Service Chain Placement in SDNs

Gilad Kutiel [1] Dror Rawitz [2]

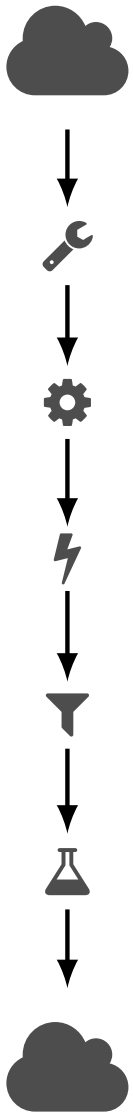
[1]Technion

[2]Bar Ilan University

ALGOCLOUD 2017

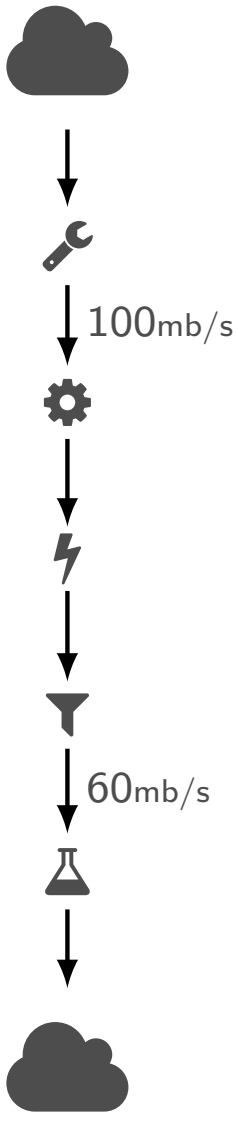
Service Chain (Customer's Perspective)

- An ordered set of Virtual Functions



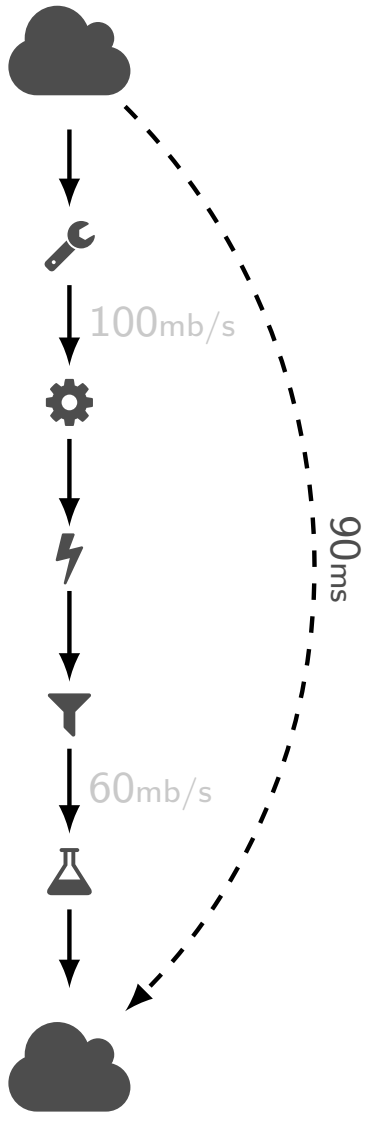
Service Chain (Customer's Perspective)

- An ordered set of Virtual Functions
- Bandwidth requirement



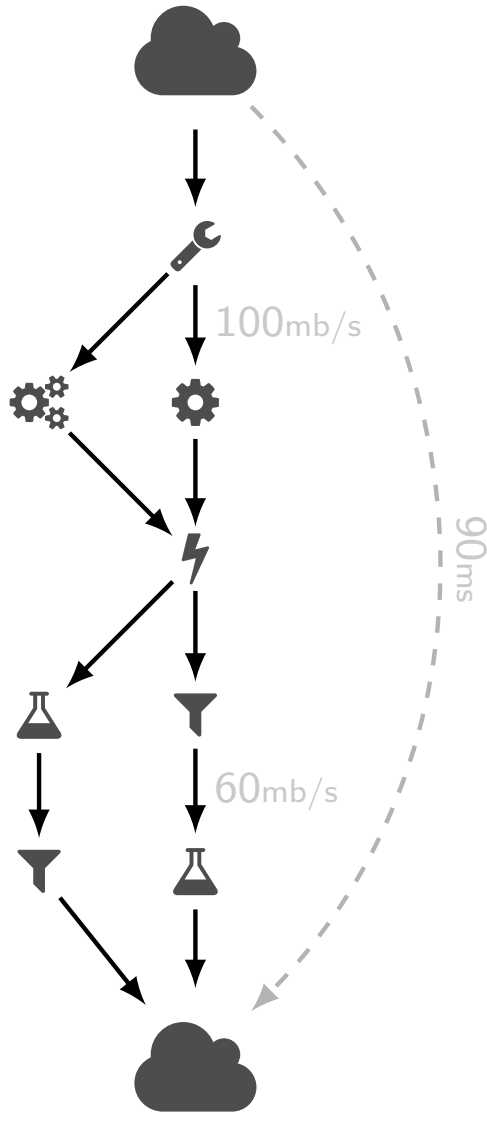
Service Chain (Customer's Perspective)

- An ordered set of Virtual Functions
- Bandwidth requirement
- Latency constraint



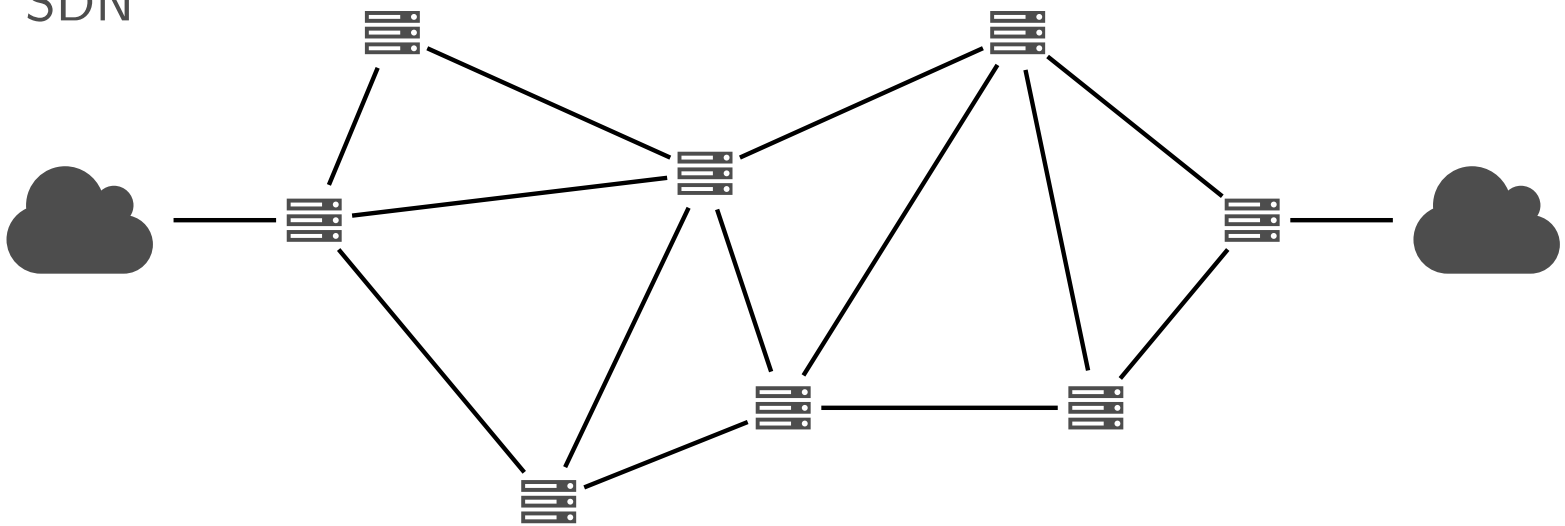
Service Chain (Customer's Perspective)

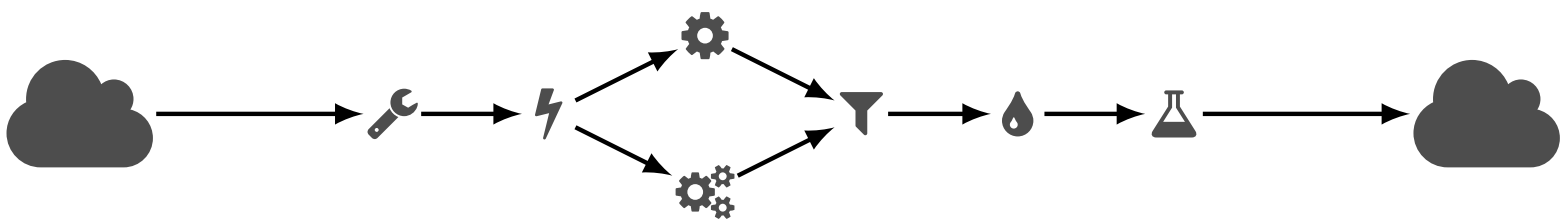
- An ordered set of Virtual Functions
- Bandwidth requirement
- Latency constraint
- Several alternatives



Problem (ISP's Perspective)

SDN

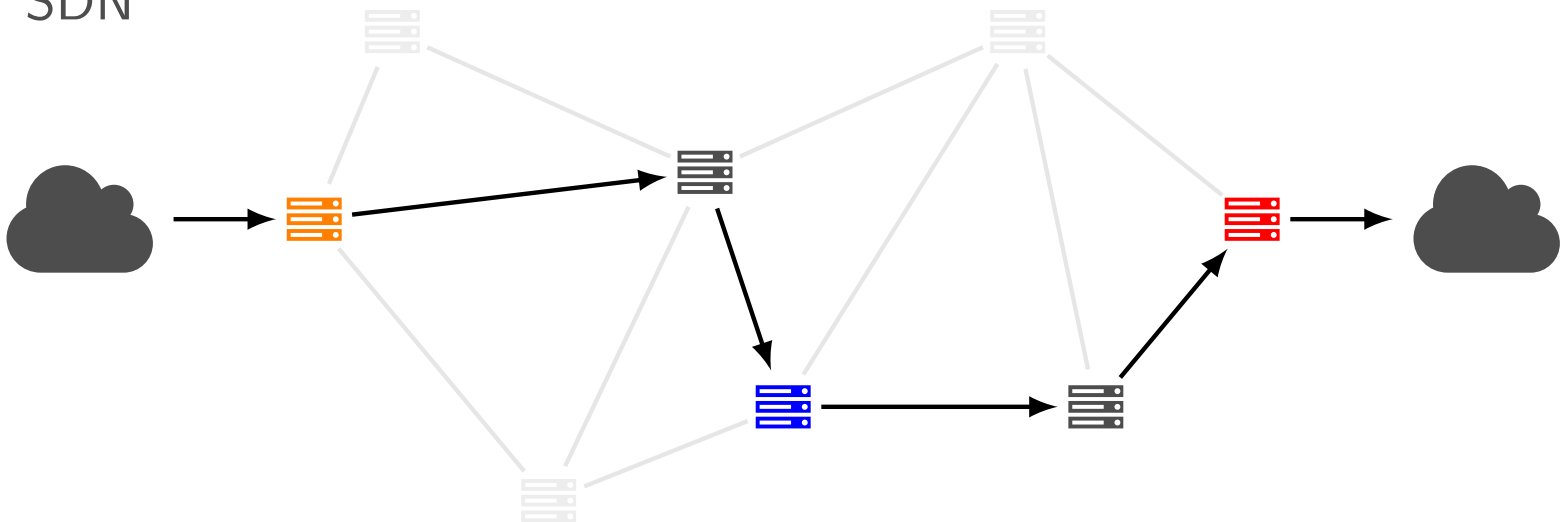




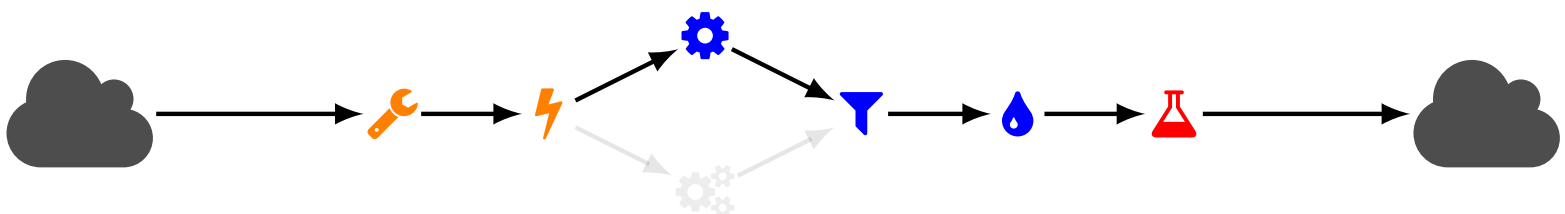
Service Chain

Problem (ISP's Perspective)

