Impact of QoS on Replica Placement in Tree Networks

Anne Benoit, Veronika Rehn and Yves Robert

GRAAL team, LIP
École Normale Supérieure de Lyon

May 2007

Introduction and motivation

- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS 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)

How many replicas required?
Which locations?
Total replica cost?

Introduction and motivation

- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS 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)

How many replicas required?
Which locations?
Total replica cost?

Introduction and motivation

- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS 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)

How many replicas required?
Which locations?
Total replica cost?

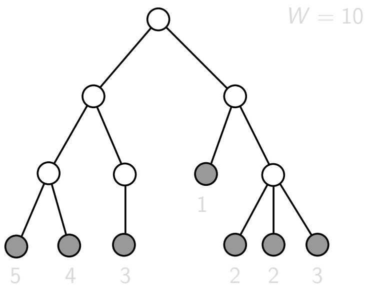
Introduction and motivation

- Replica placement in tree networks
- Set of clients (tree leaves): requests with QoS 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)

How many replicas required?
Which locations?
Total replica cost?

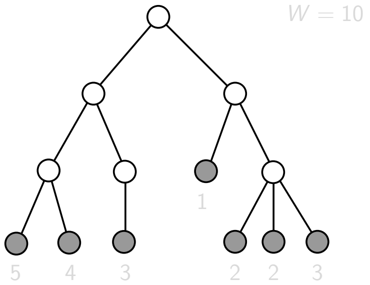
Rule of the game

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



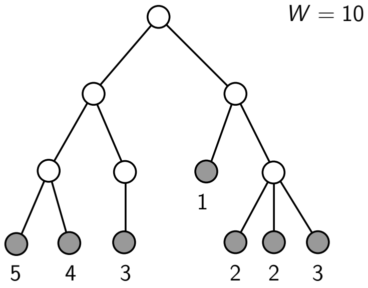
Rule of the game

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



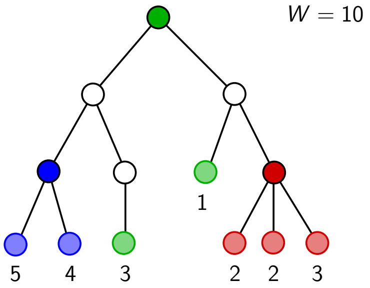
Rule of the game

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



Rule of the game

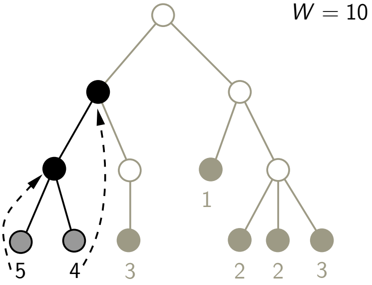
- Handle all client requests, and minimize cost of replicas
- → REPLICA PLACEMENT problem
- Several policies to assign replicas



Closest

Rule of the game

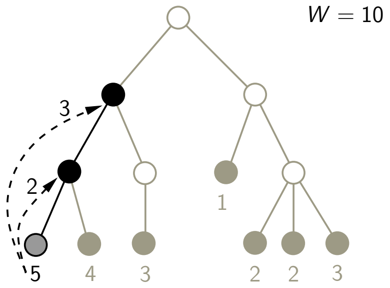
- Handle all client requests, and minimize cost of replicas
- → Replica Placement problem
- Several policies to assign replicas



Upwards

Rule of the game

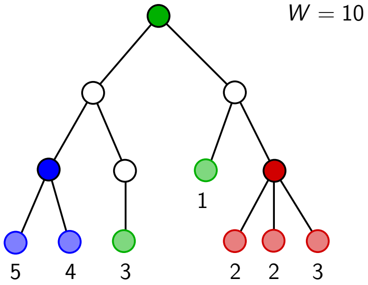
- Handle all client requests, and minimize cost of replicas
- → Replica Placement problem
- Several policies to assign replicas



Multiple

Rule of the game

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



Closest: with QoS?

