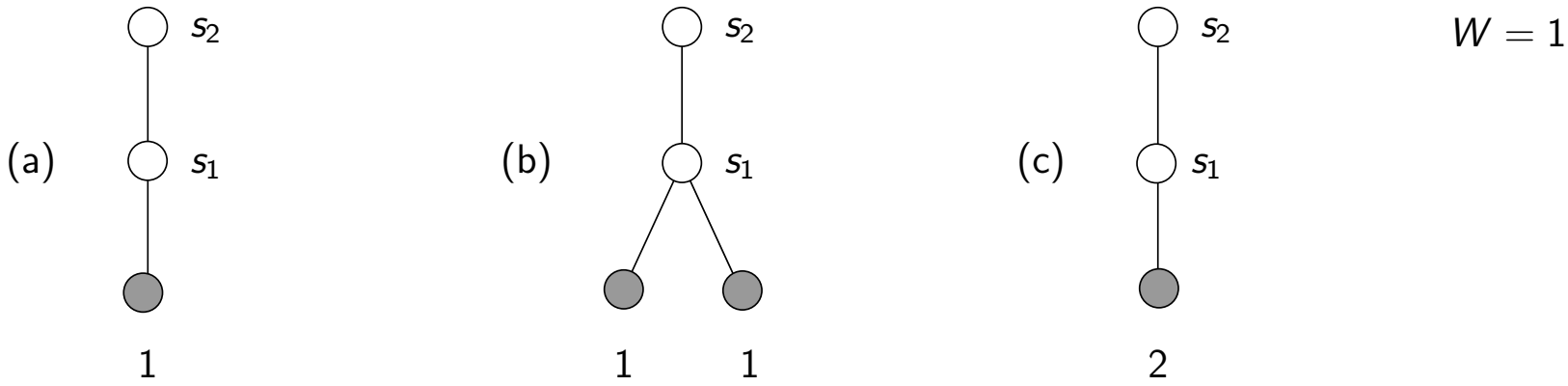
- (a): solution for all policies (*Closest*, *Upwards*, *Multiple*)
- (b): no solution with *Closest*
- (c): no solution with *Closest* nor *Upwards*

Example: existence of a solution



- (a): solution for all policies (*Closest*, *Upwards*, *Multiple*)
- (b): no solution with *Closest*
- (c): no solution with *Closest* nor *Upwards*

Example: existence of a solution



- (a): solution for all policies (*Closest*, *Upwards*, *Multiple*)
- (b): no solution with *Closest*
- (c): no solution with *Closest* nor *Upwards*

Complexity results

Homogeneous platform: Replica Counting problem, no bandwidth constraints

	No QoS	**With QoS**
Closest	polynomial [Cidon02,Liu06]	polynomial [Liu06]
Upwards	NP-hard	NP-hard
Multiple	polynomial	NP-hard

Homogeneous platforms with bandwidth and QoS constraints: *Closest* remains polynomial

Heterogeneous platforms: all problems are NP-hard

Complexity results

Homogeneous platform: Replica Counting problem, no bandwidth constraints

	No QoS	With QoS
Closest	polynomial [Cidon02,Liu06]	polynomial [Liu06]
Upwards	NP-hard	NP-hard
Multiple	polynomial	NP-hard

Homogeneous platforms with bandwidth and QoS constraints: *Closest* remains polynomial

Heterogeneous platforms: all problems are NP-hard

Complexity results

Homogeneous platform: Replica Counting problem, no bandwidth constraints

	No QoS	**With QoS**
Closest	polynomial [Cidon02,Liu06]	polynomial [Liu06]
Upwards	NP-hard	NP-hard
Multiple	polynomial	NP-hard

Homogeneous platforms with bandwidth and QoS constraints: *Closest* remains polynomial

Heterogeneous platforms: all problems are NP-hard

Complexity results

Homogeneous platform: Replica Counting problem, no bandwidth constraints

	No QoS	**With QoS**
Closest	polynomial [Cidon02,Liu06]	polynomial [Liu06]
Upwards	NP-hard	NP-hard
Multiple	polynomial	NP-hard

Homogeneous platforms with bandwidth and QoS constraints: *Closest* remains polynomial

Heterogeneous platforms: all problems are NP-hard

Algo: Homogeneous Platform with QoS and Bandwidth

Basic idea:

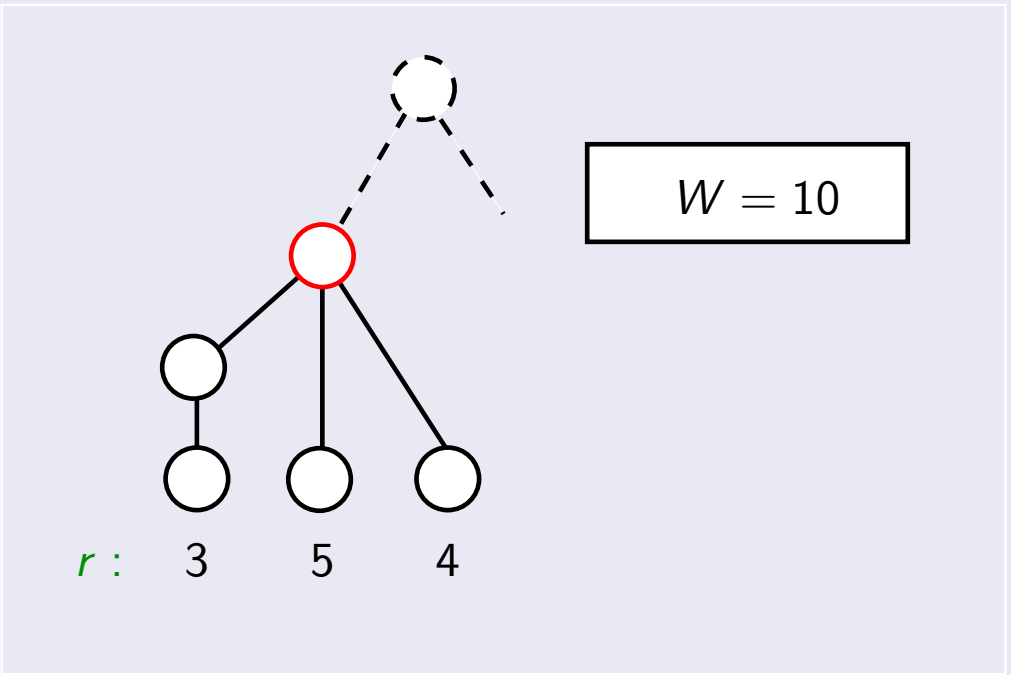
computation of the minimal necessary number of replicas in a subtree

Algo: Homogeneous Platform with QoS and Bandwidth

Basic idea:

computation of the minimal necessary number of replicas in a subtree

Case 1: too many requests

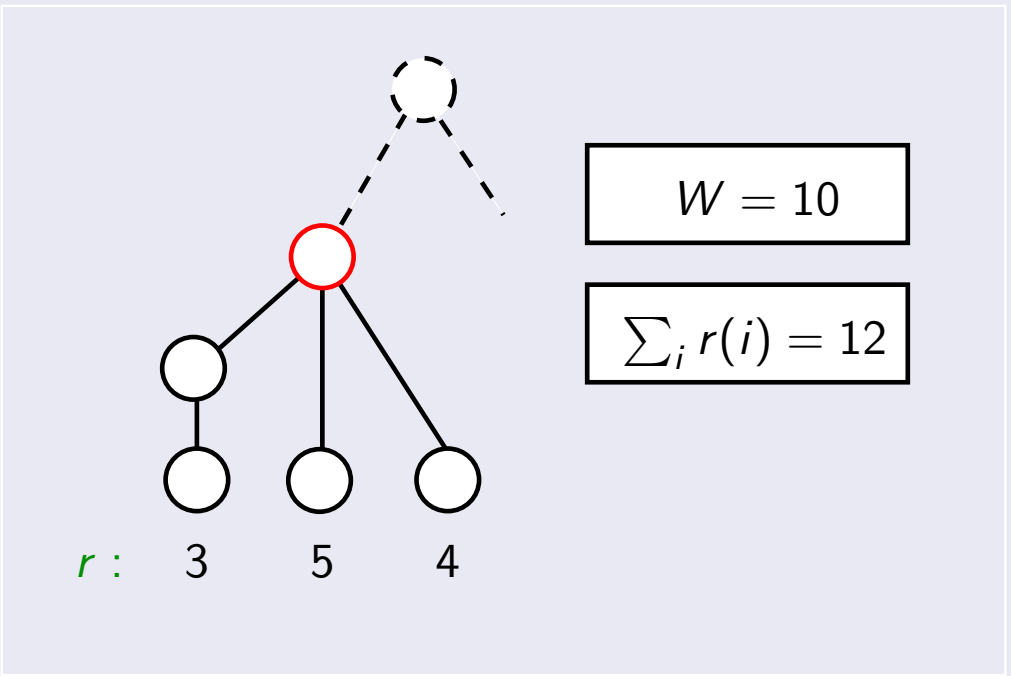


Algo: Homogeneous Platform with QoS and Bandwidth

Basic idea:

computation of the minimal necessary number of replicas in a subtree

Case 1: too many requests

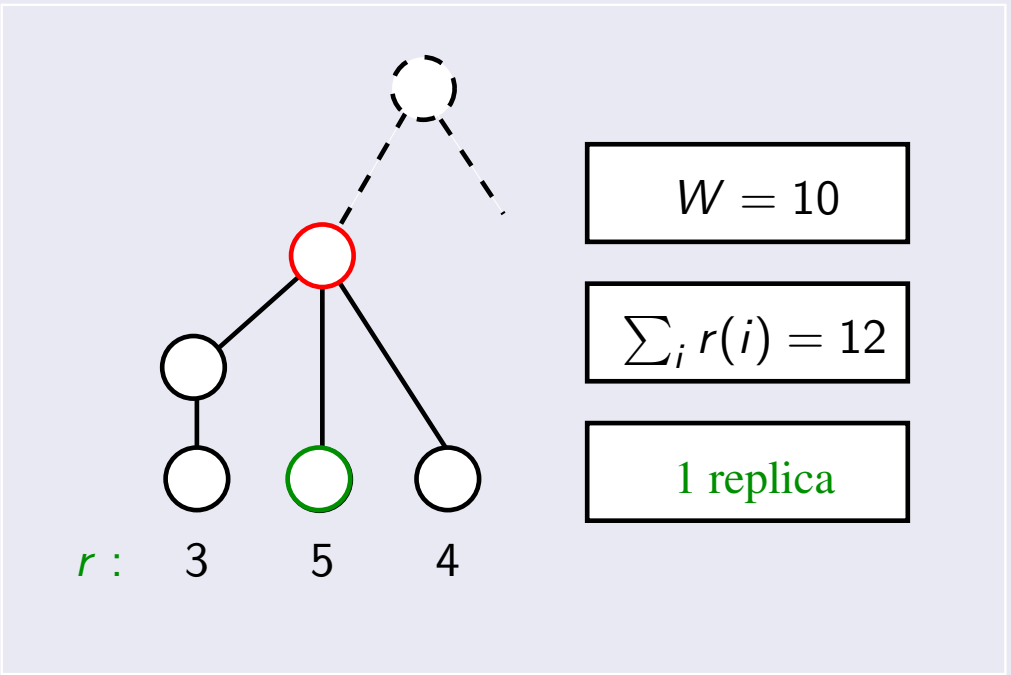


Algo: Homogeneous Platform with QoS and Bandwidth

Basic idea:

computation of the minimal necessary number of replicas in a subtree

Case 1: too many requests

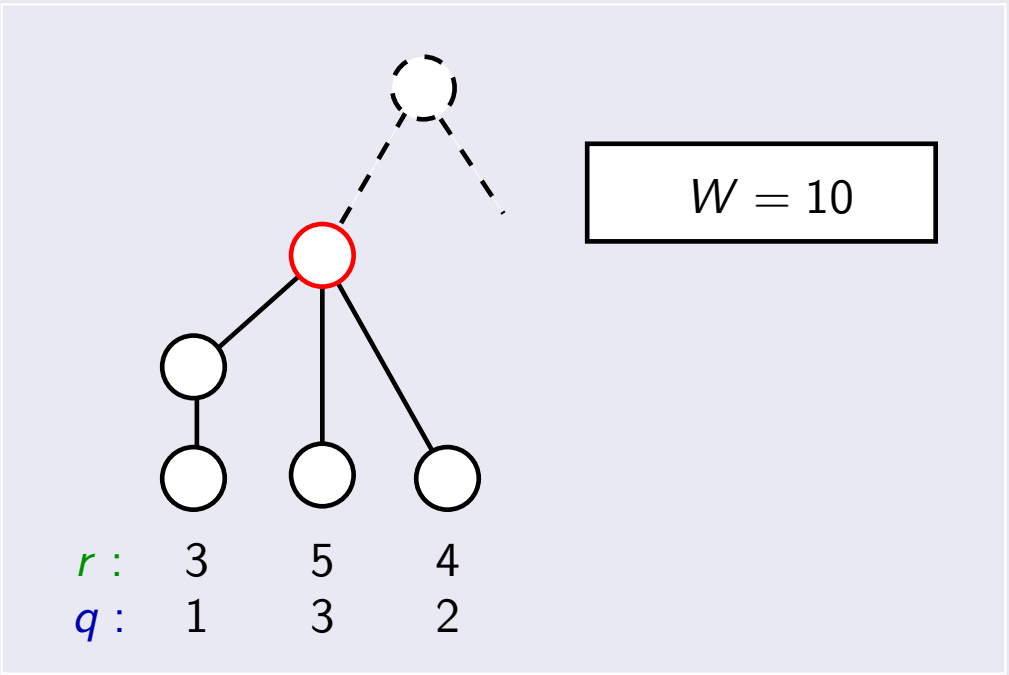


Algo: Homogeneous Platform with QoS and Bandwidth

Basic idea:

computation of the minimal necessary number of replicas in a subtree

Case 2: QoS constraints

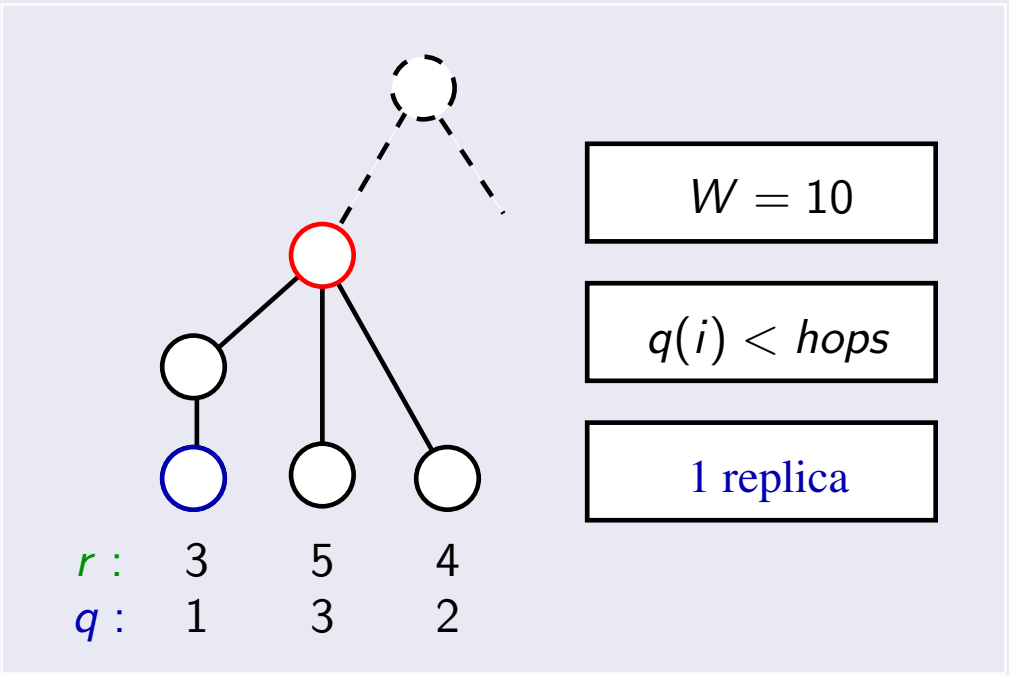


Algo: Homogeneous Platform with QoS and Bandwidth

Basic idea:

