Towards Hierarchical Autonomous Control for Elastic Data Stream Processing in the Fog

Valeria Cardellini, Francesco Lo Presti,
Matteo Nardelli, Gabriele Russo Russo

University of Rome Tor Vergata, Italy

Data Stream Processing

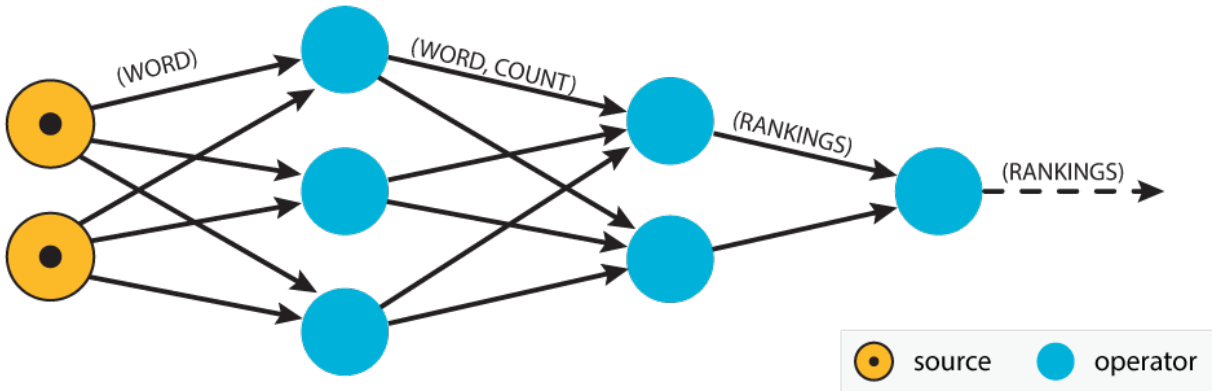
Data Stream Processing (DSP) applications:

- processing of data streams generated by distributed sources
- To extract information in a (near) real-time manner

To increase scalability and availability, reduce latency, network traffic, and power consumption

Exploit distributed and near-edge computation

(distributed cloud and Fog computing)



Old and New Challenges

Distributed Environment

- Geographic distribution, network latencies are not-negligible
- Computing and network resources can be heterogeneous (e.g., capacity, energy consumption, business constraints)
- Data cannot be quickly moved among computing nodes

DSP Applications are long running

Reconfigure the application deployment

- has a non negligible cost!
- can negatively affect application performance in the short term
  - Application freezing times, especially for stateful operators

State of the art

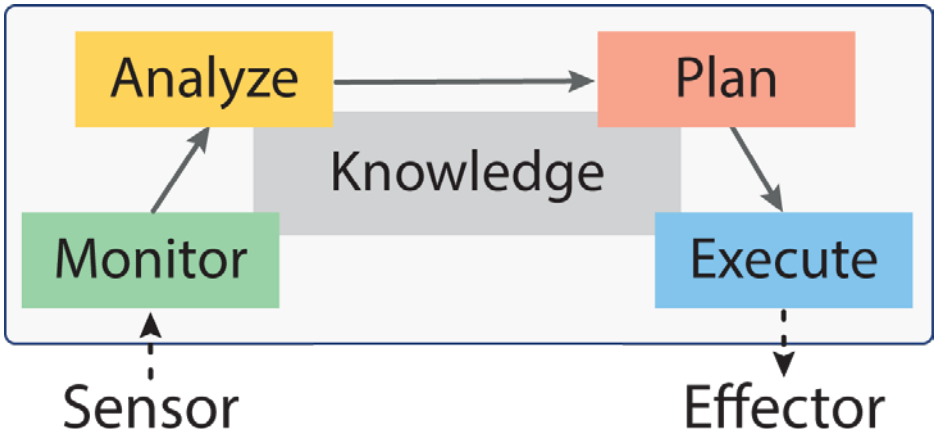
Centralized approaches:

- most of the proposed approaches designed for clusters
- do not scale well in a distributed environment

Decentralized approaches:

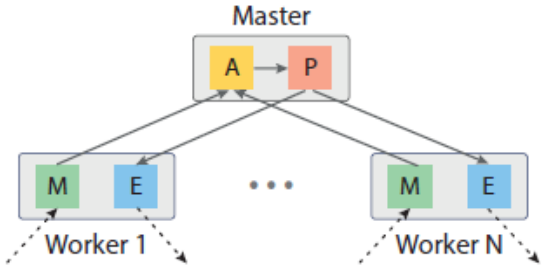
- several proposal
- their inherent lack of coordination might result in frequent reconfigurations

MAPE (**M**onitor, **A**nalyze, **P**lan and **E**xecute)

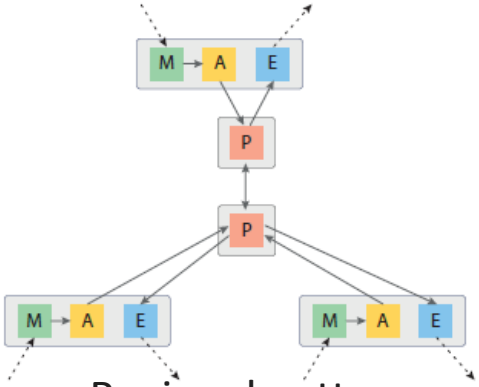


Decentralized MAPE

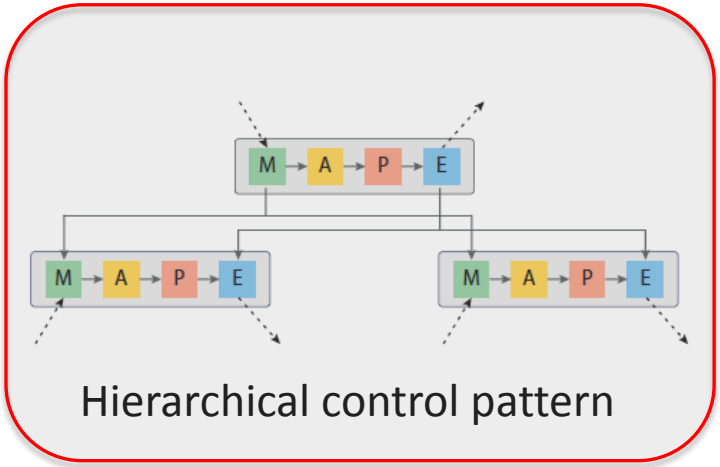
- Many Patterns, each with pro and cons



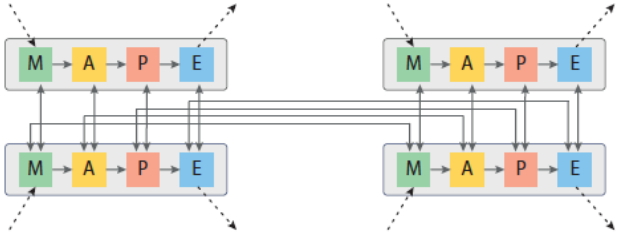
Master-worker



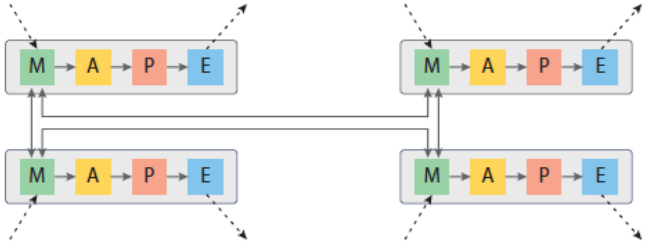
Regional pattern



Hierarchical control pattern



Coordinated control pattern



Information sharing pattern

D. Weyns, B. Schmerl, V. Grassi, S. Malek, et al. On patterns for decentralized control in self-adaptive systems. In Software Engineering for Self-Adaptive Systems II, vol. 7475 of LNCS, Springer, 2013.