Major contributions

Theory New access policies
Problem complexity with QoS
LP-based optimal solution

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

Major contributions

Theory New access policies
Problem complexity with QoS
LP-based optimal solution

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

Outline

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 = nb hops)
- **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 $comm_l = 1$

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 = nb hops)
- **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 $\mathrm{comm}_l = 1$

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 = nb hops)
- **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 $\mathsf{comm}_l = 1$

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 = nb hops)
- **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 $\text{comm}_l = 1$

Problem instances (1/2)

- Goal: place replicas to process client requests
- Client $i \in \mathcal{C}$: Servers(i) $\subseteq \mathcal{N}$ set of servers responsible for processing its requests
- $r_{i,s}$: number of requests from client i processed by server s $(\sum_{s \in \text{Servers}(i)} r_{i,s} = r_i)$
- $R = \{s \in \mathcal{N} | \; \exists i \in C \; , \; s \in \text{Servers}(i)\}$: set of replicas

Problem instances (2/2)

- Minimize $\sum_{s \in R} sc_s$ under the constraints:
- **Server capacity** – $\forall s \in R, \sum_{i \in \mathcal{C} | s \in \text{Servers}(i)} r_{i,s} \leq \text{W}_s$
- **QoS** – $\forall i \in \mathcal{C}, \forall s \in \text{Servers}(i), \sum_{l \in \text{path}[i \to s]} \text{comm}_l \leq \text{q}_i$.

- Restrict to case where $\text{sc}_s = \text{W}_s$:
  Replica Counting problem on homogeneous platforms,
  Replica Cost problem with heterogeneous servers.

Problem instances (2/2)

- Minimize $\sum_{s \in R} \mathrm{sc}_s$ under the constraints:
- **Server capacity** – $\forall s \in R, \sum_{i \in \mathcal{C} | s \in \mathrm{Servers}(i)} r_{i,s} \leq \mathrm{W}_s$
- **QoS** – $\forall i \in \mathcal{C}, \forall s \in \mathrm{Servers}(i), \sum_{l \in \mathrm{path}[i \to s]} \mathrm{comm}_l \leq \mathrm{q}_i$.

- Restrict to case where $\mathrm{sc}_s = \mathrm{W}_s$:
  Replica Counting problem on homogeneous platforms,
  Replica Cost problem with heterogeneous servers.

Problem instances (2/2)

- Minimize $\sum_{s \in R} \text{sc}_s$ under the constraints:
- **Server capacity** – $\forall s \in R, \sum_{i \in \mathcal{C} | s \in \text{Servers}(i)} r_{i,s} \leq \text{W}_s$
- **QoS** – $\forall i \in \mathcal{C}, \forall s \in \text{Servers}(i), \sum_{l \in \text{path}[i \rightarrow s]} \text{comm}_l \leq \text{q}_i$.

- Restrict to case where $\text{sc}_s = \text{W}_s$:
  Replica Counting problem on homogeneous platforms,
  Replica Cost problem with heterogeneous servers.

Access policies

Single server – Each client i is assigned a single server server(i), that is responsible for processing all its requests. *Upwards* policy.

Single server policy *Closest* with additional constraint: server of client i is constrained to be first server found on the path that goes from i upwards to the tree root.

Multiple servers – A client i may be assigned several servers in a set Servers(i). Each server $s \in$ Servers(i) will handle a fraction $r_{i,s}$ of the requests. *Multiple* policy.

Access policies

Single server – Each client i is assigned a single server server(i), that is responsible for processing all its requests. *Upwards* policy.

Single server policy *Closest* with additional constraint: server of client i is constrained to be first server found on the path that goes from i upwards to the tree root.

Multiple servers – A client i may be assigned several servers in a set Servers(i). Each server $s \in$ Servers(i) will handle a fraction $r_{i,s}$ of the requests. *Multiple* policy.

Access policies

Single server – Each client i is assigned a single server server(i), that is responsible for processing all its requests. *Upwards* policy.

Single server policy *Closest* with additional constraint: server of client i is constrained to be first server found on the path that goes from i upwards to the tree root.

Multiple servers – A client i may be assigned several servers in a set Servers(i). Each server $s \in$ Servers(i) will handle a fraction $r_{i,s}$ of the requests. *Multiple* policy.