computation of the minimal necessary number of replicas in a subtree

Case 2: QoS constraints

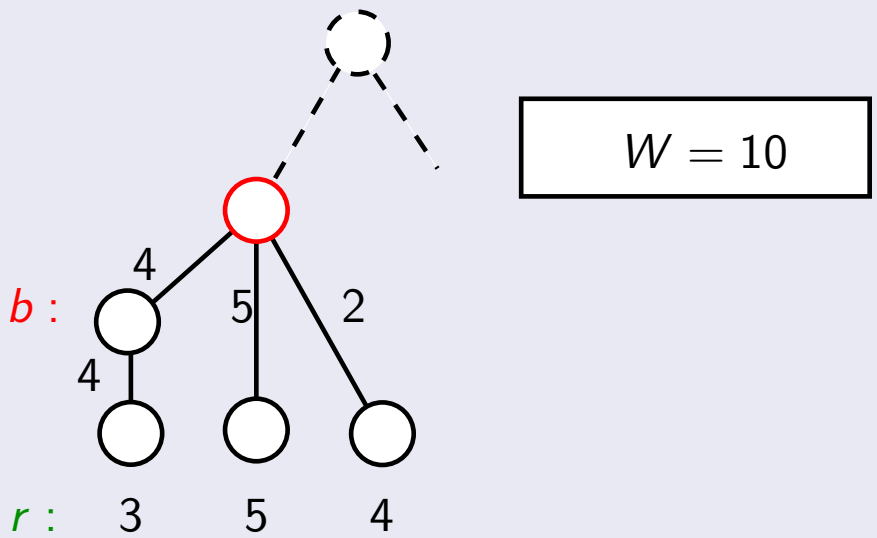


Algo: Homogeneous Platform with QoS and Bandwidth

Basic idea:

computation of the minimal necessary number of replicas in a subtree

Case 3: bandwidth constraints

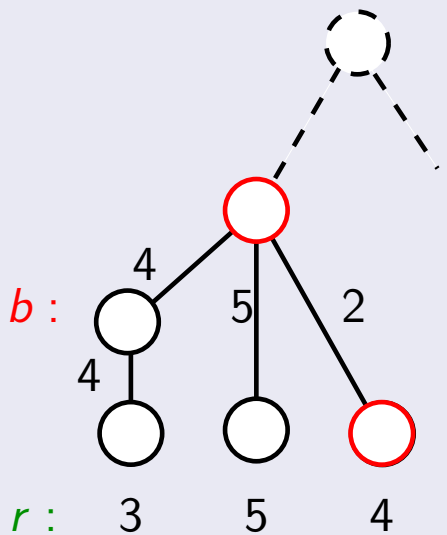


Algo: Homogeneous Platform with QoS and Bandwidth

Basic idea:

computation of the minimal necessary number of replicas in a subtree

Case 3: bandwidth constraints



$W = 10$

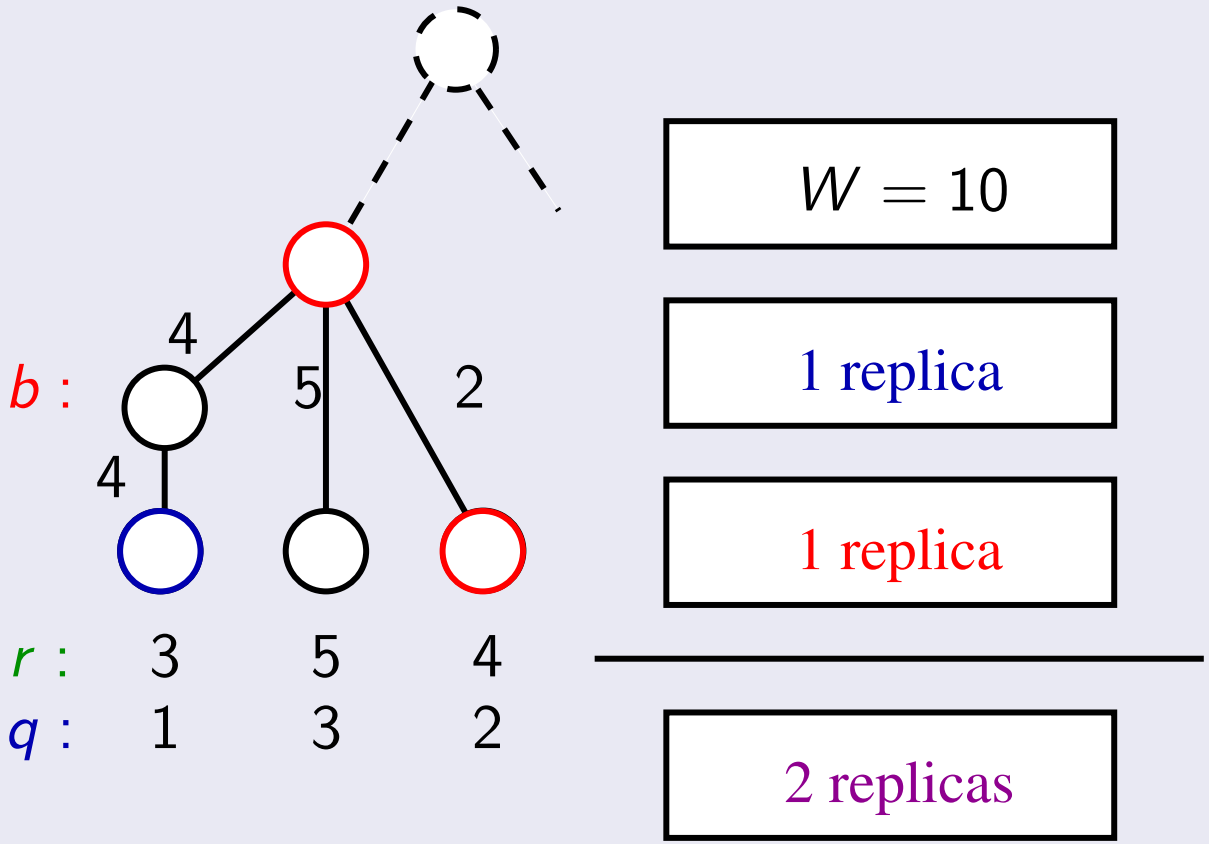
$b(l) < r(i)$

1 replica

Algo: Homogeneous Platform with QoS and Bandwidth

Basic idea:

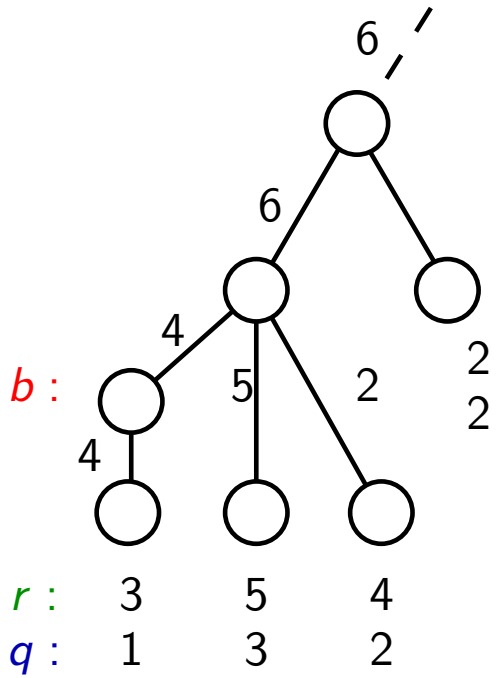
computation of the minimal necessary number of replicas in a subtree



ORP - Optimal Replica Placement Algorithm

Preparation Tree transformation

Step 1 Bottom up computation of the contribution of client requests



$C(v, i)$: the contribution of node v on its i-th ancestor

$e(v, i)$: children of v that have to be equipped with a replica to minimize the contribution on the i-th ancestor of v (respecting some additional constraints).

ORP - Optimal Replica Placement Algorithm

Preparation Tree transformation

Step 1 Bottom up computation of the contribution of client requests

Step 2 Top down replica placement

procedure **Place-replica** (v, i)

```
if v ∈ C then
  return;
end
place a replica at each node of e(v, i);
forall c ∈ children(v) do
  if c ∈ e(v, i) then
    Place-replica(c,0);
  else
    Place-replica(c,i+1);
  end
end
```

Linear programming

- General instance of the problem: Heterogeneous platform, QoS+bandwidth, *Closest*, *Upwards* and *Multiple* policies
- Solving over the rationals: solution for all practical values of the problem size
  - Not very precise bound
  - *Upwards*/*Closest* equivalent to *Multiple*
- Integer solving: limitation to $s \leq 50$ nodes and clients
- Mixed bound obtained by solving the *Upwards* formulation over the rational and imposing only the x_j being integers
  - Resolution for problem sizes $s \leq 400$
  - Improved bound: if a server is used only at 50% of its capacity, the cost of placing a replica at this node is not halved as it would be with $x_j = 0.5 \rightarrow$ optimal solution for *Multiple*

Linear programming

- General instance of the problem: Heterogeneous platform, QoS+bandwidth, *Closest*, *Upwards* and *Multiple* policies
- Solving over the rationals: solution for all practical values of the problem size
  - Not very precise bound
  - *Upwards*/*Closest* equivalent to *Multiple*
- Integer solving: limitation to $s \leq 50$ nodes and clients
- Mixed bound obtained by solving the *Upwards* formulation over the rational and imposing only the x_j being integers
  - Resolution for problem sizes $s \leq 400$
  - Improved bound: if a server is used only at 50% of its capacity, the cost of placing a replica at this node is not halved as it would be with $x_j = 0.5 \rightarrow$ optimal solution for *Multiple*

Linear programming

- General instance of the problem: Heterogeneous platform, QoS+bandwidth, *Closest*, *Upwards* and *Multiple* policies
- Solving over the rationals: solution for all practical values of the problem size
  - Not very precise bound
  - *Upwards*/*Closest* equivalent to *Multiple*
- Integer solving: limitation to $s \leq 50$ nodes and clients
- Mixed bound obtained by solving the *Upwards* formulation over the rational and imposing only the x_j being integers
  - Resolution for problem sizes $s \leq 400$
  - Improved bound: if a server is used only at 50% of its capacity, the cost of placing a replica at this node is not halved as it would be with $x_j = 0.5 \rightarrow$ optimal solution for *Multiple*

Linear programming

- General instance of the problem: Heterogeneous platform, QoS+bandwidth, *Closest*, *Upwards* and *Multiple* policies
- Solving over the rationals: solution for all practical values of the problem size
  - Not very precise bound
  - *Upwards*/*Closest* equivalent to *Multiple*
- Integer solving: limitation to $s \leq 50$ nodes and clients
- Mixed bound obtained by solving the *Upwards* formulation over the rational and imposing only the x_j being integers
  - Resolution for problem sizes $s \leq 400$
  - Improved bound: if a server is used only at 50% of its capacity, the cost of placing a replica at this node is not halved as it would be with $x_j = 0.5 \rightarrow$ optimal solution for *Multiple*

Heuristics

- Polynomial heuristics for REPLICA COST problem
  - Heterogeneous platforms
  - Heuristics with and without QoS
  - QoS constraints: QoS of client i represents the maximum distance (number of hops) between i and server(i)
- Experimental assessment of relative performance of the three policies
- Impact of QoS
- No QoS: Traversals of the tree, bottom-up or top-down
- QoS: Sorted lists
- Worst case complexity $O(s^2)$, where $s = |\mathcal{C}| + |\mathcal{N}|$ is problem size

Heuristics

- Polynomial heuristics for REPLICA COST problem
  - Heterogeneous platforms
  - Heuristics with and without QoS
  - QoS constraints: QoS of client i represents the maximum distance (number of hops) between i and server(i)
- Experimental assessment of relative performance of the three policies
- Impact of QoS
- No QoS: Traversals of the tree, bottom-up or top-down
- QoS: Sorted lists
- Worst case complexity $O(s^2)$, where $s = |\mathcal{C}| + |\mathcal{N}|$ is problem size

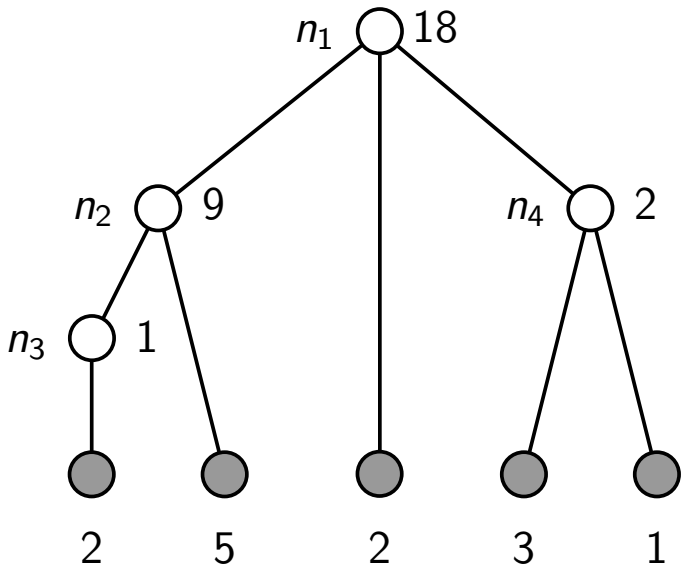
Heuristics

- Polynomial heuristics for REPLICA COST problem
  - Heterogeneous platforms
  - Heuristics with and without QoS
  - QoS constraints: QoS of client i represents the maximum distance (number of hops) between i and server(i)
- Experimental assessment of relative performance of the three policies
- Impact of QoS
- No QoS: Traversals of the tree, bottom-up or top-down
- QoS: Sorted lists
- Worst case complexity $O(s^2)$, where $s = |\mathcal{C}| + |\mathcal{N}|$ is problem size