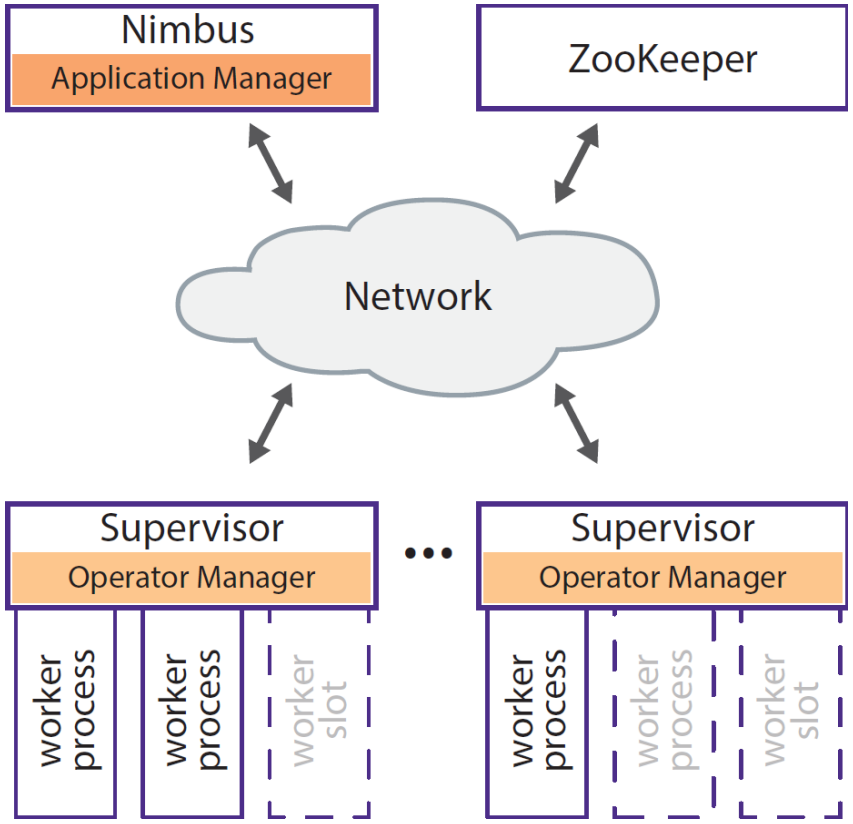
Goals

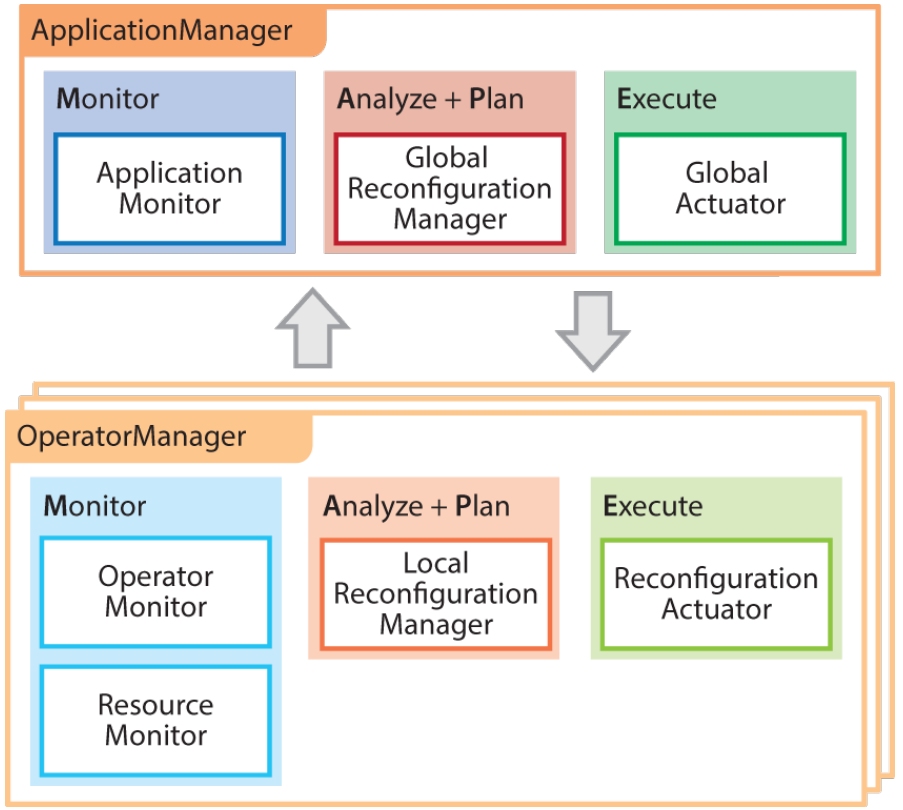
- Design a **hierarchical distributed approach** to the autonomous control of DSP applications
- Support **run-time adaptation**
  - **Elasticity**
    automatically scale in/out the number of operator instances
  - **Stateful Migration**
    relocate operators without compromising application integrity
- Design a **simple control policy**
- Integration of our solution in Storm

