北京大學
PEKING UNIVERSITY

Fast and Fine-grained Autoscaler for Streaming Jobs with Reinforcement Learning

Authors: **Mingzhe Xing**, Hangyu Mao, Zhen Xiao*

June 2022

Resource Autoscaling

- Definition
  - **Dynamically** allocating computing resources, e.g., CPU, GPU or memory;
  - **Job-level** autoscaling and **Task-level** autoscaling, i.e., assigning resources to jobs or fine-grained tasks.
- Autoscaling methods
  - Heuristic-based methods
  - Reinforcement-learning-based methods

Motivation

- Fine-grained autoscaling
  - More **precise** resource management;
  - **Better performance** in multiple computing scenarios, e.g., 11-x faster execution speed for web services and 35% gain on GPU utilization by previous work.

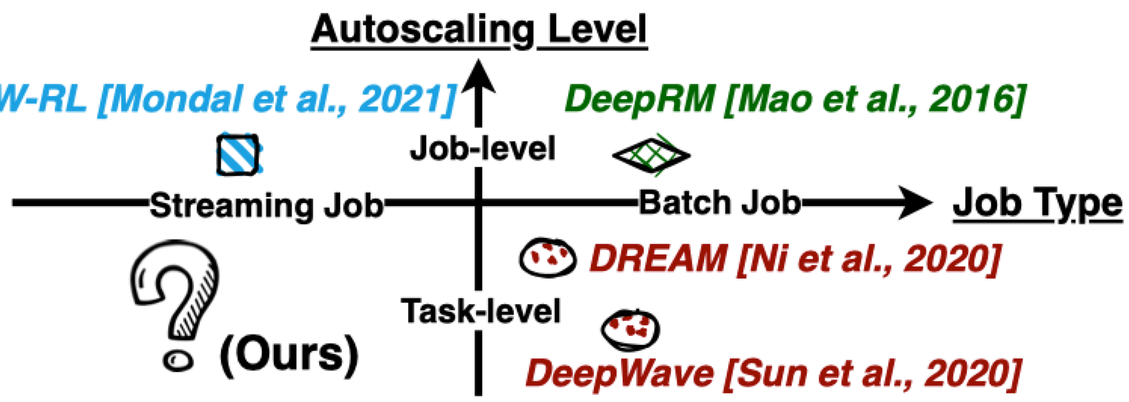


Figure 1: The categories of RL-based autoscalers, which have shown their superiority over heuristic-based methods in previous work.

Motivation

- Large temporal dimension
  - Running online for months or even years;
  - Produce massive records of job states;
  - Heavy computation overhead (stream computing is time-critical).