Heuristics for *Closest*- No QoS

Closest Top Down Largest First **CTDLF**

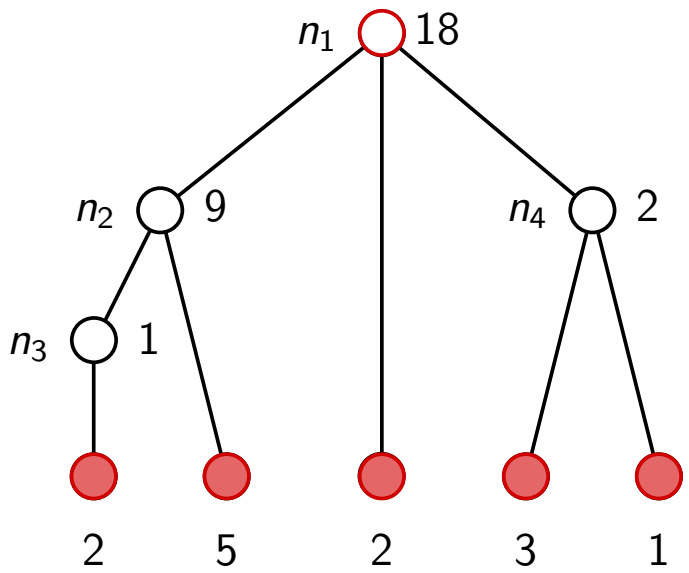
- Traversal of the tree, treating subtrees that contains most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added



Heuristics for *Closest*- No QoS

Closest Top Down Largest First **CTDLF**

- Traversal of the tree, treating subtrees that contains most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added

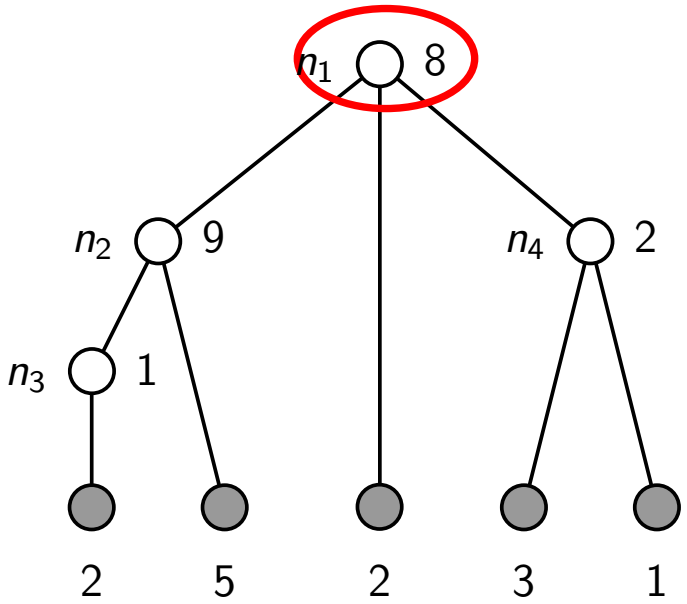


Cost: 18

Heuristics for *Closest*- No QoS

Closest Top Down Largest First **CTDLF**

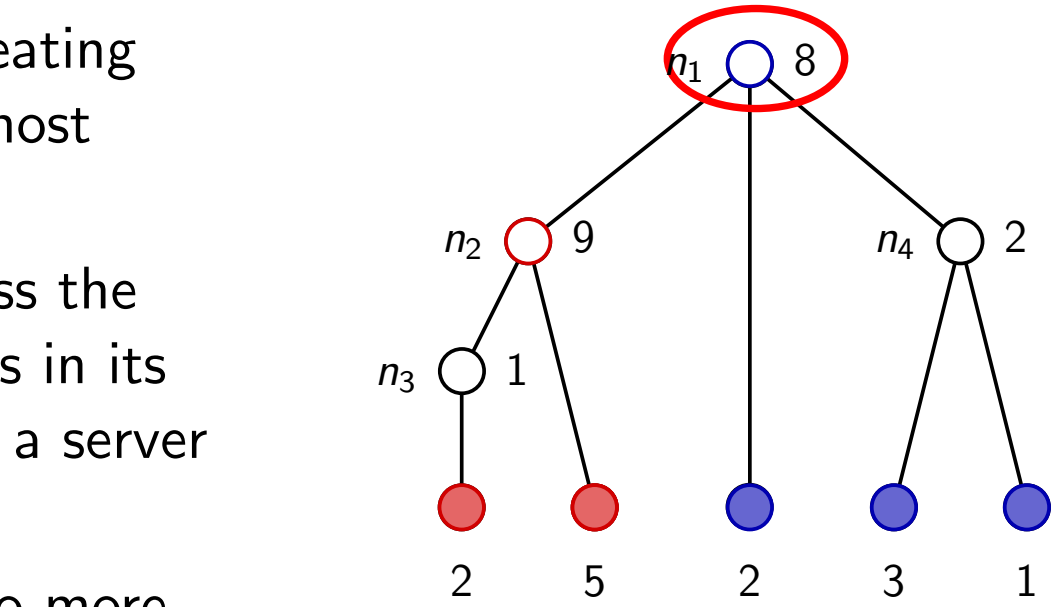
- Traversal of the tree, treating subtrees that contains most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added



Heuristics for *Closest*- No QoS

Closest Top Down Largest First **CTDLF**

- Traversal of the tree, treating subtrees that contains most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added

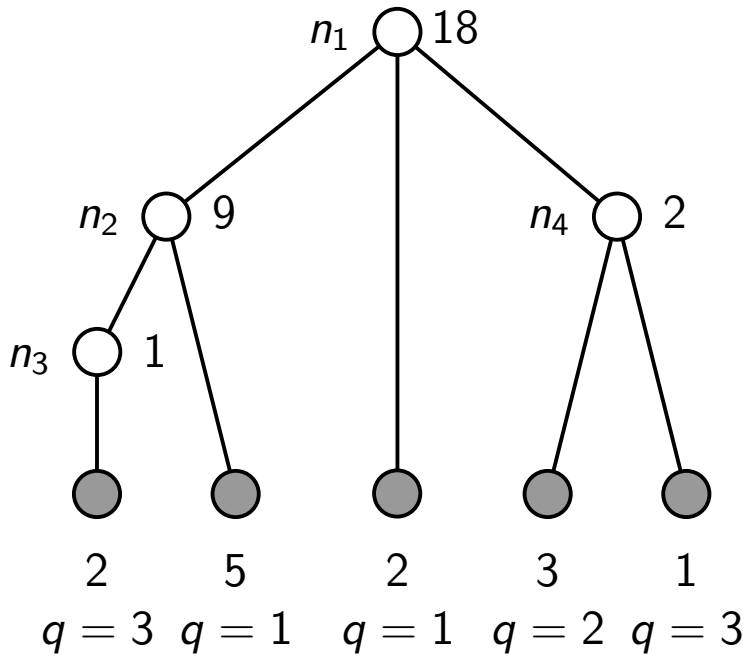


Solution cost: 17

Heuristics for *Closest*- QoS

Closest Big Subtree First **CBSF**

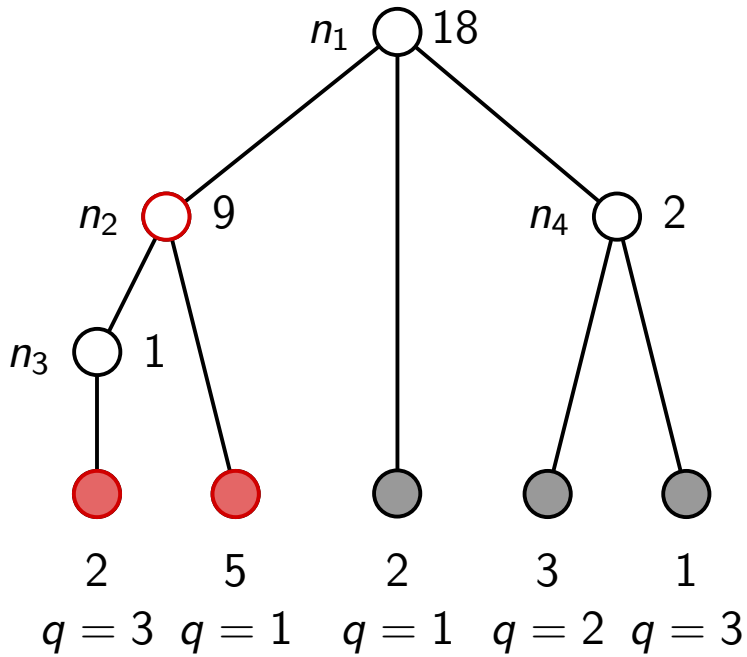
- Traversal of the tree, treating subtrees that contain most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added



Heuristics for *Closest*- QoS

Closest Big Subtree First **CBSF**

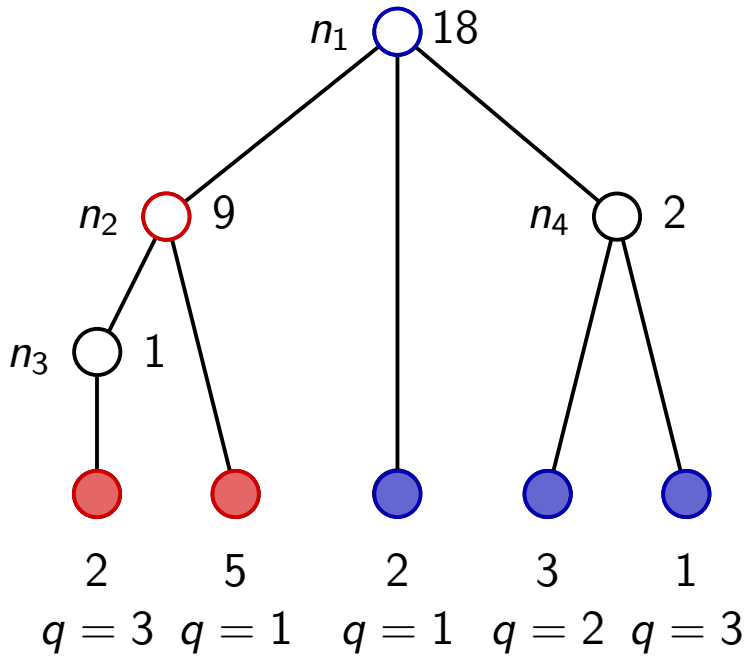
- Traversal of the tree, treating subtrees that contain most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added



Heuristics for *Closest*- QoS

Closest Big Subtree First **CBSF**

- Traversal of the tree, treating subtrees that contain most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added

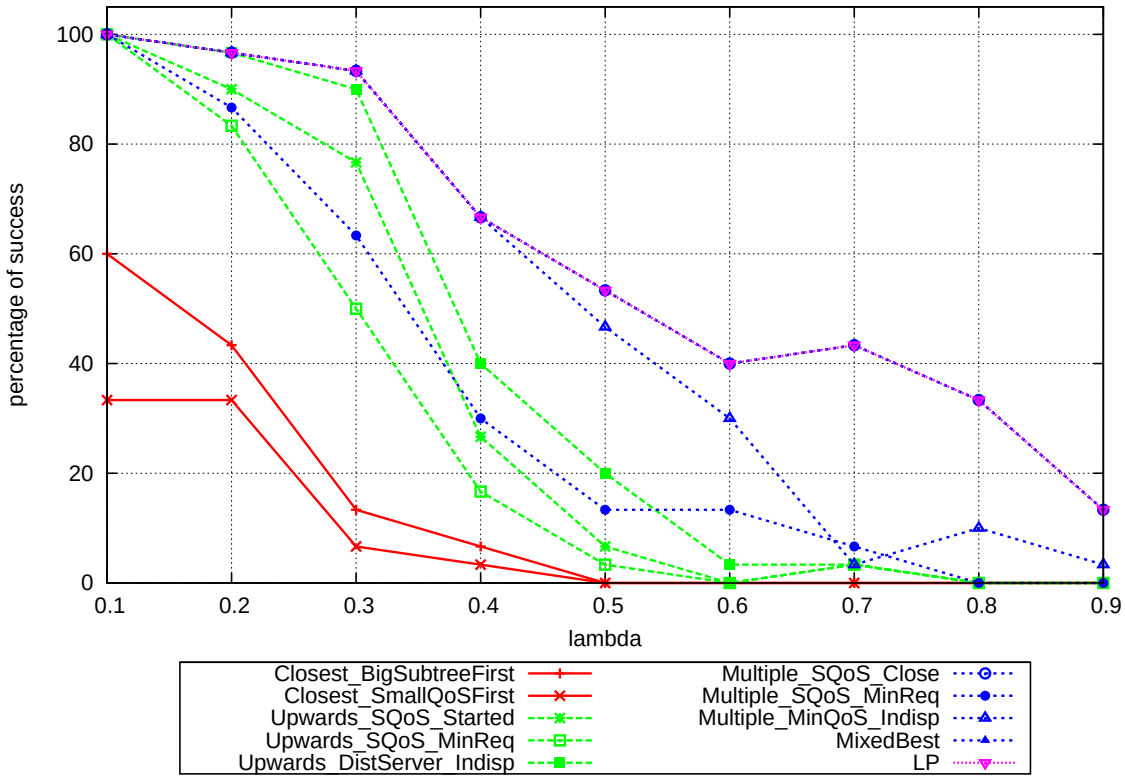


Cost: 27

Results - Percentage of success - QoS

- Number of solutions for each lambda and each heuristic
- $average(qos) = height/2$

$$\lambda = \frac{\sum_{i \in \mathcal{C}} r_i}{\sum_{j \in \mathcal{N}} W_i}$$



Results - Relative Performance

- Distance of the result (in terms of replica cost) of the heuristic to the optimal solution
- T_λ: subset of trees with a solution
- Relative performance:

$$rperf = \frac{1}{|T_\lambda|} \sum_{t \in T_\lambda} \frac{cost_{LP}(t)}{cost_h(t)}$$

- $cost_{LP(t)}$: optimal cost on tree t
- $cost_h(t)$: heuristic cost on tree t; $cost_h(t) = +\infty$ if h did not find any solution

