Multi-agent Reinforcement Learning for Job Scheduling & Data Allocation

Jan. 15, 2025

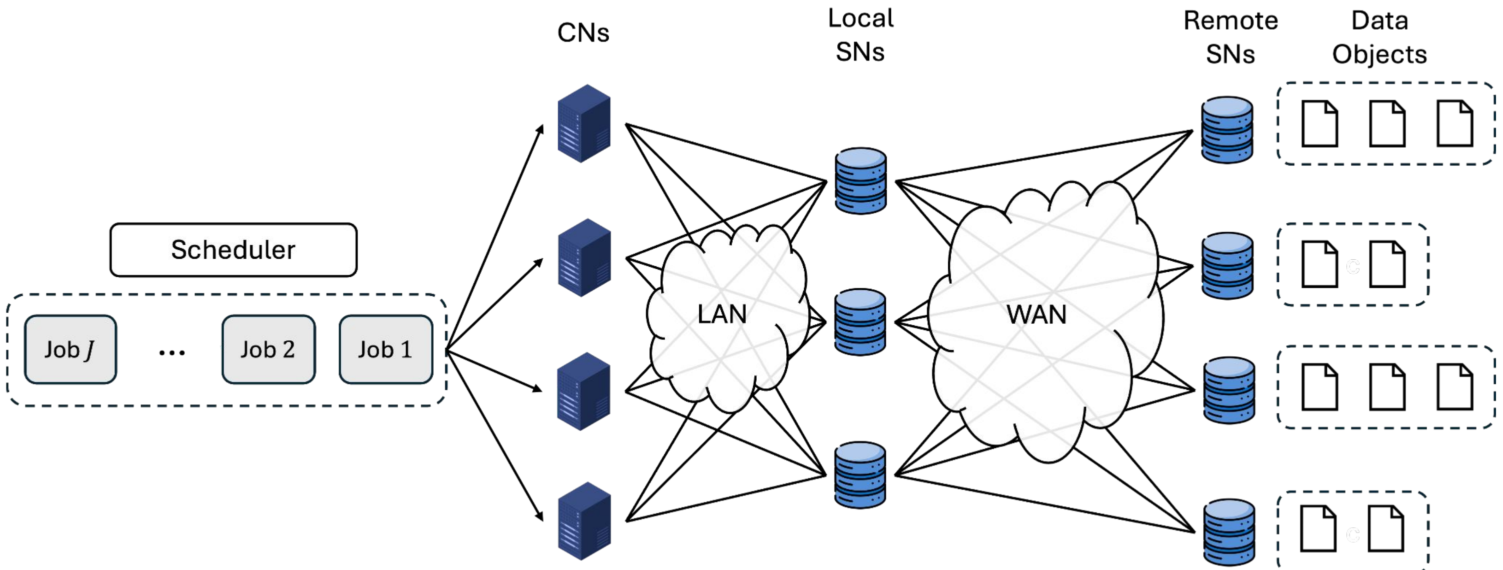
Shengyu Feng

(CMU)