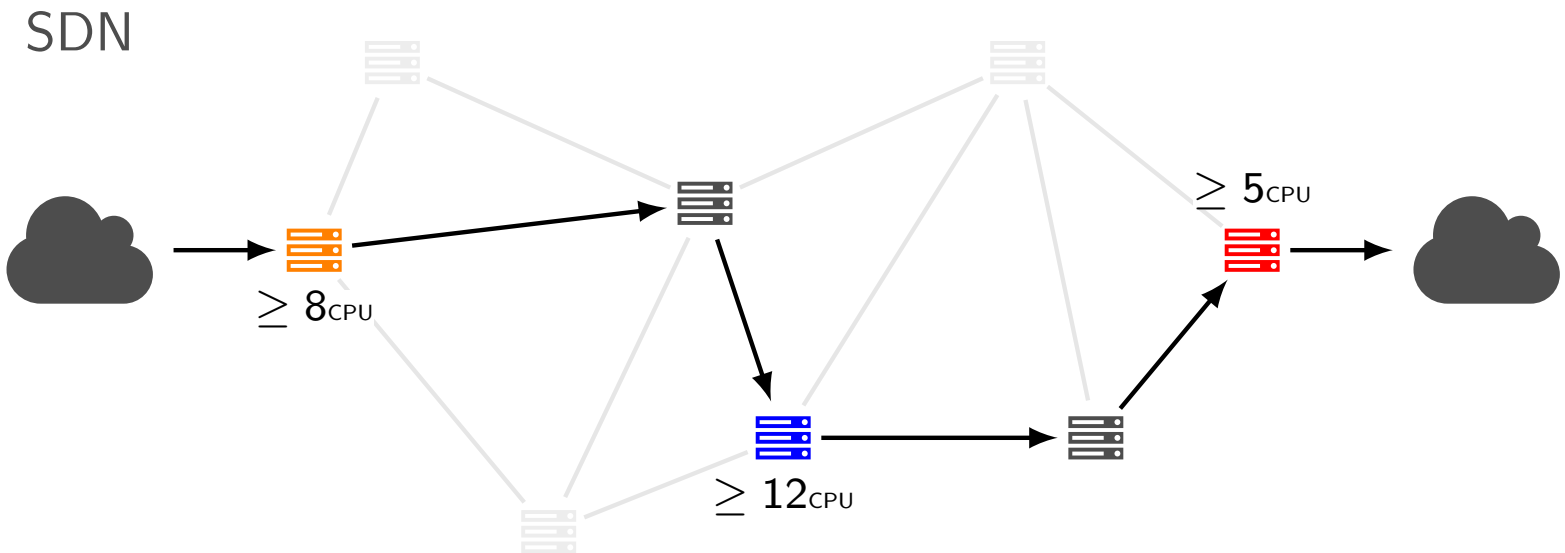
SDN



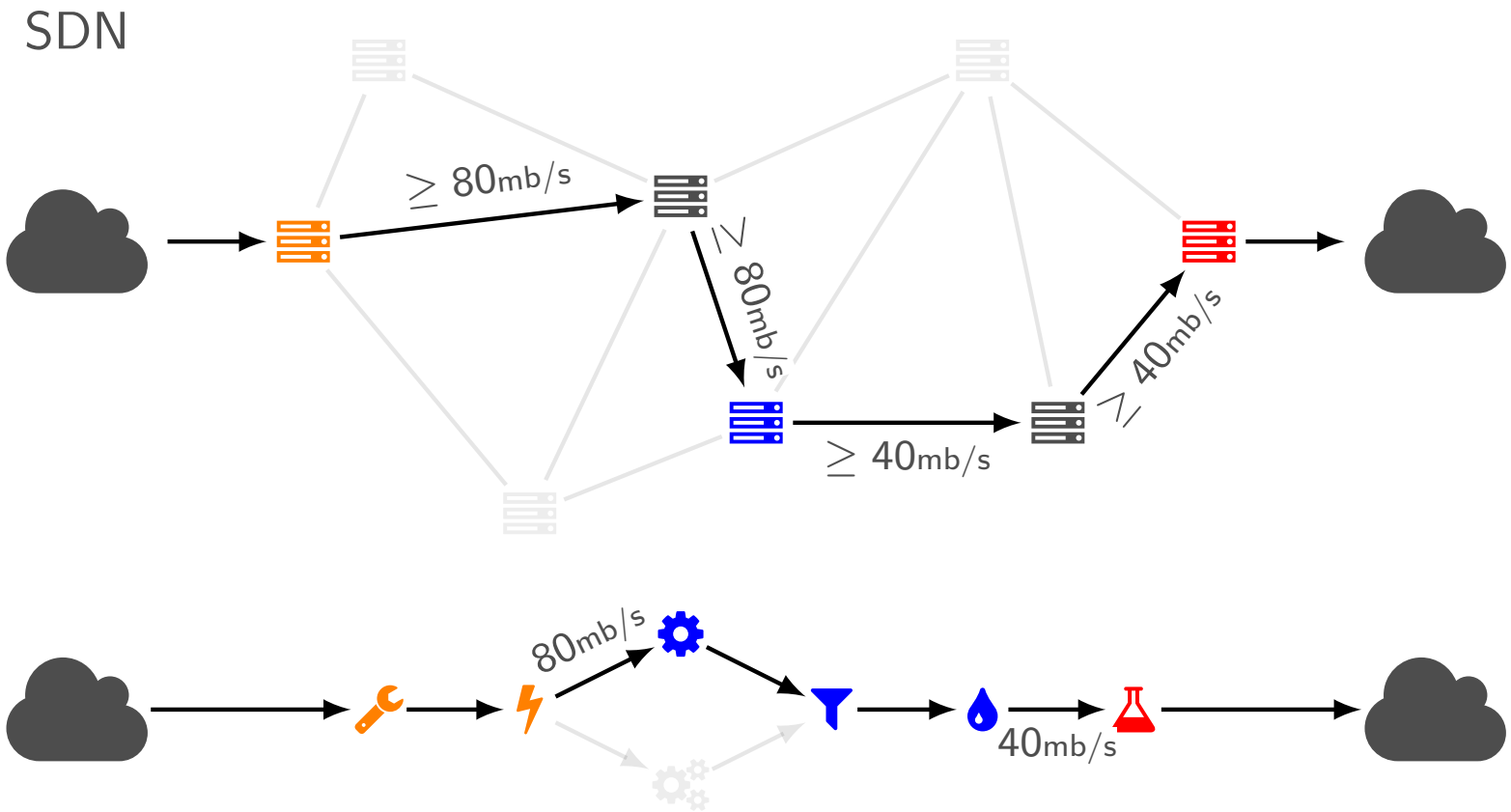
Service Chain



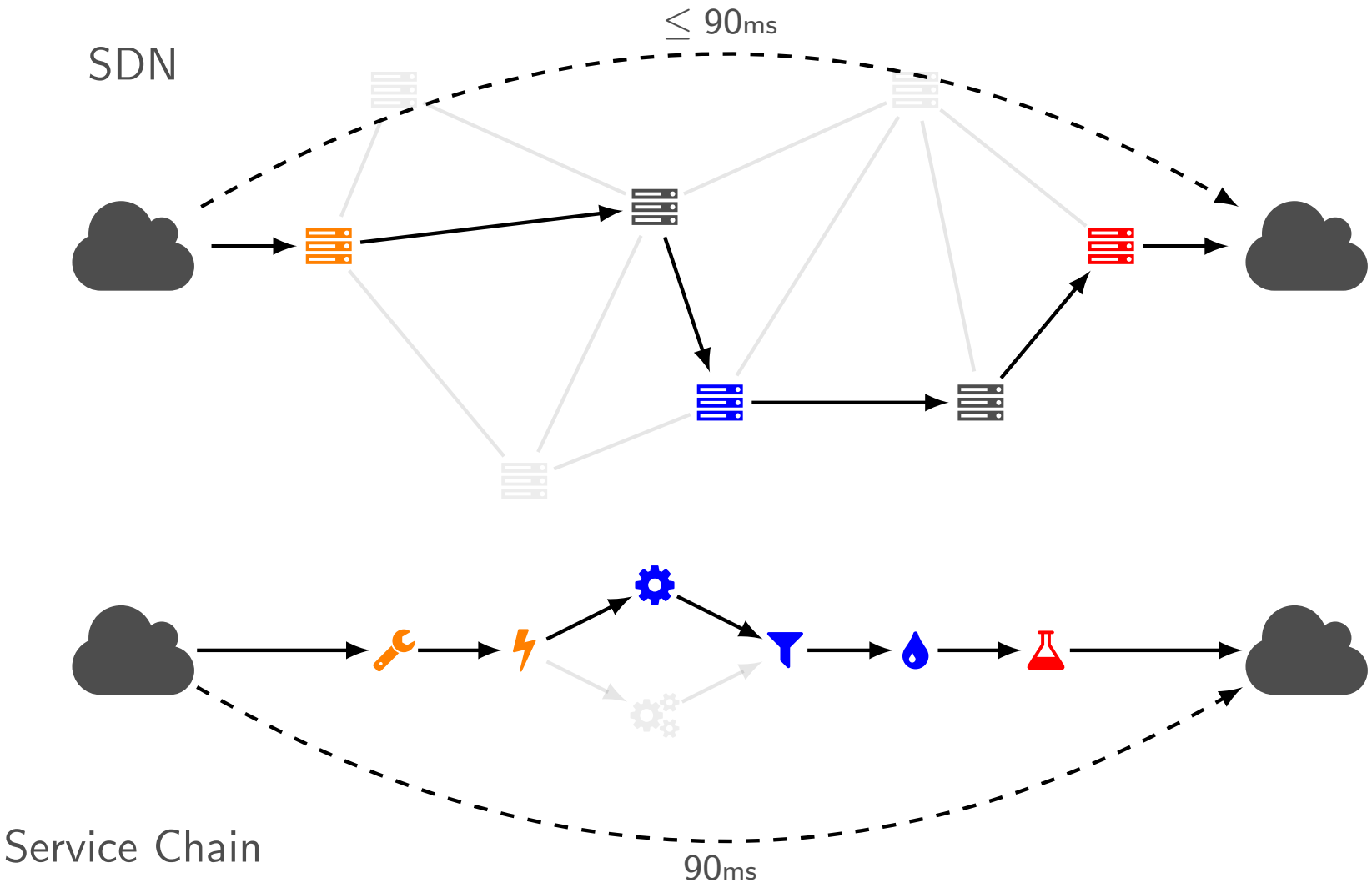
Problem (ISP's Perspective)



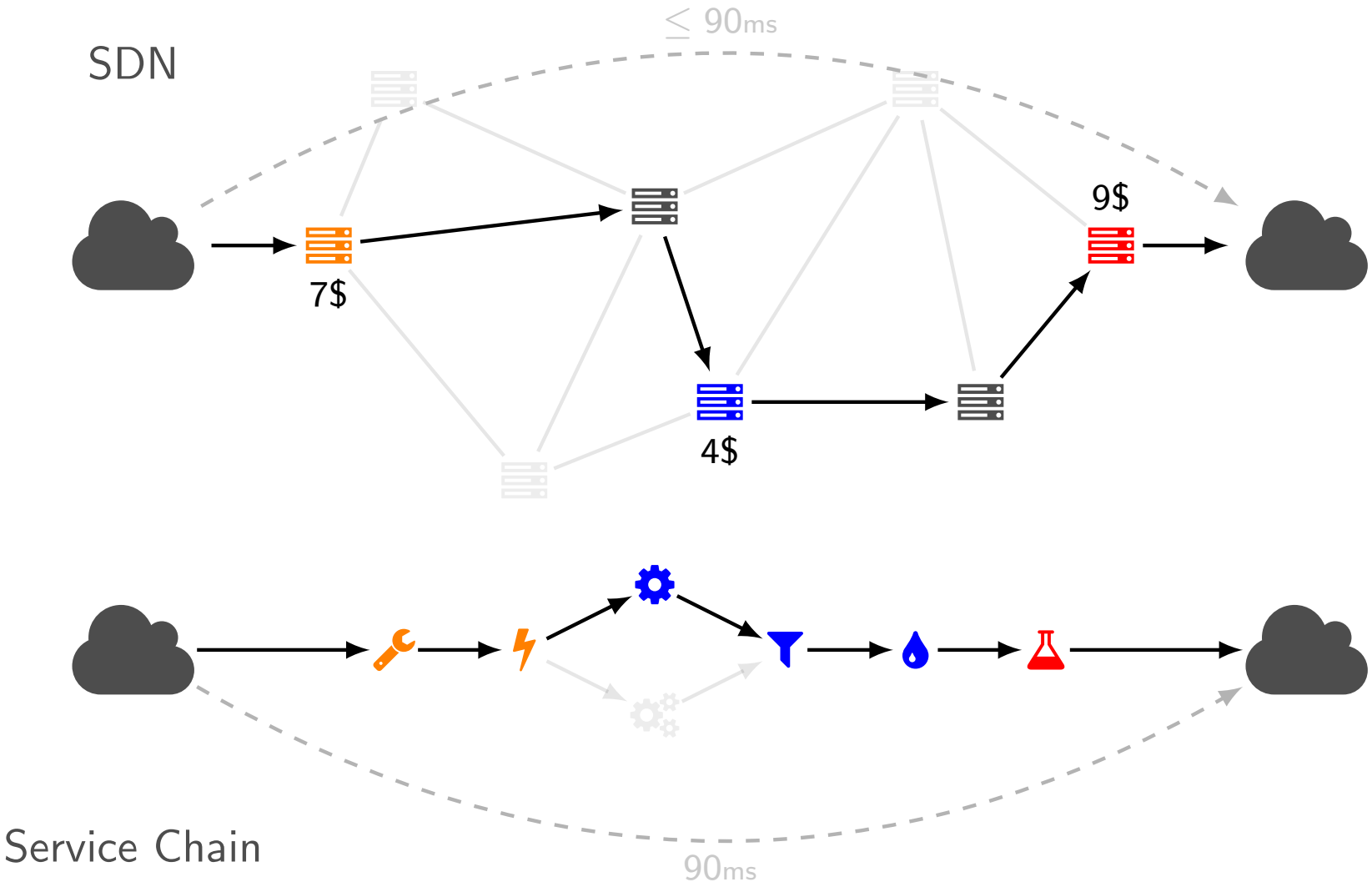
Problem (ISP's Perspective)



Problem (ISP's Perspective)



Problem (ISP's Perspective)



Problem cont.

Minimize Placement Cost

s.t.

- CPU
- Bandwidth
- Latency

Related Work

- System

Related Work

- System
  - A. Gember-Jacobson et al. 2014

Related Work

- System
  - A. Gember-Jacobson et al. 2014
  - R. Hartert et al. 2015

Related Work

- System
  - A. Gember-Jacobson et al. 2014
  - R. Hartert et al. 2015
- Online Service Chain Embedding Problem

Related Work

- System
  - A. Gember-Jacobson et al. 2014
  - R. Hartert et al. 2015
- Online Service Chain Embedding Problem
  - T. Lukovszki and S. Schmid 2015

Related Work

- System
  - A. Gember-Jacobson et al. 2014
  - R. Hartert et al. 2015
- Online Service Chain Embedding Problem
  - T. Lukovszki and S. Schmid 2015
  - Even et al. 2016

Related Work

- System
  - A. Gember-Jacobson et al. 2014
  - R. Hartert et al. 2015
- Online Service Chain Embedding Problem
  - T. Lukovszki and S. Schmid 2015
  - Even et al. 2016
- Different Model / Objective

Related Work

- System
  - A. Gember-Jacobson et al. 2014
  - R. Hartert et al. 2015
- Online Service Chain Embedding Problem
  - T. Lukovszki and S. Schmid 2015
  - Even et al. 2016
- Different Model / Objective
  - Even et al. 2016

Related Work

- System
  - A. Gember-Jacobson et al. 2014
  - R. Hartert et al. 2015
- Online Service Chain Embedding Problem
  - T. Lukovszki and S. Schmid 2015
  - Even et al. 2016
- Different Model / Objective
  - Even et al. 2016
  - Cohen et al. 2015

Our Results

- NP-hardness in many cases

1

Our Results

- NP-hardness in many cases

SDN	Costs	Our Result

1

Our Results

- NP-hardness in many cases

SDN	Costs	Our Result
DAG	Integral, Polynomial	Optimal

1

Our Results

- NP-hardness in many cases

SDN	Costs	Our Result
DAG	Integral, Polynomial	Optimal
DAG	Any	FPTAS - $(1+\varepsilon)$-apx

1

Our Results

- NP-hardness in many cases

SDN	Costs	Our Result
DAG	Integral, Polynomial	Optimal
DAG	Any	FPTAS - $(1+\varepsilon)$-apx
Any	Any	FPTAS - $(1+\varepsilon)$-apx

1

Our Results

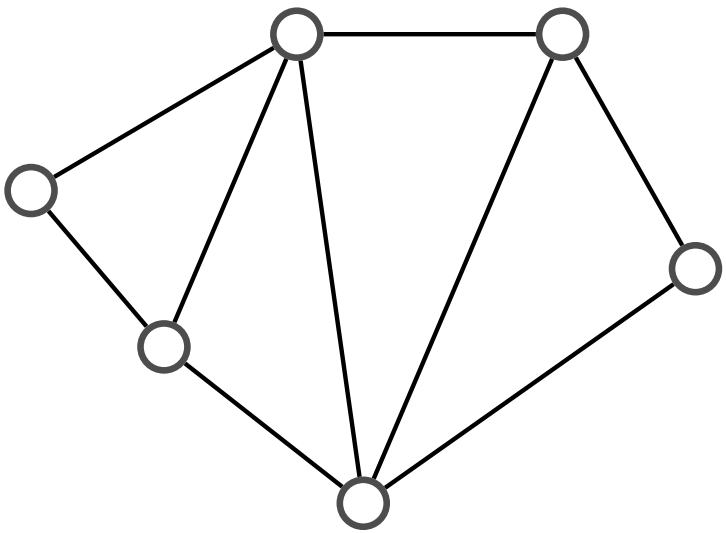
- NP-hardness in many cases

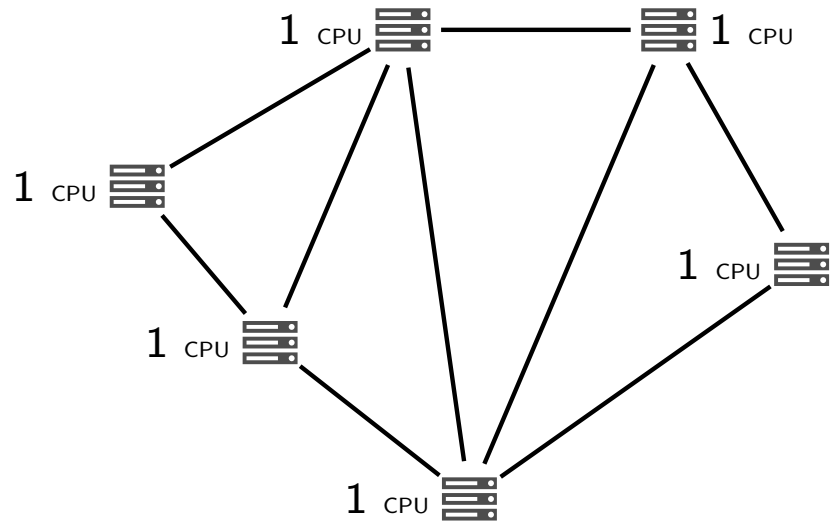
SDN	Costs	Our Result
DAG	Integral, Polynomial	Optimal
DAG	Any	FPTAS - $(1+\varepsilon)$-apx
Any	Any	FPTAS - $(1+\varepsilon)$-apx[1]