Results - Relative Performance

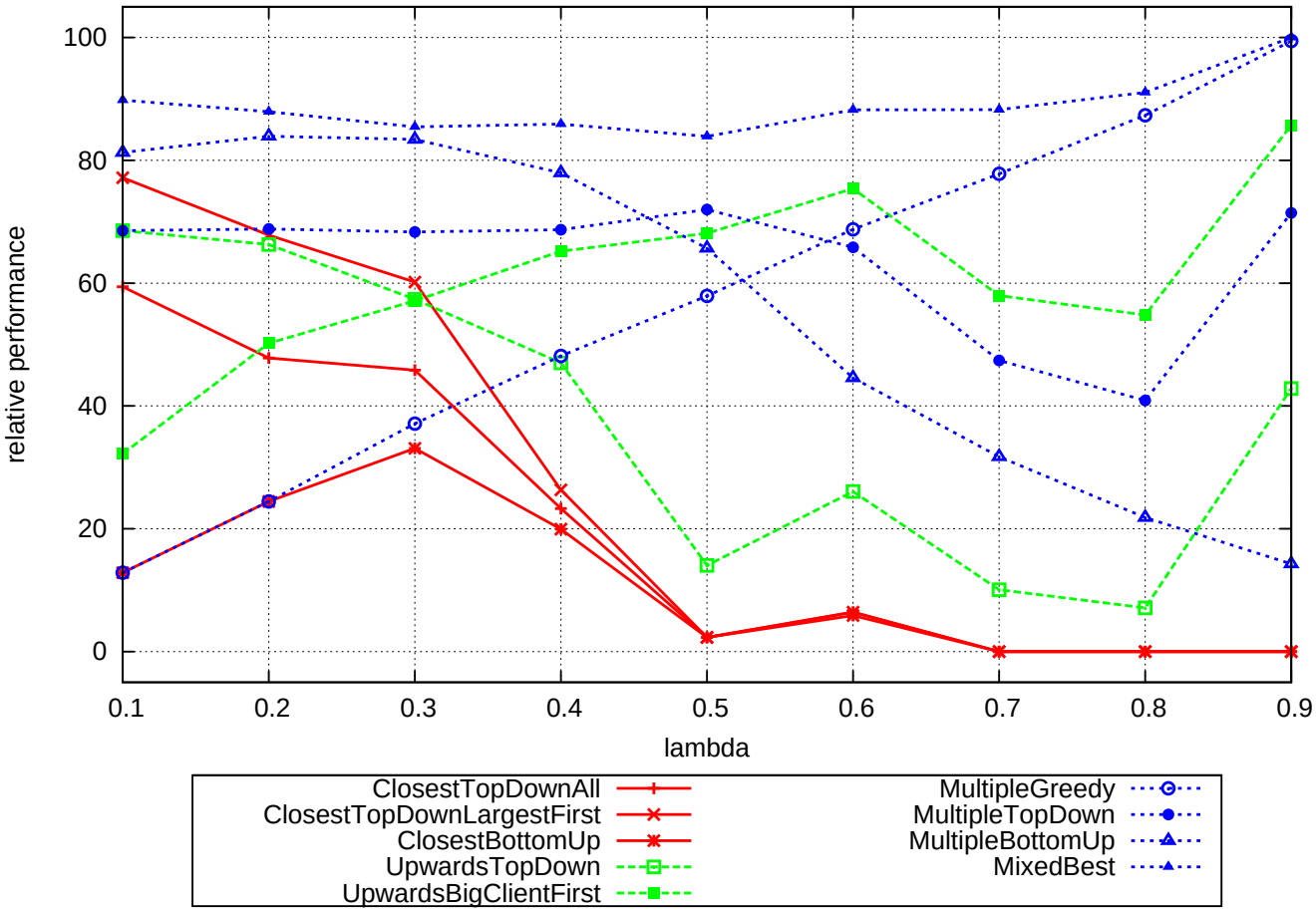
- Distance of the result (in terms of replica cost) of the heuristic to the optimal solution
- T_λ: subset of trees with a solution
- Relative performance:

$$rperf = \frac{1}{|T_\lambda|} \sum_{t \in T_\lambda} \frac{cost_{LP}(t)}{cost_h(t)}$$

- $cost_{LP(t)}$: optimal cost on tree t
- $cost_h(t)$: heuristic cost on tree t; $cost_h(t) = +\infty$ if h did not find any solution

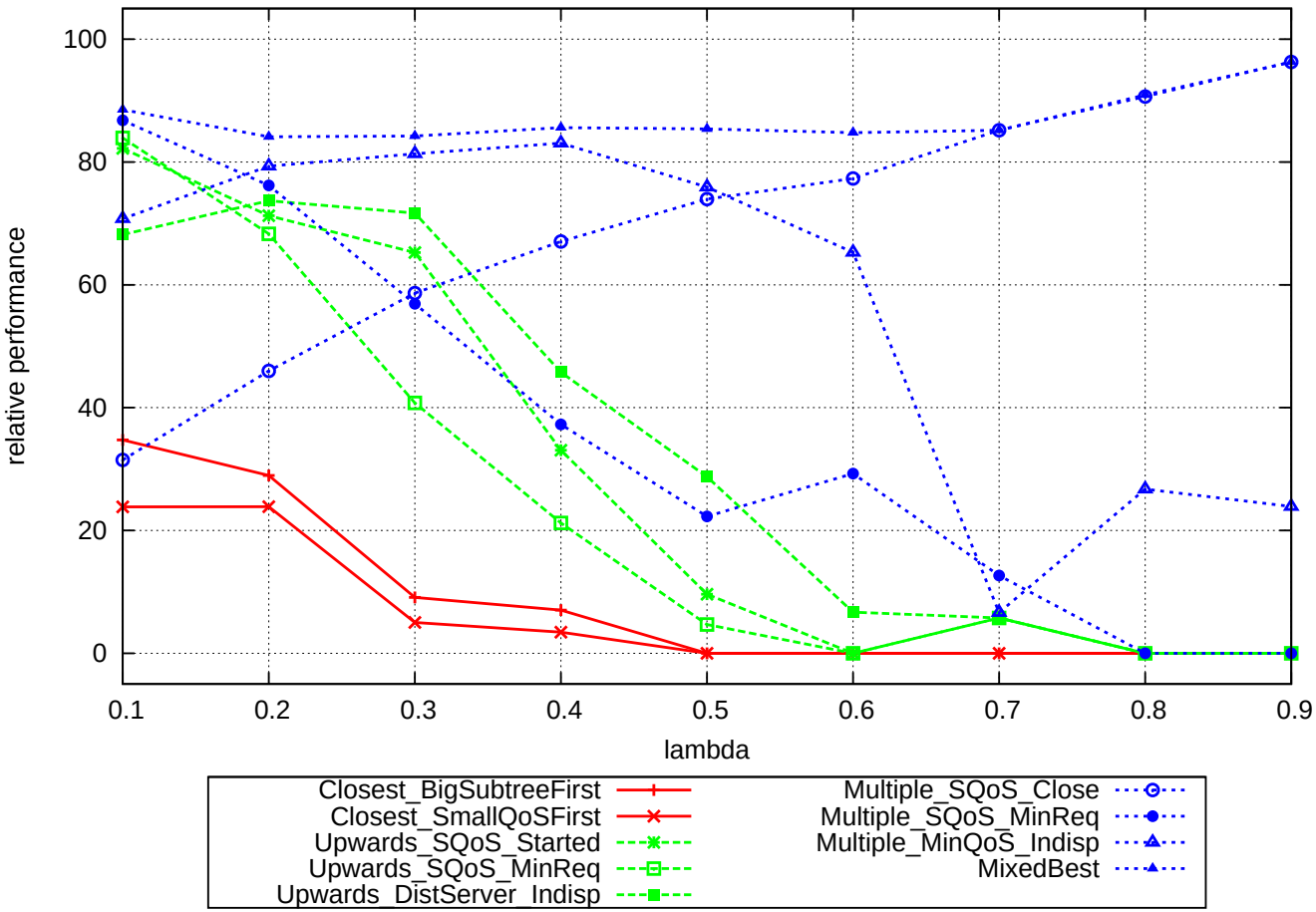
Results - Relative Performance - No QoS

- Heterogeneous results - similar to the homogeneous case



Results - Relative Performance - QoS

$average(qos) = height/2$



Summary

No QoS:

- Striking effect of new policies: many more solutions to the Replica Placement problem
- *Multiple* $\geq$ *Upwards* $\geq$ *Closest*: hierarchy observed within our heuristics
- Best *Multiple* heuristic (MB) always at 85% of the optimal: satisfactory result

QoS:

- Hierarchy also under QoS constraints
- Performance compared to the optimal solution:

$qos \in \{1, 2\}$:	95%
$average(qos) = height/2$:	85%
no qos:	85%

- Smaller trees: results slightly less good

Summary

No QoS:

- Striking effect of new policies: many more solutions to the Replica Placement problem
- *Multiple* $\geq$ *Upwards* $\geq$ *Closest*: hierarchy observed within our heuristics
- Best *Multiple* heuristic (MB) always at 85% of the optimal: satisfactory result

QoS:

- Hierarchy also under QoS constraints
- Performance compared to the optimal solution:

$qos \in \{1, 2\}$:	95%
$average(qos) = height/2$:	85%
no qos:	85%

- Smaller trees: results slightly less good

Related work

- Several papers on replica placement, but...
- ...all consider only the *Closest* policy
- Replica Placement in a general graph is NP-complete
- Wolfson and Milo: impact of the *write* cost, use of a minimum spanning tree for updates. Tree networks: polynomial solution
- Cidon et al (multiple objects) and Liu et al (QoS constraints): polynomial algorithms for homogeneous networks.
- Kalpakis et al: NP-completeness of a variant with bidirectional links (requests served by any node in the tree)
- Karlsson et al: comparison of different objective functions and several heuristics. No QoS, but several other constraints.
- Tang et al: real QoS constraints
- Rodolakis et al: *Multiple* policy but in a very different context

Related work

- Several papers on replica placement, but...
- ...all consider only the *Closest* policy

- Replica Placement in a general graph is NP-complete
- Wolfson and Milo: impact of the *write* cost, use of a minimum spanning tree for updates. Tree networks: polynomial solution
- Cidon et al (multiple objects) and Liu et al (QoS constraints): polynomial algorithms for homogeneous networks.
- Kalpakis et al: NP-completeness of a variant with bidirectional links (requests served by any node in the tree)
- Karlsson et al: comparison of different objective functions and several heuristics. No QoS, but several other constraints.
- Tang et al: real QoS constraints
- Rodolakis et al: *Multiple* policy but in a very different context

Related work

- Several papers on replica placement, but...
- ...all consider only the *Closest* policy

- Replica Placement in a general graph is NP-complete
- Wolfson and Milo: impact of the *write* cost, use of a minimum spanning tree for updates. Tree networks: polynomial solution
- Cidon et al (multiple objects) and Liu et al (QoS constraints): polynomial algorithms for homogeneous networks.
- Kalpakis et al: NP-completeness of a variant with bidirectional links (requests served by any node in the tree)
- Karlsson et al: comparison of different objective functions and several heuristics. No QoS, but several other constraints.
- Tang et al: real QoS constraints
- Rodolakis et al: *Multiple* policy but in a very different context

Outline of the Talk

Introduction and motivation

- Mapping applications onto parallel platforms
  Difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms
  Even more difficult!
- Structured programming approach
  - Easier to program (deadlocks, process starvation)
  - Range of well-known paradigms (pipeline, farm)
  - Algorithmic skeleton: help for mapping

Focus on pipeline applications

Mapping the JPEG encoder pipeline onto a cluster of workstations.

Introduction and motivation

- Mapping applications onto parallel platforms
  Difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms
  Even more difficult!
- Structured programming approach
  - Easier to program (deadlocks, process starvation)
  - Range of well-known paradigms (pipeline, farm)
  - Algorithmic skeleton: help for mapping

Focus on pipeline applications

Mapping the JPEG encoder pipeline onto a cluster of workstations.

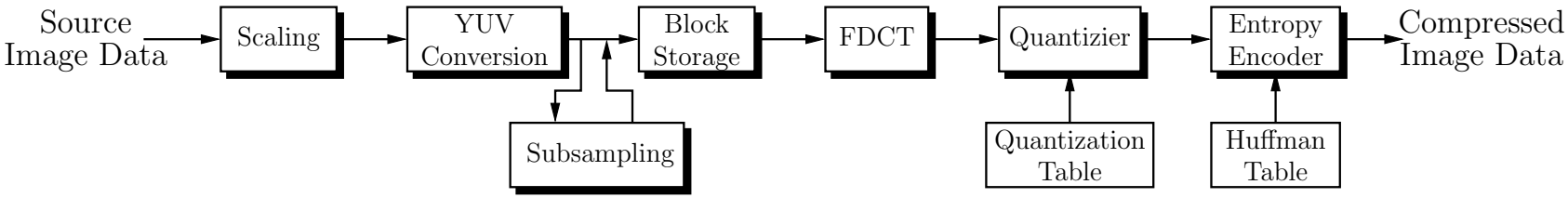
Introduction and motivation

- Mapping applications onto parallel platforms
  Difficult challenge
- Heterogeneous clusters, fully heterogeneous platforms
  Even more difficult!
- Structured programming approach
  - Easier to program (deadlocks, process starvation)
  - Range of well-known paradigms (pipeline, farm)
  - Algorithmic skeleton: help for mapping

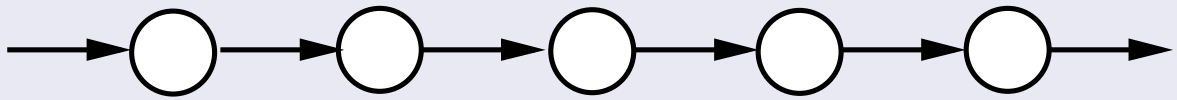
Focus on pipeline applications

Mapping the JPEG encoder pipeline onto a cluster of workstations.

Multi-criteria scheduling of workflows



Workflow



Several consecutive data-sets enter the application graph.

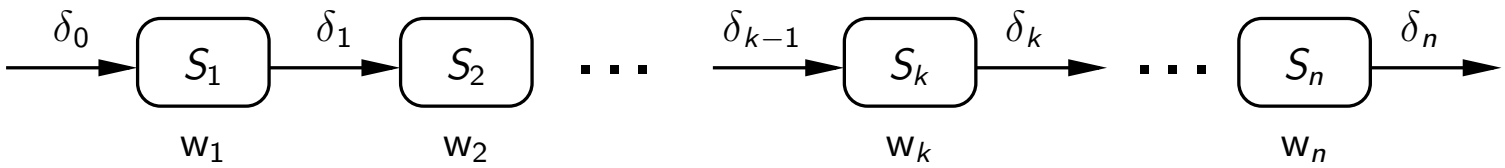
Period $\mathcal{P}$: time interval between the beginning of execution of two consecutive data-sets

Latency $\mathcal{L}$: maximal time elapsed between beginning and end of execution of a data-set

Failure probability $\mathcal{FP}$: the probability that a processor fails during execution

Rule of the game

The application



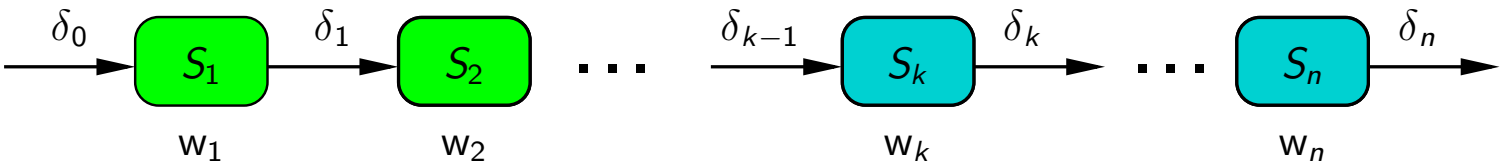
- Cut pipeline into intervals
- Map each interval on a single processor...
- ... or replicate it to improve reliability

The platform

- $\mathcal{P}$ processors
- Fully connected graph (i.e., a clique)