Hierarchical MAPEs in Storm

- New components in Apache Storm to realize a Hierarchical MAPE pattern

- Operator Manager vs Application Manager
  - Concerns and time scale separation

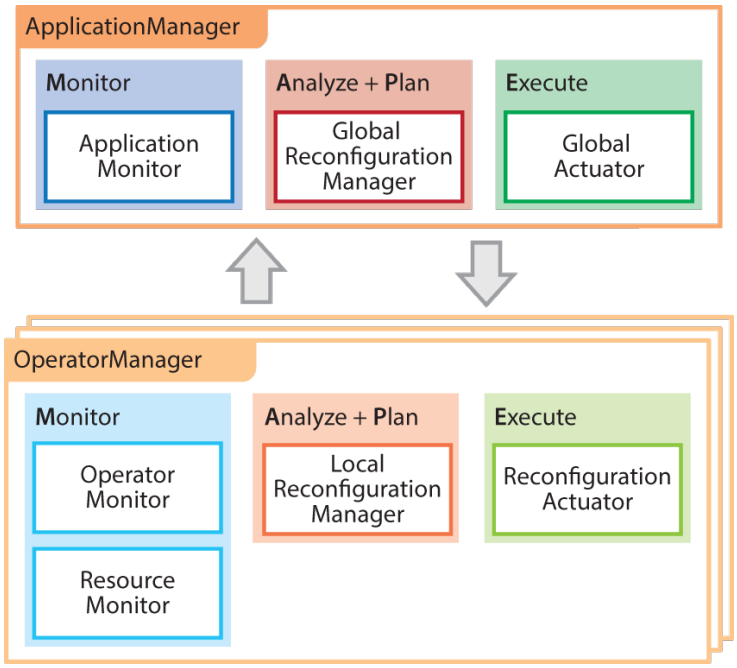




Hierarchical MAPEs in Storm

Operator Manager

- Monitors operator and local resources
  - e.g., Thread CPU utilization,
- Determines whether a Migration and/or Scale operation is needed
- Executes the reconfiguration
  - If gets the permission to



Application Manager

- Monitors Application Performance
  - SLA enforcement
- Coordinates operator reconfigurations
  - Grants permission to enact reconfigurations
  - Controls reconfiguration frequencies

General Framework for Distributed Optimization

Simple Distributed Heuristic: Operator Manager

- issues reconfiguration plans:

  action, gain, cost

- action: migrates an operator replica
  - threshold based policy on CPU utilization
  - new location: probabilistic selection from the neighborhood
  - cost: estimated stateful migration time