[1] running time depends on SDN's typology

Hardness (Feasibility)

Hardness (Feasibility)



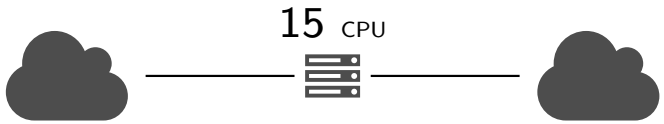


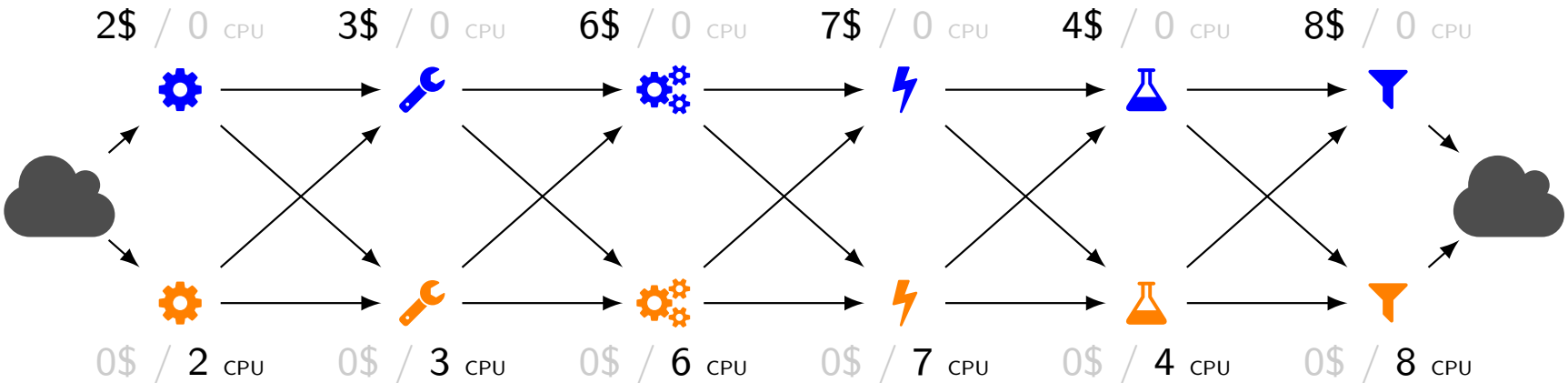
Hamiltonian path $\iff$ Feasible Placement

Hardness (SDN with a single node)

Hardness (SDN with a single node)

$$\{2, 3, 6, 7, 4, 8\}$$

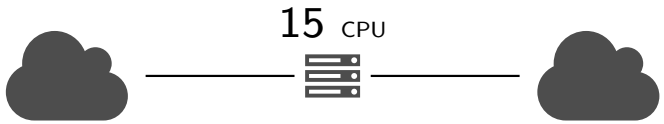


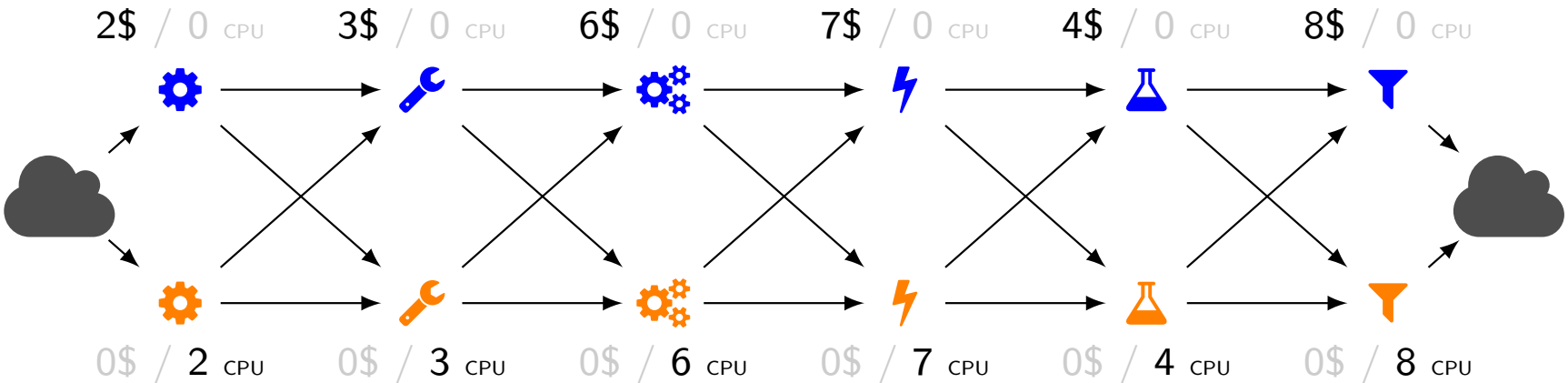


Partition $\iff$ Placement that costs 15$

Hardness (SDN with a single node)

$$\{2, 3, 6, 7, 4, 8\}$$



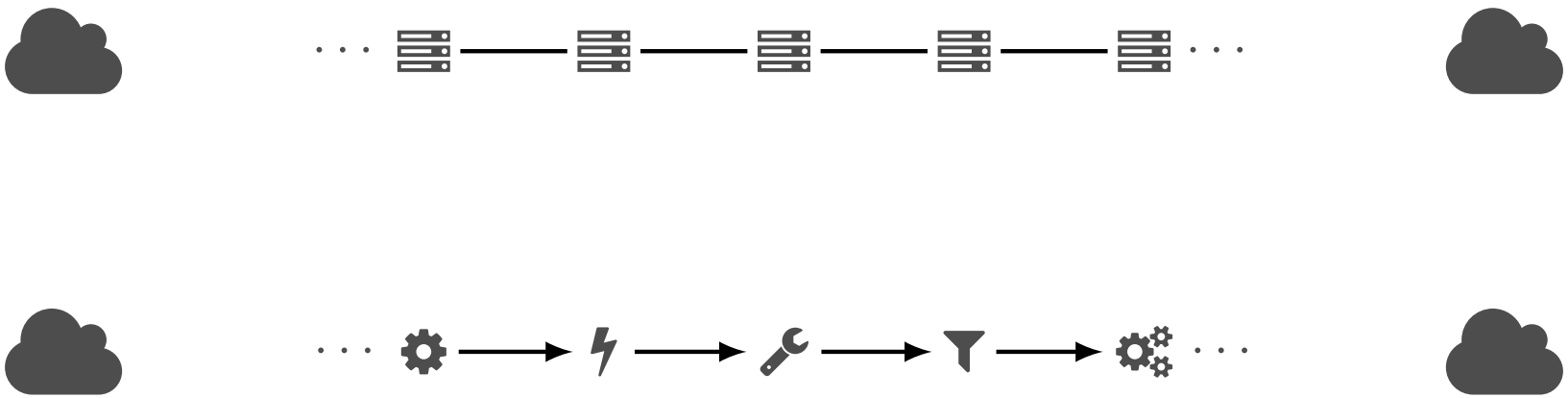


Partition $\iff$ Placement that costs 15$

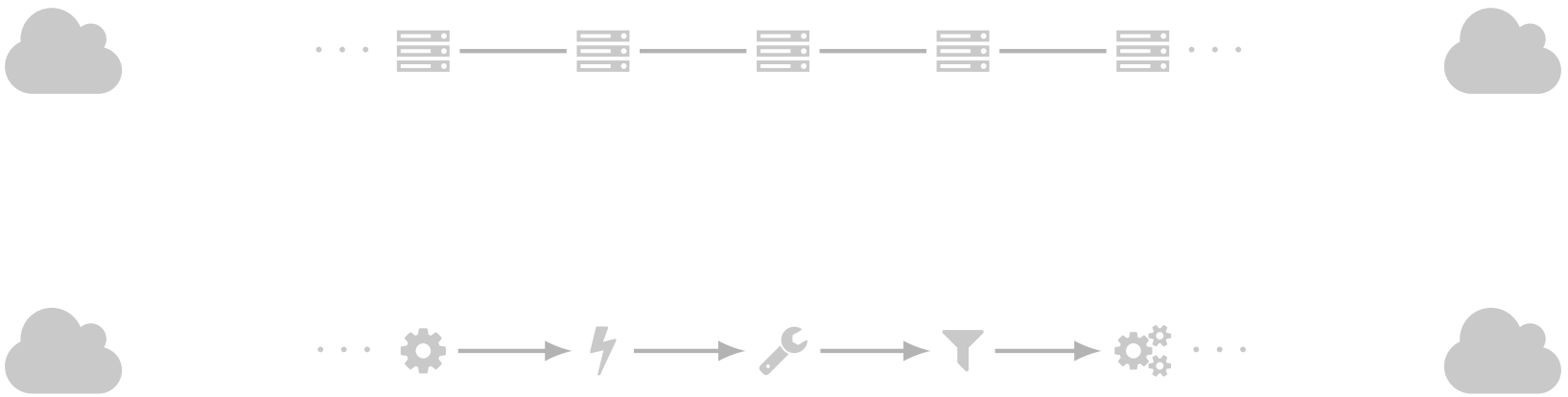
* Not hard for integral, polynomial values

Hardness (simple paths)

Hardness (simple paths)



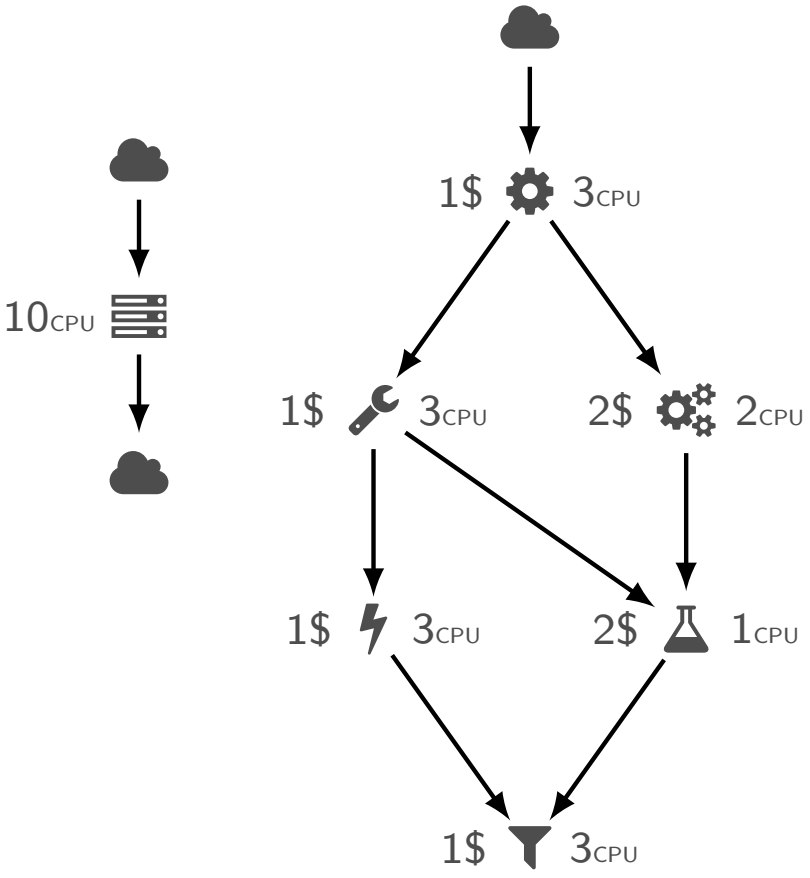
Hardness (simple paths)



See paper...

Integral Polynomial Costs, Single Node

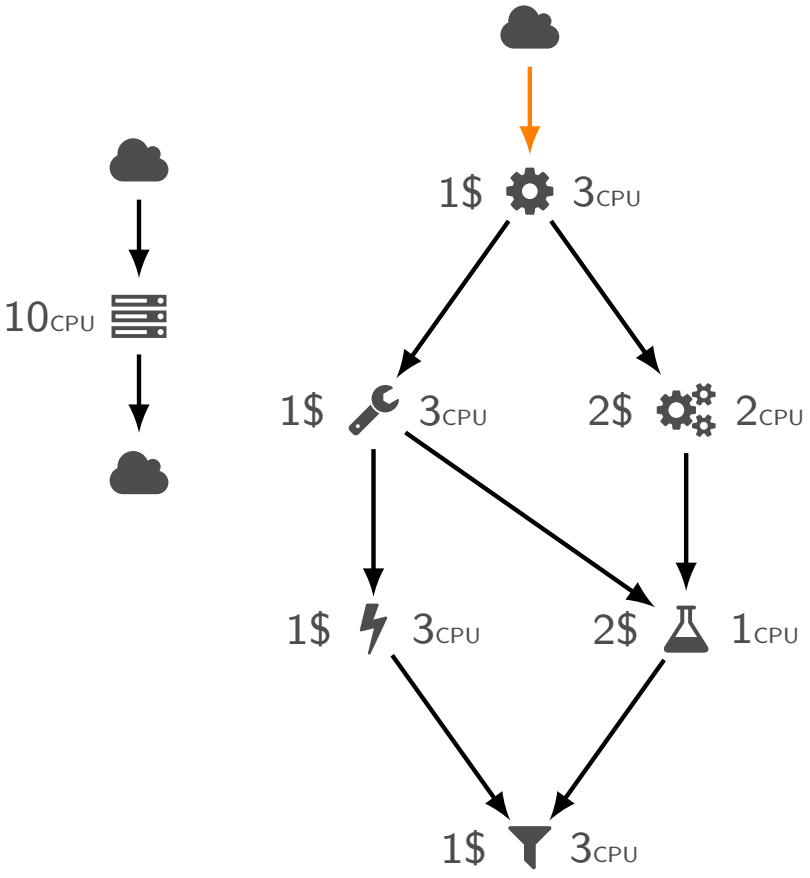
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	⏷
7	0						
6	0						
5	0						
4	0						
3	0						
2	0						
1	0						
0	0						