Rule of the game

The application



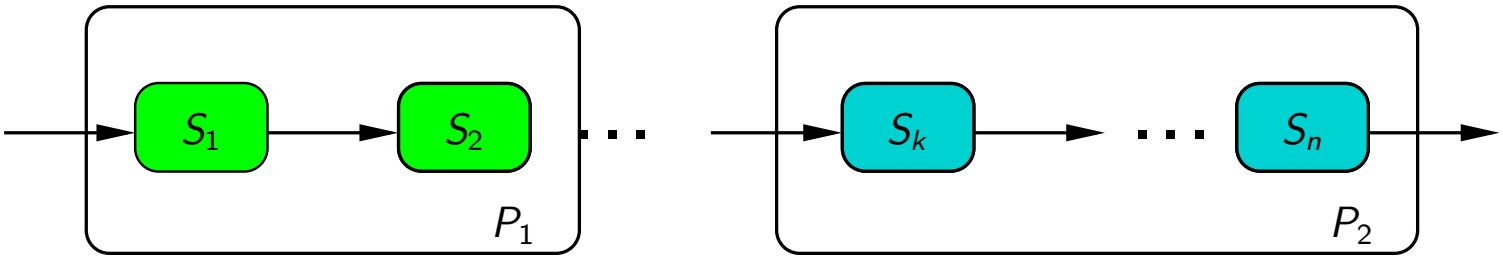
- Cut pipeline into intervals
- Map each interval on a single processor...
- ... or replicate it to improve reliability

The platform

- $\mathcal{P}$ processors
- Fully connected graph (i.e., a clique)

Rule of the game

The application



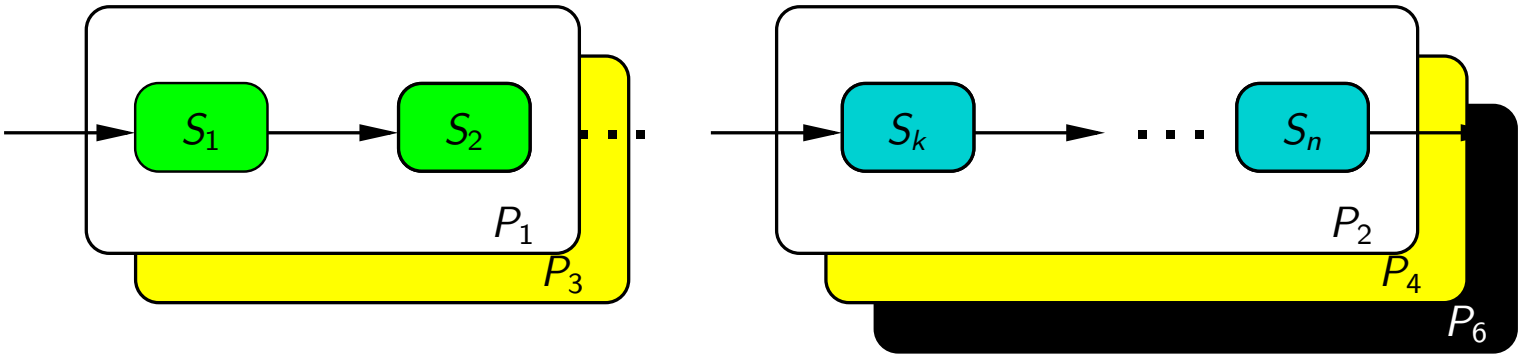
- Cut pipeline into intervals
- Map each interval on a single processor...
- ... or replicate it to improve reliability

The platform

- $\mathcal{P}$ processors
- Fully connected graph (i.e., a clique)

Rule of the game

The application



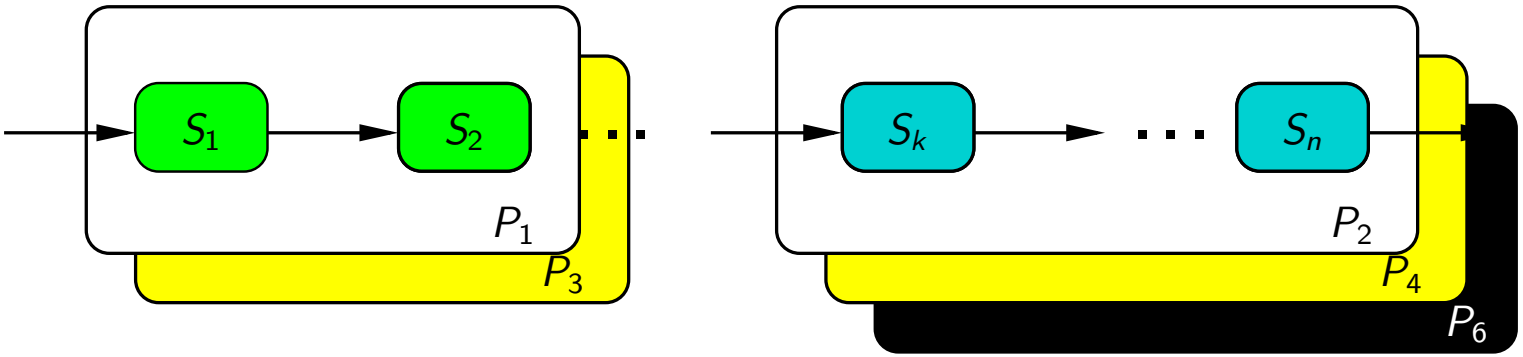
- Cut pipeline into intervals
- Map each interval on a single processor...
- ... or replicate it to improve reliability

The platform

- $\mathcal{P}$ processors
- Fully connected graph (i.e., a clique)

Rule of the game

The application



- Cut pipeline into intervals
- Map each interval on a single processor...
- ... or replicate it to improve reliability

The platform

- $\mathcal{P}$ processors
- Fully connected graph (i.e., a clique)

Objective function?

Mono-criterion

- Minimize $\mathcal{P}$
- Minimize $\mathcal{L}$
- Minimize $\mathcal{FP}$

Bi-criteria

- How to define it?
  Minimize $\alpha.\mathcal{P} + \beta.\mathcal{L}$?
  Minimize $\alpha.\mathcal{L} + \beta.\mathcal{FP}$?
- Values which are not comparable
- Minimize $\mathcal{P}$ for a fixed latency
- Minimize $\mathcal{L}$ for a fixed period
- Minimize $\mathcal{FP}$ for a fixed latency
- Minimize $\mathcal{L}$ for a fixed failure probability

Objective function?

Mono-criterion

- Minimize $\mathcal{P}$
- Minimize $\mathcal{L}$
- Minimize $\mathcal{FP}$

Bi-criteria

- How to define it?
  Minimize $\alpha.\mathcal{P} + \beta.\mathcal{L}$?
  Minimize $\alpha.\mathcal{L} + \beta.\mathcal{FP}$?
- Values which are not comparable
- Minimize $\mathcal{P}$ for a fixed latency
- Minimize $\mathcal{L}$ for a fixed period
- Minimize $\mathcal{FP}$ for a fixed latency
- Minimize $\mathcal{L}$ for a fixed failure probability

Objective function?

Mono-criterion

- Minimize $\mathcal{P}$
- Minimize $\mathcal{L}$
- Minimize $\mathcal{FP}$

Bi-criteria

- How to define it?
  Minimize $\alpha.\mathcal{P} + \beta.\mathcal{L}$?
  Minimize $\alpha.\mathcal{L} + \beta.\mathcal{FP}$?
- Values which are not comparable

- Minimize $\mathcal{P}$ for a fixed latency
- Minimize $\mathcal{L}$ for a fixed period

- Minimize $\mathcal{FP}$ for a fixed latency
- Minimize $\mathcal{L}$ for a fixed failure probability

Objective function?

Mono-criterion

- Minimize $\mathcal{P}$
- Minimize $\mathcal{L}$
- Minimize $\mathcal{FP}$

Bi-criteria

- How to define it?
  Minimize $\alpha.\mathcal{P} + \beta.\mathcal{L}$?
  Minimize $\alpha.\mathcal{L} + \beta.\mathcal{FP}$?
- Values which are not comparable
- Minimize $\mathcal{P}$ for a fixed latency
- Minimize $\mathcal{L}$ for a fixed period
- Minimize $\mathcal{FP}$ for a fixed latency
- Minimize $\mathcal{L}$ for a fixed failure probability

An Optimal Algorithm

Minimize L with fixed P - Homogeneous platform

$L(i,q)$: min. latency with exactly q procs mapping stages 1 to i

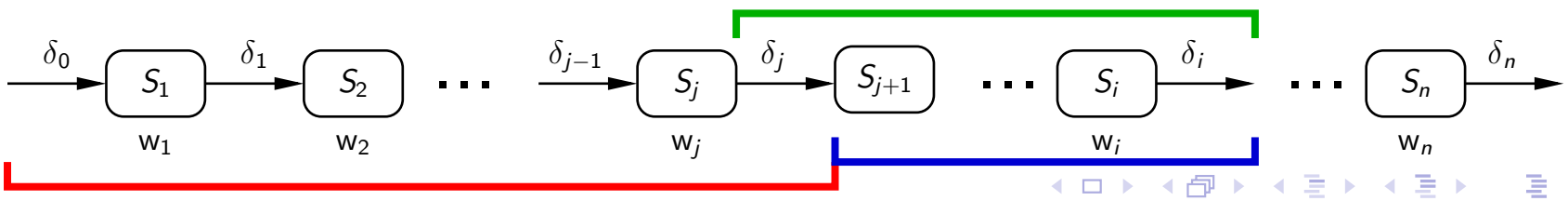
$L(n,q)$ for $1 \leq q \leq p$

Init:

$$L(i,1) = \begin{cases} \frac{\delta_0}{b} + \sum_{k=1}^{i} \frac{w_k}{s} + \frac{\delta_i}{b} & \text{if} \leq P \\ \infty & \text{else} \end{cases}, \quad L(1,q) = \infty \text{ if } q > 1$$

Recursion:

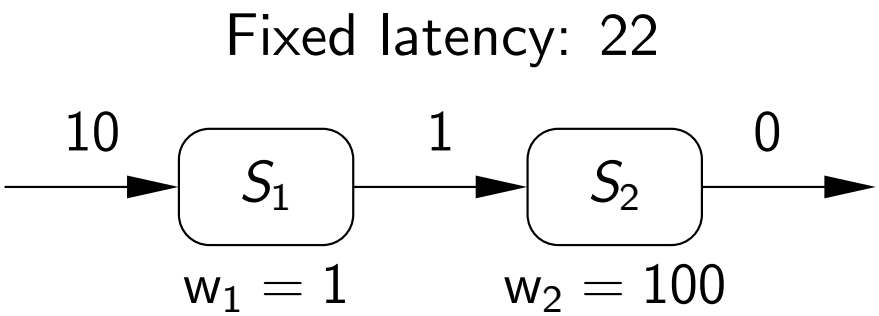
$$L(i,q) = \min_{j<i} \left\{ L(j,q-1) + \sum_{k=j+1}^{i} \frac{w_k}{s} + \frac{\delta_i}{b} \;\middle|\; \frac{\delta_j}{b} + \sum_{k=j+1}^{i} \frac{w_k}{s} + \frac{\delta_i}{b} \leq P \right\}$$

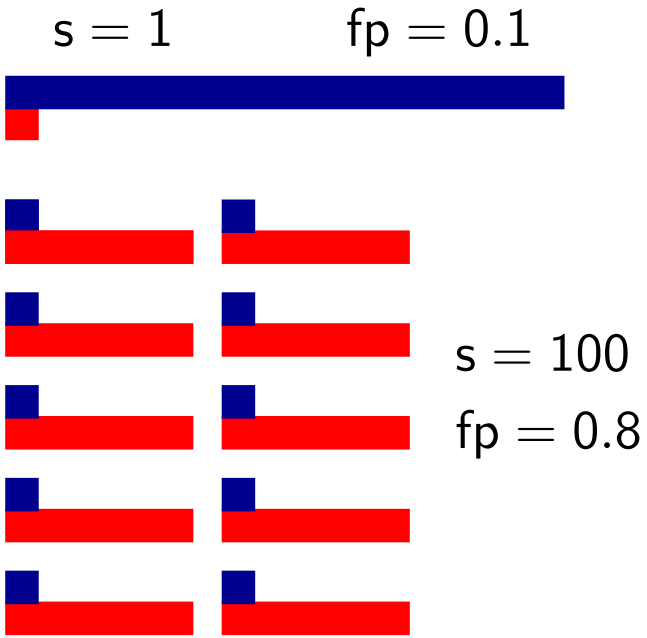


Exemple: Minimizing $\mathcal{FP}$ with fixed latency

Minimize $\mathcal{FP}$ with fixed latency

Different speed processors - Failure heterogeneous



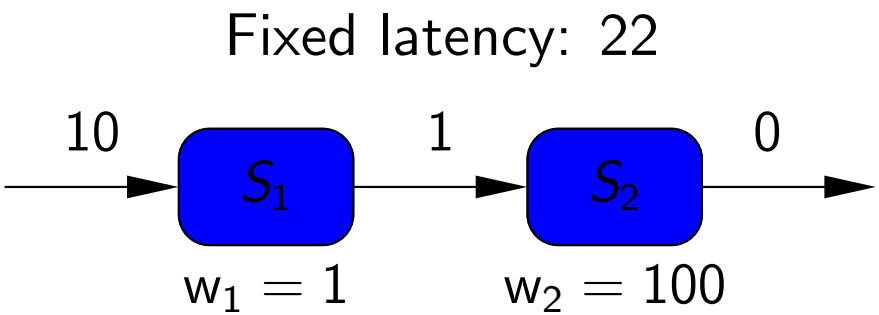


Open complexity!

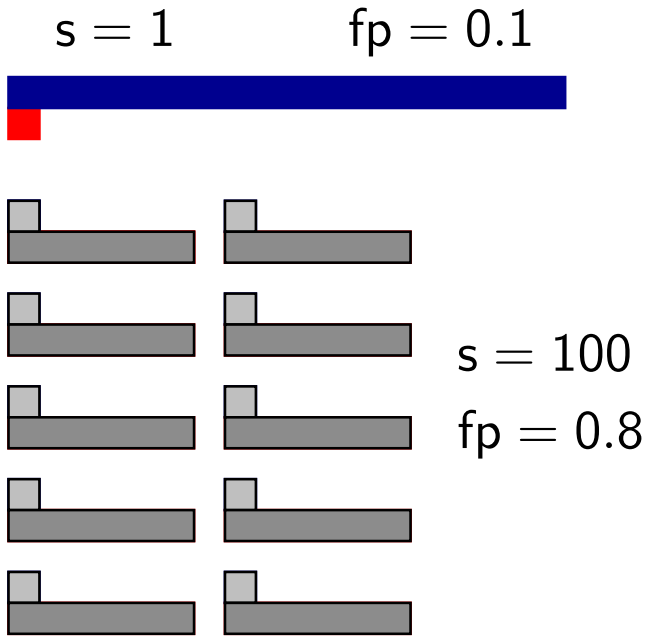
Exemple: Minimizing $\mathcal{FP}$ with fixed latency

Minimize $\mathcal{FP}$ with fixed latency

Different speed processors - Failure heterogeneous



$10 + 101 \gg 22$

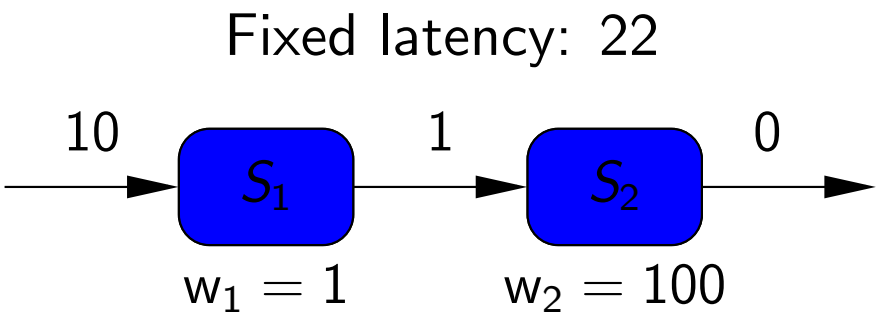


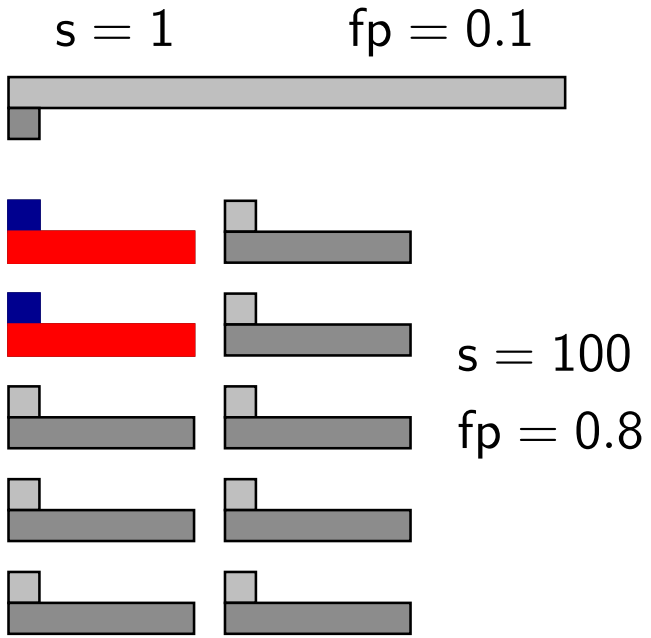
Open complexity!