Markov Decision Process Definition

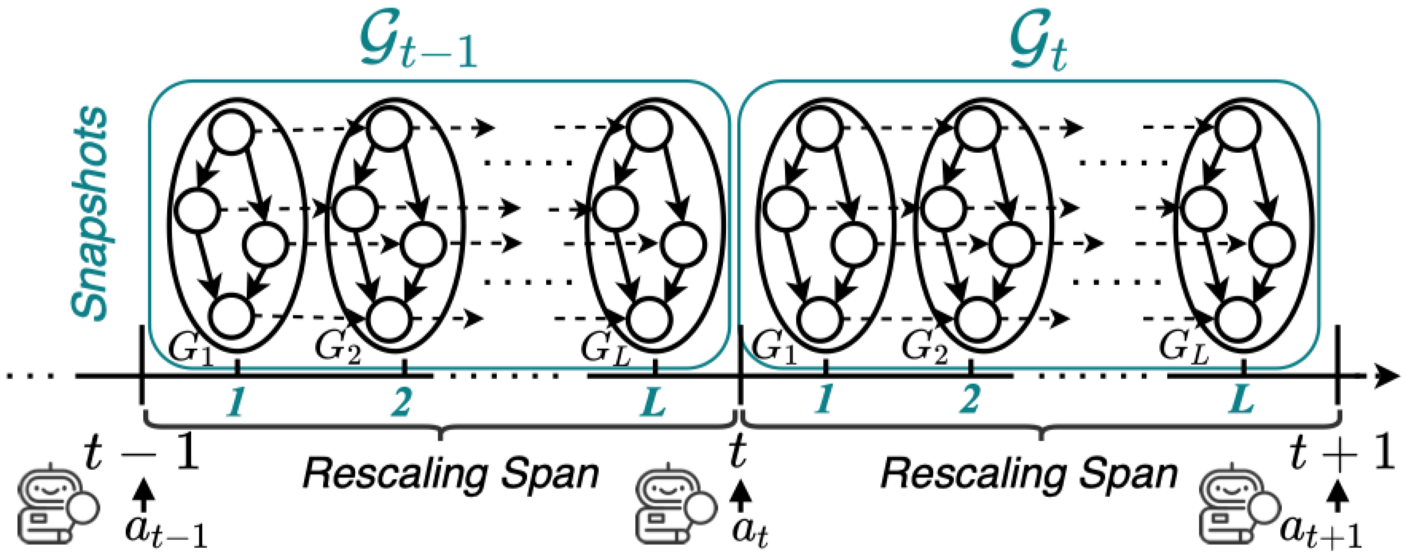


Figure 2: The MDP formulation of autoscaling process of streaming jobs. The running states of jobs (*i.e.*, snapshots) can be formatted as spatio-temporal graphs $\mathcal{G}$.

Optimization objection:

- minimize **latency**
- maximize **resource utilization ratio**

$$r_t = -\lambda l_t + (1 - \lambda) u_t$$

Neural Variational Subgraph Sampler

- Motivation: It is unnecessary to model all job state snapshots
- Subgraph sampling
  - Temporal dimension:
    - Weighted video stream sampling
    - Underlying **importance weights distribution** along temporal dimension
  - Spatial dimension:
    - Graph Neural Network
    - A **subset of spatial neighbors** is most relevant
- Pros:
  - Reduce redundant or noisy information
  - Lower computation cost