Integral Polynomial Costs, Single Node

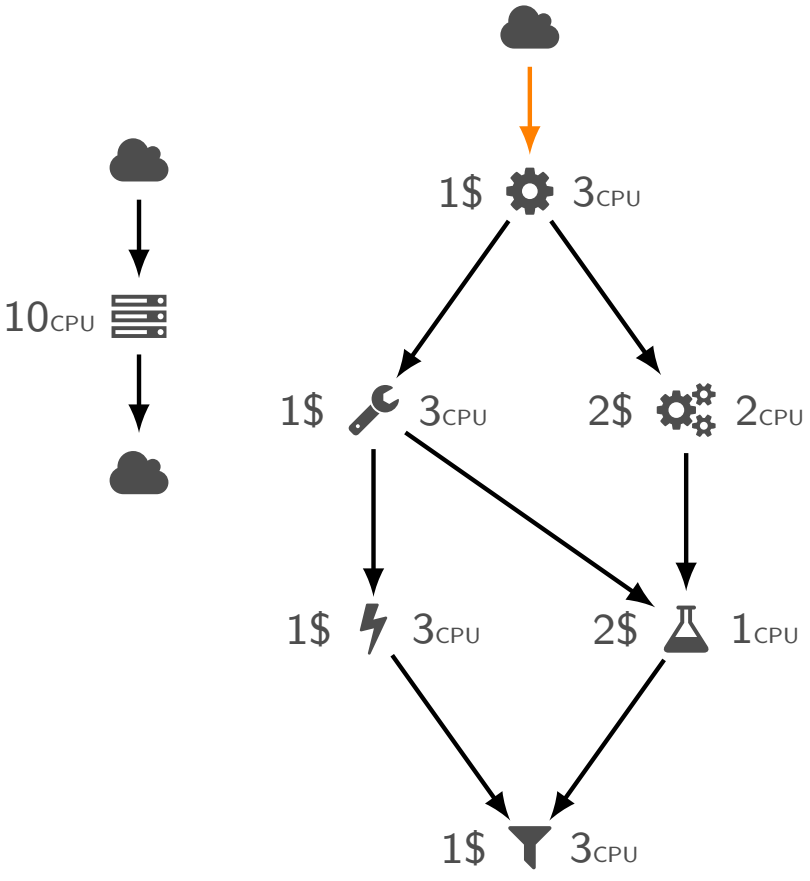
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	▼
7	0						
6	0						
5	0						
4	0						
3	0						
2	0						
1	0						
0	0						

Integral Polynomial Costs, Single Node

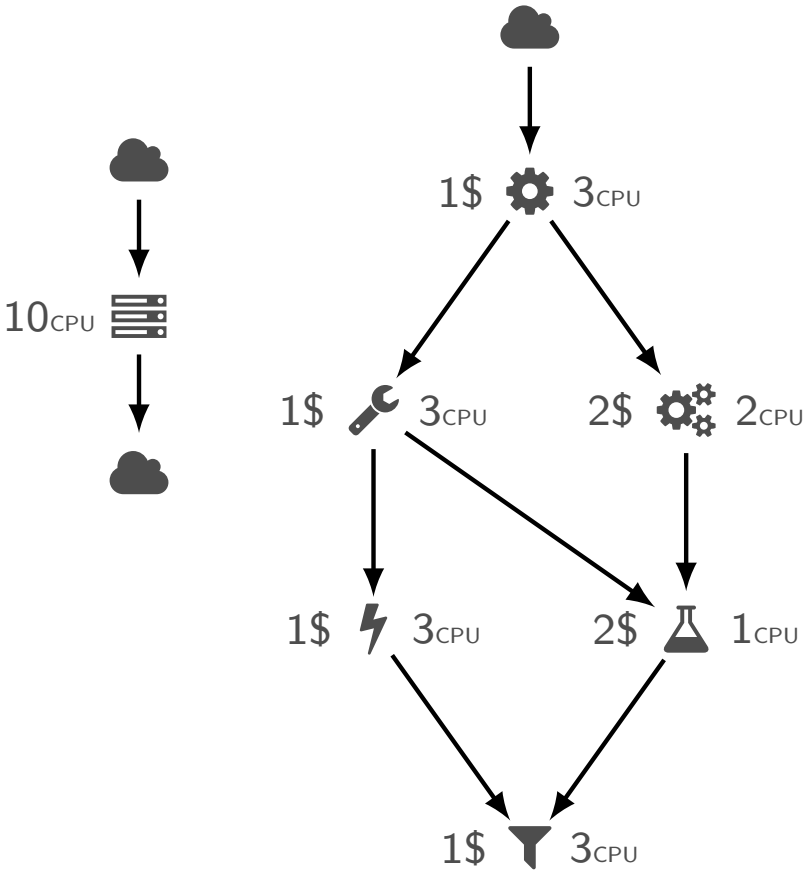
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	▼
7	0	3					
6	0						
5	0						
4	0						
3	0						
2	0						
1	0						
0	0						

Integral Polynomial Costs, Single Node

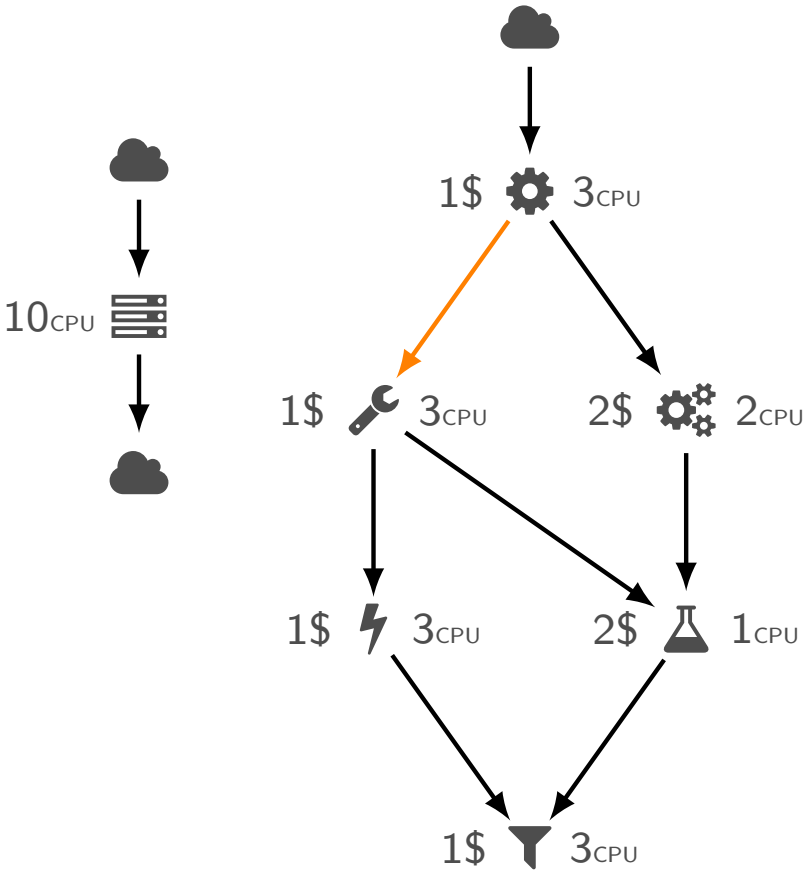
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	▼
7	0	3					
6	0	3					
5	0	3					
4	0	3					
3	0	3					
2	0	3					
1	0	3					
0	0	-					

Integral Polynomial Costs, Single Node

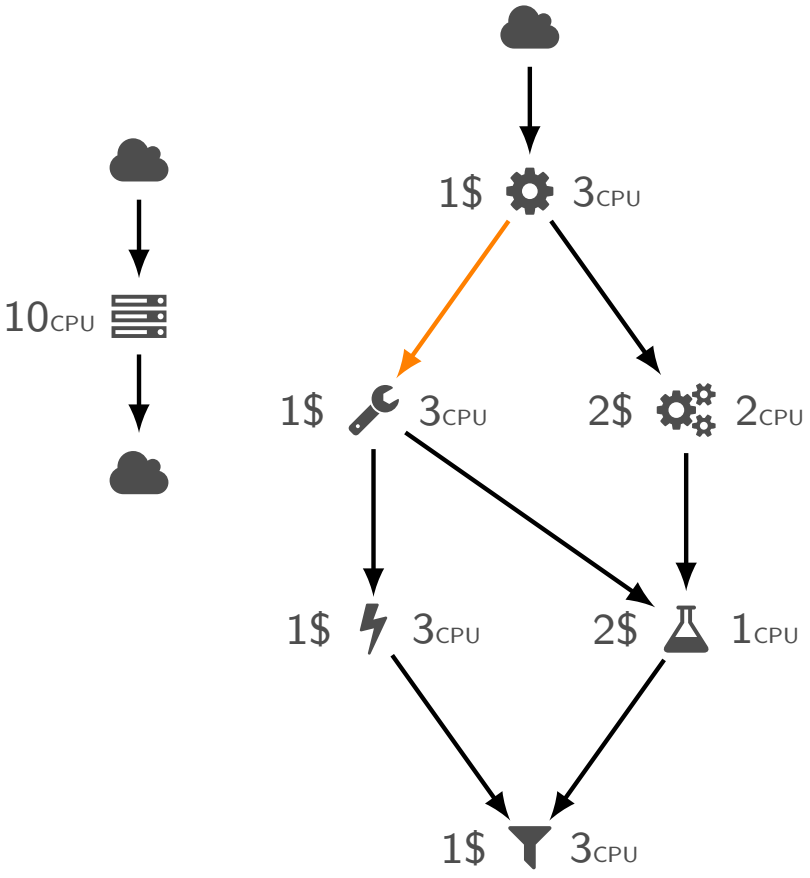
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	⏷
7	0	3					
6	0	3					
5	0	3					
4	0	3					
3	0	3					
2	0	3					
1	0	3					
0	0	-					

Integral Polynomial Costs, Single Node

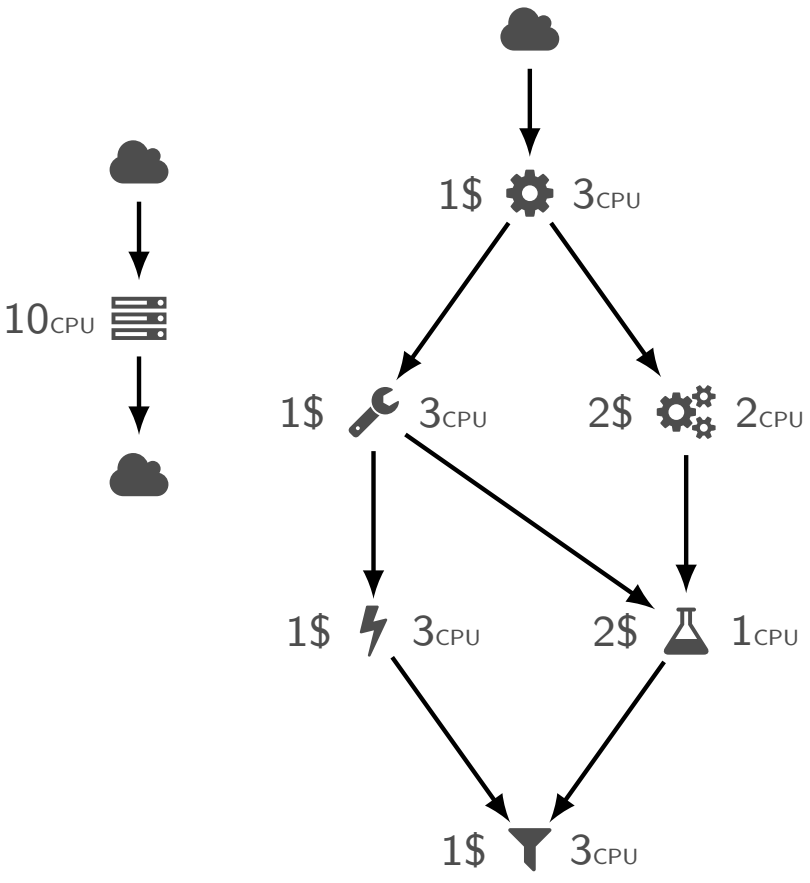
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	⏷
7	0	3	6				
6	0	3					
5	0	3					
4	0	3					
3	0	3					
2	0	3					
1	0	3					
0	0	-					

Integral Polynomial Costs, Single Node

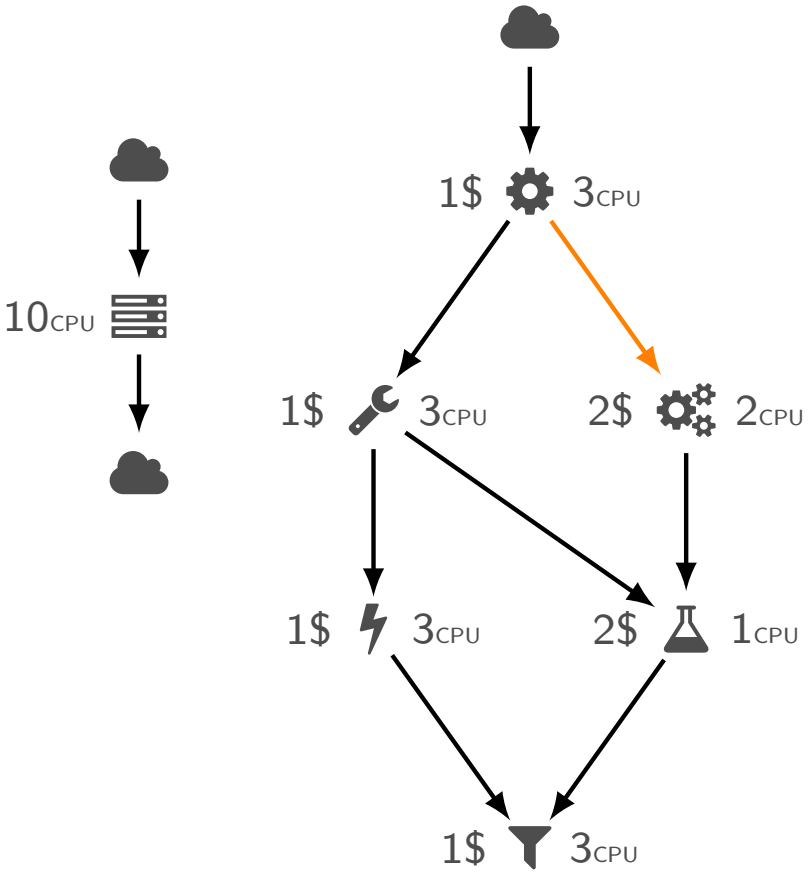
Min CPU per Cost Budget.