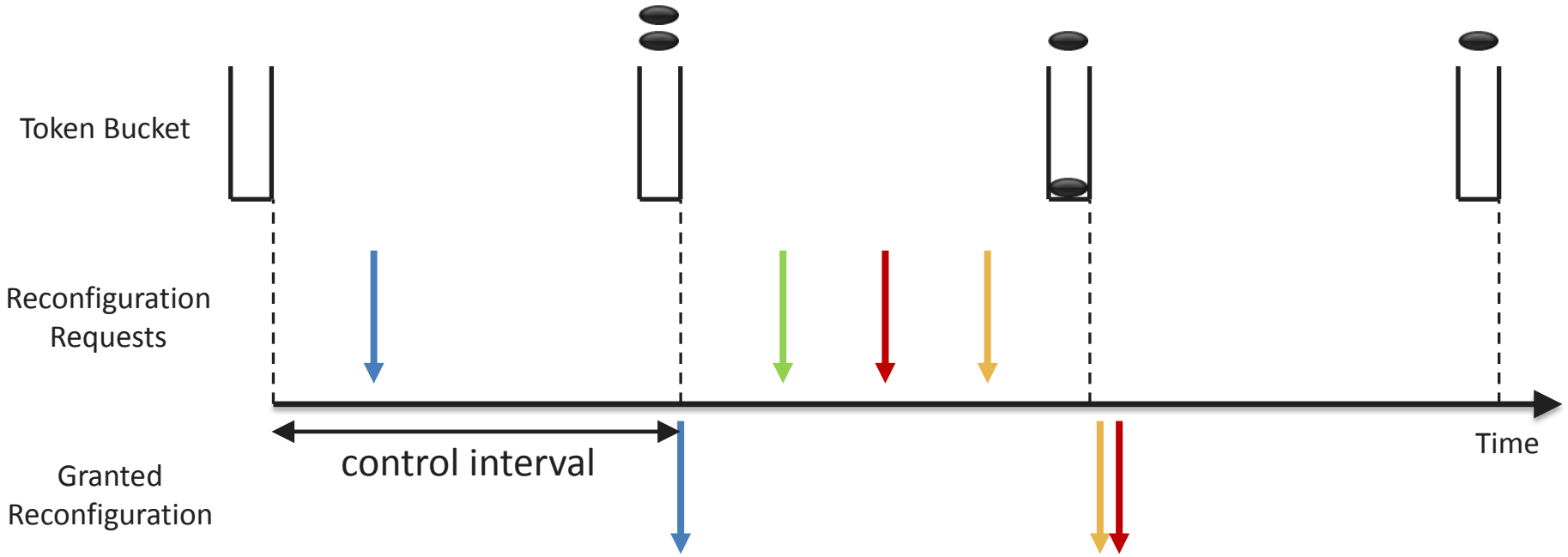
- action: operator scaling
  - threshold based policy on S_α percent of CPU time used by the replica α
  - scale in: if removing a replica does not significantly increase load on other replicas $\sum_{\alpha=1}^{n} S_\alpha/(n-1) < cS_{\text{s-out}}$
  - cost: estimated time to relocate the operator state (if any)

- gain function: `scale-out > migration > scale-in`

Simple Distributed Heuristic: Application Manager

Token-based policy

- Considers time divided in intervals
- Generates reconfiguration tokens based on application performance
- Grants as many reconfigurations as available tokens
  - Prioritizing by gain to cost ratio



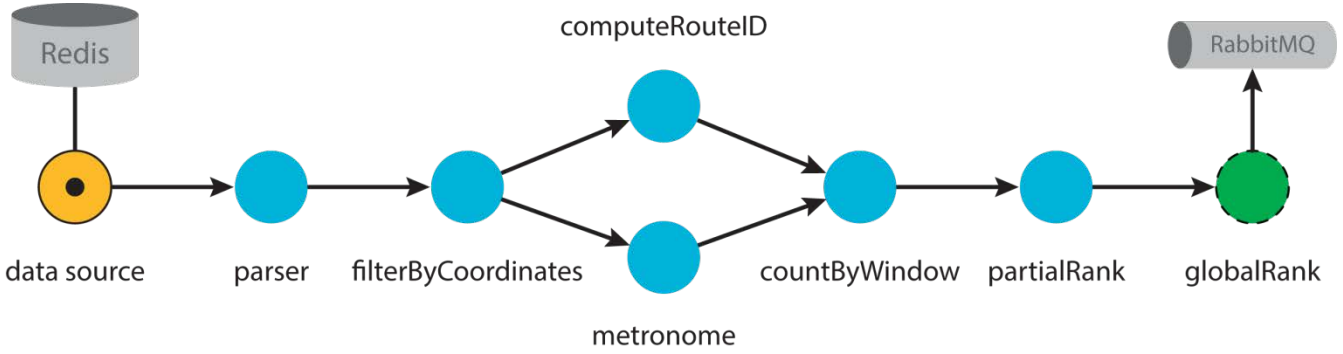
Evaluation

Infrastructure

- 5 worker nodes + 1 host for Nimbus and ZooKeeper
- each node Intel Xeon 8 cores@2Ghz, 16 GB RAM

Application

- DEBS 2015 Grand Challenge: **top10 frequent routes** NYC taxis in the last 30 min
- Requires: max Response Time R_{max} = 200 ms