Problem Setup

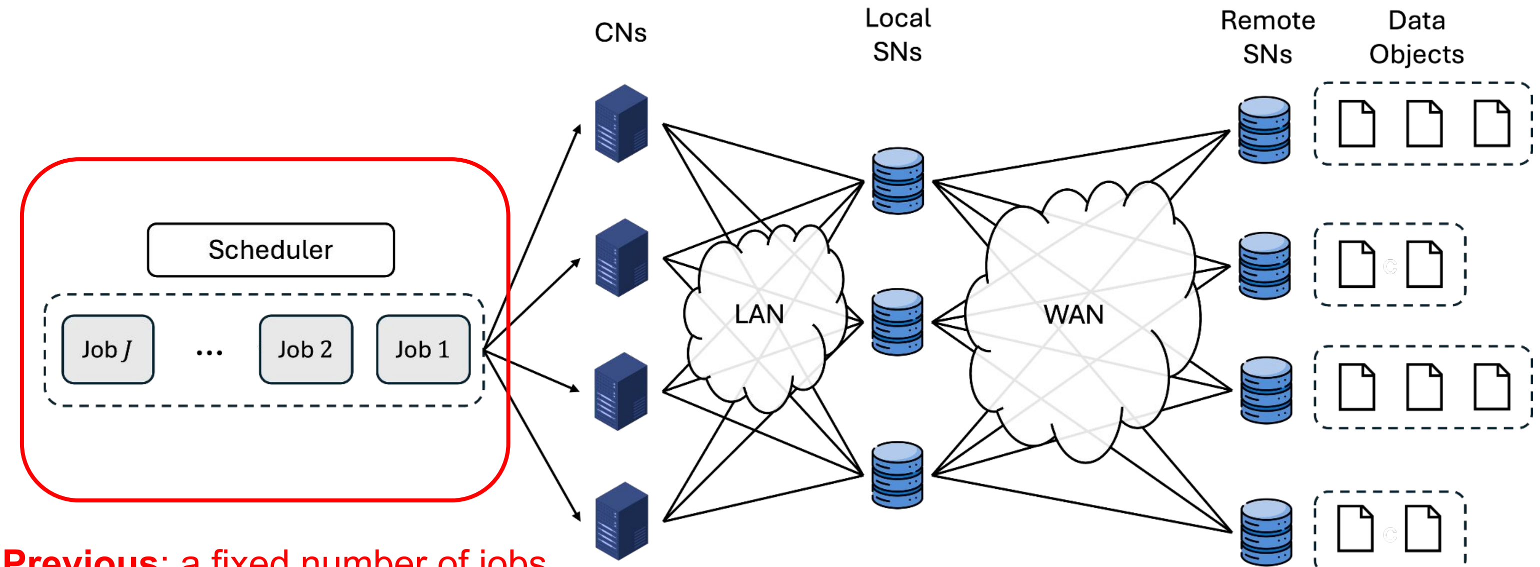
previous: completion time

- **Goal**: maximize the throughput (# data processed/time unit) of the system



Problem Setup

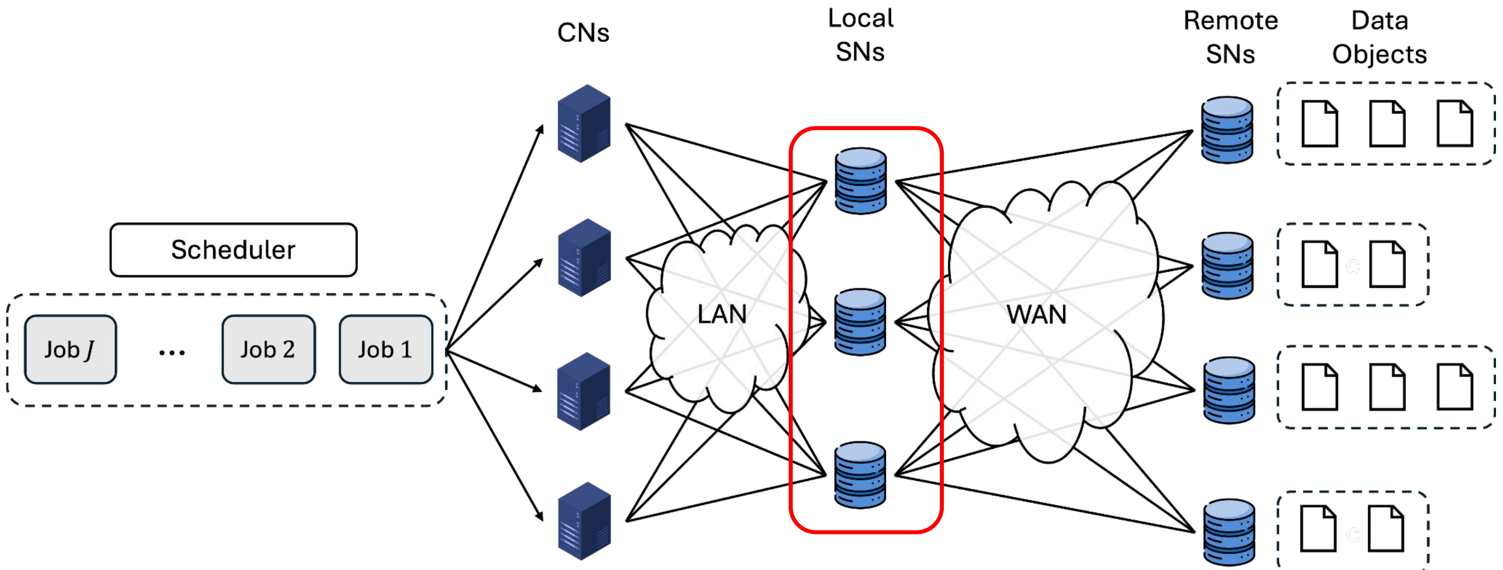
- **Goal**: maximize the throughput (# data processed/time unit) of the system