$							
7	0	3	6				
6	0	3	6				
5	0	3	6				
4	0	3	6				
3	0	3	6				
2	0	3	6				
1	0	3	-				
0	0	-	-				

Integral Polynomial Costs, Single Node

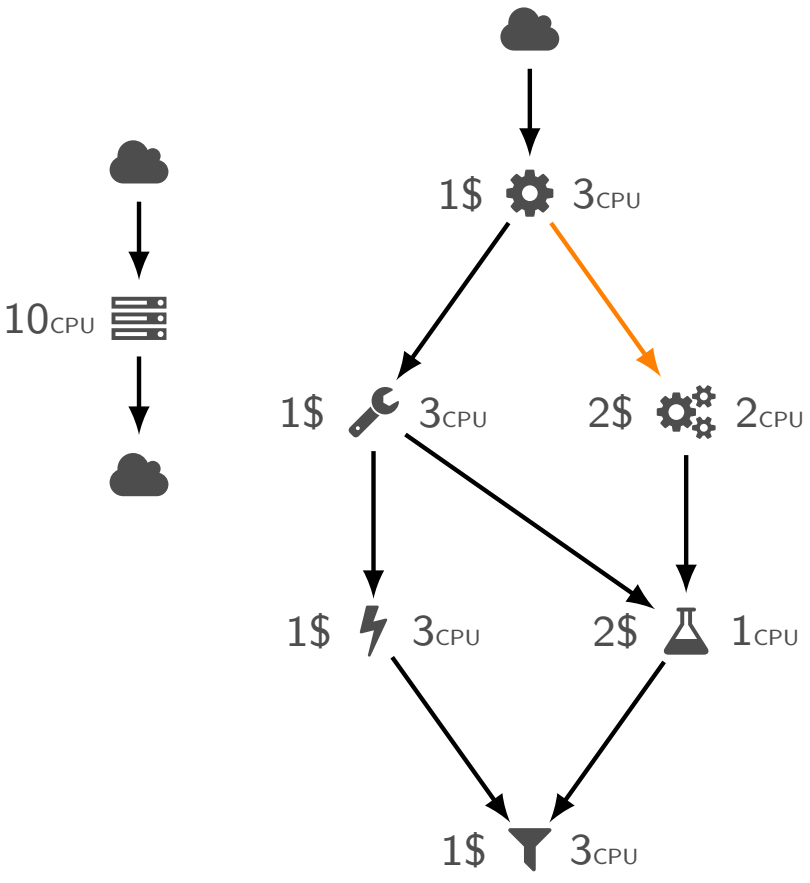
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	▼
7	0	3	6				
6	0	3	6				
5	0	3	6				
4	0	3	6				
3	0	3	6				
2	0	3	6				
1	0	3	-				
0	0	-	-				

Integral Polynomial Costs, Single Node

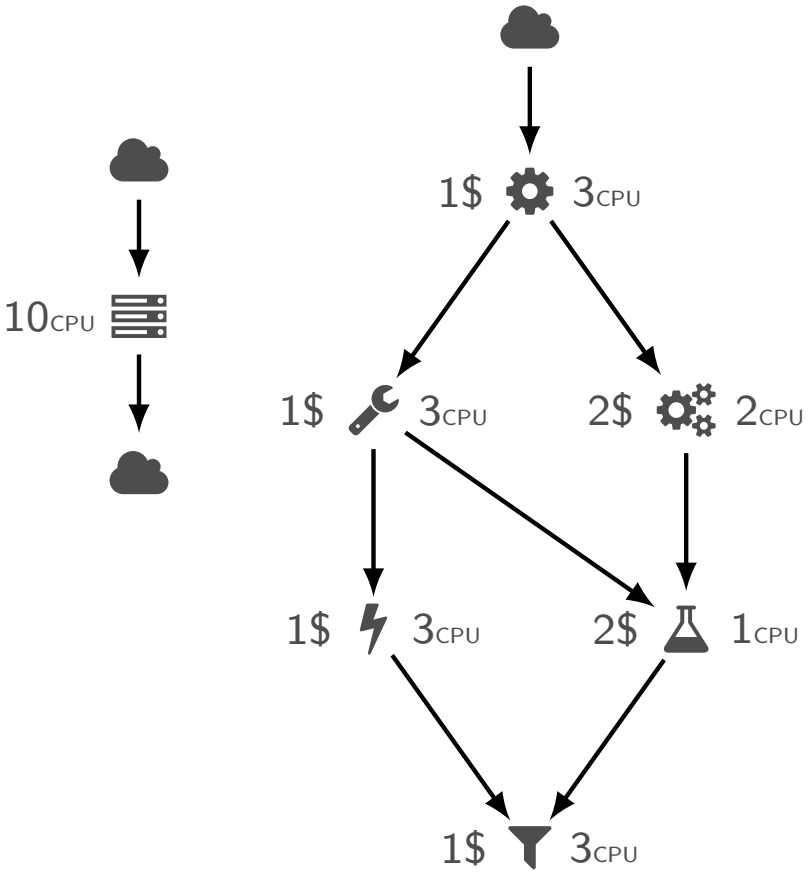
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	▼
7	0	3	6	5			
6	0	3	6				
5	0	3	6				
4	0	3	6				
3	0	3	6				
2	0	3	6				
1	0	3	-				
0	0	-	-				

Integral Polynomial Costs, Single Node

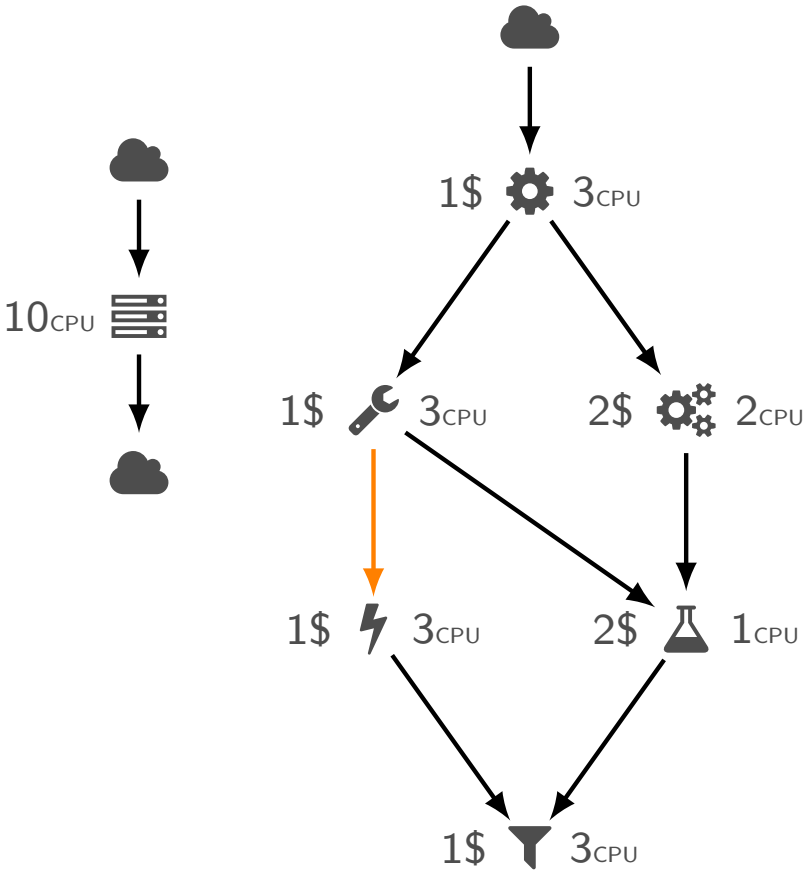
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	ϟ	⚗	⏷
7	0	3	6	5			
6	0	3	6	5			
5	0	3	6	5			
4	0	3	6	5			
3	0	3	6	5			
2	0	3	6	-			
1	0	3	-	-			
0	0	-	-	-			

Integral Polynomial Costs, Single Node

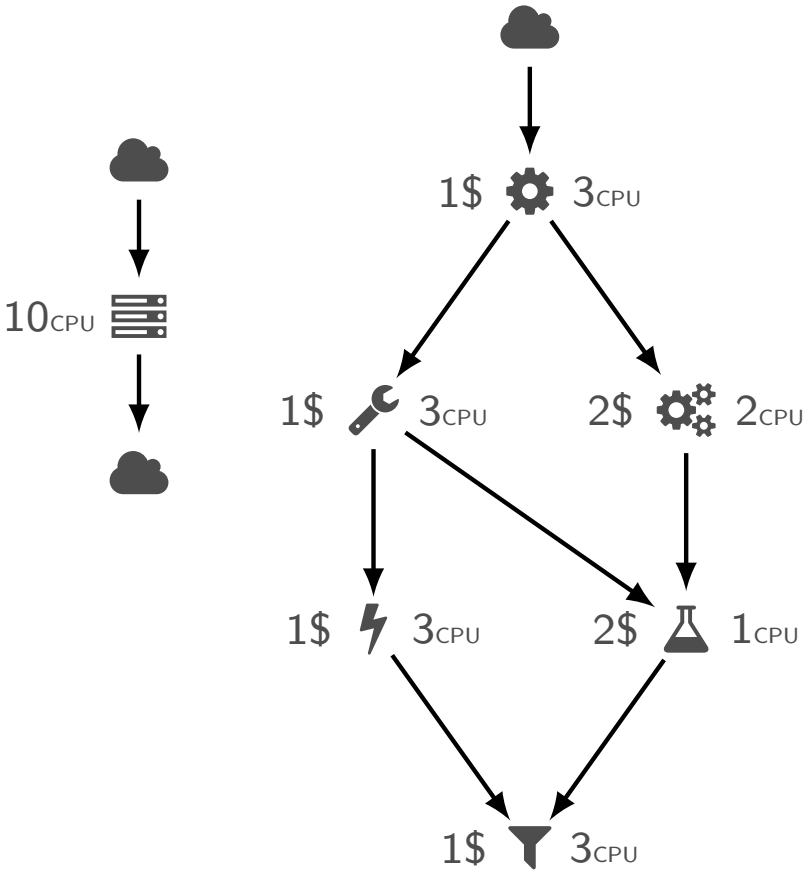
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	▼
7	0	3	6	5			
6	0	3	6	5			
5	0	3	6	5			
4	0	3	6	5			
3	0	3	6	5			
2	0	3	6	-			
1	0	3	-	-			
0	0	-	-	-			

Integral Polynomial Costs, Single Node

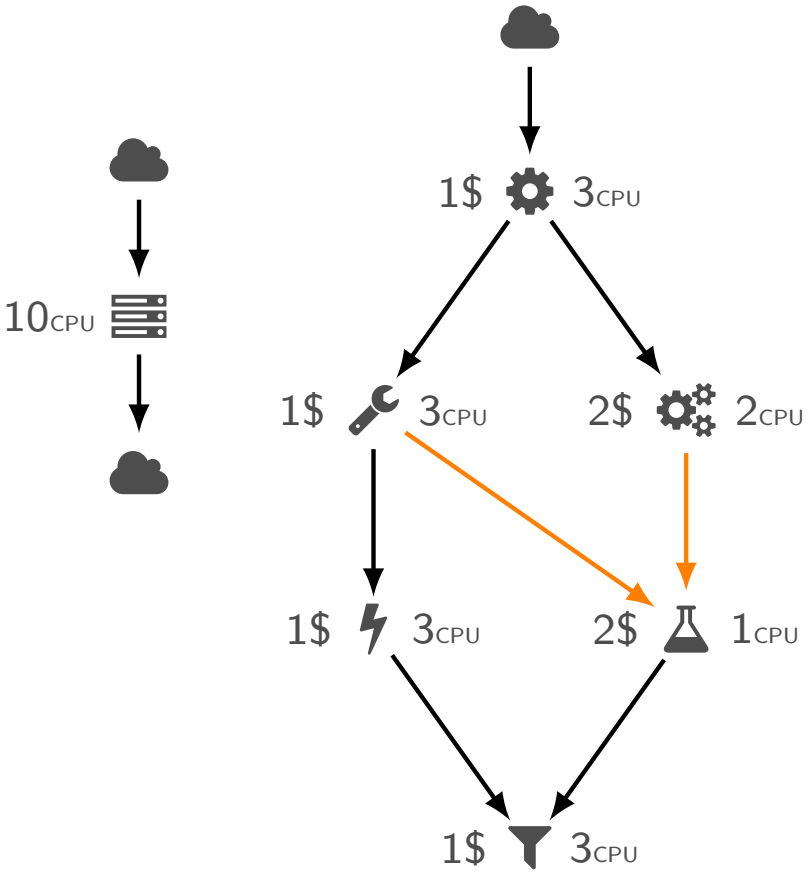
Min CPU per Cost Budget.



$							
7	0	3	6	5	9		
6	0	3	6	5	9		
5	0	3	6	5	9		
4	0	3	6	5	9		
3	0	3	6	5	9		
2	0	3	6	-	-		
1	0	3	-	-	-		
0	0	-	-	-	-		

Integral Polynomial Costs, Single Node

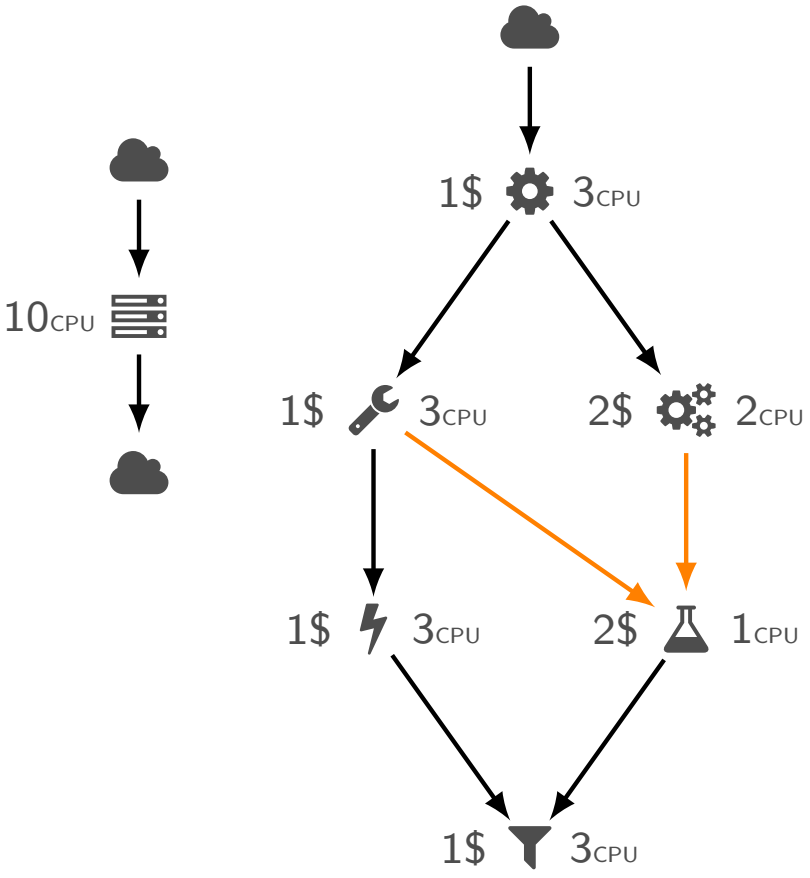
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	▼
7	0	3	6	5	9		
6	0	3	6	5	9		
5	0	3	6	5	9		
4	0	3	6	5	9		
3	0	3	6	5	9		
2	0	3	6	-	-		
1	0	3	-	-	-		
0	0	-	-	-	-		

Integral Polynomial Costs, Single Node

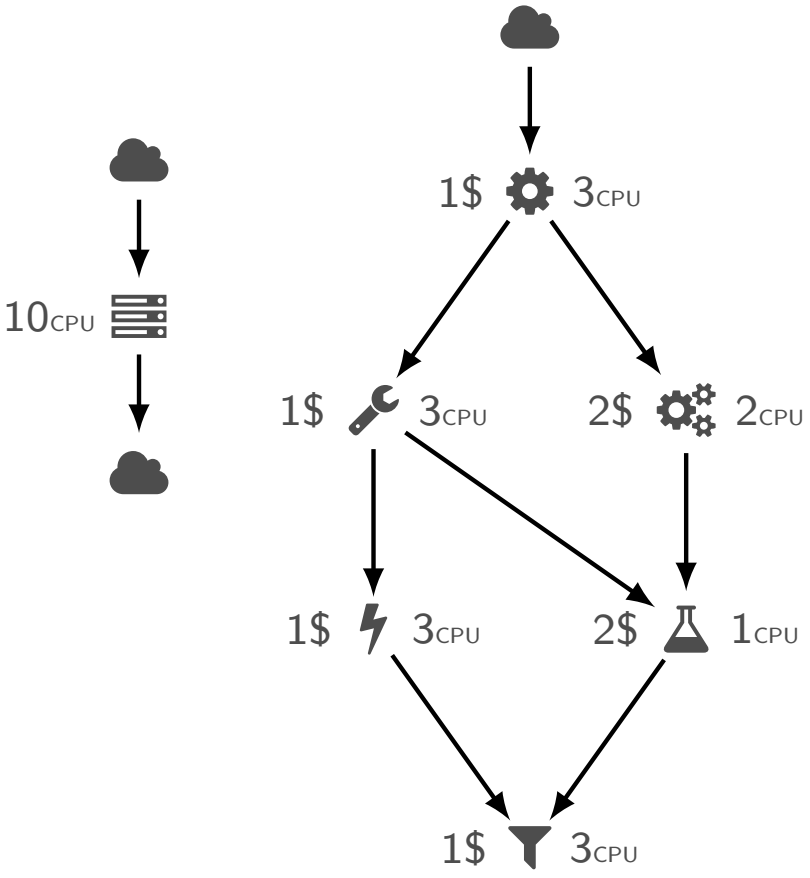
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	▼
7	0	3	6	5	9	6	
6	0	3	6	5	9	6	
5	0	3	6	5	9	6	
4	0	3	6	5	9		
3	0	3	6	5	9		
2	0	3	6	-	-		
1	0	3	-	-	-		
0	0	-	-	-	-		

Integral Polynomial Costs, Single Node

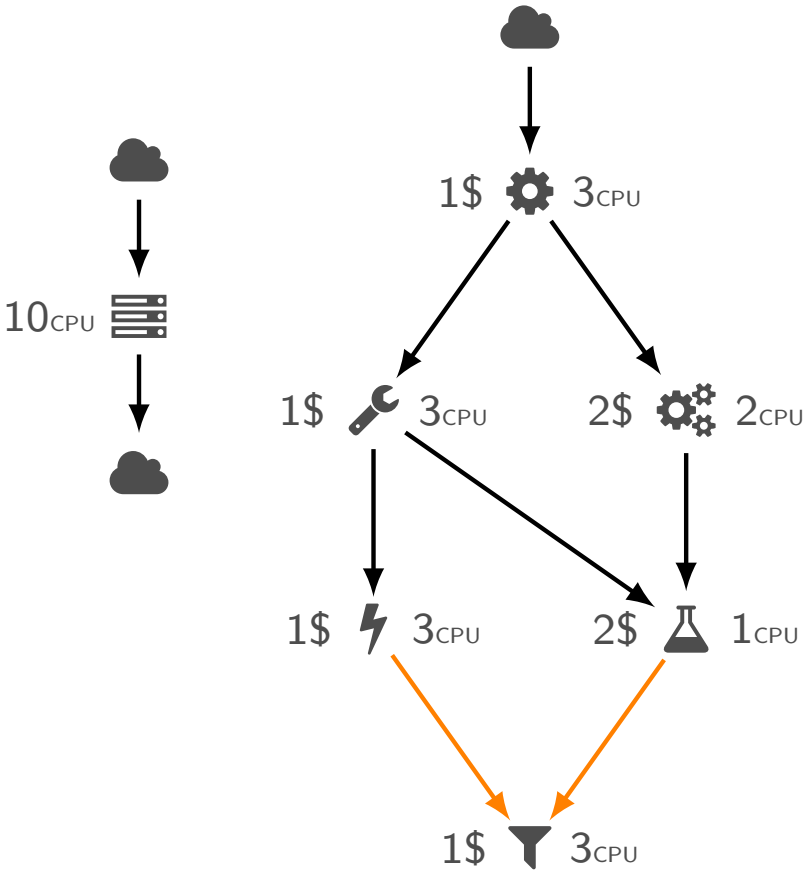
Min CPU per Cost Budget.