Exemple: Minimizing $\mathcal{FP}$ with fixed latency

Minimize $\mathcal{FP}$ with fixed latency

Different speed processors - Failure heterogeneous





$20 + 101/100 < 22$

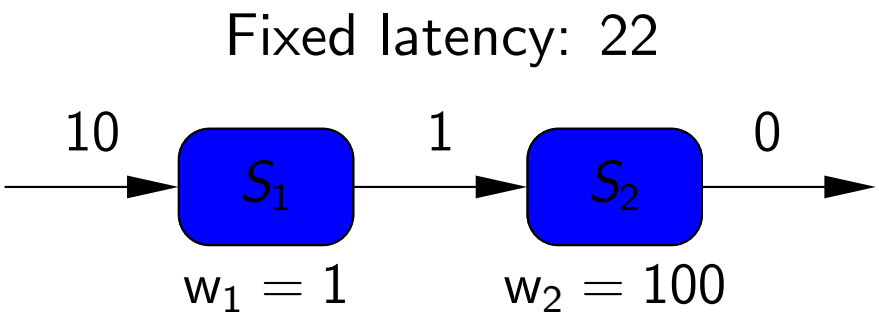
$\mathcal{FP} = (1 - (1 - 0.8^2)) = 0.64$

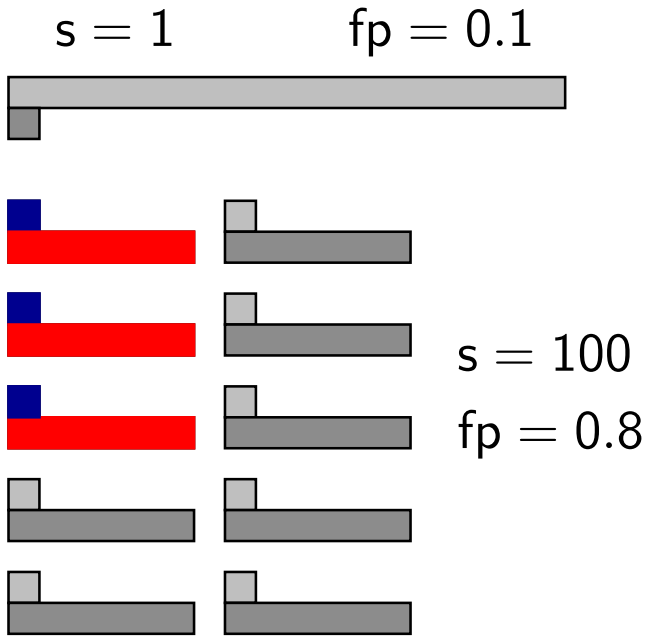
Open complexity!

Exemple: Minimizing $\mathcal{FP}$ with fixed latency

Minimize $\mathcal{FP}$ with fixed latency

Different speed processors - Failure heterogeneous





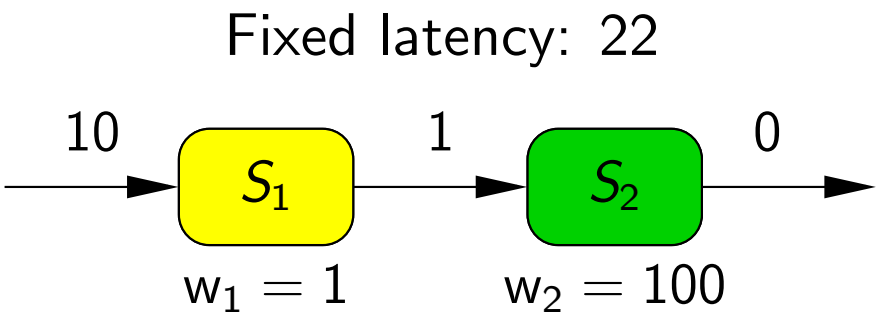
$30 + 101/100 > 22$

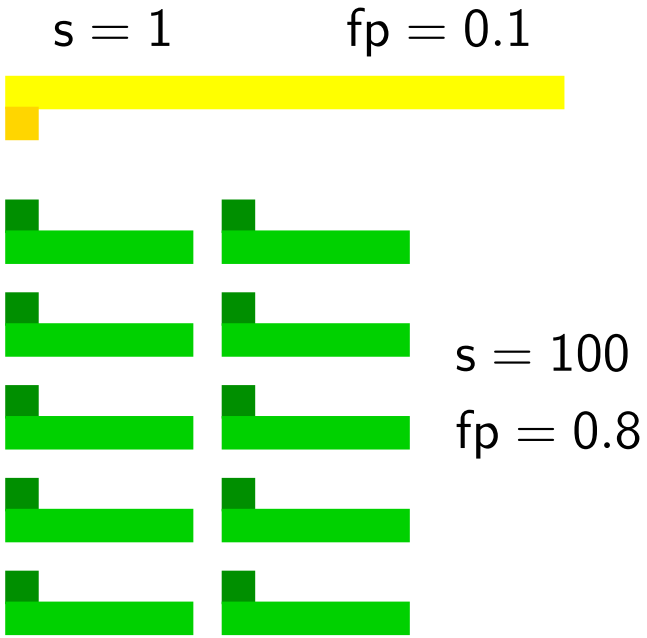
Open complexity!

Exemple: Minimizing $\mathcal{FP}$ with fixed latency

Minimize $\mathcal{FP}$ with fixed latency

Different speed processors - Failure heterogeneous





$10 + 1/1 + 10 \times 1 + 100/100 = 22$

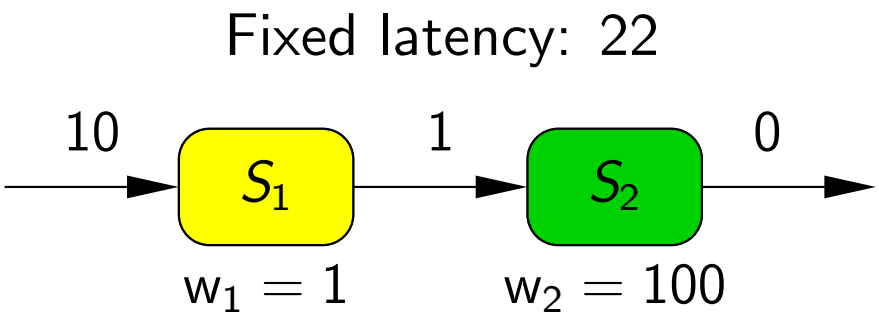
$\mathcal{FP} : 1-(1-0.1)\times(1-0.8^{10}) < 0.2$

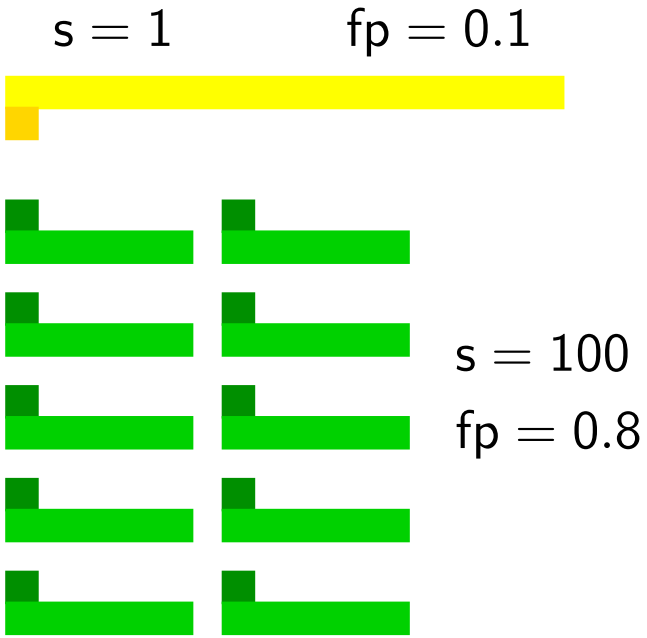
Open complexity!

Exemple: Minimizing $\mathcal{FP}$ with fixed latency

Minimize $\mathcal{FP}$ with fixed latency

Different speed processors - Failure heterogeneous





Open complexity!

Complexity Results

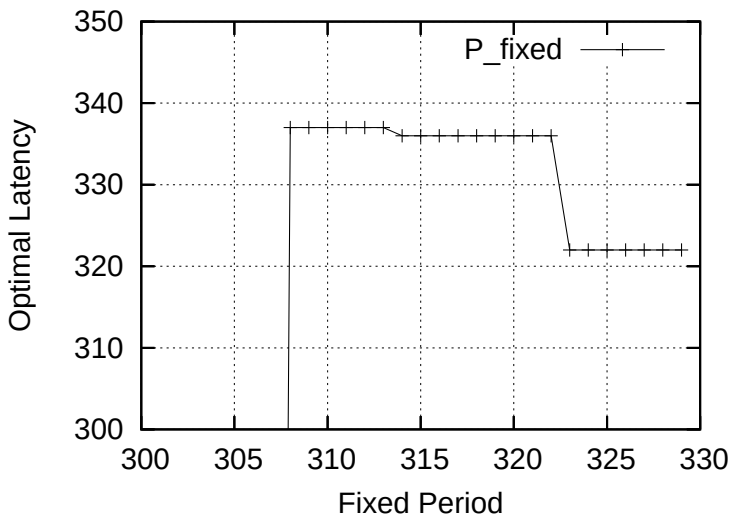
Bi-criteria interval mapping

Objective	Failure	Hom.	Com. Hom.	Het.
P & L	/	polynomial	NP-hard	NP-hard
FP & L	hom.	polynomial	polynomial	NP-hard
FP & L	het.	polynomial	open	NP-hard

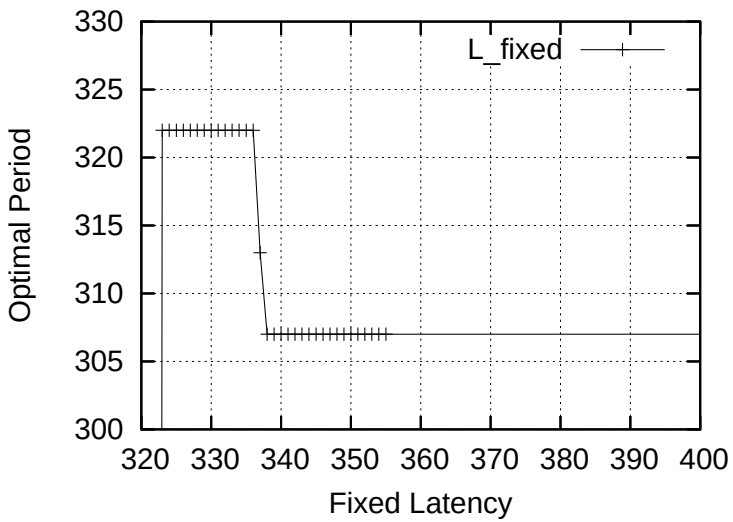
Integer linear programming

- Integer LP to solve Interval Mapping on *Communication Homogeneous* platforms
- Many integer variables: no efficient algorithm to solve
- Approach limited to small problem instances
- Absolute performance of the heuristics for such instances

Bucket behavior of LP solutions



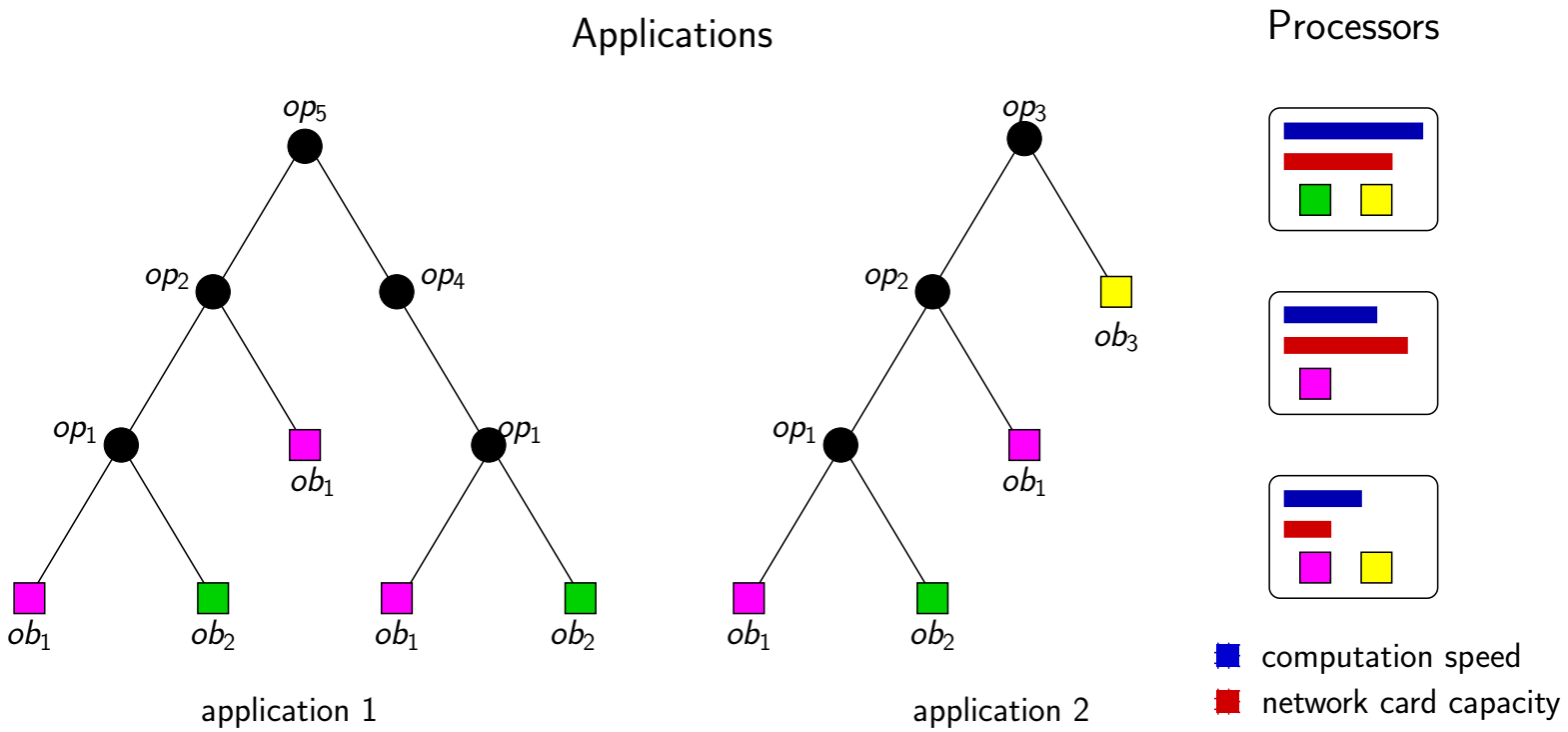
(a) Fixed P.



