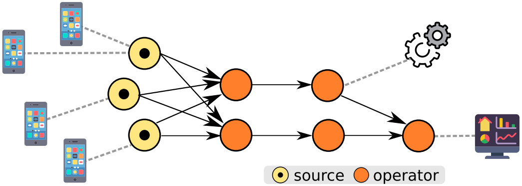
Reinforcement Learning Based Policies for Elastic Stream Processing on Heterogeneous Resources

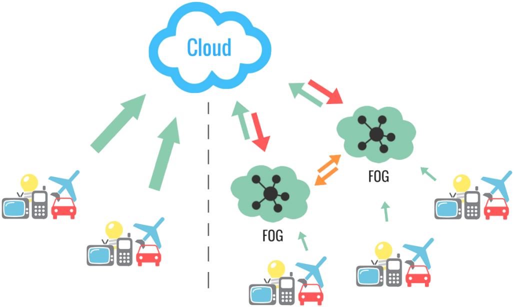
Gabriele Russo Russo, Valeria Cardellini, Francesco Lo Presti

University of Rome Tor Vergata, Italy

Distributed Data Stream Processing

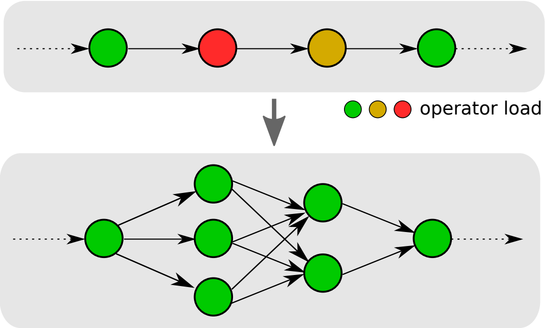


New pervasive services enabled by real-time stream processing



New trend:
moving applications towards users (and data!)

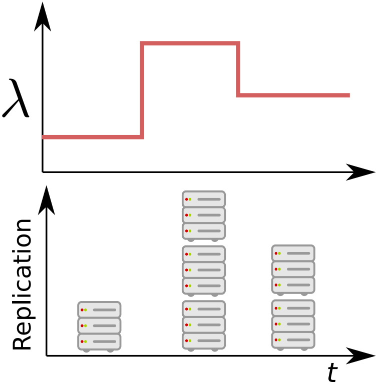
Elasticity for DSP



- A key feature for modern DSP systems
- Many approaches in the literature: queueing theory, control theory, threshold-based heuristics, ...
- Common assumption: homogeneous computing resources

Elasticity on Heterogeneous Resources

- Computing resources in Fog/Edge environments can be highly heterogeneous
- Trade-offs between cost, capacity, energy consumption, ...
- Elasticity policies should take it into account!



Goals

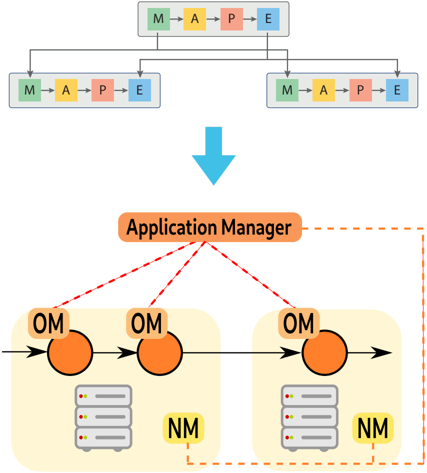
Decentralized elasticity on heterogeneous resources

- Problem formulation based on Markov Decision Process
- Efficient resolution through Function Approximation techniques
- Dealing with uncertainty: reinforcement learning

A Framework for Decentralized Elasticity

Based on Hierarchical MAPE:

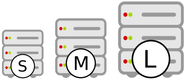
- An Application Manager for each application
- An Operator Manager for each operator



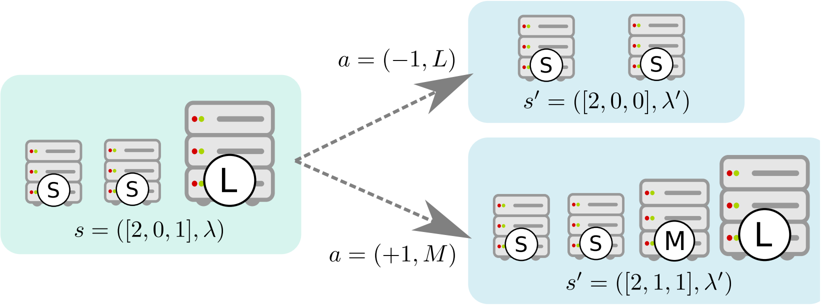
V. Cardellini, F. Lo Presti, M. Nardelli, G. Russo Russo, "Decentralized self-adaptation for elastic data stream processing", *Future Generation Computing Systems*, Vol. 87, pp. 171-185, October 2018.