Previous: a fixed number of jobs
Now: jobs keep coming as a function of time

Problem Setup

- **Goal**: maximize the throughput (# data processed/time unit) of the system



Previous: one copy of each data object
Now: limited storage + unlimited copies

Job Distribution

Job parameters (file size, job type, et al.)

- Start with simple distributions like normal or exponential distributions
- Further extended to learnt distributions like in David's work

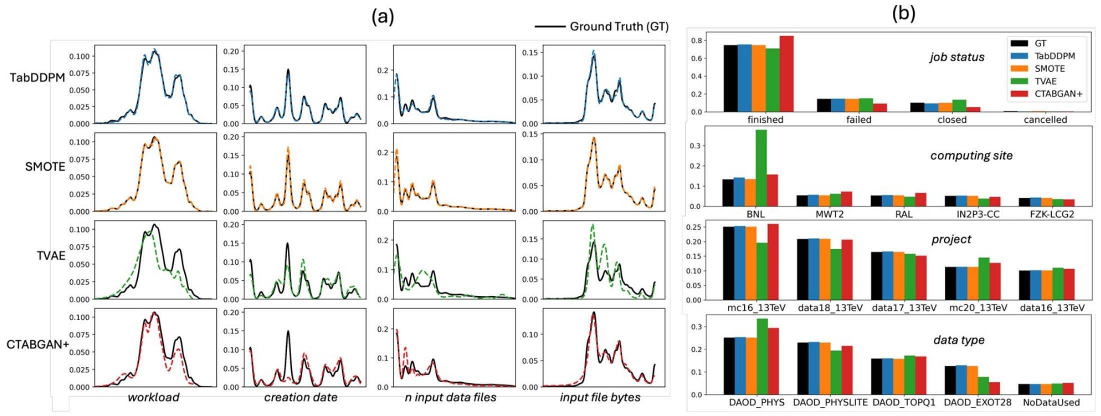
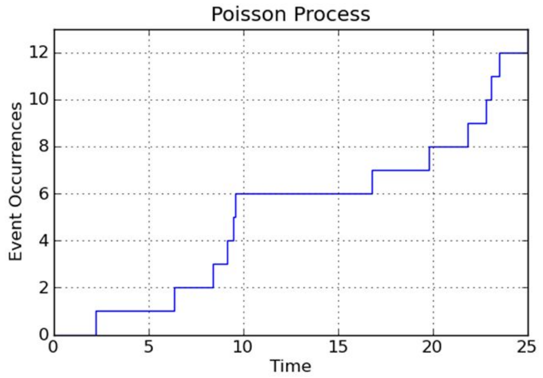


Fig. 4. Comparisons of generative performances based on distributional similarities of individual features. (a) Distinct columns show each of all four numerical features used as training inputs, while individual rows correspond to a model. Black and dotted color lines correspond to ground truth (GT) and synthetic data, respectively. (b) The graphs are comparing if distributions are similar for unique entries with top counts across four categorical features.