Outline

1. Framework
2. Complexity results
3. Linear programming formulation
4. Heuristics
5. Experiments
6. Conclusion

Complexity results

Homogeneous platform: Replica Counting problem

	No QoS	With QoS
Closest		
Upwards		
Multiple		

- Liu et al.: *Closest* remains polynomial with QoS
- Benoit et al. (HCW'07): *Upwards* is NP-complete even without QoS, and *Multiple* is polynomial without QoS
- New result: *Multiple* becomes NP-complete with QoS (reduction from 2-Partition)

Complexity results

Homogeneous platform: REPLICA COUNTING problem

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

- Liu et al.: *Closest* remains polynomial with QoS
- Benoit et al. (HCW'07): *Upwards* is NP-complete even without QoS, and *Multiple* is polynomial without QoS
- New result: *Multiple* becomes NP-complete with QoS (reduction from 2-Partition)

Complexity results

Homogeneous platform: Replica Counting problem

	No QoS	With QoS
Closest	polynomial [Cidon02,Liu06]	polynomial [Liu06]
Upwards	NP-complete [Be07]	NP-complete [Be07]
Multiple	polynomial [Be07]	

- Liu et al.: *Closest* remains polynomial with QoS
- Benoit et al. (HCW'07): *Upwards* is NP-complete even without QoS, and *Multiple* is polynomial without QoS
- New result: *Multiple* becomes NP-complete with QoS (reduction from 2-Partition)

Complexity results

Homogeneous platform: REPLICA COUNTING problem

	No QoS	**With QoS**
Closest	polynomial [Cidon02,Liu06]	polynomial [Liu06]
Upwards	NP-complete [Be07]	NP-complete [Be07]
Multiple	polynomial [Be07]	NP-complete

- Liu et al.: *Closest* remains polynomial with QoS
- Benoit et al. (HCW'07): *Upwards* is NP-complete even without QoS, and *Multiple* is polynomial without QoS
- New result: *Multiple* becomes NP-complete with QoS (reduction from 2-Partition)

Complexity results

Homogeneous platform: Replica Counting problem

	No QoS	With QoS
Closest	polynomial [Cidon02,Liu06]	polynomial [Liu06]
Upwards	NP-complete [Be07]	NP-complete [Be07]
Multiple	polynomial [Be07]	NP-complete

- Liu et al.: *Closest* remains polynomial with QoS
- Benoit et al. (HCW'07): *Upwards* is NP-complete even without QoS, and *Multiple* is polynomial without QoS
- New result: *Multiple* becomes NP-complete with QoS (reduction from 2-Partition)

Heterogeneous platforms: all problems are NP-complete

Outline

Linear programming

- General instance of the problem: Heterogeneous platform, QoS, *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 $x_j = 0.5$ → optimal solution for *Multiple*

Linear programming

- General instance of the problem: Heterogeneous platform, QoS, *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 $x_j = 0.5$ → optimal solution for *Multiple*

Linear programming

- General instance of the problem: Heterogeneous platform, QoS, *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, *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 $x_j = 0.5$ → optimal solution for *Multiple*

Outline

1. Framework
2. Complexity results
3. Linear programming formulation
4. Heuristics
5. Experiments
6. Conclusion

Heuristics

- Polynomial heuristics for the REPLICA COST problem
  - Heterogeneous platforms
  - QoS constraints: QoS of client i represents the maximum distance (number of hops) between i and server(i)
- Experimental assessment of the impact of QoS constraints on performance
- Sorted lists of clients or servers: trade-off between large r_i and small q_i
- Worst case complexity $O(s^2)$, where $s = |\mathcal{C}| + |\mathcal{N}|$ is the problem size

Heuristics

- Polynomial heuristics for the REPLICA COST problem
  - Heterogeneous platforms
  - QoS constraints: QoS of client i represents the maximum distance (number of hops) between i and server(i)
- Experimental assessment of the impact of QoS constraints on performance
- Sorted lists of clients or servers: trade-off between large r_i and small q_i
- Worst case complexity $O(s^2)$, where $s = |\mathcal{C}| + |\mathcal{N}|$ is the problem size

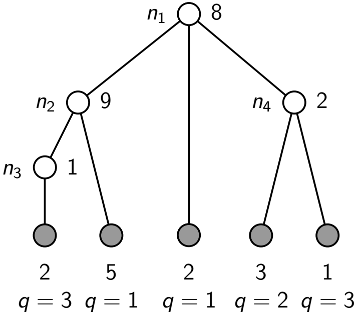
Heuristics

- Polynomial heuristics for the Replica Cost problem
  - Heterogeneous platforms
  - QoS constraints: QoS of client i represents the maximum distance (number of hops) between i and server(i)
- Experimental assessment of the impact of QoS constraints on performance
- Sorted lists of clients or servers: trade-off between large r_i and small q_i
- Worst case complexity $O(s^2)$, where $s = |\mathcal{C}| + |\mathcal{N}|$ is the problem size