$							
7	0	3	6	5	9	6	
6	0	3	6	5	9	6	
5	0	3	6	5	9	6	
4	0	3	6	5	9	7	
3	0	3	6	5	9	-	
2	0	3	6	-	-	-	
1	0	3	-	-	-	-	
0	0	-	-	-	-	-	

Integral Polynomial Costs, Single Node

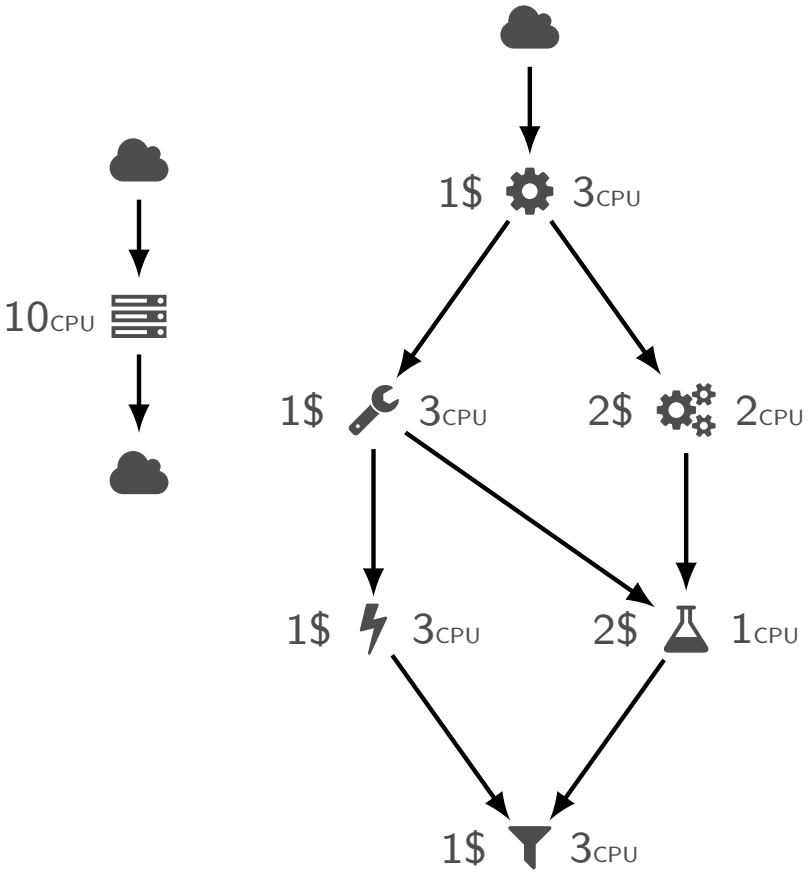
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	▼
7	0	3	6	5	9	6	
6	0	3	6	5	9	6	
5	0	3	6	5	9	6	
4	0	3	6	5	9	7	
3	0	3	6	5	9	-	
2	0	3	6	-	-	-	
1	0	3	-	-	-	-	
0	0	-	-	-	-	-	

Integral Polynomial Costs, Single Node

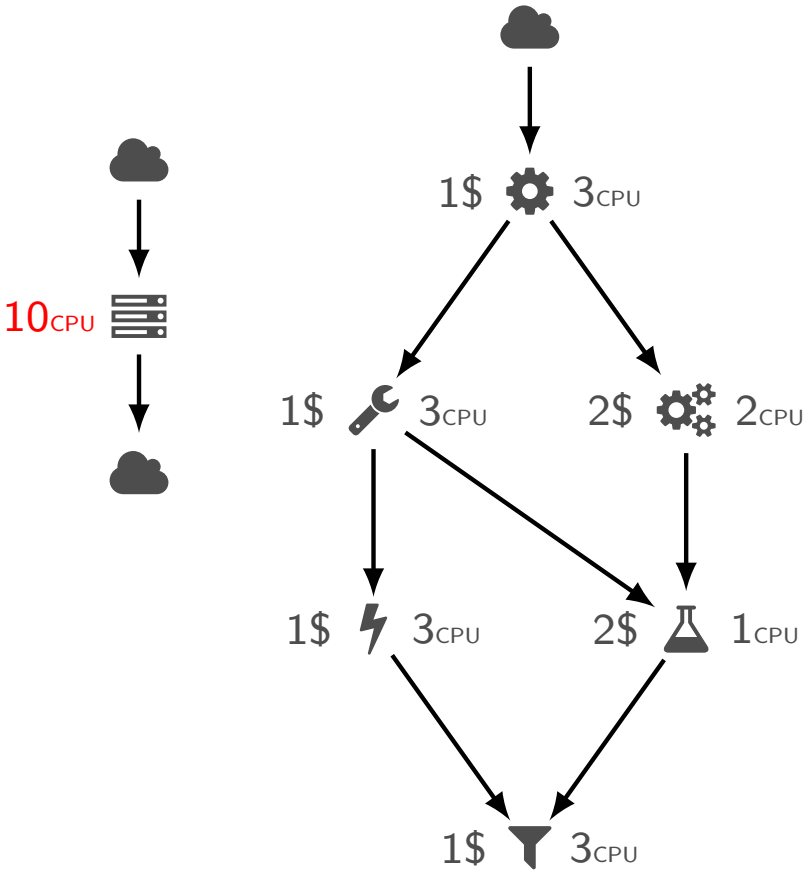
Min CPU per Cost Budget.



$							
7	0	3	6	5	9	6	9
6	0	3	6	5	9	6	9
5	0	3	6	5	9	6	10
4	0	3	6	5	9	7	12
3	0	3	6	5	9	-	-
2	0	3	6	-	-	-	-
1	0	3	-	-	-	-	-
0	0	-	-	-	-	-	-

Integral Polynomial Costs, Single Node

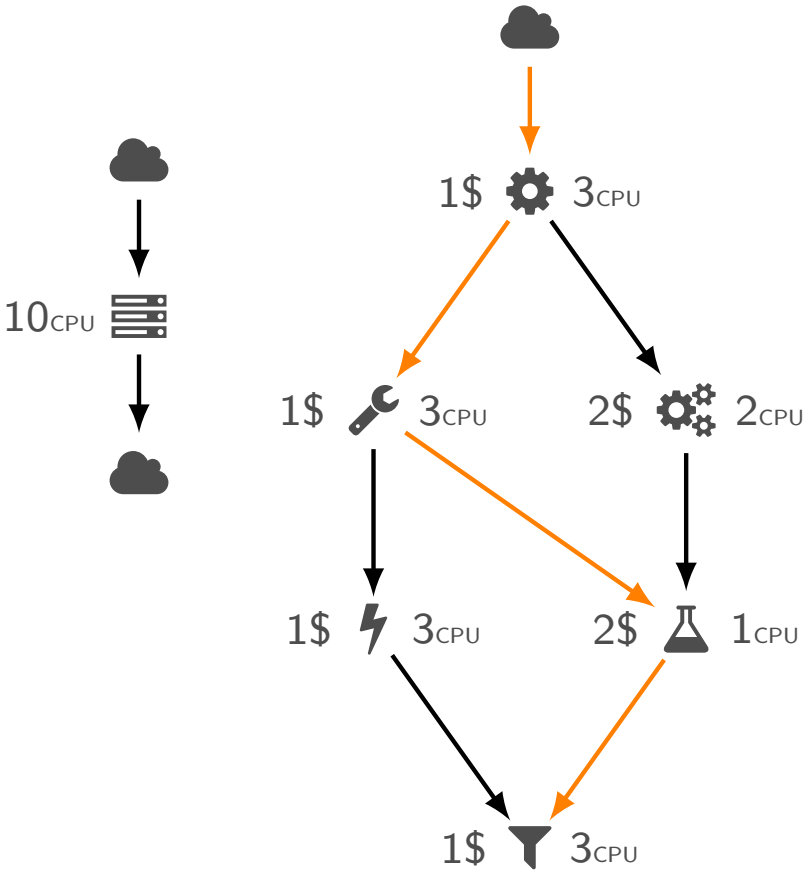
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	▼
7	0	3	6	5	9	6	9
6	0	3	6	5	9	6	9
5	0	3	6	5	9	6	10
4	0	3	6	5	9	7	12
3	0	3	6	5	9	-	-
2	0	3	6	-	-	-	-
1	0	3	-	-	-	-	-
0	0	-	-	-	-	-	-

Integral Polynomial Costs, Single Node

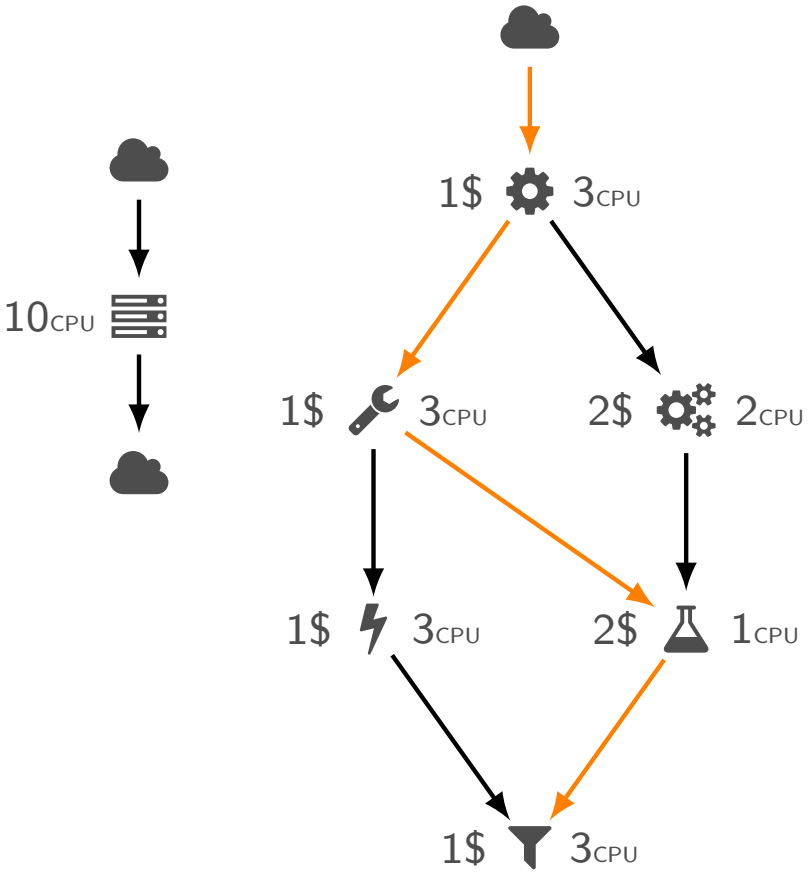
Min CPU per Cost Budget.



$	☁	⚙	🔧	⚙⚙	⚡	⚗	▼
7	0	3	6	5	9	6	9
6	0	3	6	5	9	6	9
5	0	3	6	5	9	6	10
4	0	3	6	5	9	7	12
3	0	3	6	5	9	-	-
2	0	3	6	-	-	-	-
1	0	3	-	-	-	-	-
0	0	-	-	-	-	-	-

Integral Polynomial Costs, Single Node

Min CPU per Cost Budget.

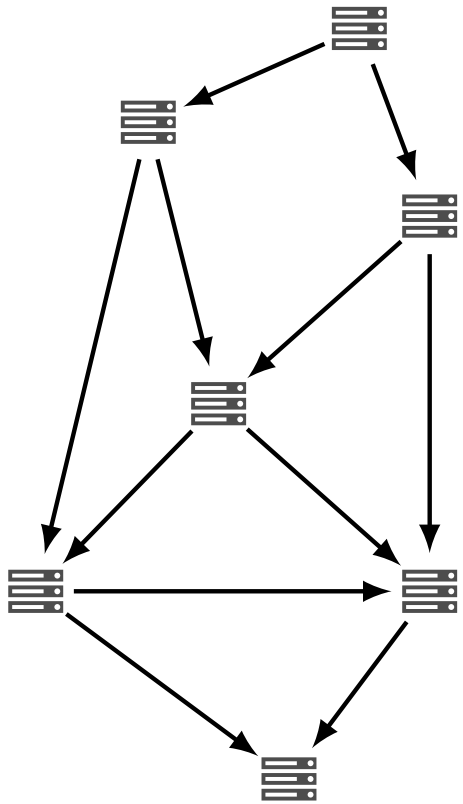


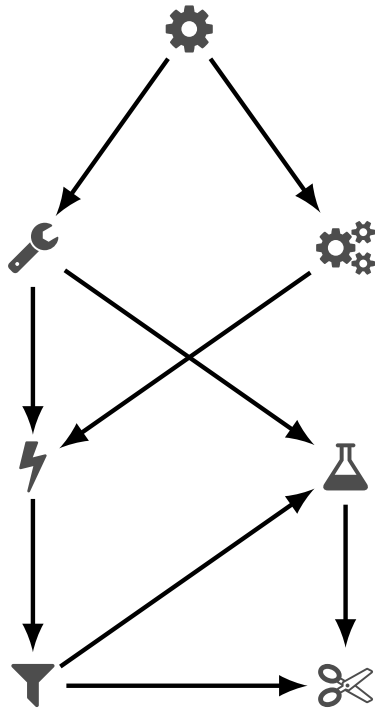
$							
7	0	3	6	5	9	6	9
6	0	3	6	5	9	6	9
5	0	3	6	5	9	6	10
4	0	3	6	5	9	7	12
3	0	3	6	5	9	-	-
2	0	3	6	-	-	-	-
1	0	3	-	-	-	-	-
0	0	-	-	-	-	-	-

Search for min Budget s.t. CPU constraint.