Job Distribution

Arriving time

- Assume the jobs submitted by each user follow the **Poisson process**



- Also assume there are N users, each has different job parameter distributions

Previous: AlterMILP

- **Idea: Alternating optimization** by fixing one variable as constant
    - If **variables are splitted** ($A_{j,c}$ vs. $H_{i,j}$,$B_{d,s}$), then problem **becomes MILP again**

$$w_d = \sum_{s=1}^{S} td_1(d,s)B_{d,s}, \quad \forall d \in [D]; \tag{5}$$

$$l_j = \max_{d \in O_j} \left(\max\{w_d, f_j\} + \sum_{s=1}^{S} \sum_{c=1}^{C} \boxed{td_2(d,s,c)A_{j,c}B_{d,s}} \right), \quad \forall j \in [J]; \tag{6}$$

$$f_j \geq V\left(\boxed{H_{i,j}(A_{j,c} + A_{i,c} - 1)} - 1\right) + (l_i + e_i), \; \forall i \neq j, \; i,j \in [J], c \in [C] \tag{7}$$

$$e_j = \sum_{c=1}^{C} exe(j,c)A_{j,c}, \quad \forall j \in [J]; \tag{8}$$

$$T \geq l_j + e_j, \quad \forall j \in [J]; \tag{9}$$

$$H_{i,j} \in \{0,1\}, \quad \forall i \neq j, \; i,j \in [J]; \tag{10}$$

$$A_{j,c} \in \{0,1\}, \; f_j, l_j, e_j \geq 0, \quad \forall j \in [J], \; c \in [C]; \tag{11}$$

$$B_{d,s} \in \{0,1\}, \; w_d \geq 0, V >> 0, \quad \forall d \in [D], \; s \in [S]. \tag{12}$$

Difficulty in Mixed Integer Linear Programming

Problem parameters are probabilistic, intractable to compute

$$w_d = \sum_{s=1}^{S} td_1(d,s)B_{d,s}, \quad \forall d \in [D]; \tag{5}$$

$$l_j = \max_{d \in O_j} \left(\max\{w_d, f_j\} + \sum_{s=1}^{S}\sum_{c=1}^{C} td_2(d,s,c)A_{j,c}B_{d,s} \right), \quad \forall j \in [J]; \tag{6}$$

$$f_j \geq V\big(H_{i,j}(A_{j,c} + A_{i,c} - 1) - 1\big) + (l_i + e_i), \; \forall i \neq j, \; i,j \in [J], c \in [C] \tag{7}$$

$$e_j = \sum_{c=1}^{C} exe(j,c)A_{j,c}, \quad \forall j \in [J]; \tag{8}$$

$$T \geq l_j + e_j, \quad \forall j \in [J]; \tag{9}$$

$$H_{i,j} \in \{0,1\}, \quad \forall i \neq j, \; i,j \in [J]; \tag{10}$$

$$A_{j,c} \in \{0,1\}, \; f_j, l_j, e_j \geq 0, \quad \forall j \in [J], \; c \in [C]; \tag{11}$$

$$B_{d,s} \in \{0,1\}, \; w_d \geq 0, V >> 0, \quad \forall d \in [D], \; s \in [S]. \tag{12}$$

Switch to Reinforcement learning (RL)

Challenges

C1: The job arriving time is a continuous function of time

Solution: We would discretize it into sequential time intervals (approximation error happens here)

C2: The current state is dependent on previous state (known as Partially Observable Markov Decision Process)

Solution: We can take the historical information into account (trade-off between the computational cost & information)

C3: Heterogeneous action space (job scheduling, data replication)

Solution: Use multi-agent RL (complex than single-agent RL)

Discretization

Divide the time into discrete time slots (question: what would be a proper unit for the time interval here, i.e., on avg. how long does a job come?)