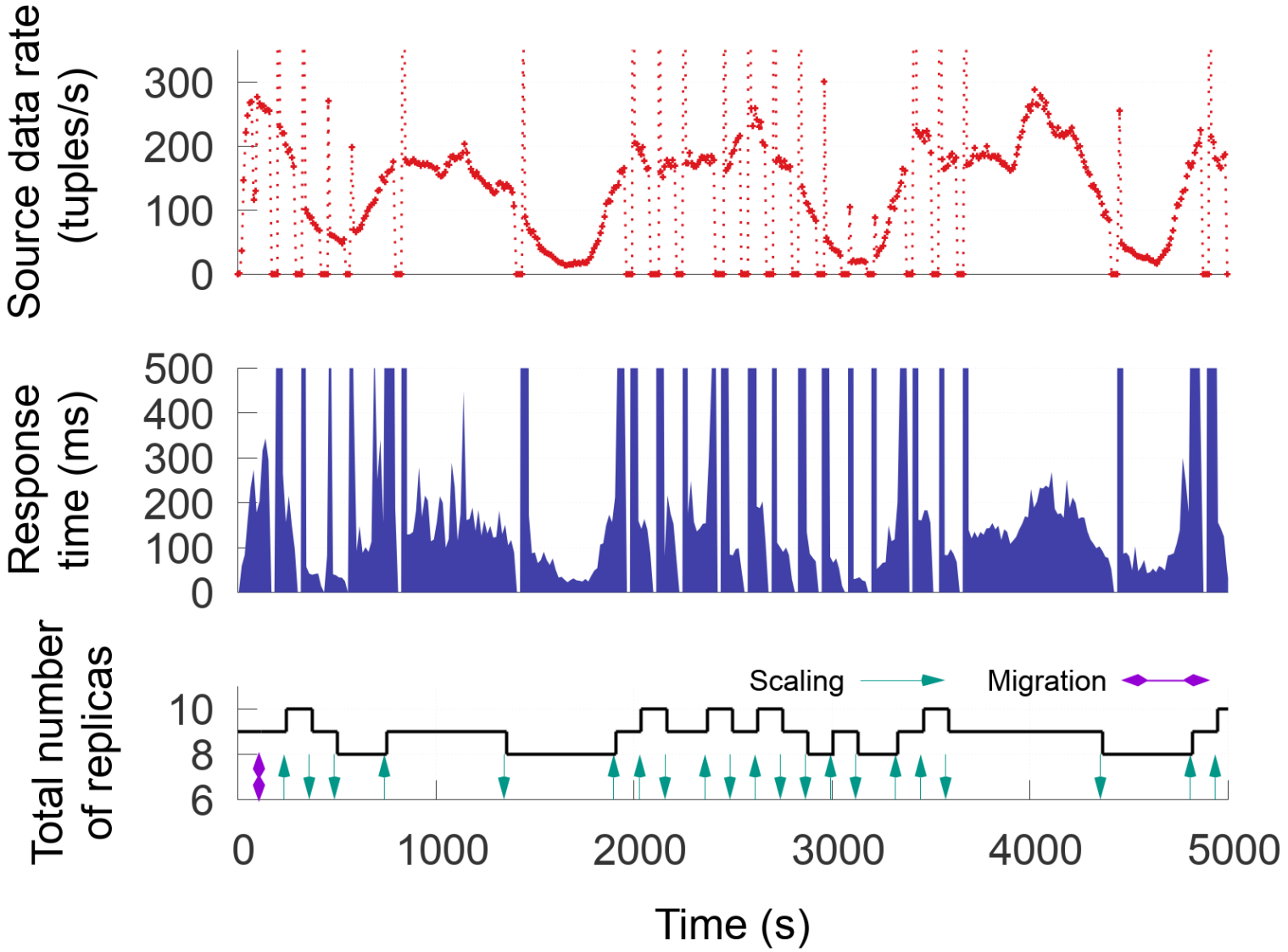


Policy parameters

- Operator Manager policy: thresholds on utilization to 70% (c = 0.75)
- Application Manager policy: token bucket capacity = 1 token