Heuristics for *Closest*

Closest Big Subtree First **CBSF**

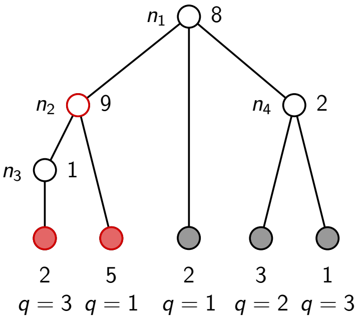
- Traversal of the tree, treating subtrees that contain most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added



Heuristics for *Closest*

Closest Big Subtree First **CBSF**

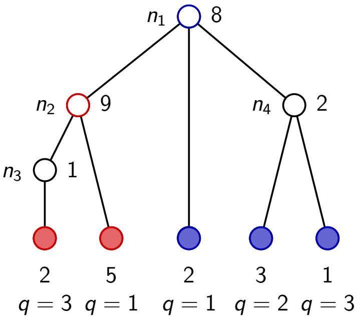
- Traversal of the tree, treating subtrees that contain most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added



Heuristics for *Closest*

Closest Big Subtree First **CBSF**

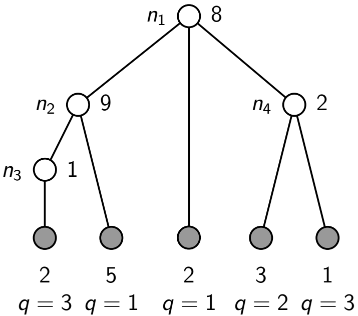
- Traversal of the tree, treating subtrees that contain most requests first
- When a node can process the requests of all the clients in its subtree, node chosen as a server and traversal stopped
- Procedure called until no more servers are added



Heuristics for *Upwards*

Upwards Small QoS **USQoS**

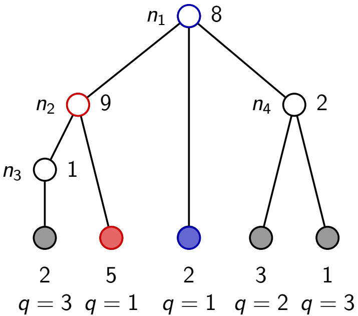
- Treating clients in non-decreasing order of QoS
- Choosing appropriate server: several variants



Heuristics for *Upwards*

Upwards Small QoS **USQoS**

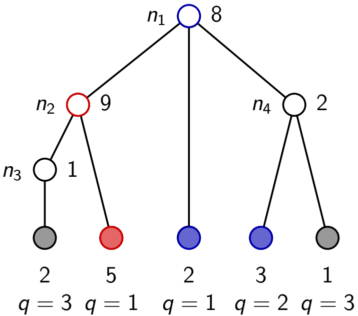
- Treating clients in non-decreasing order of QoS
- Choosing appropriate server: several variants



Heuristics for *Upwards*

Upwards Small QoS **USQoS**

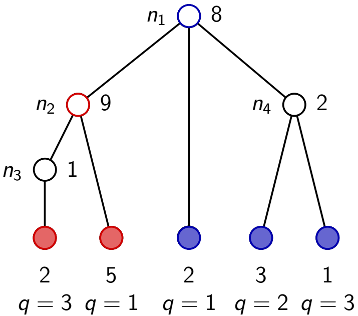
- Treating clients in non-decreasing order of QoS
- Choosing appropriate server: several variants



Heuristics for *Upwards*

Upwards Small QoS **USQoS**

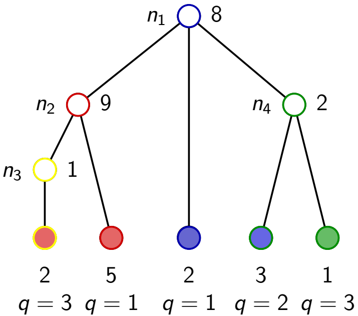
- Treating clients in non-decreasing order of QoS
- Choosing appropriate server: several variants



Heuristics for *Multiple*

Multiple MinQoS Indisp **MMQoSI**

- Choose indispensable servers
- Sort servers by non-decreasing value of reachable request numbers
- Delete clients requests by $\min(QoS, dist(root))$



Outline

Plan of experiments

- Assess impact of different access policies
- Assess impact of QoS constraints on the performance
- Important parameter:

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

- 30 trees for each $\lambda = 0.1, 0.2, ..., 0.9$
- Problem size $s = |\mathcal{C}| + |\mathcal{N}|$ such that $15 \leq s \leq 400$
- Computation of the LP optimal solution for each tree

Plan of experiments

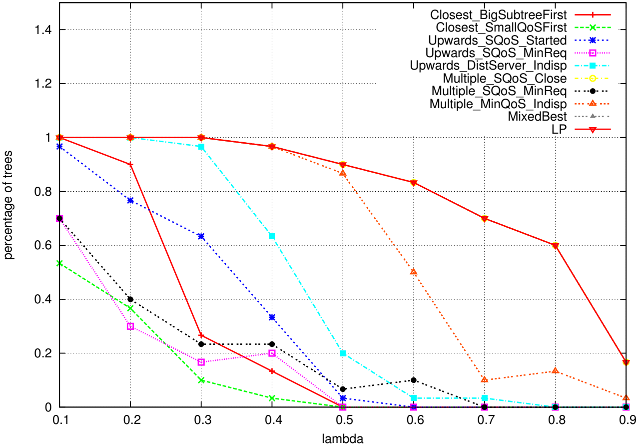
- Assess impact of different access policies
- Assess impact of QoS constraints on the performance
- Important parameter:

$$\lambda = \frac{\sum_{i \in \mathcal{C}} r_i}{\sum_{j \in \mathcal{N}} \mathsf{W}_i}$$

- 30 trees for each $\lambda = 0.1, 0.2, ..., 0.9$
- Problem size $s = |\mathcal{C}| + |\mathcal{N}|$ such that $15 \leq s \leq 400$
- Computation of the LP optimal solution for each tree

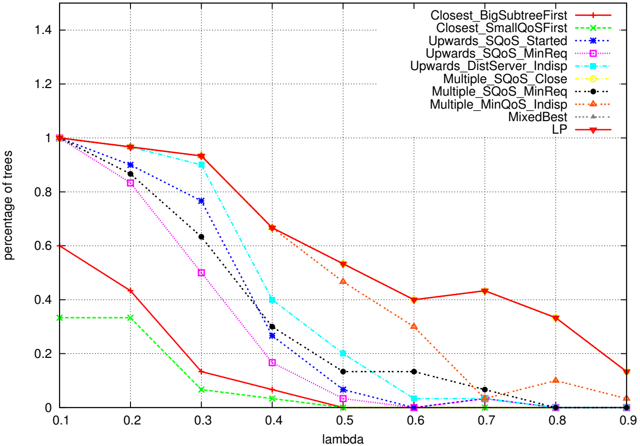
Results - Percentage of success

- Number of solutions for each lambda and each heuristic
- $qos = height + 1 \longrightarrow$ no qos



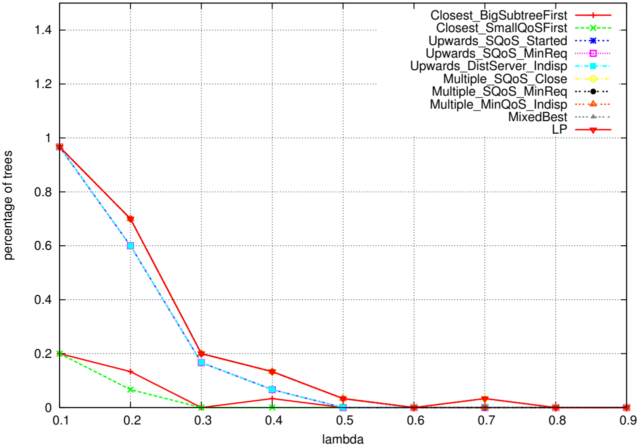
Results - Percentage of success

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



Results - Percentage of success

- Number of solutions for each lambda and each heuristic
- $qos \in \{1, 2\}$



Results - Solution cost

- 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 cost:

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

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

Results - Solution cost

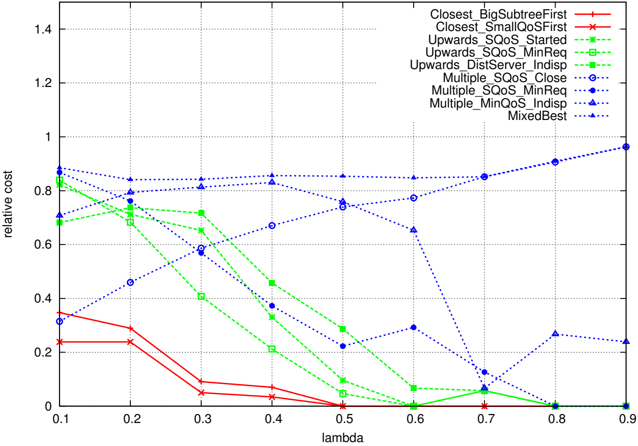
- Distance of the result (in terms of replica cost) of the heuristic to the optimal solution
- T_λ: subset of trees with a solution
- Relative cost:

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

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

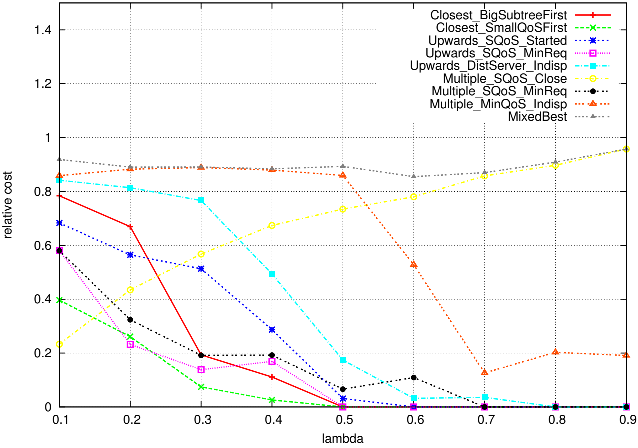
Results - Solution cost

$average(qos) = height/2$



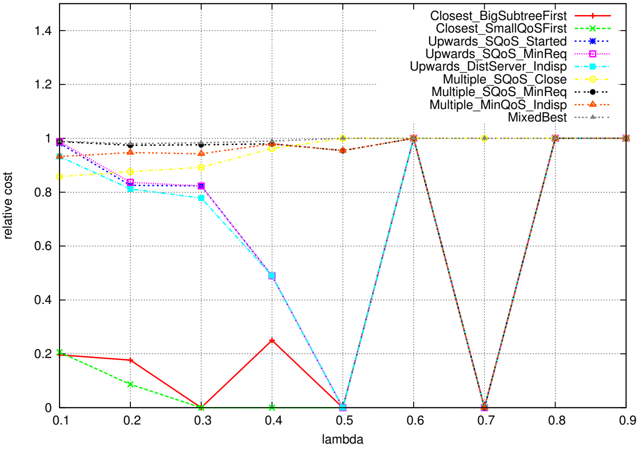
Results - Solution cost

$qos = height + 1 \longrightarrow$ no qos



Results - Solution cost

$qos \in \{1, 2\}$



Summary

- *Multiple* $\geq$ *Upwards* $\geq$ *Closest*:
  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

- Good performance of the heuristics for strongly to loosely constrained trees
- The trade-off worked well ☺

Summary

- *Multiple* $\geq$ *Upwards* $\geq$ *Closest*: 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
- Good performance of the heuristics for strongly to loosely constrained trees
- The trade-off worked well ☺

Summary

- *Multiple* $\geq$ *Upwards* $\geq$ *Closest*: hierarchy also under QoS constraints
- Performance compared to the optimal solution:

$$
\begin{aligned}
qos \in \{1,2\}: &\quad 95\% \\
average(qos) = height/2: &\quad 85\% \\
\text{no qos}: &\quad 85\%
\end{aligned}
$$

- Smaller trees: results slightly less good

- Good performance of the heuristics for strongly to loosely constrained trees
- The trade-off worked well ☺

Outline

Conclusion

Theoretical side

- Complexity of each policy, for homogeneous and heterogeneous platforms, with and without QoS
- NP-completeness of *Multiple* +QoS on homogeneous platforms

Practical side

- Design of several heuristics for each policy, taking QoS into account
- Striking impact of the policy on the result
- Use of a LP-based optimal solution to assess the absolute performance, which turns out to be quite good.

Future work

Short term

- More simulations for the REPLICA COST problem: shape of the trees, distribution law of the requests, degree of heterogeneity of the platforms
- Add bandwidth constraints (another trade-off) to the heuristics

Longer term

- Consider the problem with several object types
- Extension with more complex objective functions