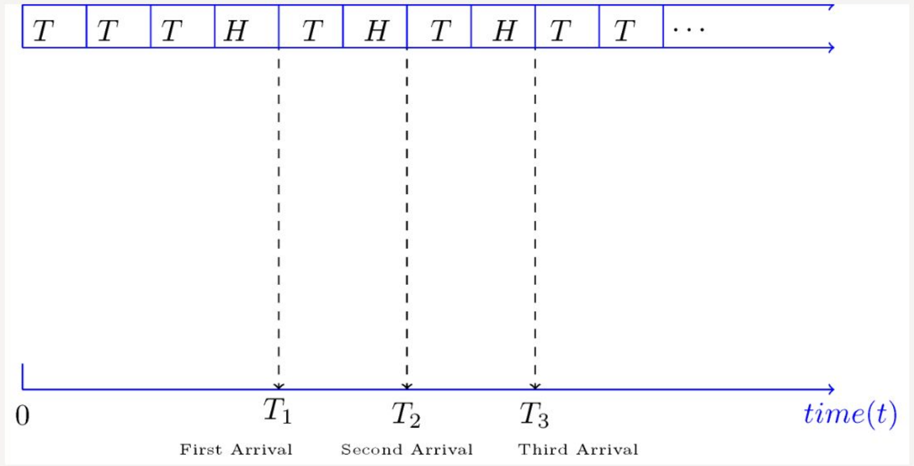
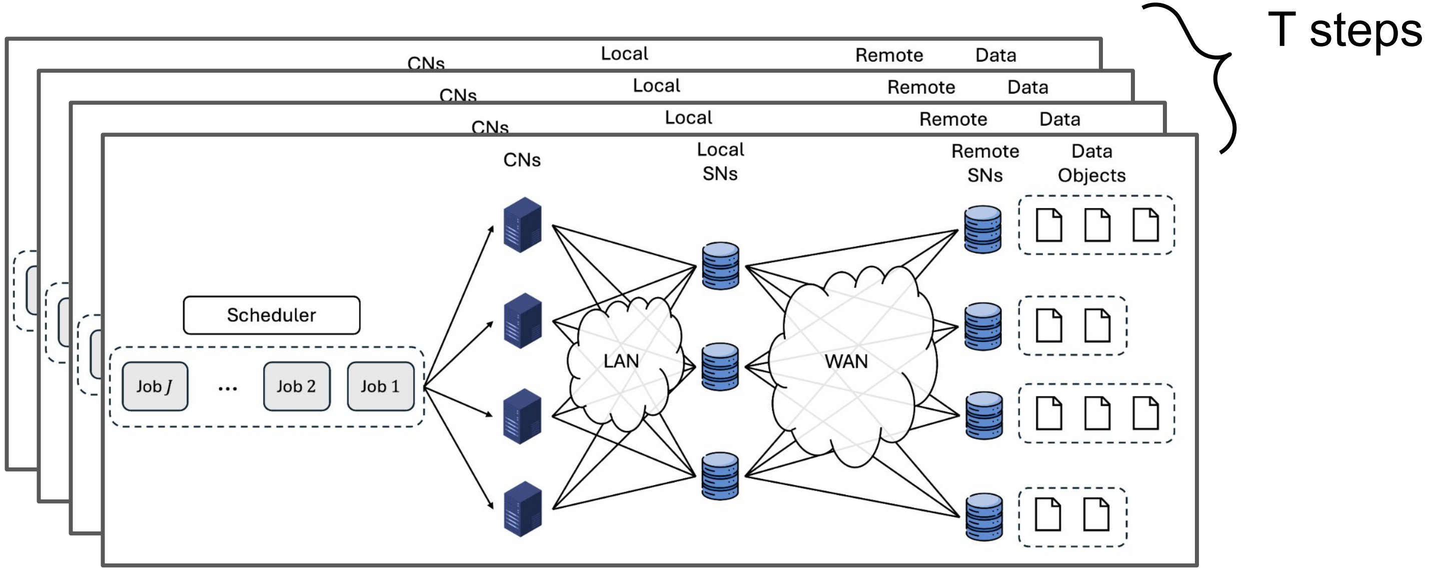


Figure 11.3 - Poisson process as a limit of a Bernoulli process.