Small, Integral Costs, DAG

Min Latency s.t. Cost Budget.

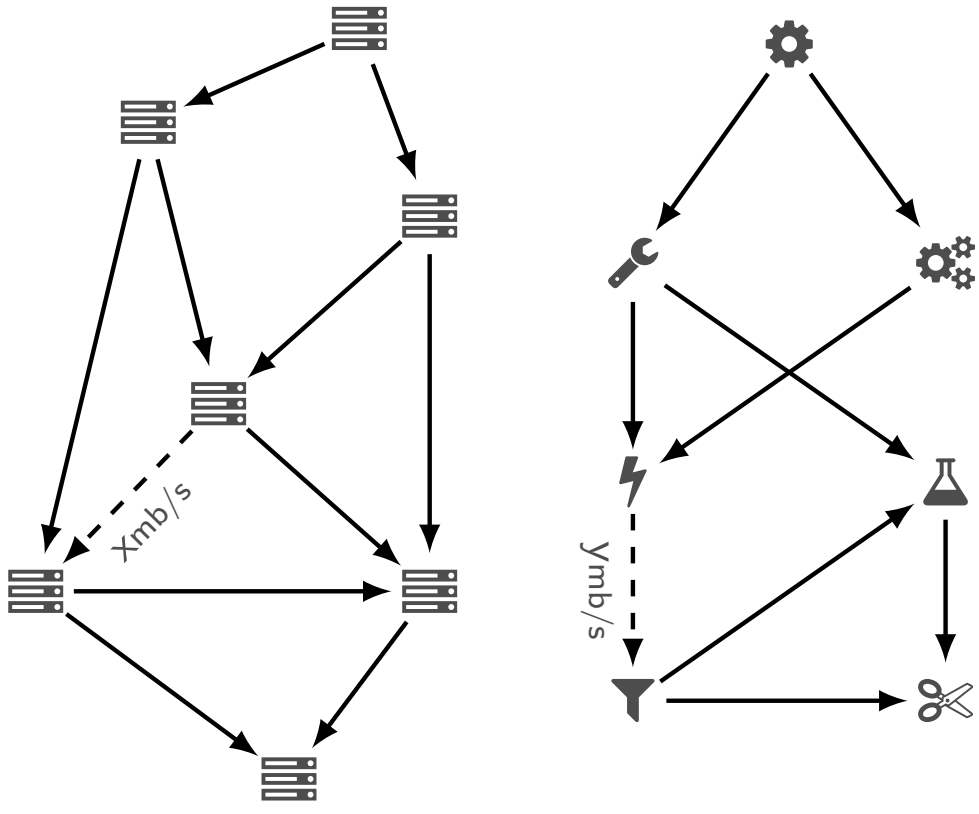




Small, Integral Costs, DAG

Min Latency s.t. Cost Budget.

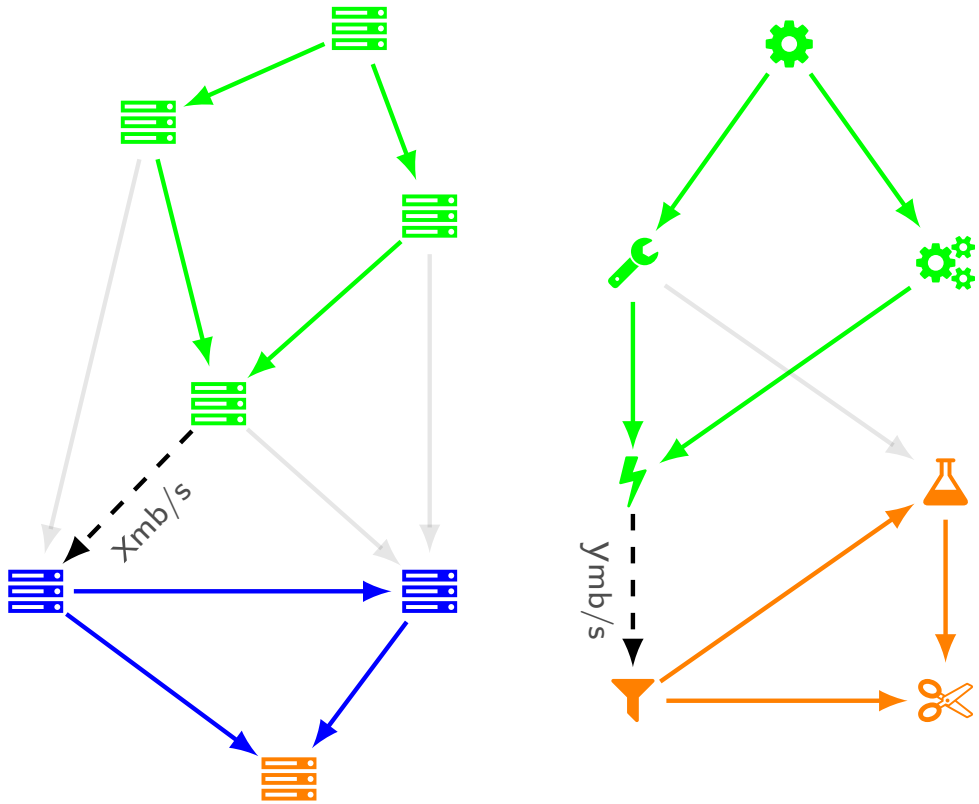
- Pick 2 arcs ($x \geq y$)



Small, Integral Costs, DAG

Min Latency s.t. Cost Budget.

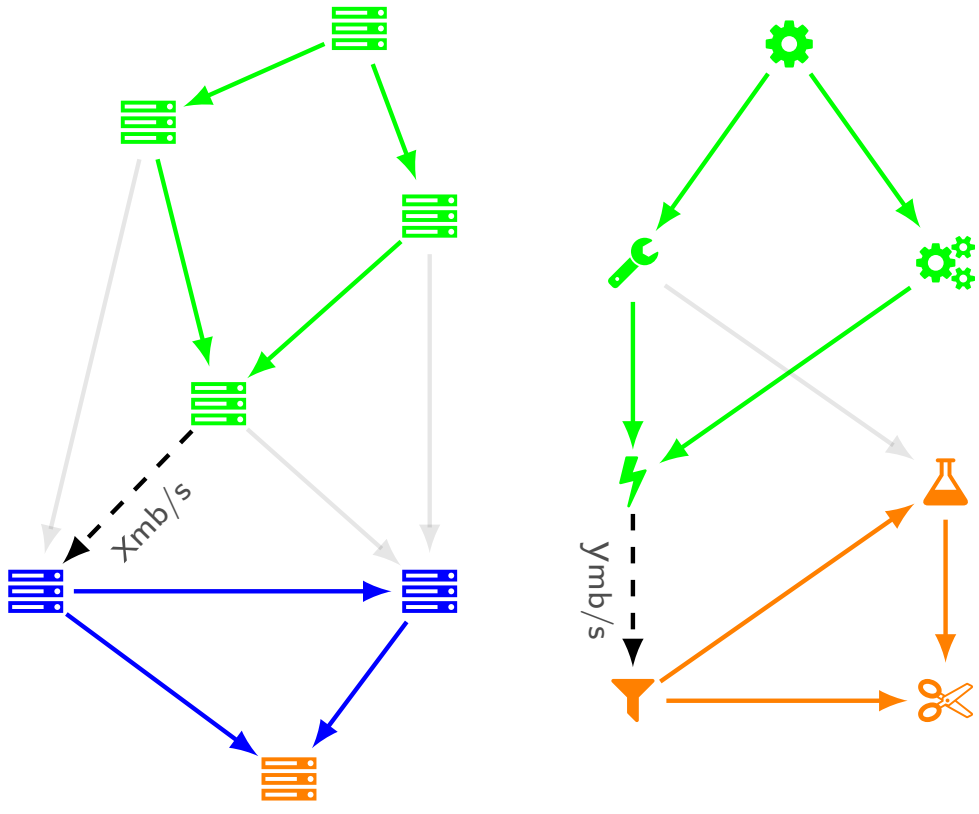
- Pick 2 arcs ($x \geq y$)
- Orange Problem - Single Node



Small, Integral Costs, DAG

Min Latency s.t. Cost Budget.

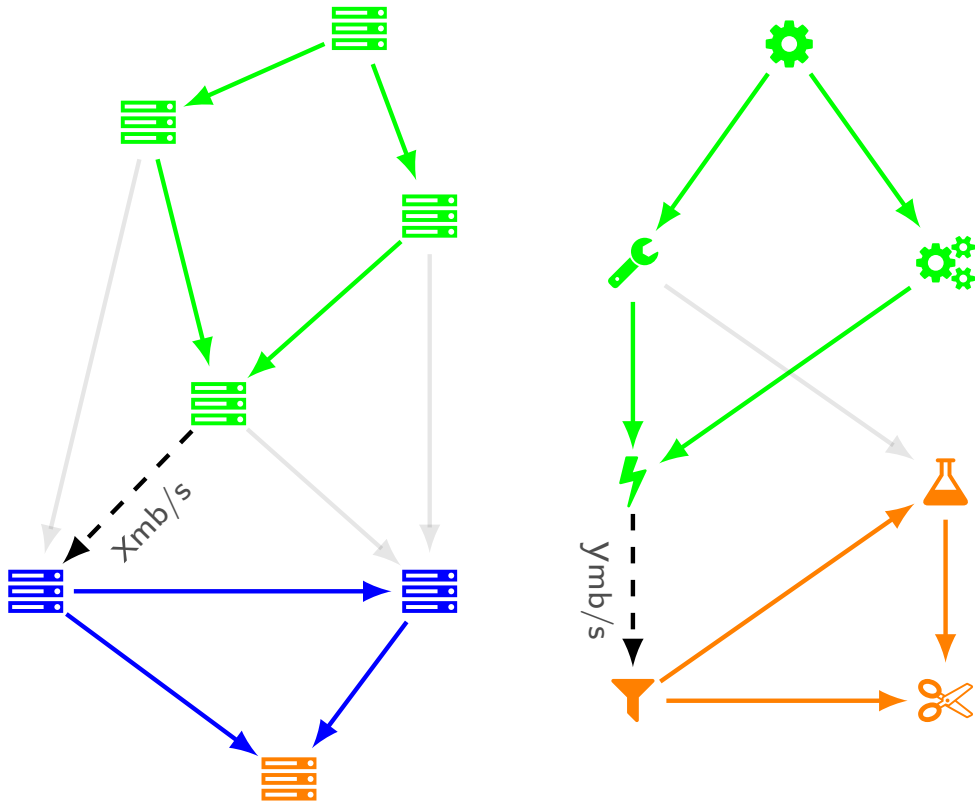
- Pick 2 arcs ($x \geq y$)
- Orange Problem - Single Node
- Blue Problem - Shortest Path



Small, Integral Costs, DAG

Min Latency s.t. Cost Budget.

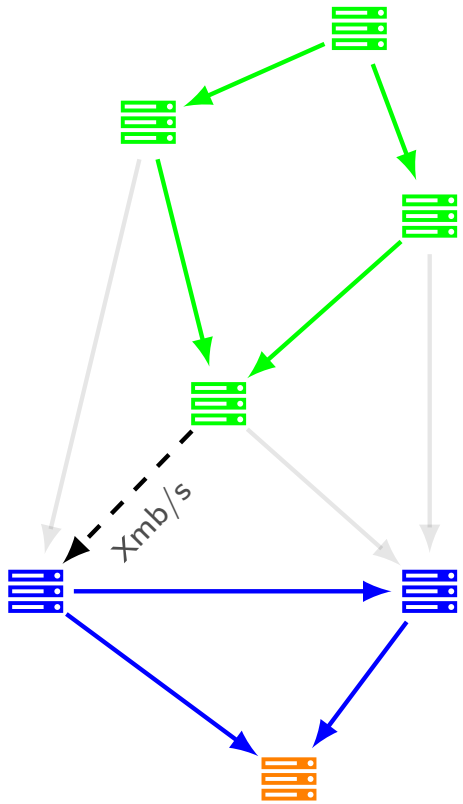
- Pick 2 arcs ($x \geq y$)
- Orange Problem - Single Node
- Blue Problem - Shortest Path
- Green Problem - Recursively

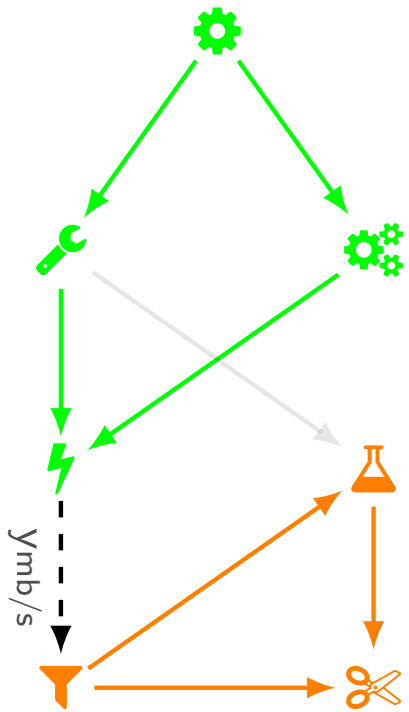


Small, Integral Costs, DAG

Min Latency s.t. Cost Budget.

- Pick 2 arcs ($x \geq y$)
- Orange Problem - Single Node
- Blue Problem - Shortest Path
- Green Problem - Recursively
  - At most $|V| \times |F| \times Cost$

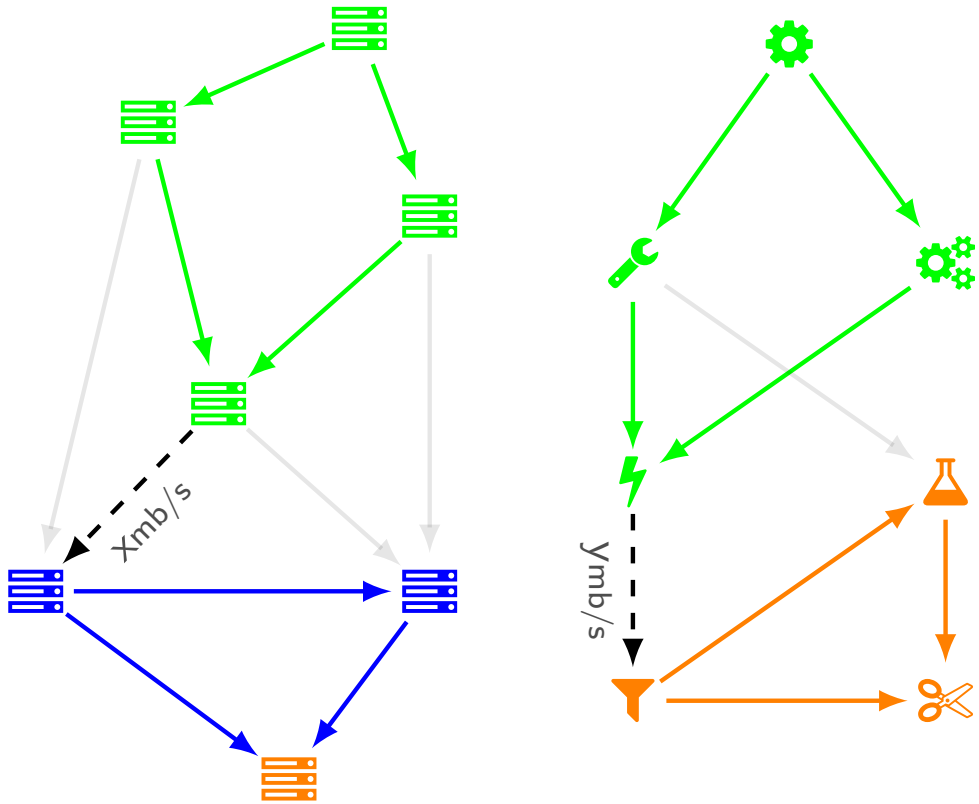




Small, Integral Costs, DAG

Min Latency s.t. Cost Budget.