Evaluation

Application Manager policy: grants all reconfiguration requests

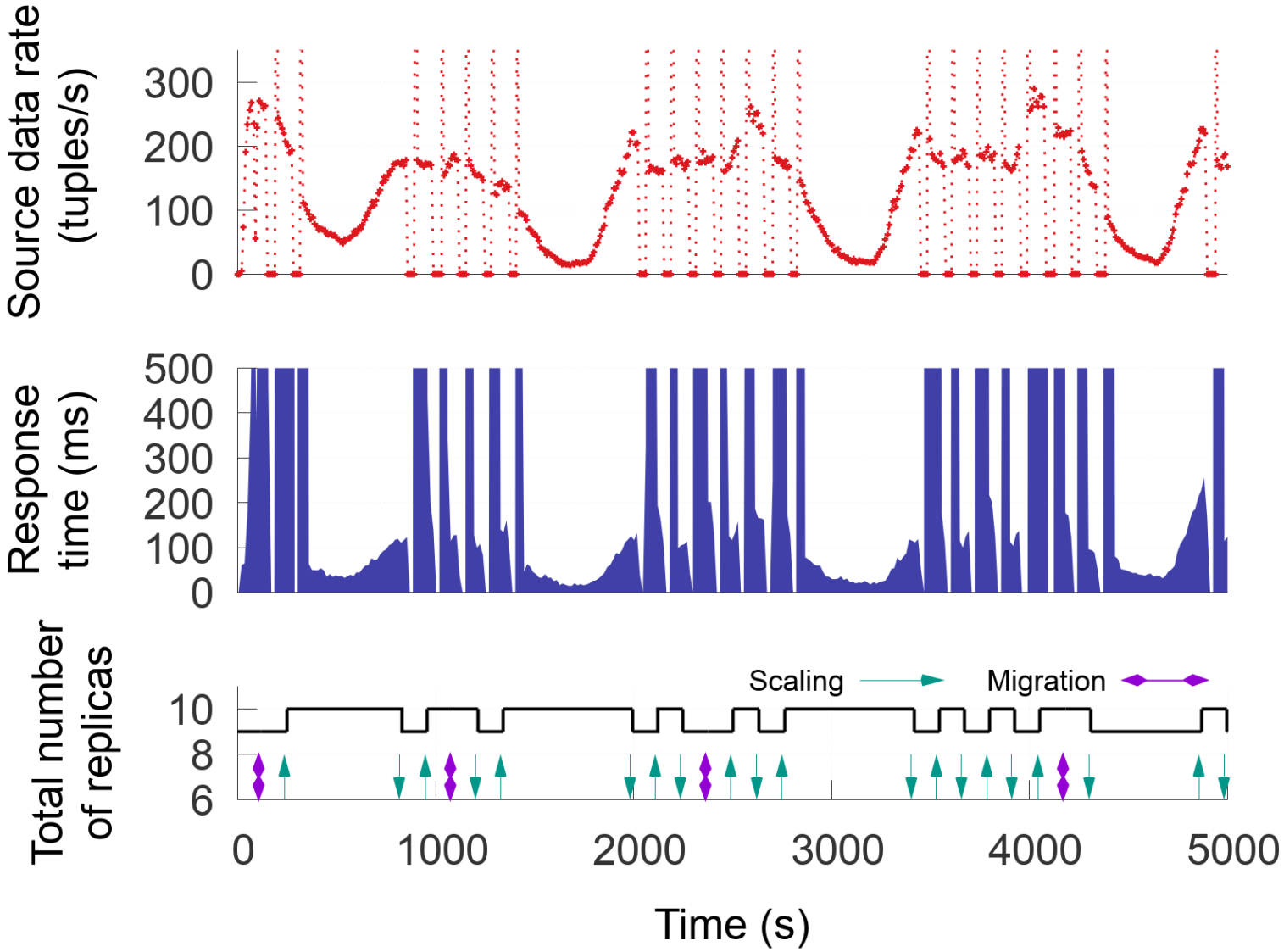


median of response time 130.6 ms

up and running 93.7% of time

Evaluation

Application Manager policy: 1 token/min if response time > **50%** R_{max}

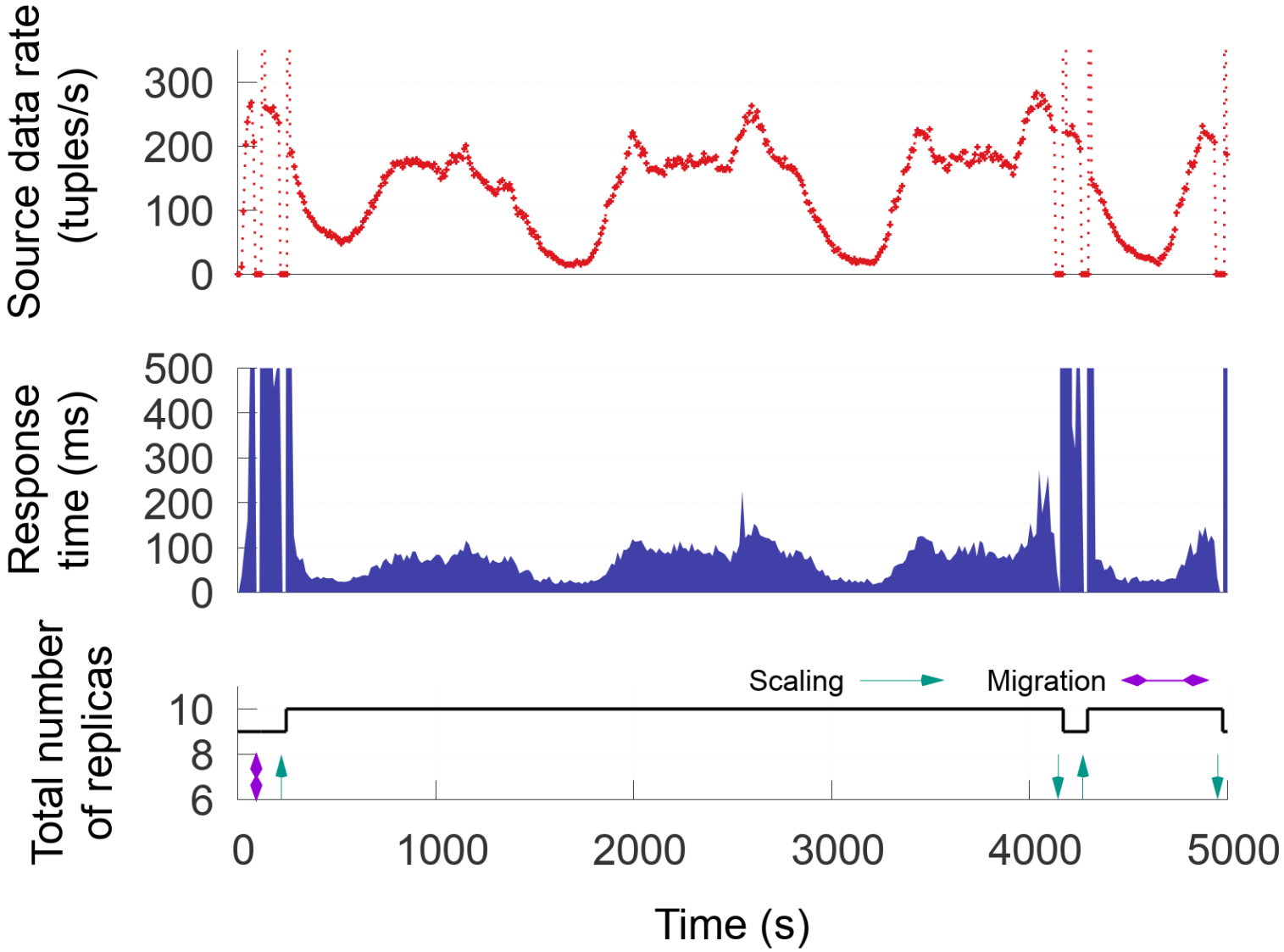


median of response time 117.0 ms

up and running 93.4% of time

Evaluation

Application Manager policy: 1 token/min if response time > **75%** R_{max}



median of response time 80.4 ms

up and running 98.3% of time

Conclusions

- We designed a hierarchical distributed architecture for the autonomous control of DSP applications
- We developed a simple control policy
- We integrated our solution in Storm
- We evaluated the effectiveness of our solution

Future Works

- Extend distributed heuristic: reduce oscillations, without compromising scalability
- Design new multi-time scale heuristics which capture the system dynamics (e.g., MDP) or learn from experience (e.g., Reinf. Learning)

Thank you!

Matteo Nardelli

nardelli@ing.uniroma2.it

http://www.ce.uniroma2.it/~nardelli