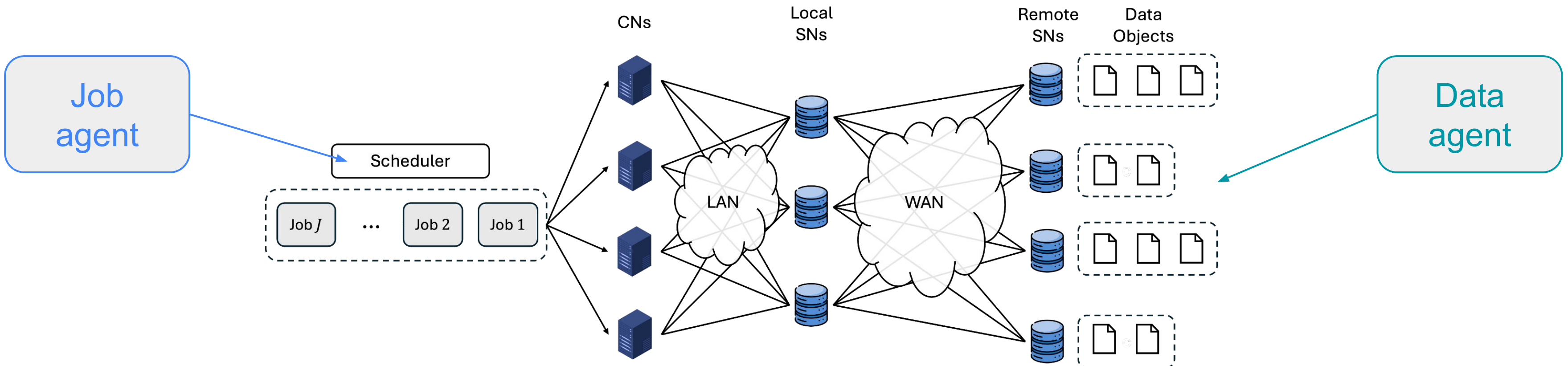
Reinforcement Learning Formulation: State

job information in the past time T steps



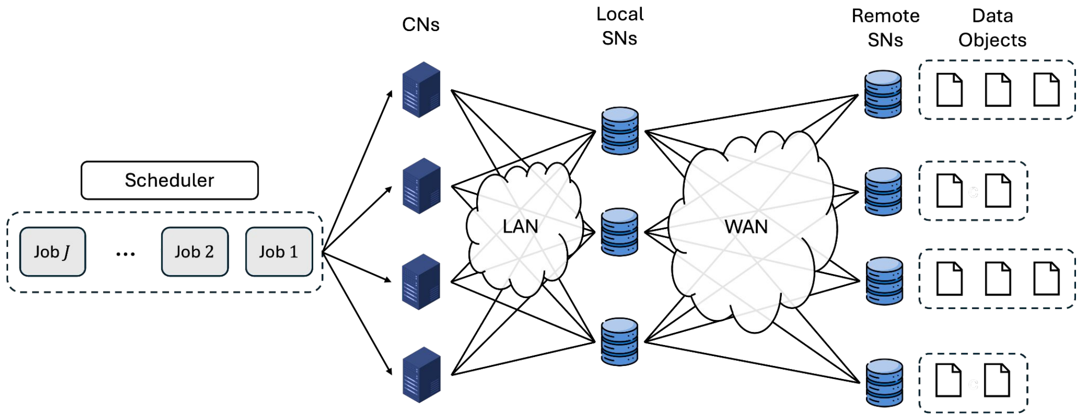
Reinforcement Learning Formulation: Action

- Job agent
  - Decide the priority of a job
  - Assign the computational node of a job
  - Wait (make the decision until more jobs come)
- Data agent
  - Replicate the copy of a data object on a local storage node
  - Wait (make the decision until more jobs come)



Reinforcement Learning Formulation: Rewards

- **Joint:** throughput within a time period (sparse)
- **Job agent:** negative CN idle time/job waiting time
- **Data agent:** negative data transfer time



Baselines (Tentative)

Method	**Type**
First Come First Served	Static heuristic
First Scheduling	Static heuristic
SLAQ	Dynamic heuristic
TRADL	Dynamic heuristic
RLPTO	Dynamic heuristic (RL)