(b) Fixed L.

Outline of the Talk

Rule of the Game

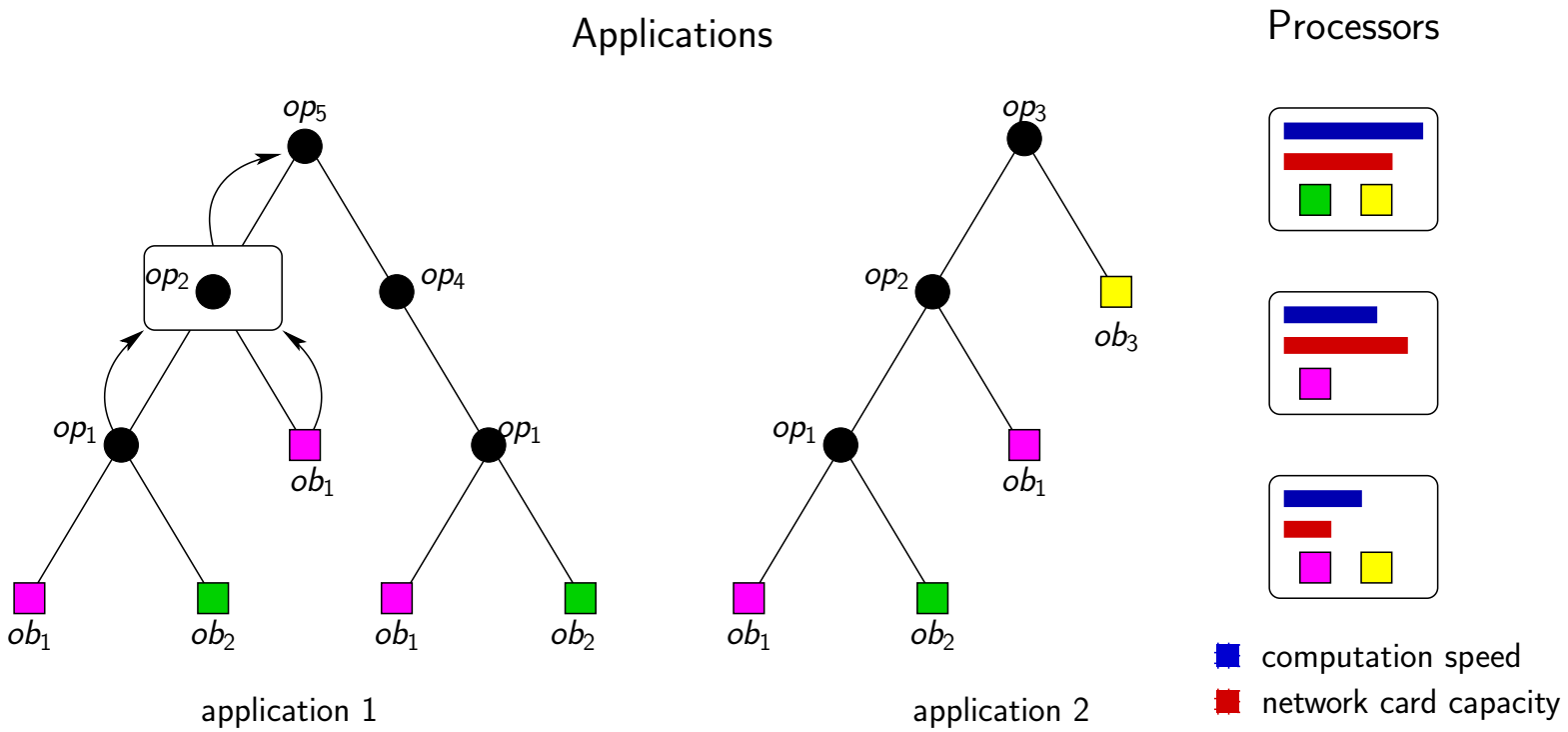


Goal

Minimize total processing power of the target platform while matching all application requirements.

Assess impact of reusing intermediate results.

Rule of the Game

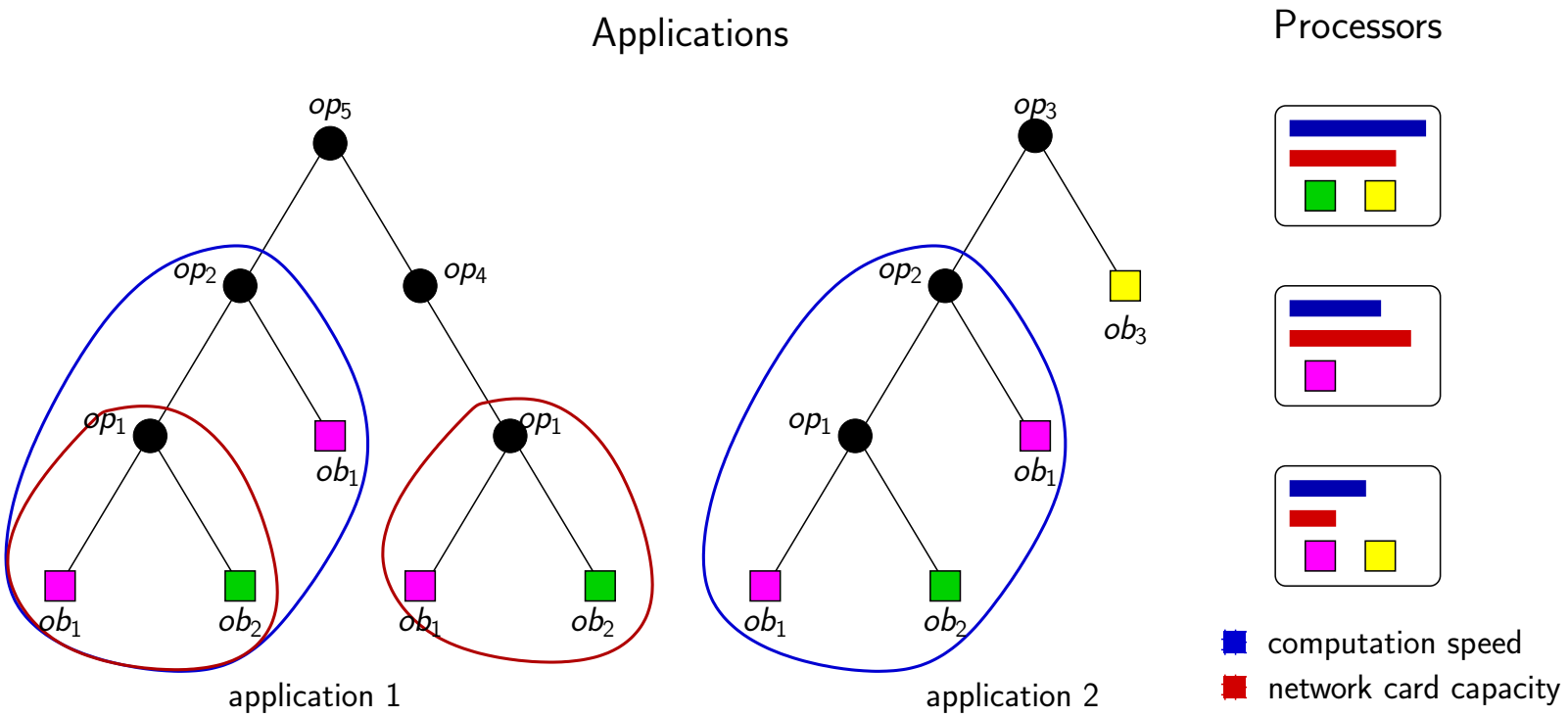


Goal

Minimize total processing power of the target platform while matching all application requirements.

Assess impact of reusing intermediate results.

Rule of the Game

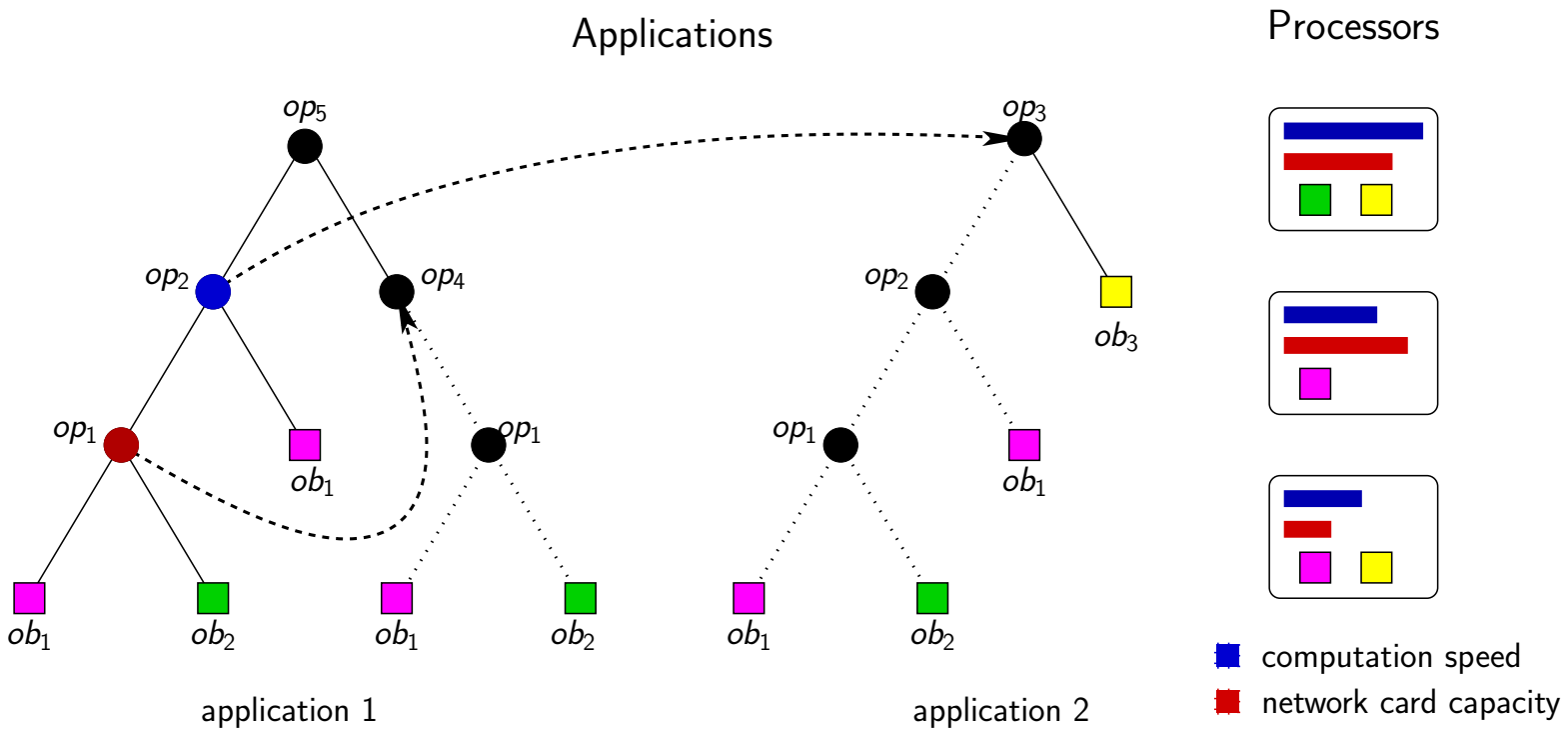


Goal

Minimize total processing power of the target platform while matching all application requirements.

Assess impact of reusing intermediate results.

Rule of the Game

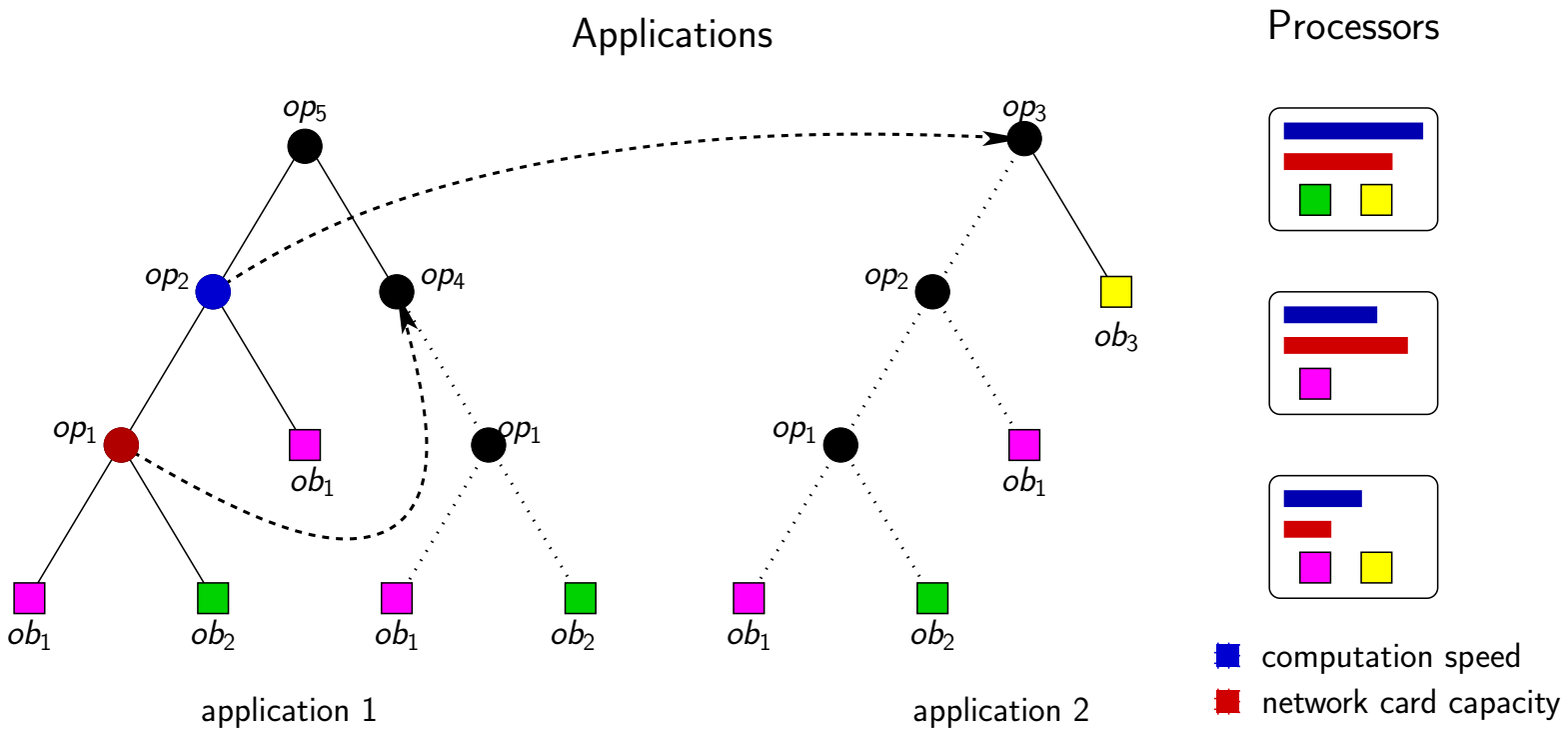


Goal

Minimize total processing power of the target platform while matching all application requirements.