- Pick 2 arcs ($x \geq y$)
- Orange Problem - Single Node
- Blue Problem - Shortest Path
- Green Problem - Recursively
  - At most $|V| \times |F| \times Cost$



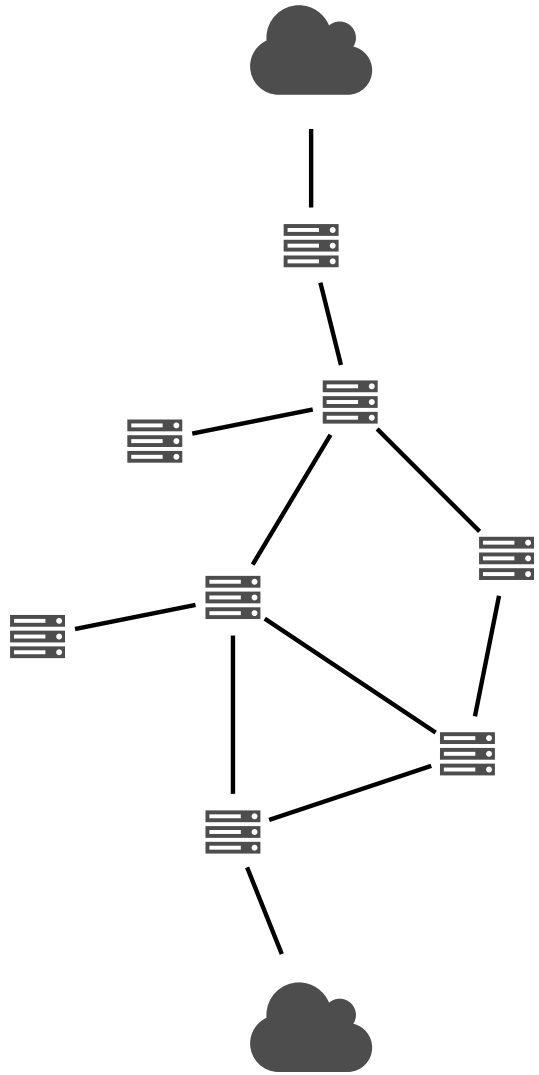
Search for the minimum budget s.t. Latency constraint.

Theorem

Costs can be scaled and rounded such that for any $\varepsilon > 0$ an optimal solution w.r.t the rounded value is a $1 + \varepsilon$-approximation for the original costs.

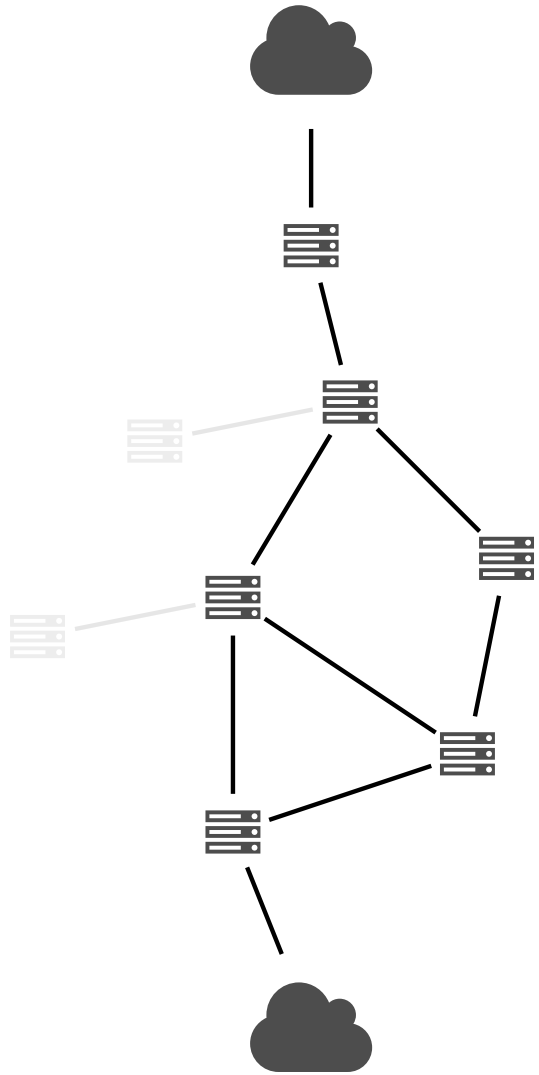
General Case

- Find unreachable nodes



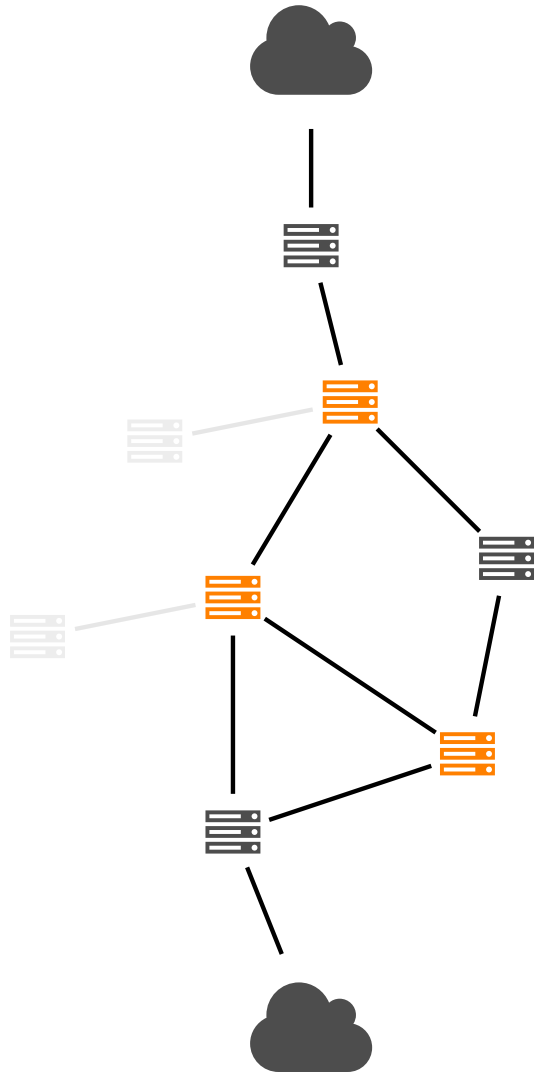
General Case

- Find unreachable nodes
- Mark nodes with deg > 2



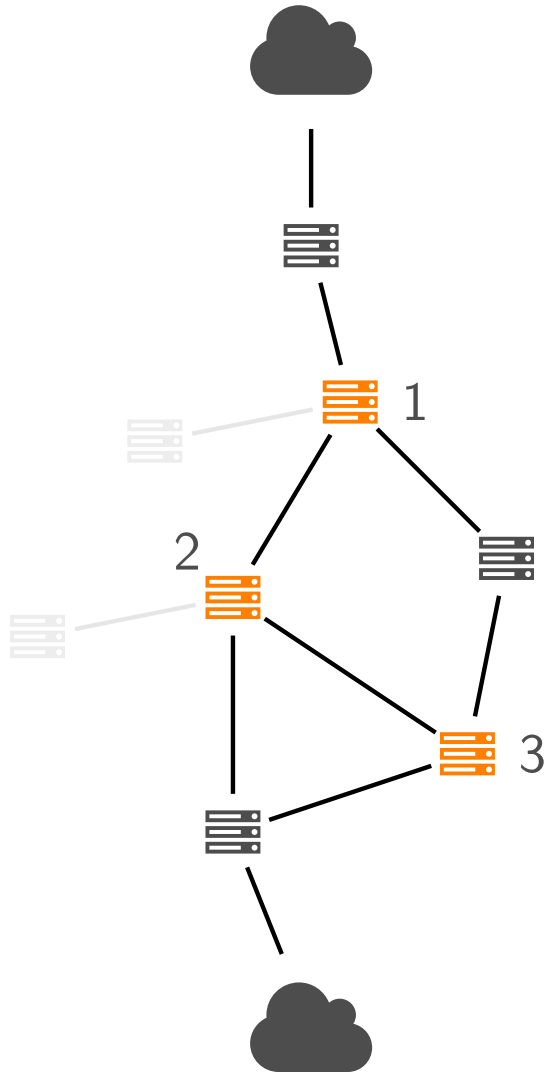
General Case

- Find unreachable nodes
- Mark nodes with deg > 2
- Choose a permutation



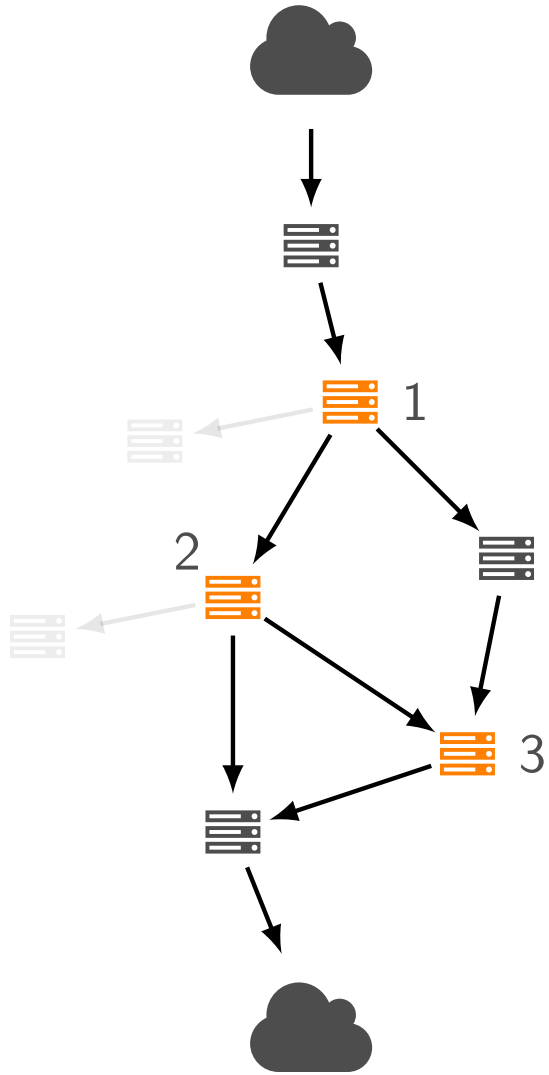
General Case

- Find unreachable nodes
- Mark nodes with deg > 2
- Choose a permutation
- Orient the network accordingly



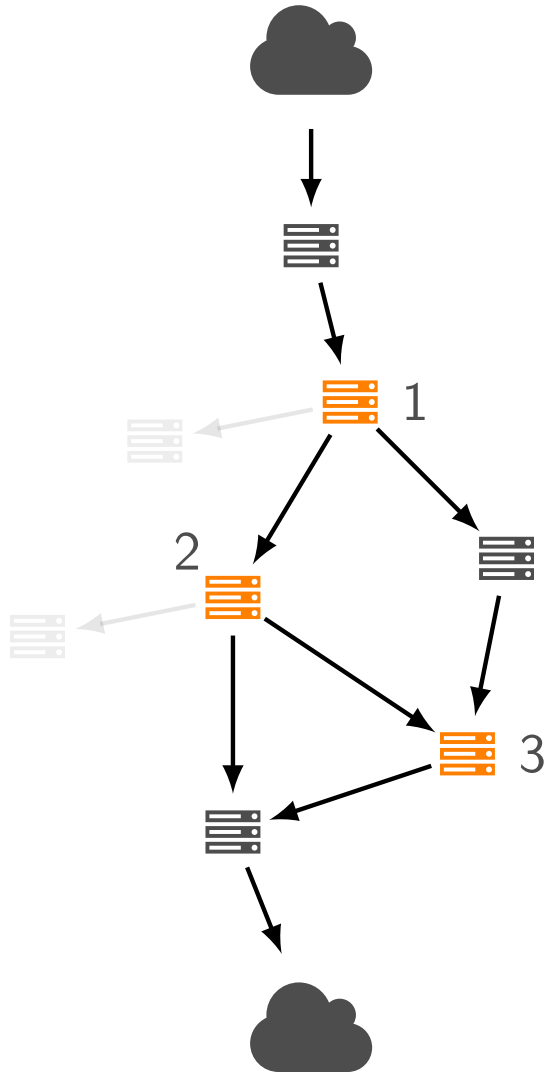
General Case

- Find unreachable nodes
- Mark nodes with deg > 2
- Choose a permutation
- Orient the network accordingly
- Use the previous algorithm



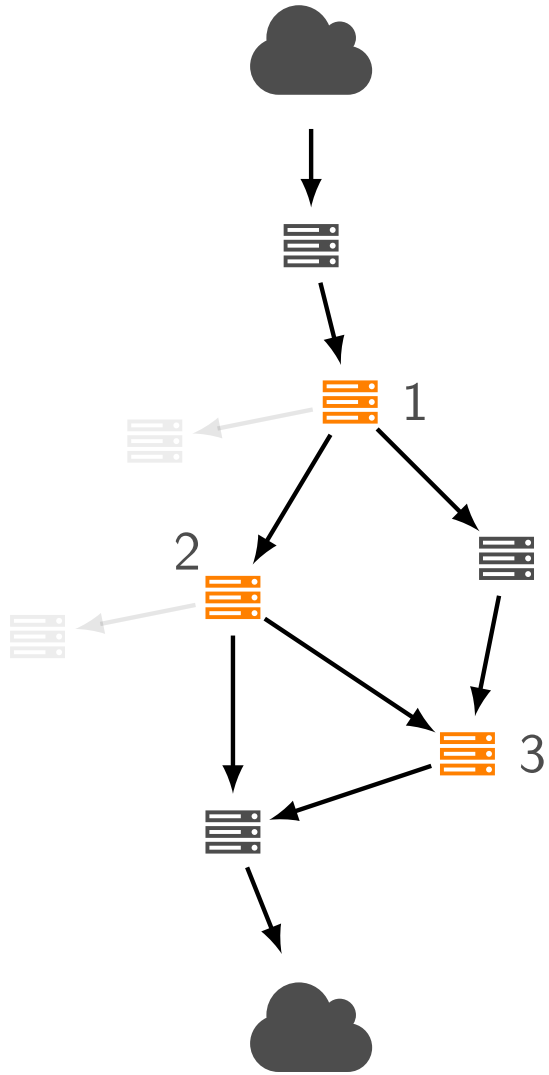
General Case

- Find unreachable nodes
- Mark nodes with deg > 2
- Choose a permutation
- Orient the network accordingly
- Use the previous algorithm
- Running time is $O(k!t(n))$ [1]



General Case

- Find unreachable nodes
- Mark nodes with deg > 2
- Choose a permutation
- Orient the network accordingly
- Use the previous algorithm
- Running time is $O(k!t(n))$ [1]



[1] Can be reduced to $O(2^k t(n))$

Conclusion

- Attracts Attention

Conclusion

- Attracts Attention
  - System

Conclusion

- Attracts Attention
  - System
  - Algorithmically

Conclusion

- Attracts Attention
  - System
  - Algorithmically
- Best we can hope solution for DAGs

Conclusion

- Attracts Attention
  - System
  - Algorithmically
- Best we can hope solution for DAGs
- Open questions for general SDNs

Conclusion

- Attracts Attention
  - System
  - Algorithmically
- Best we can hope solution for DAGs
- Open questions for general SDNs
  - Best we can hope solution under some assumptions

Conclusion

- Attracts Attention
  - System
  - Algorithmically
- Best we can hope solution for DAGs
- Open questions for general SDNs
  - Best we can hope solution under some assumptions
- Fault Tolerance

Conclusion

- Attracts Attention
  - System
  - Algorithmically
- Best we can hope solution for DAGs
- Open questions for general SDNs
  - Best we can hope solution under some assumptions
- Fault Tolerance
- In practice?

Thank You.