Neural Variational Subgraph Sampler

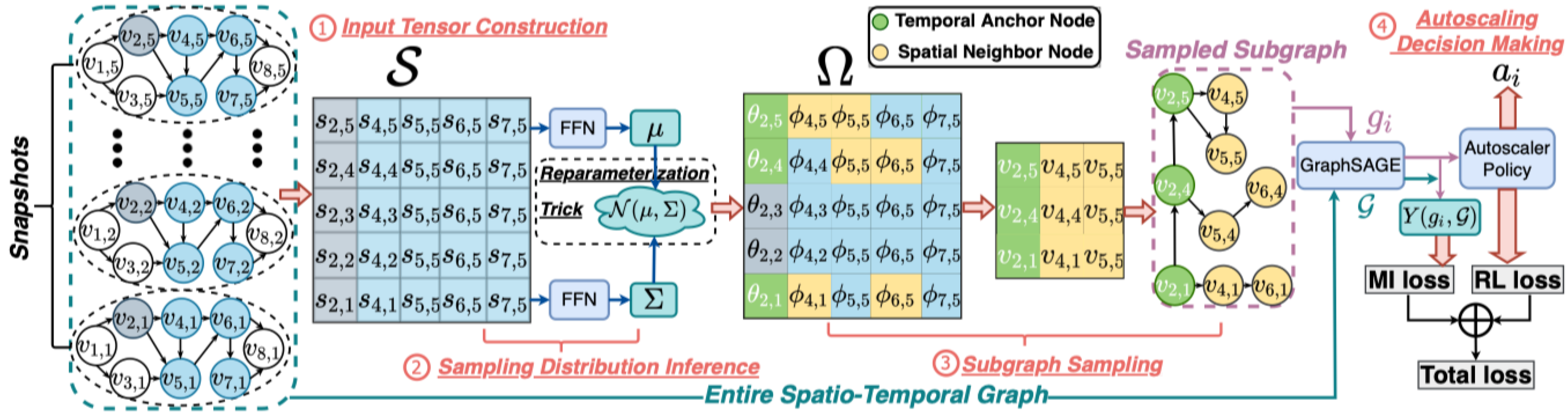


Figure 3: The overall architecture of our proposed approach. It shows an example to sample a subgraph for task node v_2, and then make autoscaling decision for this task node. L, K, k_1 and k_2 are set as 5, 4, 3 and 2 in this example. "FFN" denotes the feed forward network. The steps labeled with ①, ②, ③ and ④ correspond to the four steps introduced in Section 4.1.

Under this sampling procedure, the marginal likelihood of subgraph is

$$p(g_i|\mathcal{S}_i) = \prod_{l=1}^{k_1} \prod_{s=1}^{k_2} p(v_{l_s}|\phi_i^l) p(v_l|\theta_i) p(\theta_i, \phi_i|\mathcal{S}_i))$$

Subgraph Mutual Information

- Motivation: **explicitly** encourage to sample representative subgraphs
- Larger MI indicates that the two variables are more correlated

$$\max I\big(f(g), f(\mathcal{G})\big) = H\big(f(g)\big) - H\big(f(g)|f(\mathcal{G})\big)$$

- Optimization lower bound

$$\begin{aligned}
Y(g, \mathcal{G}) &= log\mathbb{E}_{g\sim p(g|\mathcal{S})} I\big(f(g), f(\mathcal{G})\big) \\
&\geq \sum_g \Big(-\mathcal{KL}\big(q(\Omega|\mathcal{S})||p(\Omega|g, \mathcal{S})\big) \\
&- \big(\frac{1}{2}log|\Sigma| + (\Omega-\mu)^T\Sigma^{-1}(\Omega-\mu) + CE(\Omega, \hat{\Omega})\big) \\
&+ \mathbb{E}_q logI\big(f(g), f(\mathcal{G})\big)\Big),
\end{aligned}$$

Training with Reinforcement Learning

- RL objective function:

$$J(\psi) = \frac{1}{N}\sum_{n=1}^{N} log\pi_{\psi} R$$

- Total loss:

$$\mathcal{L} = -J - \lambda_1 \sum_{i=1}^{|V|} Y_i$$

Experiments

- Implement a **simulation environment** for stream computing.
- Use ClarkNet Trace as **workloads**, which describes the number of HTTP requests to the servers.
- Select **jobs** in Alibaba Cluster Dataset that were running for more than 2,000 minutes.
- Sample six jobs with **different task numbers**:
  - Small-1, Small-2, Medium-1, Medium-2, Large-1 and Large2