Assess impact of reusing intermediate results.

Rule of the Game



Goal

Minimize total processing power of the target platform while matching all application requirements.

Assess impact of reusing intermediate results.

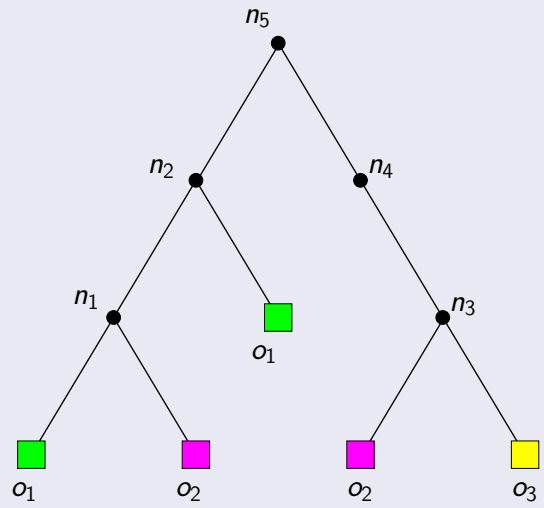
The Application Model

- $\mathcal{K}$ applications
- $\mathcal{OP} = \{op_1, op_2, \ldots\}$ set of operators
- $\mathcal{OB} = \{ob_1, ob_2, ob_3, \ldots\}$ basic objects
- Computation of operator op_p: w_p operations, δ_p size of output

For application k:
$\rho^{(k)}$ application throughput

Object ob_j
- d_j size of ob_j
- $f_j^{(k)}$ download frequency

Application tree



The Platform Model

The platform

- $\mathcal{P}$ processors
- Fully connected graph (i.e., a clique)

Objective

Map operators onto processors such that processing power is minimized and all application throughputs are achieved.

Overview of Heuristics (1)

Server selection strategies:

- (S1) Select the fastest processor (blocking);
- (S2) Select the processor with the fastest network card (blocking);
- (S3) Select the fastest processor (non-blocking);
- (S4) Select the processor with the fastest network card (non-blocking).

Overview of Heuristics (2)

Heuristics: Reuse of intermediate results

(H1) RandomNoReuse

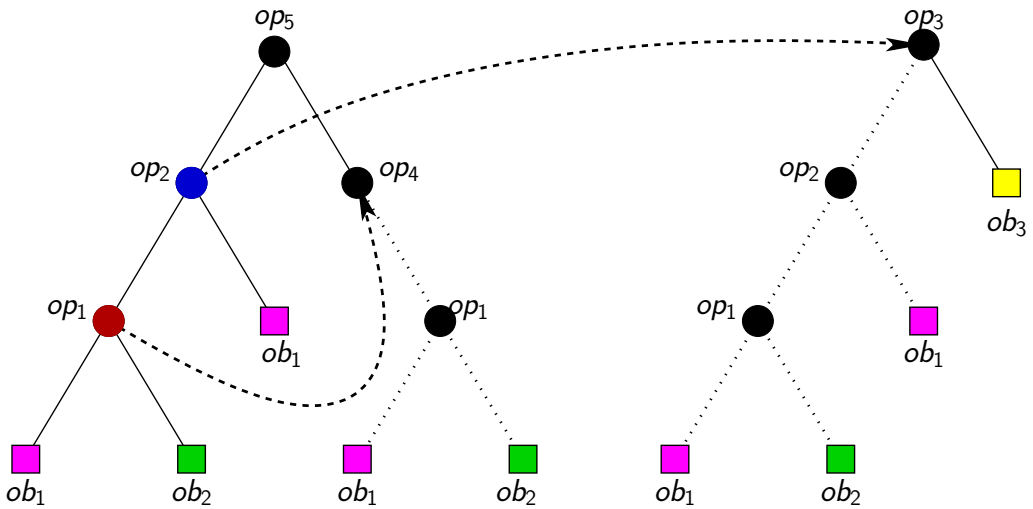
(H2) Random

(H3) TopDownBFS

(H4) TopDownDFS

(H5) BottomUpBFS

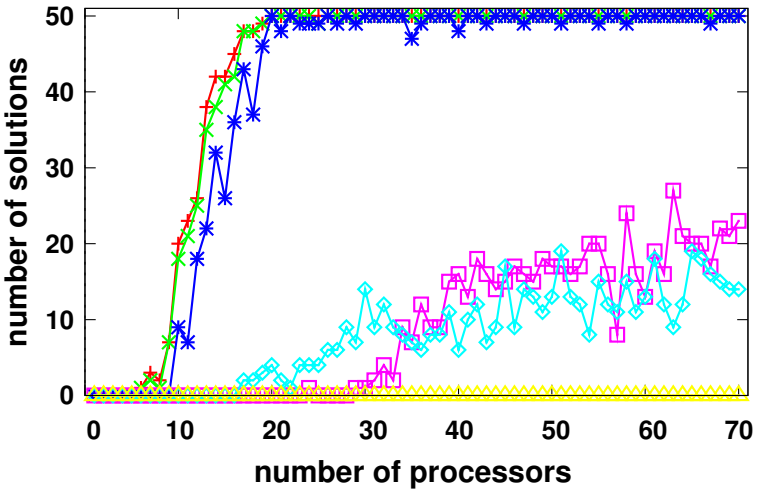
(H6) BottomUpDFS



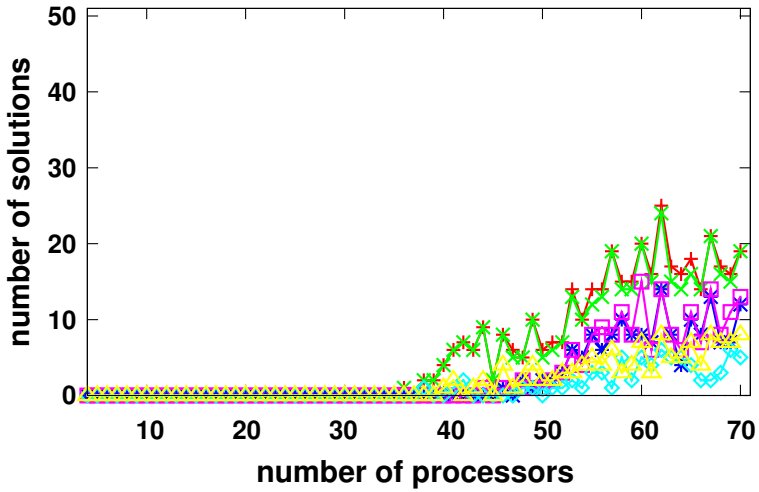
Results

Number of processors increases.
50 runs. 5 applications. 50 operators.

Successful runs.



(S3) Fastest proc.

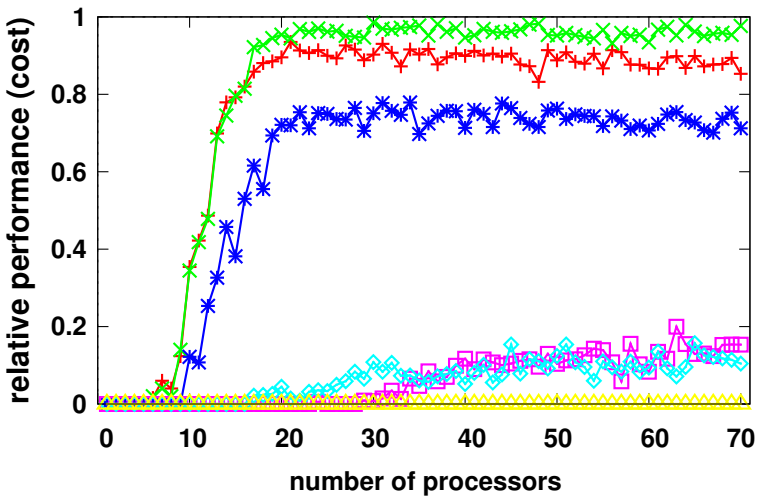


(S3) Fastest proc - no reuse.

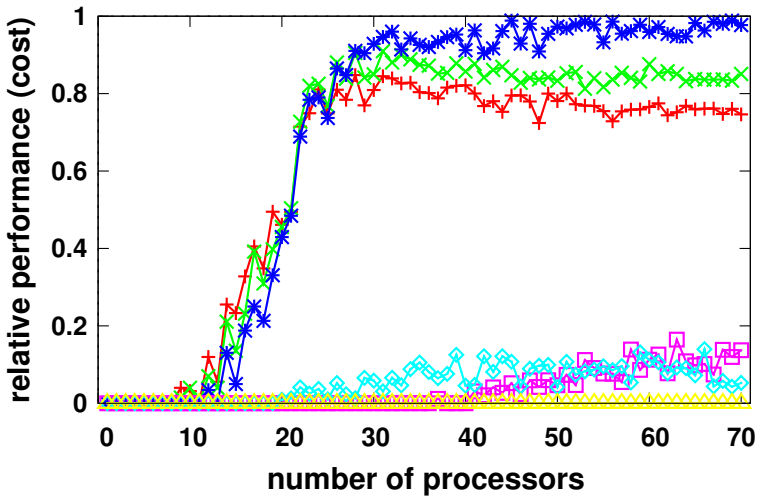
Results

Number of processors increases.
50 runs. 5 applications. 50 operators.

Relative performance.



(S3) Fastest proc.



(S1) Fastest proc - blocking.

Summary

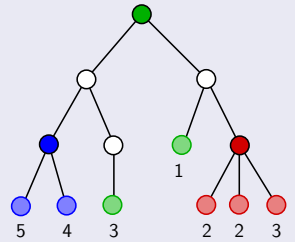
- Random approach dramatically bad
- Neglecting reuse limits success rate and quality of solution in terms of cost
- Top Down approach turns out to be the best 
- BottomUp only with BFS competitive 
- DFS unable to reuse results efficiently (bandwidth)
- Strong dependency of processor selection strategy on solution quality
- Solid combination: TopDownBFS with fastest proc - non-blocking 

Outline of the Talk

Summary

Study of three mapping and scheduling problems

- Replica placement
- Pipeline workflows
- In-network stream processing



Complexity study:

- Influence of heterogeneity
- Multi-criteria optimization

Algorithms:

- Optimal algorithms and NP-completeness proofs
- Heuristics for NP-complete instances

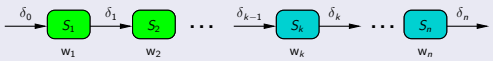
Experiments:

- Absolute performance via comparison to LP
- MPI-application of JPEG encoder

Summary

Study of three mapping and scheduling problems

- Replica placement
- Pipeline workflows
- In-network stream processing



Complexity study:

- Influence of heterogeneity
- Multi-criteria optimization

Algorithms:

- Optimal algorithms and NP-completeness proofs
- Heuristics for NP-complete instances

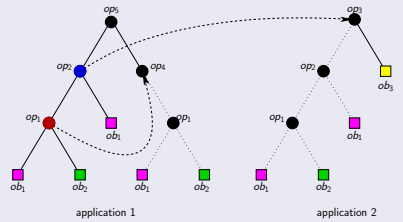
Experiments:

- Absolute performance via comparison to LP
- MPI-application of JPEG encoder

Summary

Study of three mapping and scheduling problems

- Replica placement
- Pipeline workflows
- In-network stream processing



Complexity study:

- Influence of heterogeneity
- Multi-criteria optimization

Algorithms:

- Optimal algorithms and NP-completeness proofs
- Heuristics for NP-complete instances

Experiments:

- Absolute performance via comparison to LP
- MPI-application of JPEG encoder

Summary

Study of three mapping and scheduling problems

- Replica placement
- Pipeline workflows
- In-network stream processing

Complexity study:

- Influence of heterogeneity
- Multi-criteria optimization

Algorithms:

- Optimal algorithms and NP-completeness proofs
- Heuristics for NP-complete instances

Experiments:

- Absolute performance via comparison to LP
- MPI-application of JPEG encoder

Ongoing Work

Tri-criteria optimization on pipelines:

- Latency - reliability - period
- Heuristics
- More ambitious application: MPEG4

Perspectives

Replica Placement

- More simulations for the REPLICA COST problem: shape of the trees, distribution law of the requests, degree of heterogeneity of the platforms
- Consider the problem with several object types
- Extension to more complex objective functions

Pipeline Workflow

- Extension to fork, fork-join and tree workflows
- Multi-criteria: new objectives like power consumption and rental cost

Stream Processing

- Mutable applications: Operators can be rearranged based on operator associativity and commutativity rules

Journals

A. Benoit, H. Kosch, V. Rehn-Sonigo, and Y. Robert.
Multi-criteria Scheduling of Pipeline Workflows (and Application to the JPEG Encoder)
Int. Journal of High Performance Computing Applications, 23, 2009

A.Benoit, V. Rehn-Sonigo, and Y. Robert.
Replica Placement and Access Policies in Tree Networks.
IEEE Transactions on Parallel and Distributed Systems, 19(12), 2008.

L. Marchal, V. Rehn, Y. Robert, and F. Vivien.
Scheduling algorithms for data redistribution and load-balancing on master-slave platforms.
Parallel Processing Letters, 17(1), 2007.

Conferences

Replica Placement
CoreGRID'2007, Core GRID Symposium 2007
ICCS'07, the 2007 International Conference on Computational Science.
HCW'07, the 16th Heterogeneity in Computing Workshop.

Pipeline Workflow Applications
ICCS'08, the 2008 International Conference on Computational Science.
HCW'08, the 17th Heterogeneity in Computing Workshop. HeteroPar'07, Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks (in conjunction with Cluster 2007).

In-Network Stream Processing
APDCM'09, the 11th Workshop on Advances in Parallel and Distributed Computational Models.

Scheduling Strategies on Star Platforms
PDP'2007, 15th Euromicro Workshop on Parallel, Distributed and Network-based Processing. HCW'06, the 15th Heterogeneous Computing Workshop.