Operator Manager: controlling elasticity

- N_{res} types of resources: $\tau_1, \tau_2, \ldots, \tau_{N_{res}}$



- We model the problem as a Markov Decision Process (MDP)
- System **state**: $s = (\mathbf{k}, \lambda)$
  k_τ = num. of replicas deployed on resources of type τ
  λ = current input data rate
- **Actions**: possible deployment adaptations



Operator Manager: controlling elasticity (2)

- **Cost** $c(s, a, s')$ paid after executing action a in state s, entering s'
- $c(s, a, s')$ weighted sum of normalized cost terms
- Resources cost

$$c_{res}(s, a, s') = \sum_{\tau \in T_{res}} k'_\tau c_\tau$$

- Reconfiguration cost

$$c_{rcf}(s, a, s') = \mathbb{1}_{\{\text{deployment changed}\}}$$

- SLO violation penalty

$$c_{SLO}(s, a, s') = \mathbb{1}_{\{R(s') > R_{max}\}}$$

- The optimal **policy** minimizes $\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t, s_{t+1}) \quad \gamma \in (0, 1)$

Scalability issues

- The optimal policy can be computed under different settings:
  - the model is completely known (e.g., using Value Iteration)
  - the model is (partially) unknown (using reinforcement learning)
- Most algorithms rely on the **Q function**:
  expected long-term cost of every action in every state
- Standard algorithms use the Q table to represent Q:
  an entry for each state-action pair in memory ... cannot scale!

State	**Action**	**Q**
s_1	a_1	$Q(s_1, a_1)$
s_2	a_2	$Q(s_2, a_2)$
...	...	...

Function Approximation for MDPs

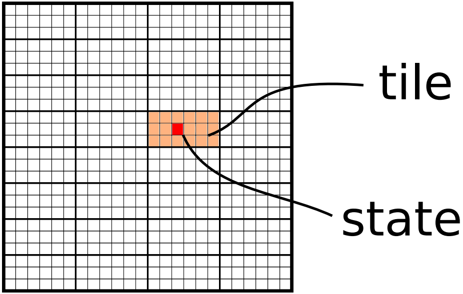
- Idea: replacing the Q table with a parametric function $\hat{Q}(s, a, \theta)$
- Need to store (and compute) only the parameters θ
- We focus on linear Function Approximation:

$$\hat{Q}(s, a, \boldsymbol{\theta}) = \sum_i \phi_i(s, a)\theta_i$$

- Weights $\boldsymbol{\theta}$: updated using Stochastic Gradient Descent
- Features $\boldsymbol{\phi}$: critical choice for good accuracy!

Defining features: Tile Coding

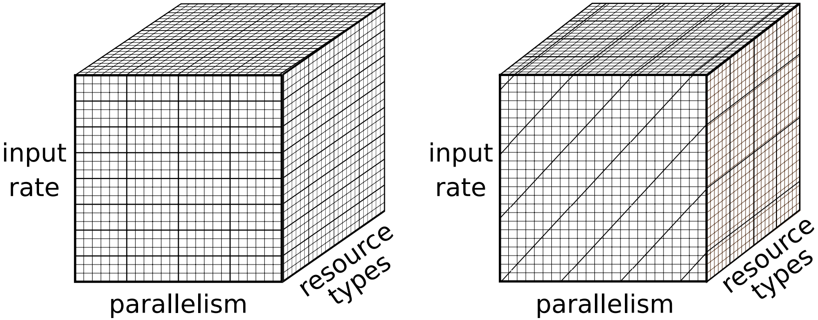
- Manually defining a good set of features is not feasible
- **Tile Coding**: cover the state space with “tilings”
- “similar” states covered by a single tile (i.e., a single feature)
- different number and shape of tiles
- multiple overlapping tilings combined for increased accuracy



Defining features: Tile Coding (2)

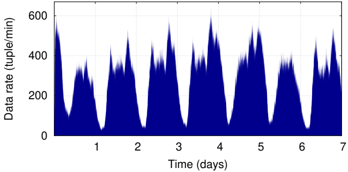
We aggregate "similar" states along 3 dimensions:

- input rate
- parallelism
- set of used resource types



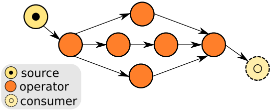
Evaluation

- We consider different sets of resource types
  - characterized by speedup and cost
- Standard and FA-based algorithms (including Q-learning) compared through a numerical evaluation
- Two threshold-based heuristic policies included in the comparison
  - CPU utilization threshold used for scaling
  - **TH-cost** picks the cheapest resource when needed
  - **TH-speedup** picks the resource with max speedup when needed
- Realistic workload, from DEBS 2015 Grand Challenge



Results: comparing algorithms

We compare SLO violations, deployment reconfigurations and resources cost when using different policies



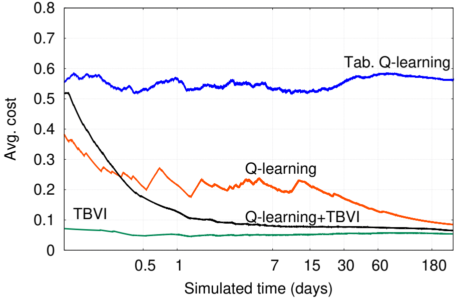
3 types of resources

Algorithm	Viol (%)	Reconf. (%)	Res.	
TH-cost	100.0	3.31	2.8	✗
TH-speedup	0.12	0.01	90.6	✗
VI	0.0	0.0	12.0	✓
TBVI (VI + FA)	0.0	0.30	11.4	✓

10 types of resources

Algorithm	Viol (%)	Reconf. (%)	Res.	
TH-cost	100.0	4.11	2.8	✗
TH-speedup	0.12	0.01	90.6	✗
VI	-	-	-	✗
TBVI (VI + FA)	0.10	0.03	17.7	✓

Results: learning algorithms

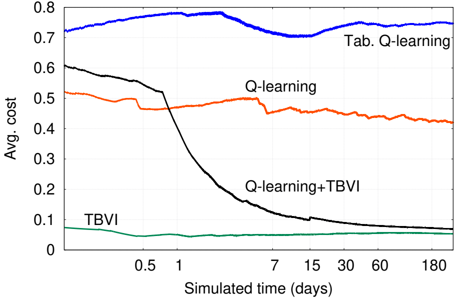


Average cost during a single experiment

← 3 types of resources

TBVI (model-based)

Tabular Q-learning
Q-learning with FA
Q-learning initialized with an approximate model



← 10 types of resources

Results: different sets of features

We solve the MDP using different tiling configurations, varying the size of tiles:

- coarse-grained ($\approx$ 1000 features)
- standard ($\approx$ 2500 features)
- fine-grained ($\approx$ 6000 features)

Features	**Avg. cost**
Tile Coding (coarser)	0.059
Tile Coding	0.054
Tile Coding (finer)	0.070

Conclusion

- Decentralized policies for elasticity on heterogeneous resources
- Reinforcement learning allows to deal with model uncertainty
- Function Approximation techniques required for scalability
- An approach likely re-usable for solving similar problems with self-adaptive distributed systems

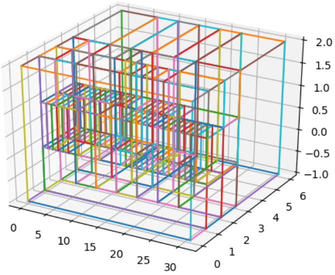
Future work:

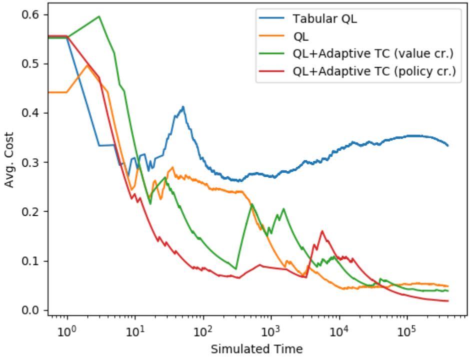
- Implementation on top of existing DSP framework
- Non-linear FA, including Neural Networks
- Adaptive Tile Coding

Adaptive Tile Coding (preview)

- Tile Coding still requires expertise to choose size/shape of tiles
- If the problem changes, may need new tilings

- Adaptive Tile Coding: identify best partitioning in an automated way
- Start with one large tile, then iteratively split to increase accuracy





Thanks for your attention!

russo.russo@ing.uniroma2.it
www.ce.uniroma2.it/~russorusso