Experiments

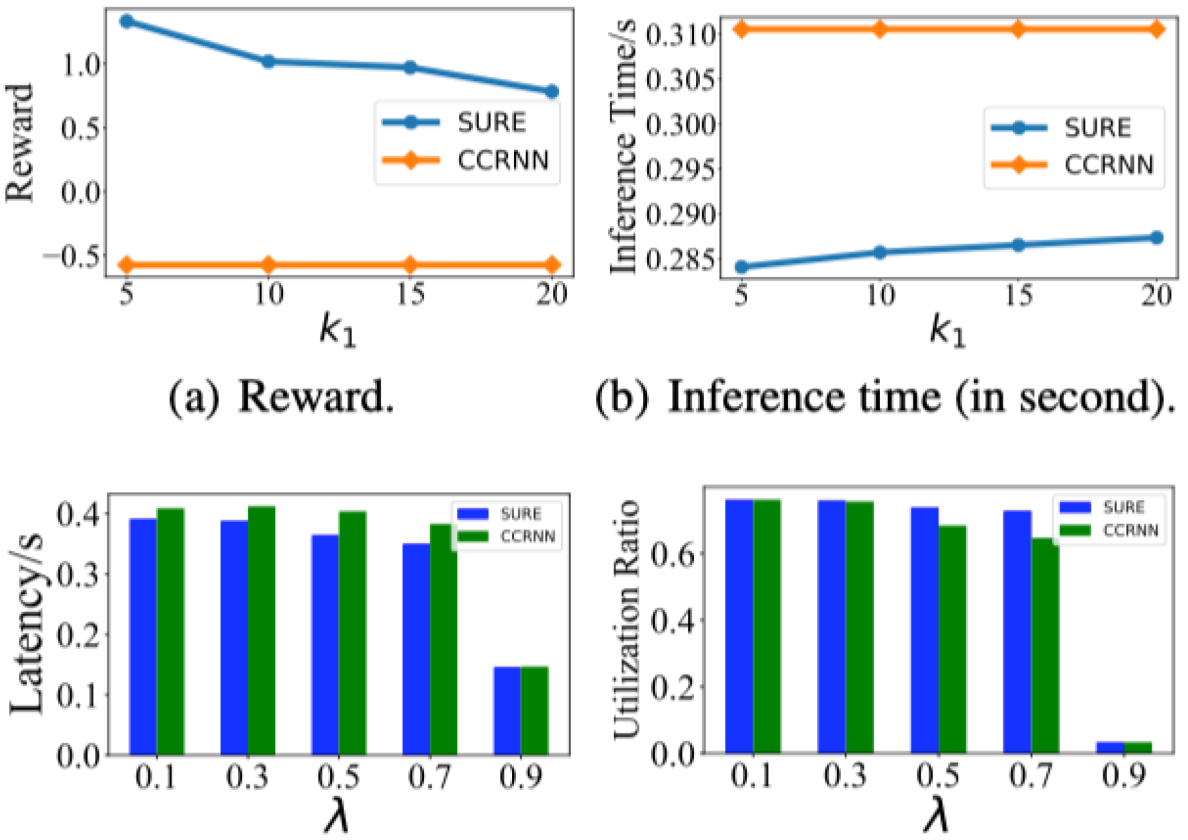
- Performance comparisons

		Small-1	Small-2	Medium-1	Medium-2	Large-1	Large-2	Average
Heuristic-based	HPA	-0.17	1.16	-2.69	-0.90	-1.28	-2.35	-1.04
RL-based	DeepWave	-2.77	-1.23	0.16	-0.97	0.69	0.32	-0.63
	DREAM	0.50	-0.23	0.23	-0.11	0.92	-1.14	0.03
	TVW-RL	0.26	0.66	0.08	-0.40	0.95	0.85	0.40
Spatial-temporal GNN	ASTGCN	0.26	-0.66	0.36	1.09	-1.24	0.29	0.02
	CCRNN	0.48	0.97	0.24	1.12	-0.57	0.46	0.45
Ours	SURE	**0.52**	**1.41**	**1.19**	**1.81**	**1.02**	**0.95**	**1.15**

Table 1: Performance comparison with baselines on *Small*, *Medium* and *Large* job settings, respectively. The best, second best and third best results are in bold, underline and gray cell, respectively.

Experiments

- Parameter sensitivity
  - Size of subgraphs
  - Weights of latency and utilization ratio



(a) Reward. (b) Inference time (in second).

(c) Latency (lower is better). (d) Utilization Ratio (higher is better).

Figure 4: Sensitivity analysis by varying k_1 and λ for Large-1 job.

Conclusion

- Contributions:
  - We are the first to give an **MDP formulation of autoscaling streaming jobs.**
  - We design a **Neural Variational Subgraph Sampler**, which can greatly **save the graph learning time.**
  - We propose an objective function based on **mutual information** to guide the sampler to **extract more representative subgraphs.**
- Future Work:
  - We will apply our method to solve other classical spatio-temporal graph modeling tasks, such as traffic forecasting and pose detection, which also **suffer from the large temporal dimension issue.**

Thank you